WMT –

"Whac-A-Mole" Like Game

CSEE 4840 Embedded System Design

Lingchuan Mei: lm2908@columbia.edu

Jian Lu: jl3946@columbia.edu

Shuting Yang: sy2489@columbia.edu

Si Wang: sw2778@columbia.edu

Acknowledgement

We WMT team would thank dear Prof. Stephen A. Edwards and Professor David Lariviere for all the professional instructions and kindly help.

We really appreciate the hard work of the TAs.

Table of Contents

Table of Figures

1. Overview

Throughout this semester, we designed and implemented a “Whac-A-Mole” like game system using a TRDB-LTM touch panel. The user will interact directly with the panel while playing the game. Apart from the main game logic, we also introduced audio output along with the game to make it more fancy and commercial oriented.

Figure 1 LTM Panel

The game consists of four different scenarios. First one is a welcome page that enables the user to choose from one of the three difficulty levels and press “enter” to go to the main game page. The second page is where all the funny part happens. There are nine holes from which the moles might come out randomly at a certain rate. Not only the moles, we have also introduced baby bottles for supplement and evils that trying to destroy you. The game will run for two minutes for a round, then the last two background will show up to notify you result depending on your performance previously. After 5 seconds, the user will see the welcome page again and the system is ready for the next round.

2. Project Design

Overall system architecture is shown as follows:

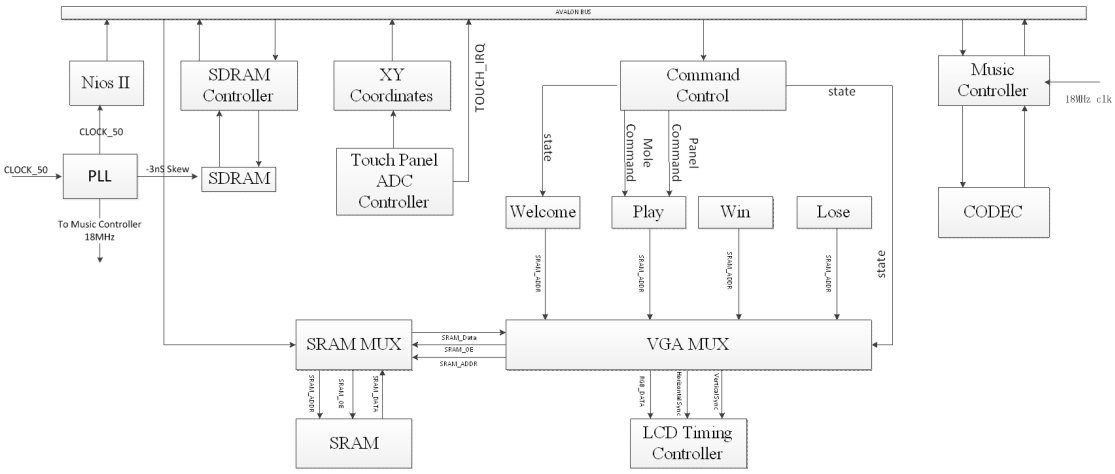


Figure 2 System Architecture

3. System Implementation

3.1 Hardware

1. Overview of the system

   The system contains three basic functions that are NIOS II CPU, ADC controller that detects the touch position and generates interrupt and an 800 x 480 resolution display. The CPU core operates at 50 MHz and the display operates at 25MHz. The CPU used SDRAM as memory and SRAM as storage. In order to use the SDRAM as memory, a PLL is needed to generate the -3nS skew clock for the CPU. Another 18MHz is also generated from the PLL.

2. Modules Used in the Hardware System

Module	Function
Adc_spi_controller.v	Control the ADC output mode
Audio_controller.v	Control the audio output
Command.v	Used to receive and distribute

	command to sub-sequent modules
Control_panel.v	Controls all the output on the control panel
DE2_TOP_PROJECT.v	Design TOP-Level
interrupt.v	Generate interrupt for touching
irq_gen.v	Generate interrupt for touching
lcd_controller.v	Lcd timing controller
lcd_spi_controller.v	Used to set the display mode of the display
lcd_timing_controller.v	Control the output HS, VS and RGB to the LCD display
letter.v	Used to display the letter
life.v	Used to display the life
lose.v	Used to display the lose message
mole_controller.v	Used to control the action of mole
musiscontroller.vhd	Used to control the music
No_GEN.v	Used to display the touch position x and y coordinate on DE2 LEDs
number.v	Used to display numbers

panel_control.v	Used to display all panel information
pause.v	Implement the pause button
play.v	Implement the Play state interface
quit.v	Implement the quit button
score.v	Used to display the score
reset_delay.v	Set the reset delay for different modules
SEG7_LUT_8.v	Display the numbers on 8 LEDs
SEG_LUT.v	Display the numbers on LED
sram_mux.v	Used to select which vga signal to output
sram.v	Sram controller
start.v	Start button
symbol.v	Used to display symbol
three_wire_controller.v	Used to set up the LCD display mode
Timmer.v	Timer on the display
welcome.v	Welcome state vga
win.v	Win state vga

xycoord_interface.v	Transmit x and y coordinates through Avalon bus.

3. Memory requirement

Content	Size	Location stored
Background(Mole Position)	64000	SRAM
Background(Control Panel)	28800	SRAM
Mole	28800	SRAM
Milk Bottle	28800	SRAM
Bomb	28800	SRAM
Indicator(Level selection)	7200	SRAM
Red Heart	7200	SRAM
Gold coin	7200	SRAM
Clock	7200	SRAM
Letters(26 letters)	2048 x 26	SRAM

4. Sprites

Sprites Catalog	Number of Sprites

Mole	2
Milk Bottle	1
Bomb	1
Gold Coin	1
Red Heart	1
Clock	1
Letters	26
Background	2

All the sprites were pre-processed using MATLAB. In our case, the SRAM only output 16 bits number at a time. So all the image sprites used in the game was transformed from 24 bit true color to 16 bit high-color. The MATLAB code can be found in the project package. To get rid of the white back ground, a matrix of 1s and 0s were generated to represent if it's a white spot. If it's a white spot, it will be replaced by the background color.

5. Peripherals

    1. LCD timing controller

        LCD timing controller is used to generate Hcount and Vcount， which generates Hsync and Vsync. The RGB data from VGA MUX is passed to the display through LCD timing controller.

    2. Touch Panel ADC Controller

        Touch panel takes the pressure when user presses the screen and generate

two 12 bit digital numbers which represent the x and y coordinates. It also generates an IRQ which will be used in the software to detect state transition or user actions.

3. XY Coordinates

   XY Coordinates is used to take the data from touch panel ADC controller and pass the data on to the Avalon bus. Software will take the data and do the next step procession.

4. Command Control

   Command control takes the data from the CPU and writes them into the assigned registers. There two type of commands. One is Mole_ctrl, which is used to control the mole activity. In other words, it controls whether there is a mole, milk bottle or bomb and if it's hit or not. The other command, which we call Panel_ctrl, it contains all the information that we need on the control panel, like score, life and current state. By decoding these two commands, the system will be able to switch between different state and conduct the game logic.

5. Welcome, Play, Win, Lose

   This four peripheral contains all the basic information the game need in four different states. Like in the Welcome, we have the initial interface for users to choose the game difficulty they want. In the Play state, users can play the game, pause the game and quit the game. If the user wins the game, Win state will pop up. Otherwise, lose state will show up.

6. SRAM MUX

   At the beginning of the game, we have to write all the data we need into the SRAM. Data are basically background images, sprites, and some colors we need. This was done by using the SDRAM as memory and writing all the data into the SRAM through Avalon bus at the beginning of the game. Once all the

data has been written to SRAM, the SRAM is automatically disconnect from the Avalon bus and connected to VGA MUX.

7. VGA MUX

   VGA MUX is used to choose which states to load. It's controlled only by current state. It takes the address from the proper state and uses it to read the colors from SRAM.

6. Lesson learned

- ELF error is usually due to wiring mistake in the top level connection of NIOS.
- Every time you copy an entire project, remember to recreate the nios project. Otherwise the BSP package is mapped to the old SOPC.
- Should have design the interface more easy to use. Should split all the variables instead of putting all of them together
- Build the design using small modules.

3.2 Display Dynamic Items

For the implementation of Verilog programs to control all the dynamical elements appearing and disappearing as well as the nine holes on the touch panel, we got the signal from the software, which indicate which element needs to appear or disappear. Then we used the first 27 bits of the 32-bit input for the identifier and the state of each element in the nine holes, three for each. For each three-bit, two of them represent that element is; 00 for the mole, 01 for the evil, 10 for the baby bottle. The third one is the state of the element; 1 represents going up while 0 represents going down. For all the period while a certain element need to be presented on the screen, we would get a 1 on the third bit of that hole. Once the element is hit or timed out, we would receive a 0 immediately.

The logic is as follows.

We defined three layers for each cell on the screen (a cell consists of the background, the hole and the space for the element to be placed). The lowest layer is the grass

background and the upper half of the black hole. The middle layer is the element, while the highest layer is the lower half of the black hole.

Instead of just appearing on the screen suddenly, we would make it go up one pixel by one pixel to mock the real situation when we get the signal that an element need to appear. By increasing the height of the element starting at 0 till 120 pixels could achieve this.

When a 0 on the third bit is detected, the height of the item will be set to 0, which means it will disappear on the screen.

We have downloaded the pictures of the three items from the Internet as shown below.

Figure 3 Items for the game

We used Matlab to get the matrix representations of the pictures in the first place. In order to display the item without the white edge, we wrote a Python script to calculate the edges of the item for each picture and then feed the edge information into the Verilog. When an item needs to be printed on the screen, the program will check whether the pixel being printed is inside the edge of the item, if so, the overwrite signal will be sent, otherwise, the hardware will just print the background grass or hole accordingly.

3.3 Software

In the software part, the main task is to corporate with the hardware and music to make our touching screen work.

The procedure is described as followed:

(a) User chooses the game mode among Easy, Normal and Hard on the Welcome page.

(b) After pressing "Start", the game starts. The item to be shown among mole, bottle and bomb and the position of the item is randomized. Every mole appears for one second.

(c) When a state change is detected, it goes to Pause, Quit or Item status.

(d) In the Pause status, the timer is stopped and the item is held at the original position.

(e) In the Quit status, it jumps to the Welcome page.

(f) In the Item page, there are three situations:

- Item is mole: The total score adds five points if the mole is hit, otherwise one life is deducted from the lives.

- Item is bottle: The lives add one if it is smaller than three.

- Item is bomb: One life is deducted from lives. If the lives are equal to zero, it goes to the Lose page and goes to the Welcome page after remaining for 5 seconds.

(g) Win page appears if the game last more than one minutes.

Here is the block diagram of the game mechanism:

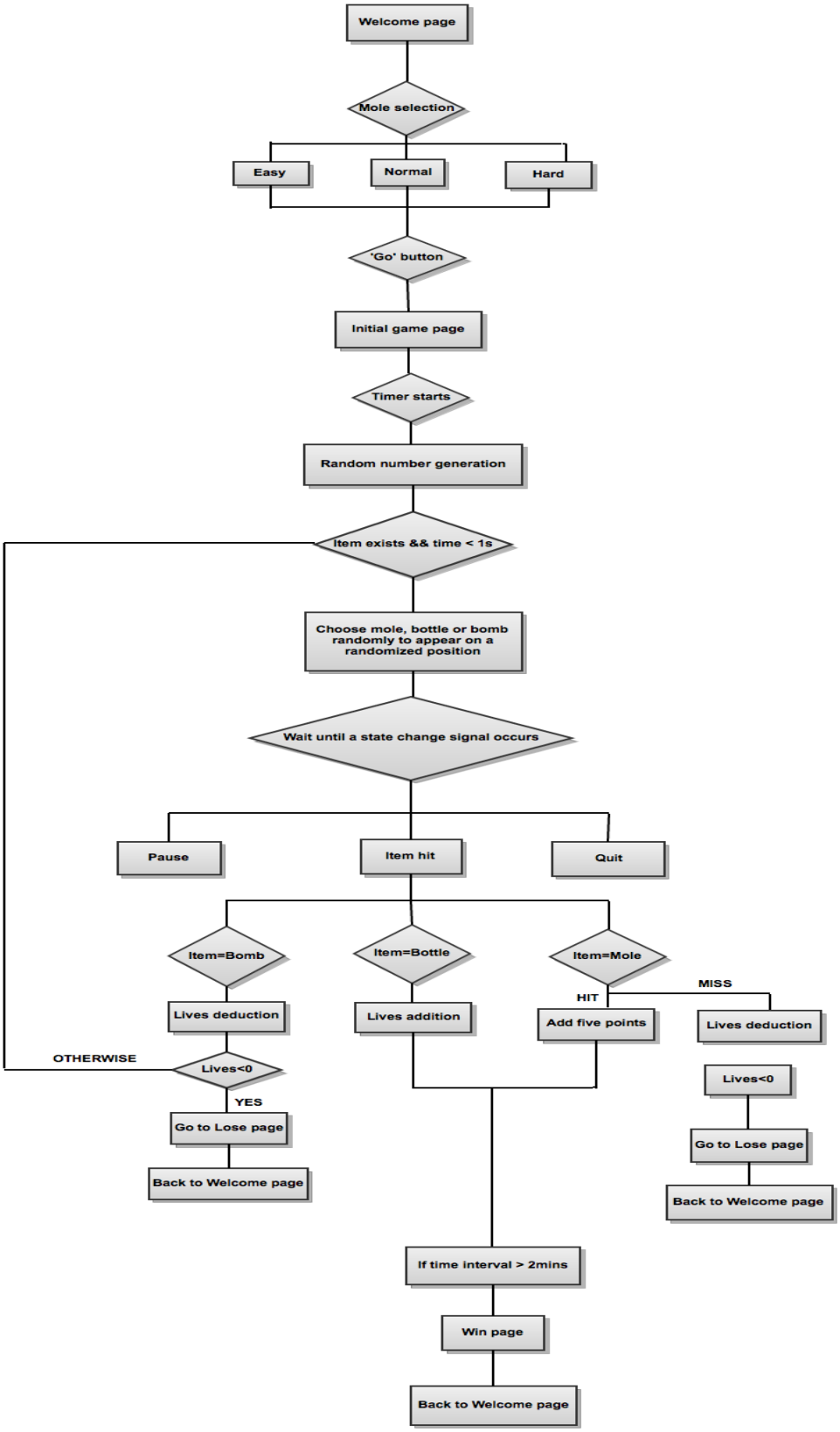


Figure 4 Software block diagram

3.4 Audio

In this game we have two kinds of audio: the game music and the game sound effects. All the sound tracks are downloaded from internet converted by matlab, and stored in the SDRAM as .h files (we used SDRAM as the CPU memory because it has enough space for the music), so that we can directly access them in the software level instead of read from other memory.

Audio Architecture:

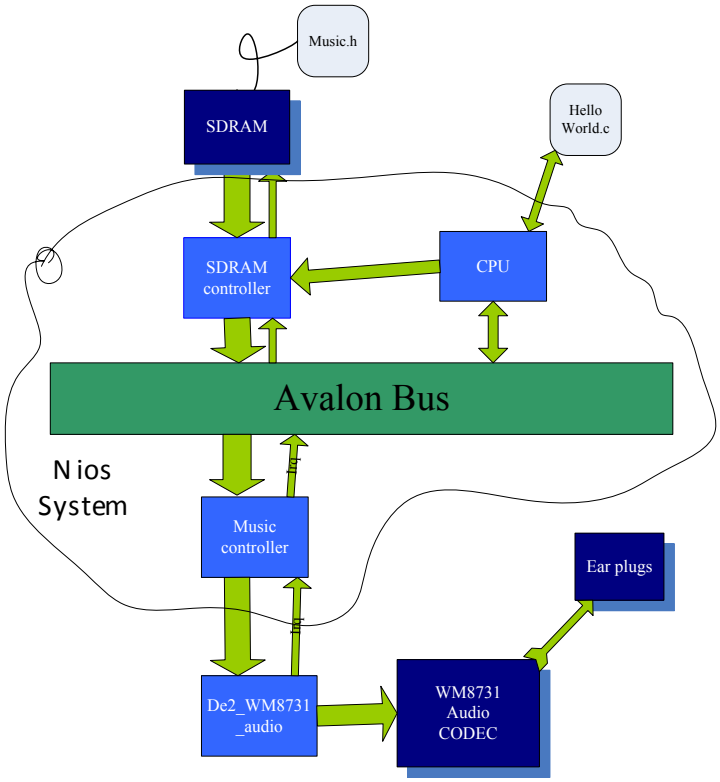


Figure 5 Audio Architecture

As shown in the picture, CPU controls SDRAM to transfer audio data to audio part through nios-system (sopc). Parts are described below:

— WM8731 CODEC is the audio function chip on the DE2 Board.

- de2_WM8731 is the vhdl code that controls the chip by providing it clocks and shifting audio data into it.
- Music controller is the interface VHDL that maps the audio part into the Nios-system.
- SDRAM controller is the interface VHDL that maps the SDRAM into the Nios-system.
- SDRAM and CPU are self-explained components which if you don't understand you'd better drop the class and choose Agriculture as your major.
- Main.c is your weapon to command audio part when to play what; Music.h is where your melody waiting to be played.

Key issues

How to make it sound.

Our digital sound data are actually arrays of 16-bit sample values, if we align the sample value in sequence, they will appear as sound wave. WM8731 audio chip makes sounds by align sample values in sequence and convert them into analog signals.

To make the WM8731 work, understanding its controller de2_WM8731_audio is critical. de2_WM8731_audio.vhd has two main jobs: provide clocks, and shift audio data bits into the WM8731 audio chip. Data bits are shifted one bit at a time into the Audio chip(left and right channel), change between channels every 16 bits(same 16 bits), after the 16 bits are shifted into both L/R channel, it reads another 16-bit sample and keep on shift in. There're two important clocks needed to ensure the bits are shifted properly---LRCK and BCK. BCK is Bit-stream clock, it controls the pace that data bits are shifted into the WM8731 chip, one bit per BCK clock cycle. LRCK controls the alternation between L/R channel, when LRCK = 1, bits are shifted into the L channel, and shift in to R channel when LRCK = 0. Both LRCK and BCK are generated from the proper working clock of de2_WM8731_audio.vhd---18.432M Hz.

To generate the bck and lrck, there are many other signals used, such as LRCK_divider, BCK_divider, set_lrck, etc. Words are not enough to express what I want to tell, see the timing diagram:

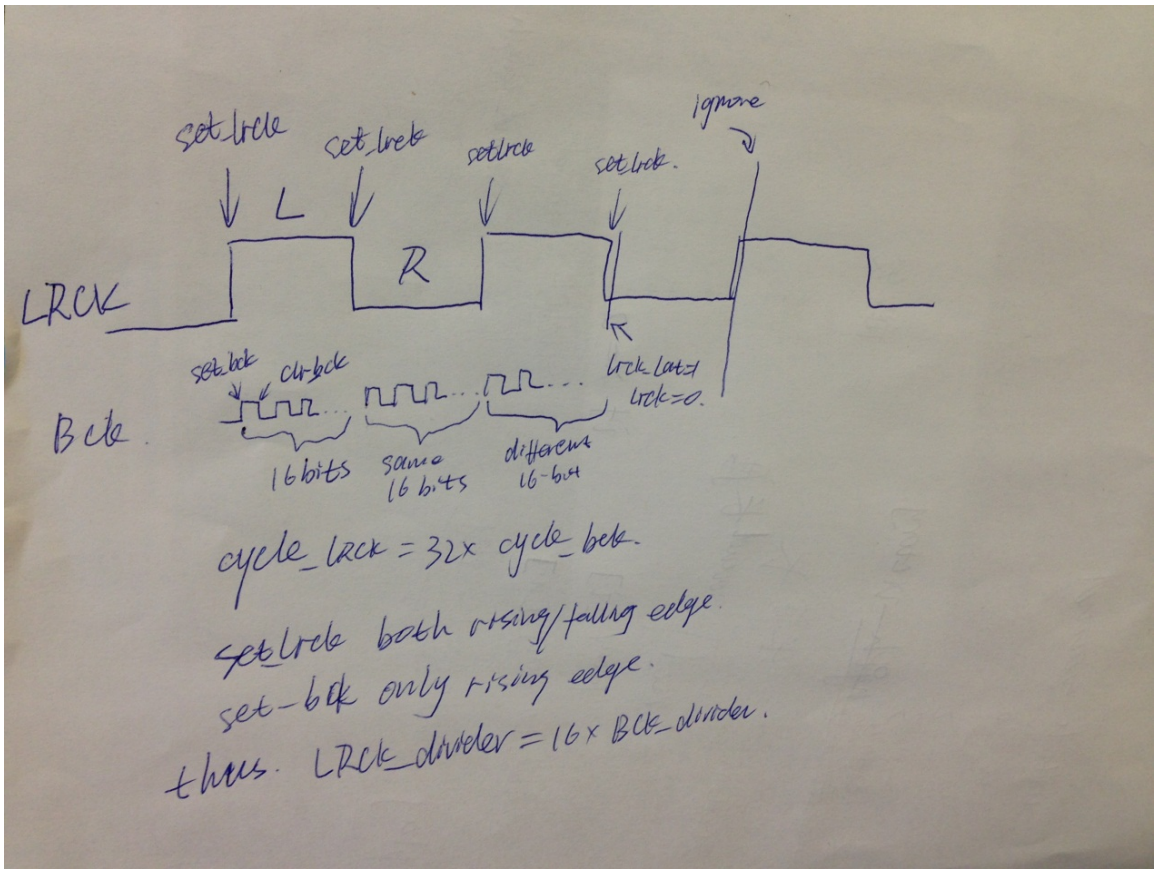


Figure 6 Timing diagram

In the diagram, you can see the relation between LRCK_divider and BCK_divider.

Another important thing is the choice of sample rate. With higher sample rate, you can get more accurate sounds, and of course, higher working clock frequency, which means smaller divider. The calculation is as follows:

LRCK_divider = 18.432M Hz/(sample rates*2)

And knowing the relation between LRCK and BCK, you can easily calculate BCK_divider and everything.

We firstly used 6000Hz sample rates, the music we get made us don't want to play the game. Thus we used 22050Hz sample rates, the game reminded us of our childhood.

Another thing to mention, there's no 18.432 M Hz clock available on chip, only 50M Hz and 27M Hz, thus, we used a .pll (phase lock loop) to generate the 18.432M hz from 50M Hz on chip.

How to send data in.

The audio play hardware is readily configured, but how to send the data in?

In order to save the on chip memory, and prevent any discrepancy in data transfer between two different clock domains, we used a 256-2bytes buffer to buffer the music writing. The picture will explain itself.

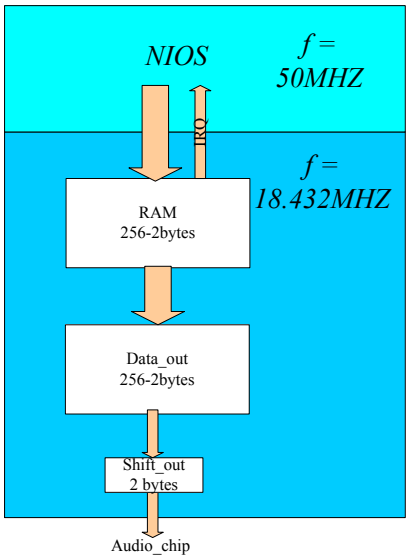


Figure 7 Data transfer block diagram

When Data_out had fully read RAM's data (address count to 256), irq =1, nios detects it and sends a new array of 256-2bytes data in.

Software control.

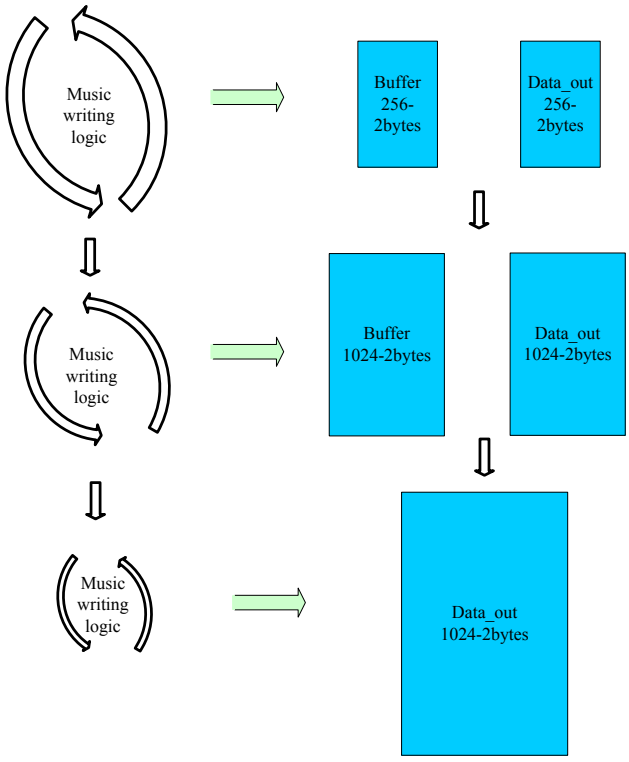
In the design, besides from the game music, we have sound effects during the game play—hit, miss, explo, life---and the stand by period---press button. We merge the music and sound effects with the function as follows:

When no sound effects are triggered, audio_to_play = music_to_play, only the music will be played. When sound effect triggered Audio_to_play = music_to_play/4

+ sound_to_play*3/4. Music's volume is decreased to 1/4 when merge with sound , because our sounds' volume itself is small, we gave it larger portion of volume. You cannot simply add the sound and music together, because the 16bits samples will overflow, the value of audio_to_play should be kept at a total scale of 1 to prevent this.

Structure Optimization.

One important issue of using SDRAM as CPU's memory is that its speed is a bit slow. Thus, when we firstly link the audio software and hardware together, the music we hear is half or even the 1/4 of the normal speed. There are two reasons that may cause this: firstly, the CPU cannot respond the audio hardware's interrupts frequently, thus the data stored in the audio buffer is not timely refreshed and is been played several times before it is refreshed; secondly, the music writing logic loop is too big, it need to be small because this part of code is executed most frequently by CPU, and will slow down the whole game if not succinct enough. The solution is, increase the buffer size in hardware so that it does not call for interruption too frequently, and optimize the music writing loop logic:

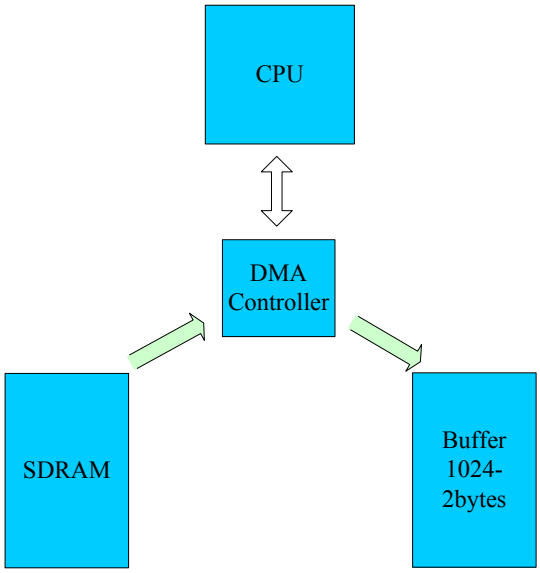


During this process, we found that the extra music buffer is not really necessary because the CPU f=50MHz is much higher than Audio chip f=18.432MHz, discrepancies in the domain boundary can be omitted, so we got rid of it to save on-chip memory and compile time.

In this way, we successfully get the right music and sound.

DMA

Another way to deal with the CPU and audio speed thing is to use a DMA (direct-memory-access) controller. It can serve itself transferring data from SDRAM to audio hardware. With DMA, CPU only need to tell it the starting address for the transfer and how many bytes (2 bytes) to transfer at the beginning of the transfer, instead of transfer every bytes (2 bytes) by itself during the whole process of transfer.



We tried this and it worked. However, we didn't include this in our final design, because we can also solve the problem by optimizing the structure, and we think that the structure optimization should always be the first choice before we resort to any other extra resources.

Audio Experience.

We tried many methods in implementing the audio during our design. At first, we

tried to synthesis the music directly in the VHDL code, similar to lab 3, and we found that the music is too simple, not exciting at all, and the sound effects are hard to generate, thus we decide to download the music; we put music in the SDRAM and sound in the VHDL, trying to merge them together in the hardware, it turns out to be a little bit complicated doing it in hardware, and due to the large size of soundtracks in ROM(high sample rates), the compile time became ridiculous. At last, we decide to put all audio tracks in the SDRAM, emerge them in the software before sending to the hardware; it turns out to be lot easier than doing it in hardware. After that, we tried to implement a Direct Memory Access controller in SOPC to control the music data transfer, so that sending audio data will not occupy CPU much and game performance can be improved. However, as reason indicated above, we didn't include it in our final design.

4. Responsibilities and Future Work

Jian was in charge of the overall hardware design and implementation; Si's main responsibility was to implement the dynamic display on the screen; Shuting managed the software and Lingchuan achieved the working audios. Throughout the project, all the team members worked together for the debugging and testing.

For the future work, we want to:

1. Make it possible for more than one items coming out the holes simultaneously.
2. Apart from the specific sounds played when you successfully whac or miss an item, implement visual efforts for all the events on the screen.

5. Code

N.B. Since the header file for the music is too large, we didn't attach it here. But it will come with the project file.

nios_system.v

```
//megafunction wizard: %Altera SOPC Builder%
//GENERATION: STANDARD
```

```
//VERSION: WM1.0


//Legal Notice: (C)2013 Altera Corporation. All rights reserved.  Your
//use of Altera Corporation's design tools, logic functions and other
//software and tools, and its AMPP partner logic functions, and any
//output files any of the foregoing (including device programming or
//simulation files), and any associated documentation or information are
//expressly subject to the terms and conditions of the Altera Program
//License Subscription Agreement or other applicable license agreement,
//including, without limitation, that your use is for the sole purpose
//of programming logic devices manufactured by Altera and sold by Altera
//or its authorized distributors.  Please refer to the applicable
//agreement for further details.

// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module audio_controller_0_avalon_slave_0_arbitrator (
                                                      // inputs:

audio_controller_0_avalon_slave_0_readdata,
                                                       clk,

cpu_0_data_master_address_to_slave,

cpu_0_data_master_byteenable,

cpu_0_data_master_dbs_address,

cpu_0_data_master_dbs_write_16,

cpu_0_data_master_no_byte_enables_and_last_term,
                                                       cpu_0_data_master_read,

cpu_0_data_master_waitrequest,
                                                       cpu_0_data_master_write,

cpu_0_instruction_master_address_to_slave,

cpu_0_instruction_master_dbs_address,

cpu_0_instruction_master_read,
                                                       reset_n,

                                                      // outputs:

audio_controller_0_avalon_slave_0_address,

audio_controller_0_avalon_slave_0_chipselect,

audio_controller_0_avalon_slave_0_read,
```

```
audio_controller_0_avalon_slave_0_readdata_from_sa,

audio_controller_0_avalon_slave_0_reset_n,

audio_controller_0_avalon_slave_0_write,

audio_controller_0_avalon_slave_0_writedata,

cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0,

cpu_0_data_master_granted_audio_controller_0_avalon_slave_0,

cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0,

cpu_0_data_master_requests_audio_controller_0_avalon_slave_0,

cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0,

cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0,

cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0,

cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0,

d1_audio_controller_0_avalon_slave_0_end_xfer
                                                                    )
;

  output  [  4: 0] audio_controller_0_avalon_slave_0_address;
  output           audio_controller_0_avalon_slave_0_chipselect;
  output           audio_controller_0_avalon_slave_0_read;
  output  [ 15: 0] audio_controller_0_avalon_slave_0_readdata_from_sa;
  output           audio_controller_0_avalon_slave_0_reset_n;
  output           audio_controller_0_avalon_slave_0_write;
  output  [ 15: 0] audio_controller_0_avalon_slave_0_writedata;
  output  [  1: 0] cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0;
  output           cpu_0_data_master_granted_audio_controller_0_avalon_slave_0;
  output
cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0;
  output
cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0;
  output           cpu_0_data_master_requests_audio_controller_0_avalon_slave_0;
  output           cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0;
  output
cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0;
  output
cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0;
  output           cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0;
  output           d1_audio_controller_0_avalon_slave_0_end_xfer;
  input   [ 15: 0] audio_controller_0_avalon_slave_0_readdata;
  input            clk;
  input   [ 24: 0] cpu_0_data_master_address_to_slave;
  input   [  3: 0] cpu_0_data_master_byteenable;
  input   [  1: 0] cpu_0_data_master_dbs_address;
  input   [ 15: 0] cpu_0_data_master_dbs_write_16;
  input            cpu_0_data_master_no_byte_enables_and_last_term;
```

```
  input              cpu_0_data_master_read;
  input              cpu_0_data_master_waitrequest;
  input              cpu_0_data_master_write;
  input     [ 24: 0] cpu_0_instruction_master_address_to_slave;
  input     [  1: 0] cpu_0_instruction_master_dbs_address;
  input              cpu_0_instruction_master_read;
  input              reset_n;

  wire      [  4: 0] audio_controller_0_avalon_slave_0_address;
  wire               audio_controller_0_avalon_slave_0_allgrants;
  wire               audio_controller_0_avalon_slave_0_allow_new_arb_cycle;
  wire               audio_controller_0_avalon_slave_0_any_bursting_master_saved_grant;
  wire               audio_controller_0_avalon_slave_0_any_continuerequest;
  reg       [  1: 0] audio_controller_0_avalon_slave_0_arb_addend;
  wire               audio_controller_0_avalon_slave_0_arb_counter_enable;
  reg       [  1: 0] audio_controller_0_avalon_slave_0_arb_share_counter;
  wire      [  1: 0] audio_controller_0_avalon_slave_0_arb_share_counter_next_value;
  wire      [  1: 0] audio_controller_0_avalon_slave_0_arb_share_set_values;
  wire      [  1: 0] audio_controller_0_avalon_slave_0_arb_winner;
  wire               audio_controller_0_avalon_slave_0_arbitration_holdoff_internal;
  wire               audio_controller_0_avalon_slave_0_beginbursttransfer_internal;
  wire               audio_controller_0_avalon_slave_0_begins_xfer;
  wire               audio_controller_0_avalon_slave_0_chipselect;
  wire      [  3: 0] audio_controller_0_avalon_slave_0_chosen_master_double_vector;
  wire      [  1: 0] audio_controller_0_avalon_slave_0_chosen_master_rot_left;
  wire               audio_controller_0_avalon_slave_0_end_xfer;
  wire               audio_controller_0_avalon_slave_0_firsttransfer;
  wire      [  1: 0] audio_controller_0_avalon_slave_0_grant_vector;
  wire               audio_controller_0_avalon_slave_0_in_a_read_cycle;
  wire               audio_controller_0_avalon_slave_0_in_a_write_cycle;
  wire      [  1: 0] audio_controller_0_avalon_slave_0_master_qreq_vector;
  wire               audio_controller_0_avalon_slave_0_non_bursting_master_requests;
  wire               audio_controller_0_avalon_slave_0_read;
  wire      [ 15: 0] audio_controller_0_avalon_slave_0_readdata_from_sa;
  reg                audio_controller_0_avalon_slave_0_reg_firsttransfer;
  wire               audio_controller_0_avalon_slave_0_reset_n;
  reg       [  1: 0] audio_controller_0_avalon_slave_0_saved_chosen_master_vector;
  reg                audio_controller_0_avalon_slave_0_slavearbiterlockenable;
  wire               audio_controller_0_avalon_slave_0_slavearbiterlockenable2;
  wire               audio_controller_0_avalon_slave_0_unreg_firsttransfer;
  wire               audio_controller_0_avalon_slave_0_waits_for_read;
  wire               audio_controller_0_avalon_slave_0_waits_for_write;
  wire               audio_controller_0_avalon_slave_0_write;
  wire      [ 15: 0] audio_controller_0_avalon_slave_0_writedata;
  wire               cpu_0_data_master_arbiterlock;
  wire               cpu_0_data_master_arbiterlock2;
  wire      [  1: 0] cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0;
  wire                                                [  1:  0]
cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0_segment_0;
  wire                                                [  1:  0]
cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0_segment_1;
  wire               cpu_0_data_master_continuerequest;
  wire               cpu_0_data_master_granted_audio_controller_0_avalon_slave_0;
  wire
cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0;
  wire
cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0;
  wire               cpu_0_data_master_requests_audio_controller_0_avalon_slave_0;
```

```
  wire        cpu_0_data_master_saved_grant_audio_controller_0_avalon_slave_0;
  wire        cpu_0_instruction_master_arbiterlock;
  wire        cpu_0_instruction_master_arbiterlock2;
  wire        cpu_0_instruction_master_continuerequest;
  wire        cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0;
  wire
cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0;
  wire
cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0;
  wire        cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0;
  wire
cpu_0_instruction_master_saved_grant_audio_controller_0_avalon_slave_0;
  reg         d1_audio_controller_0_avalon_slave_0_end_xfer;
  reg         d1_reasons_to_wait;
  reg         enable_nonzero_assertions;
  wire        end_xfer_arb_share_counter_term_audio_controller_0_avalon_slave_0;
  wire        in_a_read_cycle;
  wire        in_a_write_cycle;
  reg
last_cycle_cpu_0_data_master_granted_slave_audio_controller_0_avalon_slave_0;
  reg
last_cycle_cpu_0_instruction_master_granted_slave_audio_controller_0_avalon_slave_0;
  wire                                                [ 24: 0]
shifted_address_to_audio_controller_0_avalon_slave_0_from_cpu_0_data_master;
  wire                                                [ 24: 0]
shifted_address_to_audio_controller_0_avalon_slave_0_from_cpu_0_instruction_master;
  wire        wait_for_audio_controller_0_avalon_slave_0_counter;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_reasons_to_wait <= 0;
      else
        d1_reasons_to_wait <= ~audio_controller_0_avalon_slave_0_end_xfer;
    end


  assign audio_controller_0_avalon_slave_0_begins_xfer = ~d1_reasons_to_wait &
((cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0 |
cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0));
  //assign audio_controller_0_avalon_slave_0_readdata_from_sa =
audio_controller_0_avalon_slave_0_readdata so that symbol knows where to group signals
which may go to master only, which is an e_assign
  assign audio_controller_0_avalon_slave_0_readdata_from_sa =
audio_controller_0_avalon_slave_0_readdata;

  assign cpu_0_data_master_requests_audio_controller_0_avalon_slave_0 =
({cpu_0_data_master_address_to_slave[24 : 6] , 6'b0} == 25'h1001100) &
(cpu_0_data_master_read | cpu_0_data_master_write);
  //audio_controller_0_avalon_slave_0_arb_share_counter set values, which is an e_mux
  assign audio_controller_0_avalon_slave_0_arb_share_set_values =
(cpu_0_data_master_granted_audio_controller_0_avalon_slave_0)? 2 :
    (cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0)? 2 :
    (cpu_0_data_master_granted_audio_controller_0_avalon_slave_0)? 2 :
    (cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0)? 2 :
    1;

  //audio_controller_0_avalon_slave_0_non_bursting_master_requests mux, which is an e_mux
  assign audio_controller_0_avalon_slave_0_non_bursting_master_requests =
```

```
cpu_0_data_master_requests_audio_controller_0_avalon_slave_0 |
      cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0 |
      cpu_0_data_master_requests_audio_controller_0_avalon_slave_0 |
      cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0;

  //audio_controller_0_avalon_slave_0_any_bursting_master_saved_grant mux, which is an
e_mux
  assign audio_controller_0_avalon_slave_0_any_bursting_master_saved_grant = 0;

  //audio_controller_0_avalon_slave_0_arb_share_counter_next_value assignment, which is an
e_assign
  assign audio_controller_0_avalon_slave_0_arb_share_counter_next_value =
audio_controller_0_avalon_slave_0_firsttransfer ?
(audio_controller_0_avalon_slave_0_arb_share_set_values - 1) :
|audio_controller_0_avalon_slave_0_arb_share_counter ?
(audio_controller_0_avalon_slave_0_arb_share_counter - 1) : 0;

  //audio_controller_0_avalon_slave_0_allgrants all slave grants, which is an e_mux
  assign audio_controller_0_avalon_slave_0_allgrants =
(|audio_controller_0_avalon_slave_0_grant_vector) |
      (|audio_controller_0_avalon_slave_0_grant_vector) |
      (|audio_controller_0_avalon_slave_0_grant_vector) |
      (|audio_controller_0_avalon_slave_0_grant_vector);

  //audio_controller_0_avalon_slave_0_end_xfer assignment, which is an e_assign
  assign audio_controller_0_avalon_slave_0_end_xfer =
~(audio_controller_0_avalon_slave_0_waits_for_read |
audio_controller_0_avalon_slave_0_waits_for_write);

  //end_xfer_arb_share_counter_term_audio_controller_0_avalon_slave_0 arb share counter
enable term, which is an e_assign
  assign end_xfer_arb_share_counter_term_audio_controller_0_avalon_slave_0 =
audio_controller_0_avalon_slave_0_end_xfer &
(~audio_controller_0_avalon_slave_0_any_bursting_master_saved_grant | in_a_read_cycle |
in_a_write_cycle);

  //audio_controller_0_avalon_slave_0_arb_share_counter arbitration counter enable, which is
an e_assign
  assign audio_controller_0_avalon_slave_0_arb_counter_enable =
(end_xfer_arb_share_counter_term_audio_controller_0_avalon_slave_0 &
audio_controller_0_avalon_slave_0_allgrants) |
(end_xfer_arb_share_counter_term_audio_controller_0_avalon_slave_0 &
~audio_controller_0_avalon_slave_0_non_bursting_master_requests);

  //audio_controller_0_avalon_slave_0_arb_share_counter counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          audio_controller_0_avalon_slave_0_arb_share_counter <= 0;
      else if (audio_controller_0_avalon_slave_0_arb_counter_enable)
          audio_controller_0_avalon_slave_0_arb_share_counter <=
audio_controller_0_avalon_slave_0_arb_share_counter_next_value;
    end


  //audio_controller_0_avalon_slave_0_slavearbiterlockenable slave enables arbiterlock, which
is an e_register
  always @(posedge clk or negedge reset_n)
```

```
      begin
         if (reset_n == 0)
               audio_controller_0_avalon_slave_0_slavearbiterlockenable <= 0;
         else          if          ((|audio_controller_0_avalon_slave_0_master_qreq_vector          &
end_xfer_arb_share_counter_term_audio_controller_0_avalon_slave_0)                                  |
(end_xfer_arb_share_counter_term_audio_controller_0_avalon_slave_0                                  &
~audio_controller_0_avalon_slave_0_non_bursting_master_requests))
               audio_controller_0_avalon_slave_0_slavearbiterlockenable                            <=
|audio_controller_0_avalon_slave_0_arb_share_counter_next_value;
      end


   //cpu_0/data_master audio_controller_0/avalon_slave_0 arbiterlock, which is an e_assign
   assign                                cpu_0_data_master_arbiterlock                               =
audio_controller_0_avalon_slave_0_slavearbiterlockenable                                            &
cpu_0_data_master_continuerequest;

   //audio_controller_0_avalon_slave_0_slavearbiterlockenable2 slave enables arbiterlock2,
which is an e_assign
   assign                audio_controller_0_avalon_slave_0_slavearbiterlockenable2                   =
|audio_controller_0_avalon_slave_0_arb_share_counter_next_value;

   //cpu_0/data_master audio_controller_0/avalon_slave_0 arbiterlock2, which is an e_assign
   assign                               cpu_0_data_master_arbiterlock2                               =
audio_controller_0_avalon_slave_0_slavearbiterlockenable2                                           &
cpu_0_data_master_continuerequest;

   //cpu_0/instruction_master audio_controller_0/avalon_slave_0 arbiterlock, which is an
e_assign
   assign                              cpu_0_instruction_master_arbiterlock                           =
audio_controller_0_avalon_slave_0_slavearbiterlockenable                                            &
cpu_0_instruction_master_continuerequest;

   //cpu_0/instruction_master audio_controller_0/avalon_slave_0 arbiterlock2, which is an
e_assign
   assign                             cpu_0_instruction_master_arbiterlock2                           =
audio_controller_0_avalon_slave_0_slavearbiterlockenable2                                           &
cpu_0_instruction_master_continuerequest;

   //cpu_0/instruction_master granted audio_controller_0/avalon_slave_0 last time, which is an
e_register
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)

last_cycle_cpu_0_instruction_master_granted_slave_audio_controller_0_avalon_slave_0 <= 0;
         else

last_cycle_cpu_0_instruction_master_granted_slave_audio_controller_0_avalon_slave_0       <=
cpu_0_instruction_master_saved_grant_audio_controller_0_avalon_slave_0        ?        1        :
(audio_controller_0_avalon_slave_0_arbitration_holdoff_internal                                     |
~cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0)         ?         0         :
last_cycle_cpu_0_instruction_master_granted_slave_audio_controller_0_avalon_slave_0;
      end


   //cpu_0_instruction_master_continuerequest continued request, which is an e_mux
   assign                             cpu_0_instruction_master_continuerequest                        =
```

```
last_cycle_cpu_0_instruction_master_granted_slave_audio_controller_0_avalon_slave_0 & cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0;

  //audio_controller_0_avalon_slave_0_any_continuerequest at least one master continues requesting, which is an e_mux
  assign audio_controller_0_avalon_slave_0_any_continuerequest = cpu_0_instruction_master_continuerequest |
    cpu_0_data_master_continuerequest;

  assign cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0 = cpu_0_data_master_requests_audio_controller_0_avalon_slave_0 & ~(((~cpu_0_data_master_waitrequest | cpu_0_data_master_no_byte_enables_and_last_term | !cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0) & cpu_0_data_master_write) | cpu_0_instruction_master_arbiterlock);
  //audio_controller_0_avalon_slave_0_writedata mux, which is an e_mux
  assign audio_controller_0_avalon_slave_0_writedata = cpu_0_data_master_dbs_write_16;

  assign cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0 = (({cpu_0_instruction_master_address_to_slave[24 : 6] , 6'b0} == 25'h1001100) & (cpu_0_instruction_master_read)) & cpu_0_instruction_master_read;
  //cpu_0/data_master granted audio_controller_0/avalon_slave_0 last time, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          last_cycle_cpu_0_data_master_granted_slave_audio_controller_0_avalon_slave_0 <= 0;
      else
        last_cycle_cpu_0_data_master_granted_slave_audio_controller_0_avalon_slave_0 <= cpu_0_data_master_saved_grant_audio_controller_0_avalon_slave_0 ? 1 : (audio_controller_0_avalon_slave_0_arbitration_holdoff_internal | ~cpu_0_data_master_requests_audio_controller_0_avalon_slave_0) ? 0 : last_cycle_cpu_0_data_master_granted_slave_audio_controller_0_avalon_slave_0;
    end


  //cpu_0_data_master_continuerequest continued request, which is an e_mux
  assign cpu_0_data_master_continuerequest = last_cycle_cpu_0_data_master_granted_slave_audio_controller_0_avalon_slave_0 & cpu_0_data_master_requests_audio_controller_0_avalon_slave_0;

  assign cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0 = cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0 & ~(cpu_0_data_master_arbiterlock);
  //allow new arb cycle for audio_controller_0/avalon_slave_0, which is an e_assign
  assign audio_controller_0_avalon_slave_0_allow_new_arb_cycle = ~cpu_0_data_master_arbiterlock & ~cpu_0_instruction_master_arbiterlock;

  //cpu_0/instruction_master assignment into master qualified-requests vector for audio_controller_0/avalon_slave_0, which is an e_assign
  assign audio_controller_0_avalon_slave_0_master_qreq_vector[0] = cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0;

  //cpu_0/instruction_master grant audio_controller_0/avalon_slave_0, which is an e_assign
  assign cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0 = audio_controller_0_avalon_slave_0_grant_vector[0];

  //cpu_0/instruction_master saved-grant audio_controller_0/avalon_slave_0, which is an
```

```
e_assign
  assign cpu_0_instruction_master_saved_grant_audio_controller_0_avalon_slave_0 =
audio_controller_0_avalon_slave_0_arb_winner[0] &&
cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0;

  //cpu_0/data_master assignment into master qualified-requests vector for
audio_controller_0/avalon_slave_0, which is an e_assign
  assign audio_controller_0_avalon_slave_0_master_qreq_vector[1] =
cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0;

  //cpu_0/data_master grant audio_controller_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_granted_audio_controller_0_avalon_slave_0 =
audio_controller_0_avalon_slave_0_grant_vector[1];

  //cpu_0/data_master saved-grant audio_controller_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_saved_grant_audio_controller_0_avalon_slave_0 =
audio_controller_0_avalon_slave_0_arb_winner[1] &&
cpu_0_data_master_requests_audio_controller_0_avalon_slave_0;

  //audio_controller_0/avalon_slave_0 chosen-master double-vector, which is an e_assign
  assign audio_controller_0_avalon_slave_0_chosen_master_double_vector =
{audio_controller_0_avalon_slave_0_master_qreq_vector,
audio_controller_0_avalon_slave_0_master_qreq_vector} &
({~audio_controller_0_avalon_slave_0_master_qreq_vector,
~audio_controller_0_avalon_slave_0_master_qreq_vector} +
audio_controller_0_avalon_slave_0_arb_addend);

  //stable onehot encoding of arb winner
  assign audio_controller_0_avalon_slave_0_arb_winner =
(audio_controller_0_avalon_slave_0_allow_new_arb_cycle & |
audio_controller_0_avalon_slave_0_grant_vector) ?
audio_controller_0_avalon_slave_0_grant_vector :
audio_controller_0_avalon_slave_0_saved_chosen_master_vector;

  //saved audio_controller_0_avalon_slave_0_grant_vector, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          audio_controller_0_avalon_slave_0_saved_chosen_master_vector <= 0;
      else if (audio_controller_0_avalon_slave_0_allow_new_arb_cycle)
          audio_controller_0_avalon_slave_0_saved_chosen_master_vector <=
|audio_controller_0_avalon_slave_0_grant_vector ?
audio_controller_0_avalon_slave_0_grant_vector :
audio_controller_0_avalon_slave_0_saved_chosen_master_vector;
    end


  //onehot encoding of chosen master
  assign audio_controller_0_avalon_slave_0_grant_vector =
{(audio_controller_0_avalon_slave_0_chosen_master_double_vector[1] |
audio_controller_0_avalon_slave_0_chosen_master_double_vector[3]),
    (audio_controller_0_avalon_slave_0_chosen_master_double_vector[0] |
audio_controller_0_avalon_slave_0_chosen_master_double_vector[2])};

  //audio_controller_0/avalon_slave_0 chosen master rotated left, which is an e_assign
  assign audio_controller_0_avalon_slave_0_chosen_master_rot_left =
(audio_controller_0_avalon_slave_0_arb_winner << 1) ?
(audio_controller_0_avalon_slave_0_arb_winner << 1) : 1;
```

```
  //audio_controller_0/avalon_slave_0's addend for next-master-grant
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          audio_controller_0_avalon_slave_0_arb_addend <= 1;
      else if (|audio_controller_0_avalon_slave_0_grant_vector)
          audio_controller_0_avalon_slave_0_arb_addend                                    <=
audio_controller_0_avalon_slave_0_end_xfer?
audio_controller_0_avalon_slave_0_chosen_master_rot_left                                  :
audio_controller_0_avalon_slave_0_grant_vector;
    end


  //audio_controller_0_avalon_slave_0_reset_n assignment, which is an e_assign
  assign audio_controller_0_avalon_slave_0_reset_n = reset_n;

  assign                  audio_controller_0_avalon_slave_0_chipselect                    =
cpu_0_data_master_granted_audio_controller_0_avalon_slave_0                               |
cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0;
  //audio_controller_0_avalon_slave_0_firsttransfer first transaction, which is an e_assign
  assign                audio_controller_0_avalon_slave_0_firsttransfer                   =
audio_controller_0_avalon_slave_0_begins_xfer                                             ?
audio_controller_0_avalon_slave_0_unreg_firsttransfer                                     :
audio_controller_0_avalon_slave_0_reg_firsttransfer;

  //audio_controller_0_avalon_slave_0_unreg_firsttransfer first transaction, which is an
e_assign
  assign              audio_controller_0_avalon_slave_0_unreg_firsttransfer             =
~(audio_controller_0_avalon_slave_0_slavearbiterlockenable                                &
audio_controller_0_avalon_slave_0_any_continuerequest);

  //audio_controller_0_avalon_slave_0_reg_firsttransfer first transaction, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          audio_controller_0_avalon_slave_0_reg_firsttransfer <= 1'b1;
      else if (audio_controller_0_avalon_slave_0_begins_xfer)
          audio_controller_0_avalon_slave_0_reg_firsttransfer                             <=
audio_controller_0_avalon_slave_0_unreg_firsttransfer;
    end


  //audio_controller_0_avalon_slave_0_beginbursttransfer_internal begin burst transfer, which
is an e_assign
  assign          audio_controller_0_avalon_slave_0_beginbursttransfer_internal       =
audio_controller_0_avalon_slave_0_begins_xfer;

  //audio_controller_0_avalon_slave_0_arbitration_holdoff_internal arbitration_holdoff, which
is an e_assign
  assign          audio_controller_0_avalon_slave_0_arbitration_holdoff_internal      =
audio_controller_0_avalon_slave_0_begins_xfer                                             &
audio_controller_0_avalon_slave_0_firsttransfer;

  //audio_controller_0_avalon_slave_0_read assignment, which is an e_mux
  assign                  audio_controller_0_avalon_slave_0_read                          =
(cpu_0_data_master_granted_audio_controller_0_avalon_slave_0  &  cpu_0_data_master_read)
|          (cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0          &
```

```
cpu_0_instruction_master_read);

  //audio_controller_0_avalon_slave_0_write assignment, which is an e_mux
  assign audio_controller_0_avalon_slave_0_write =
cpu_0_data_master_granted_audio_controller_0_avalon_slave_0 & cpu_0_data_master_write;

  assign shifted_address_to_audio_controller_0_avalon_slave_0_from_cpu_0_data_master =
{cpu_0_data_master_address_to_slave >> 2,
    cpu_0_data_master_dbs_address[1],
    {1 {1'b0}}};

  //audio_controller_0_avalon_slave_0_address mux, which is an e_mux
  assign audio_controller_0_avalon_slave_0_address =
(cpu_0_data_master_granted_audio_controller_0_avalon_slave_0)?
(shifted_address_to_audio_controller_0_avalon_slave_0_from_cpu_0_data_master >> 1) :

(shifted_address_to_audio_controller_0_avalon_slave_0_from_cpu_0_instruction_master >> 1);

  assign
shifted_address_to_audio_controller_0_avalon_slave_0_from_cpu_0_instruction_master =
{cpu_0_instruction_master_address_to_slave >> 2,
    cpu_0_instruction_master_dbs_address[1],
    {1 {1'b0}}};

  //d1_audio_controller_0_avalon_slave_0_end_xfer register, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_audio_controller_0_avalon_slave_0_end_xfer <= 1;
      else
        d1_audio_controller_0_avalon_slave_0_end_xfer <=
audio_controller_0_avalon_slave_0_end_xfer;
    end


  //audio_controller_0_avalon_slave_0_waits_for_read in a cycle, which is an e_mux
  assign audio_controller_0_avalon_slave_0_waits_for_read =
audio_controller_0_avalon_slave_0_in_a_read_cycle &
audio_controller_0_avalon_slave_0_begins_xfer;

  //audio_controller_0_avalon_slave_0_in_a_read_cycle assignment, which is an e_assign
  assign audio_controller_0_avalon_slave_0_in_a_read_cycle =
(cpu_0_data_master_granted_audio_controller_0_avalon_slave_0 & cpu_0_data_master_read)
|          (cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0 &
cpu_0_instruction_master_read);

  //in_a_read_cycle assignment, which is an e_mux
  assign in_a_read_cycle = audio_controller_0_avalon_slave_0_in_a_read_cycle;

  //audio_controller_0_avalon_slave_0_waits_for_write in a cycle, which is an e_mux
  assign audio_controller_0_avalon_slave_0_waits_for_write =
audio_controller_0_avalon_slave_0_in_a_write_cycle & 0;

  //audio_controller_0_avalon_slave_0_in_a_write_cycle assignment, which is an e_assign
  assign audio_controller_0_avalon_slave_0_in_a_write_cycle =
cpu_0_data_master_granted_audio_controller_0_avalon_slave_0 & cpu_0_data_master_write;

  //in_a_write_cycle assignment, which is an e_mux
```

```
  assign in_a_write_cycle = audio_controller_0_avalon_slave_0_in_a_write_cycle;

  assign wait_for_audio_controller_0_avalon_slave_0_counter = 0;
  assign {cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0_segment_1,
cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0_segment_0}                =
cpu_0_data_master_byteenable;
  assign            cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0          =
((cpu_0_data_master_dbs_address[1]                                  ==                  0))?
cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0_segment_0 :
      cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0_segment_1;


//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
  //audio_controller_0/avalon_slave_0 enable non-zero assertions, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          enable_nonzero_assertions <= 0;
      else
        enable_nonzero_assertions <= 1'b1;
    end


  //grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if            (cpu_0_data_master_granted_audio_controller_0_avalon_slave_0            +
cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0 > 1)
        begin
          $write("%0d ns: > 1 of grant signals are active simultaneously", $time);
          $stop;
        end
    end


  //saved_grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if          (cpu_0_data_master_saved_grant_audio_controller_0_avalon_slave_0          +
cpu_0_instruction_master_saved_grant_audio_controller_0_avalon_slave_0 > 1)
        begin
          $write("%0d ns: > 1 of saved_grant signals are active simultaneously", $time);
          $stop;
        end
    end



//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
```

```
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module command_0_avalon_slave_0_arbitrator (
                                             // inputs:
                                              clk,
                                              command_0_avalon_slave_0_readdata,
                                              cpu_0_data_master_address_to_slave,
                                              cpu_0_data_master_read,
                                              cpu_0_data_master_waitrequest,
                                              cpu_0_data_master_write,
                                              cpu_0_data_master_writedata,

cpu_0_instruction_master_address_to_slave,
                                              cpu_0_instruction_master_read,
                                              reset_n,

                                             // outputs:
                                              command_0_avalon_slave_0_address,

command_0_avalon_slave_0_chipselect,
                                              command_0_avalon_slave_0_read,

command_0_avalon_slave_0_readdata_from_sa,
                                              command_0_avalon_slave_0_reset_n,
                                              command_0_avalon_slave_0_write,
                                              command_0_avalon_slave_0_writedata,

cpu_0_data_master_granted_command_0_avalon_slave_0,

cpu_0_data_master_qualified_request_command_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_command_0_avalon_slave_0,

cpu_0_data_master_requests_command_0_avalon_slave_0,

cpu_0_instruction_master_granted_command_0_avalon_slave_0,

cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0,

cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0,

cpu_0_instruction_master_requests_command_0_avalon_slave_0,

d1_command_0_avalon_slave_0_end_xfer
                                                    )
;

  output  [  4: 0] command_0_avalon_slave_0_address;
  output           command_0_avalon_slave_0_chipselect;
  output           command_0_avalon_slave_0_read;
  output  [ 31: 0] command_0_avalon_slave_0_readdata_from_sa;
  output           command_0_avalon_slave_0_reset_n;
  output           command_0_avalon_slave_0_write;
  output  [ 31: 0] command_0_avalon_slave_0_writedata;
```

```
  output              cpu_0_data_master_granted_command_0_avalon_slave_0;
  output              cpu_0_data_master_qualified_request_command_0_avalon_slave_0;
  output              cpu_0_data_master_read_data_valid_command_0_avalon_slave_0;
  output              cpu_0_data_master_requests_command_0_avalon_slave_0;
  output              cpu_0_instruction_master_granted_command_0_avalon_slave_0;
  output
cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0;
  output
cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0;
  output              cpu_0_instruction_master_requests_command_0_avalon_slave_0;
  output              d1_command_0_avalon_slave_0_end_xfer;
  input               clk;
  input    [ 31: 0] command_0_avalon_slave_0_readdata;
  input    [ 24: 0] cpu_0_data_master_address_to_slave;
  input               cpu_0_data_master_read;
  input               cpu_0_data_master_waitrequest;
  input               cpu_0_data_master_write;
  input    [ 31: 0] cpu_0_data_master_writedata;
  input    [ 24: 0] cpu_0_instruction_master_address_to_slave;
  input               cpu_0_instruction_master_read;
  input               reset_n;

  wire     [  4: 0] command_0_avalon_slave_0_address;
  wire              command_0_avalon_slave_0_allgrants;
  wire              command_0_avalon_slave_0_allow_new_arb_cycle;
  wire              command_0_avalon_slave_0_any_bursting_master_saved_grant;
  wire              command_0_avalon_slave_0_any_continuerequest;
  reg      [  1: 0] command_0_avalon_slave_0_arb_addend;
  wire              command_0_avalon_slave_0_arb_counter_enable;
  reg      [  1: 0] command_0_avalon_slave_0_arb_share_counter;
  wire     [  1: 0] command_0_avalon_slave_0_arb_share_counter_next_value;
  wire     [  1: 0] command_0_avalon_slave_0_arb_share_set_values;
  wire     [  1: 0] command_0_avalon_slave_0_arb_winner;
  wire              command_0_avalon_slave_0_arbitration_holdoff_internal;
  wire              command_0_avalon_slave_0_beginbursttransfer_internal;
  wire              command_0_avalon_slave_0_begins_xfer;
  wire              command_0_avalon_slave_0_chipselect;
  wire     [  3: 0] command_0_avalon_slave_0_chosen_master_double_vector;
  wire     [  1: 0] command_0_avalon_slave_0_chosen_master_rot_left;
  wire              command_0_avalon_slave_0_end_xfer;
  wire              command_0_avalon_slave_0_firsttransfer;
  wire     [  1: 0] command_0_avalon_slave_0_grant_vector;
  wire              command_0_avalon_slave_0_in_a_read_cycle;
  wire              command_0_avalon_slave_0_in_a_write_cycle;
  wire     [  1: 0] command_0_avalon_slave_0_master_qreq_vector;
  wire              command_0_avalon_slave_0_non_bursting_master_requests;
  wire              command_0_avalon_slave_0_read;
  wire     [ 31: 0] command_0_avalon_slave_0_readdata_from_sa;
  reg               command_0_avalon_slave_0_reg_firsttransfer;
  wire              command_0_avalon_slave_0_reset_n;
  reg      [  1: 0] command_0_avalon_slave_0_saved_chosen_master_vector;
  reg               command_0_avalon_slave_0_slavearbiterlockenable;
  wire              command_0_avalon_slave_0_slavearbiterlockenable2;
  wire              command_0_avalon_slave_0_unreg_firsttransfer;
  wire              command_0_avalon_slave_0_waits_for_read;
  wire              command_0_avalon_slave_0_waits_for_write;
  wire              command_0_avalon_slave_0_write;
  wire     [ 31: 0] command_0_avalon_slave_0_writedata;
```

```
  wire                    cpu_0_data_master_arbiterlock;
  wire                    cpu_0_data_master_arbiterlock2;
  wire                    cpu_0_data_master_continuerequest;
  wire                    cpu_0_data_master_granted_command_0_avalon_slave_0;
  wire                    cpu_0_data_master_qualified_request_command_0_avalon_slave_0;
  wire                    cpu_0_data_master_read_data_valid_command_0_avalon_slave_0;
  wire                    cpu_0_data_master_requests_command_0_avalon_slave_0;
  wire                    cpu_0_data_master_saved_grant_command_0_avalon_slave_0;
  wire                    cpu_0_instruction_master_arbiterlock;
  wire                    cpu_0_instruction_master_arbiterlock2;
  wire                    cpu_0_instruction_master_continuerequest;
  wire                    cpu_0_instruction_master_granted_command_0_avalon_slave_0;
  wire
cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0;
  wire
cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0;
  wire                    cpu_0_instruction_master_requests_command_0_avalon_slave_0;
  wire                    cpu_0_instruction_master_saved_grant_command_0_avalon_slave_0;
  reg                     d1_command_0_avalon_slave_0_end_xfer;
  reg                     d1_reasons_to_wait;
  reg                     enable_nonzero_assertions;
  wire                    end_xfer_arb_share_counter_term_command_0_avalon_slave_0;
  wire                    in_a_read_cycle;
  wire                    in_a_write_cycle;
  reg
last_cycle_cpu_0_data_master_granted_slave_command_0_avalon_slave_0;
  reg
last_cycle_cpu_0_instruction_master_granted_slave_command_0_avalon_slave_0;
  wire      [ 24: 0] shifted_address_to_command_0_avalon_slave_0_from_cpu_0_data_master;
  wire                                                    [ 24: 0]
shifted_address_to_command_0_avalon_slave_0_from_cpu_0_instruction_master;
  wire                    wait_for_command_0_avalon_slave_0_counter;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_reasons_to_wait <= 0;
      else
        d1_reasons_to_wait <= ~command_0_avalon_slave_0_end_xfer;
    end


  assign      command_0_avalon_slave_0_begins_xfer      =      ~d1_reasons_to_wait      &
((cpu_0_data_master_qualified_request_command_0_avalon_slave_0                          |
cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0));
  //assign                    command_0_avalon_slave_0_readdata_from_sa                 =
command_0_avalon_slave_0_readdata so that symbol knows where to group signals which may
go to master only, which is an e_assign
  assign                    command_0_avalon_slave_0_readdata_from_sa                 =
command_0_avalon_slave_0_readdata;

  assign                cpu_0_data_master_requests_command_0_avalon_slave_0           =
({cpu_0_data_master_address_to_slave[24 : 7] , 7'b0} == 25'h100) & (cpu_0_data_master_read
| cpu_0_data_master_write);
  //command_0_avalon_slave_0_arb_share_counter set values, which is an e_mux
  assign command_0_avalon_slave_0_arb_share_set_values = 1;

  //command_0_avalon_slave_0_non_bursting_master_requests mux, which is an e_mux
  assign                command_0_avalon_slave_0_non_bursting_master_requests         =
```

```
cpu_0_data_master_requests_command_0_avalon_slave_0 |
      cpu_0_instruction_master_requests_command_0_avalon_slave_0 |
      cpu_0_data_master_requests_command_0_avalon_slave_0 |
      cpu_0_instruction_master_requests_command_0_avalon_slave_0;

  //command_0_avalon_slave_0_any_bursting_master_saved_grant mux, which is an e_mux
  assign command_0_avalon_slave_0_any_bursting_master_saved_grant = 0;

  //command_0_avalon_slave_0_arb_share_counter_next_value assignment, which is an
e_assign
  assign command_0_avalon_slave_0_arb_share_counter_next_value =
command_0_avalon_slave_0_firsttransfer ?
(command_0_avalon_slave_0_arb_share_set_values - 1) :
|command_0_avalon_slave_0_arb_share_counter ?
(command_0_avalon_slave_0_arb_share_counter - 1) : 0;

  //command_0_avalon_slave_0_allgrants all slave grants, which is an e_mux
  assign command_0_avalon_slave_0_allgrants = (|command_0_avalon_slave_0_grant_vector)
|
      (|command_0_avalon_slave_0_grant_vector) |
      (|command_0_avalon_slave_0_grant_vector) |
      (|command_0_avalon_slave_0_grant_vector);

  //command_0_avalon_slave_0_end_xfer assignment, which is an e_assign
  assign command_0_avalon_slave_0_end_xfer =
~(command_0_avalon_slave_0_waits_for_read |
command_0_avalon_slave_0_waits_for_write);

  //end_xfer_arb_share_counter_term_command_0_avalon_slave_0 arb share counter enable
term, which is an e_assign
  assign end_xfer_arb_share_counter_term_command_0_avalon_slave_0 =
command_0_avalon_slave_0_end_xfer &
(~command_0_avalon_slave_0_any_bursting_master_saved_grant | in_a_read_cycle |
in_a_write_cycle);

  //command_0_avalon_slave_0_arb_share_counter arbitration counter enable, which is an
e_assign
  assign command_0_avalon_slave_0_arb_counter_enable =
(end_xfer_arb_share_counter_term_command_0_avalon_slave_0 &
command_0_avalon_slave_0_allgrants) |
(end_xfer_arb_share_counter_term_command_0_avalon_slave_0 &
~command_0_avalon_slave_0_non_bursting_master_requests);

  //command_0_avalon_slave_0_arb_share_counter counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          command_0_avalon_slave_0_arb_share_counter <= 0;
      else if (command_0_avalon_slave_0_arb_counter_enable)
          command_0_avalon_slave_0_arb_share_counter <=
command_0_avalon_slave_0_arb_share_counter_next_value;
    end


  //command_0_avalon_slave_0_slavearbiterlockenable slave enables arbiterlock, which is an
e_register
  always @(posedge clk or negedge reset_n)
    begin
```

```
        if (reset_n == 0)
            command_0_avalon_slave_0_slavearbiterlockenable <= 0;
        else if ((|command_0_avalon_slave_0_master_qreq_vector &
end_xfer_arb_share_counter_term_command_0_avalon_slave_0) |
(end_xfer_arb_share_counter_term_command_0_avalon_slave_0 &
~command_0_avalon_slave_0_non_bursting_master_requests))
            command_0_avalon_slave_0_slavearbiterlockenable <=
|command_0_avalon_slave_0_arb_share_counter_next_value;
      end


   //cpu_0/data_master command_0/avalon_slave_0 arbiterlock, which is an e_assign
   assign cpu_0_data_master_arbiterlock = command_0_avalon_slave_0_slavearbiterlockenable
& cpu_0_data_master_continuerequest;

   //command_0_avalon_slave_0_slavearbiterlockenable2 slave enables arbiterlock2, which is an
e_assign
   assign command_0_avalon_slave_0_slavearbiterlockenable2 =
|command_0_avalon_slave_0_arb_share_counter_next_value;

   //cpu_0/data_master command_0/avalon_slave_0 arbiterlock2, which is an e_assign
   assign cpu_0_data_master_arbiterlock2 =
command_0_avalon_slave_0_slavearbiterlockenable2 & cpu_0_data_master_continuerequest;

   //cpu_0/instruction_master command_0/avalon_slave_0 arbiterlock, which is an e_assign
   assign cpu_0_instruction_master_arbiterlock =
command_0_avalon_slave_0_slavearbiterlockenable &
cpu_0_instruction_master_continuerequest;

   //cpu_0/instruction_master command_0/avalon_slave_0 arbiterlock2, which is an e_assign
   assign cpu_0_instruction_master_arbiterlock2 =
command_0_avalon_slave_0_slavearbiterlockenable2 &
cpu_0_instruction_master_continuerequest;

   //cpu_0/instruction_master granted command_0/avalon_slave_0 last time, which is an
e_register
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
            last_cycle_cpu_0_instruction_master_granted_slave_command_0_avalon_slave_0
<= 0;
         else
           last_cycle_cpu_0_instruction_master_granted_slave_command_0_avalon_slave_0 <=
cpu_0_instruction_master_saved_grant_command_0_avalon_slave_0 ? 1 :
(command_0_avalon_slave_0_arbitration_holdoff_internal |
~cpu_0_instruction_master_requests_command_0_avalon_slave_0) ? 0 :
last_cycle_cpu_0_instruction_master_granted_slave_command_0_avalon_slave_0;
      end


   //cpu_0_instruction_master_continuerequest continued request, which is an e_mux
   assign cpu_0_instruction_master_continuerequest =
last_cycle_cpu_0_instruction_master_granted_slave_command_0_avalon_slave_0 &
cpu_0_instruction_master_requests_command_0_avalon_slave_0;

   //command_0_avalon_slave_0_any_continuerequest at least one master continues
requesting, which is an e_mux
   assign command_0_avalon_slave_0_any_continuerequest =
```

```
cpu_0_instruction_master_continuerequest |
      cpu_0_data_master_continuerequest;

   assign cpu_0_data_master_qualified_request_command_0_avalon_slave_0 =
cpu_0_data_master_requests_command_0_avalon_slave_0 &
~(((~cpu_0_data_master_waitrequest) & cpu_0_data_master_write) |
cpu_0_instruction_master_arbiterlock);
   //command_0_avalon_slave_0_writedata mux, which is an e_mux
   assign command_0_avalon_slave_0_writedata = cpu_0_data_master_writedata;

   assign cpu_0_instruction_master_requests_command_0_avalon_slave_0 =
(({cpu_0_instruction_master_address_to_slave[24 : 7] , 7'b0} == 25'h100) &
(cpu_0_instruction_master_read)) & cpu_0_instruction_master_read;
   //cpu_0/data_master granted command_0/avalon_slave_0 last time, which is an e_register
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               last_cycle_cpu_0_data_master_granted_slave_command_0_avalon_slave_0 <= 0;
         else
            last_cycle_cpu_0_data_master_granted_slave_command_0_avalon_slave_0 <=
cpu_0_data_master_saved_grant_command_0_avalon_slave_0 ? 1 :
(command_0_avalon_slave_0_arbitration_holdoff_internal |
~cpu_0_data_master_requests_command_0_avalon_slave_0) ? 0 :
last_cycle_cpu_0_data_master_granted_slave_command_0_avalon_slave_0;
      end


   //cpu_0_data_master_continuerequest continued request, which is an e_mux
   assign cpu_0_data_master_continuerequest =
last_cycle_cpu_0_data_master_granted_slave_command_0_avalon_slave_0 &
cpu_0_data_master_requests_command_0_avalon_slave_0;

   assign cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0 =
cpu_0_instruction_master_requests_command_0_avalon_slave_0 &
~(cpu_0_data_master_arbiterlock);
   //allow new arb cycle for command_0/avalon_slave_0, which is an e_assign
   assign command_0_avalon_slave_0_allow_new_arb_cycle = ~cpu_0_data_master_arbiterlock
& ~cpu_0_instruction_master_arbiterlock;

   //cpu_0/instruction_master assignment into master qualified-requests vector for
command_0/avalon_slave_0, which is an e_assign
   assign command_0_avalon_slave_0_master_qreq_vector[0] =
cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0;

   //cpu_0/instruction_master grant command_0/avalon_slave_0, which is an e_assign
   assign cpu_0_instruction_master_granted_command_0_avalon_slave_0 =
command_0_avalon_slave_0_grant_vector[0];

   //cpu_0/instruction_master saved-grant command_0/avalon_slave_0, which is an e_assign
   assign cpu_0_instruction_master_saved_grant_command_0_avalon_slave_0 =
command_0_avalon_slave_0_arb_winner[0] &&
cpu_0_instruction_master_requests_command_0_avalon_slave_0;

   //cpu_0/data_master assignment into master qualified-requests vector for
command_0/avalon_slave_0, which is an e_assign
   assign command_0_avalon_slave_0_master_qreq_vector[1] =
cpu_0_data_master_qualified_request_command_0_avalon_slave_0;
```

```
  //cpu_0/data_master grant command_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_granted_command_0_avalon_slave_0 =
command_0_avalon_slave_0_grant_vector[1];

  //cpu_0/data_master saved-grant command_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_saved_grant_command_0_avalon_slave_0 =
command_0_avalon_slave_0_arb_winner[1] &&
cpu_0_data_master_requests_command_0_avalon_slave_0;

  //command_0/avalon_slave_0 chosen-master double-vector, which is an e_assign
  assign command_0_avalon_slave_0_chosen_master_double_vector =
{command_0_avalon_slave_0_master_qreq_vector,
command_0_avalon_slave_0_master_qreq_vector} &
({~command_0_avalon_slave_0_master_qreq_vector,
~command_0_avalon_slave_0_master_qreq_vector} +
command_0_avalon_slave_0_arb_addend);

  //stable onehot encoding of arb winner
  assign command_0_avalon_slave_0_arb_winner =
(command_0_avalon_slave_0_allow_new_arb_cycle & |
command_0_avalon_slave_0_grant_vector) ? command_0_avalon_slave_0_grant_vector :
command_0_avalon_slave_0_saved_chosen_master_vector;

  //saved command_0_avalon_slave_0_grant_vector, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          command_0_avalon_slave_0_saved_chosen_master_vector <= 0;
      else if (command_0_avalon_slave_0_allow_new_arb_cycle)
          command_0_avalon_slave_0_saved_chosen_master_vector <=
|command_0_avalon_slave_0_grant_vector ? command_0_avalon_slave_0_grant_vector :
command_0_avalon_slave_0_saved_chosen_master_vector;
    end


  //onehot encoding of chosen master
  assign command_0_avalon_slave_0_grant_vector =
{(command_0_avalon_slave_0_chosen_master_double_vector[1] |
command_0_avalon_slave_0_chosen_master_double_vector[3]),
    (command_0_avalon_slave_0_chosen_master_double_vector[0] |
command_0_avalon_slave_0_chosen_master_double_vector[2])};

  //command_0/avalon_slave_0 chosen master rotated left, which is an e_assign
  assign command_0_avalon_slave_0_chosen_master_rot_left =
(command_0_avalon_slave_0_arb_winner << 1) ? (command_0_avalon_slave_0_arb_winner <<
1) : 1;

  //command_0/avalon_slave_0's addend for next-master-grant
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          command_0_avalon_slave_0_arb_addend <= 1;
      else if (|command_0_avalon_slave_0_grant_vector)
          command_0_avalon_slave_0_arb_addend <=
command_0_avalon_slave_0_end_xfer? command_0_avalon_slave_0_chosen_master_rot_left :
command_0_avalon_slave_0_grant_vector;
    end
```

```
  //command_0_avalon_slave_0_reset_n assignment, which is an e_assign
  assign command_0_avalon_slave_0_reset_n = reset_n;

  assign                          command_0_avalon_slave_0_chipselect                          =
cpu_0_data_master_granted_command_0_avalon_slave_0                                             |
cpu_0_instruction_master_granted_command_0_avalon_slave_0;
  //command_0_avalon_slave_0_firsttransfer first transaction, which is an e_assign
  assign command_0_avalon_slave_0_firsttransfer = command_0_avalon_slave_0_begins_xfer ?
command_0_avalon_slave_0_unreg_firsttransfer                                                   :
command_0_avalon_slave_0_reg_firsttransfer;

  //command_0_avalon_slave_0_unreg_firsttransfer first transaction, which is an e_assign
  assign                    command_0_avalon_slave_0_unreg_firsttransfer                       =
~(command_0_avalon_slave_0_slavearbiterlockenable                                              &
command_0_avalon_slave_0_any_continuerequest);

  //command_0_avalon_slave_0_reg_firsttransfer first transaction, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          command_0_avalon_slave_0_reg_firsttransfer <= 1'b1;
      else if (command_0_avalon_slave_0_begins_xfer)
          command_0_avalon_slave_0_reg_firsttransfer                                         <=
command_0_avalon_slave_0_unreg_firsttransfer;
    end


  //command_0_avalon_slave_0_beginbursttransfer_internal begin burst transfer, which is an
e_assign
  assign              command_0_avalon_slave_0_beginbursttransfer_internal                   =
command_0_avalon_slave_0_begins_xfer;

  //command_0_avalon_slave_0_arbitration_holdoff_internal arbitration_holdoff, which is an
e_assign
  assign              command_0_avalon_slave_0_arbitration_holdoff_internal                  =
command_0_avalon_slave_0_begins_xfer & command_0_avalon_slave_0_firsttransfer;

  //command_0_avalon_slave_0_read assignment, which is an e_mux
  assign                          command_0_avalon_slave_0_read                              =
(cpu_0_data_master_granted_command_0_avalon_slave_0   &   cpu_0_data_master_read)   |
(cpu_0_instruction_master_granted_command_0_avalon_slave_0                                     &
cpu_0_instruction_master_read);

  //command_0_avalon_slave_0_write assignment, which is an e_mux
  assign                          command_0_avalon_slave_0_write                             =
cpu_0_data_master_granted_command_0_avalon_slave_0 & cpu_0_data_master_write;

  assign      shifted_address_to_command_0_avalon_slave_0_from_cpu_0_data_master      =
cpu_0_data_master_address_to_slave;
  //command_0_avalon_slave_0_address mux, which is an e_mux
  assign                          command_0_avalon_slave_0_address                           =
(cpu_0_data_master_granted_command_0_avalon_slave_0)?
(shifted_address_to_command_0_avalon_slave_0_from_cpu_0_data_master >> 2) :
    (shifted_address_to_command_0_avalon_slave_0_from_cpu_0_instruction_master >> 2);

  assign   shifted_address_to_command_0_avalon_slave_0_from_cpu_0_instruction_master   =
cpu_0_instruction_master_address_to_slave;
```

```
  //d1_command_0_avalon_slave_0_end_xfer register, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_command_0_avalon_slave_0_end_xfer <= 1;
      else
        d1_command_0_avalon_slave_0_end_xfer <= command_0_avalon_slave_0_end_xfer;
    end


  //command_0_avalon_slave_0_waits_for_read in a cycle, which is an e_mux
  assign command_0_avalon_slave_0_waits_for_read =
command_0_avalon_slave_0_in_a_read_cycle & command_0_avalon_slave_0_begins_xfer;

  //command_0_avalon_slave_0_in_a_read_cycle assignment, which is an e_assign
  assign command_0_avalon_slave_0_in_a_read_cycle =
(cpu_0_data_master_granted_command_0_avalon_slave_0 & cpu_0_data_master_read) |
(cpu_0_instruction_master_granted_command_0_avalon_slave_0 &
cpu_0_instruction_master_read);

  //in_a_read_cycle assignment, which is an e_mux
  assign in_a_read_cycle = command_0_avalon_slave_0_in_a_read_cycle;

  //command_0_avalon_slave_0_waits_for_write in a cycle, which is an e_mux
  assign command_0_avalon_slave_0_waits_for_write =
command_0_avalon_slave_0_in_a_write_cycle & 0;

  //command_0_avalon_slave_0_in_a_write_cycle assignment, which is an e_assign
  assign command_0_avalon_slave_0_in_a_write_cycle =
cpu_0_data_master_granted_command_0_avalon_slave_0 & cpu_0_data_master_write;

  //in_a_write_cycle assignment, which is an e_mux
  assign in_a_write_cycle = command_0_avalon_slave_0_in_a_write_cycle;

  assign wait_for_command_0_avalon_slave_0_counter = 0;

//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
  //command_0/avalon_slave_0 enable non-zero assertions, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          enable_nonzero_assertions <= 0;
      else
        enable_nonzero_assertions <= 1'b1;
    end


  //grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if (cpu_0_data_master_granted_command_0_avalon_slave_0 +
cpu_0_instruction_master_granted_command_0_avalon_slave_0 > 1)
        begin
          $write("%0d ns: > 1 of grant signals are active simultaneously", $time);
          $stop;
        end
    end
```

```
  //saved_grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if (cpu_0_data_master_saved_grant_command_0_avalon_slave_0 +
cpu_0_instruction_master_saved_grant_command_0_avalon_slave_0 > 1)
        begin
          $write("%0d ns: > 1 of saved_grant signals are active simultaneously", $time);
          $stop;
        end
    end



//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_jtag_debug_module_arbitrator (
                                            // inputs:
                                              clk,
                                              cpu_0_data_master_address_to_slave,
                                              cpu_0_data_master_byteenable,
                                              cpu_0_data_master_debugaccess,
                                              cpu_0_data_master_read,
                                              cpu_0_data_master_waitrequest,
                                              cpu_0_data_master_write,
                                              cpu_0_data_master_writedata,

cpu_0_instruction_master_address_to_slave,
                                              cpu_0_instruction_master_read,
                                              cpu_0_jtag_debug_module_readdata,

cpu_0_jtag_debug_module_resetrequest,
                                              reset_n,

                                            // outputs:

cpu_0_data_master_granted_cpu_0_jtag_debug_module,

cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module,

cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module,

cpu_0_data_master_requests_cpu_0_jtag_debug_module,
```

```
cpu_0_instruction_master_granted_cpu_0_jtag_debug_module,

cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module,

cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module,

cpu_0_instruction_master_requests_cpu_0_jtag_debug_module,
                                                    cpu_0_jtag_debug_module_address,

cpu_0_jtag_debug_module_begintransfer,
                                                    cpu_0_jtag_debug_module_byteenable,
                                                    cpu_0_jtag_debug_module_chipselect,
                                                    cpu_0_jtag_debug_module_debugaccess,

cpu_0_jtag_debug_module_readdata_from_sa,
                                                    cpu_0_jtag_debug_module_reset_n,

cpu_0_jtag_debug_module_resetrequest_from_sa,
                                                    cpu_0_jtag_debug_module_write,
                                                    cpu_0_jtag_debug_module_writedata,
                                                    d1_cpu_0_jtag_debug_module_end_xfer
                                                )
;

  output           cpu_0_data_master_granted_cpu_0_jtag_debug_module;
  output           cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module;
  output           cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module;
  output           cpu_0_data_master_requests_cpu_0_jtag_debug_module;
  output           cpu_0_instruction_master_granted_cpu_0_jtag_debug_module;
  output
cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module;
  output           cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module;
  output           cpu_0_instruction_master_requests_cpu_0_jtag_debug_module;
  output  [  8: 0] cpu_0_jtag_debug_module_address;
  output           cpu_0_jtag_debug_module_begintransfer;
  output  [  3: 0] cpu_0_jtag_debug_module_byteenable;
  output           cpu_0_jtag_debug_module_chipselect;
  output           cpu_0_jtag_debug_module_debugaccess;
  output  [ 31: 0] cpu_0_jtag_debug_module_readdata_from_sa;
  output           cpu_0_jtag_debug_module_reset_n;
  output           cpu_0_jtag_debug_module_resetrequest_from_sa;
  output           cpu_0_jtag_debug_module_write;
  output  [ 31: 0] cpu_0_jtag_debug_module_writedata;
  output           d1_cpu_0_jtag_debug_module_end_xfer;
  input            clk;
  input   [ 24: 0] cpu_0_data_master_address_to_slave;
  input   [  3: 0] cpu_0_data_master_byteenable;
  input            cpu_0_data_master_debugaccess;
  input            cpu_0_data_master_read;
  input            cpu_0_data_master_waitrequest;
  input            cpu_0_data_master_write;
  input   [ 31: 0] cpu_0_data_master_writedata;
  input   [ 24: 0] cpu_0_instruction_master_address_to_slave;
  input            cpu_0_instruction_master_read;
  input   [ 31: 0] cpu_0_jtag_debug_module_readdata;
  input            cpu_0_jtag_debug_module_resetrequest;
  input            reset_n;
```

```
  wire              cpu_0_data_master_arbiterlock;
  wire              cpu_0_data_master_arbiterlock2;
  wire              cpu_0_data_master_continuerequest;
  wire              cpu_0_data_master_granted_cpu_0_jtag_debug_module;
  wire              cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module;
  wire              cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module;
  wire              cpu_0_data_master_requests_cpu_0_jtag_debug_module;
  wire              cpu_0_data_master_saved_grant_cpu_0_jtag_debug_module;
  wire              cpu_0_instruction_master_arbiterlock;
  wire              cpu_0_instruction_master_arbiterlock2;
  wire              cpu_0_instruction_master_continuerequest;
  wire              cpu_0_instruction_master_granted_cpu_0_jtag_debug_module;
  wire
cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module;
  wire              cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module;
  wire              cpu_0_instruction_master_requests_cpu_0_jtag_debug_module;
  wire              cpu_0_instruction_master_saved_grant_cpu_0_jtag_debug_module;
  wire    [  8: 0] cpu_0_jtag_debug_module_address;
  wire              cpu_0_jtag_debug_module_allgrants;
  wire              cpu_0_jtag_debug_module_allow_new_arb_cycle;
  wire              cpu_0_jtag_debug_module_any_bursting_master_saved_grant;
  wire              cpu_0_jtag_debug_module_any_continuerequest;
  reg     [  1: 0] cpu_0_jtag_debug_module_arb_addend;
  wire              cpu_0_jtag_debug_module_arb_counter_enable;
  reg     [  1: 0] cpu_0_jtag_debug_module_arb_share_counter;
  wire    [  1: 0] cpu_0_jtag_debug_module_arb_share_counter_next_value;
  wire    [  1: 0] cpu_0_jtag_debug_module_arb_share_set_values;
  wire    [  1: 0] cpu_0_jtag_debug_module_arb_winner;
  wire              cpu_0_jtag_debug_module_arbitration_holdoff_internal;
  wire              cpu_0_jtag_debug_module_beginbursttransfer_internal;
  wire              cpu_0_jtag_debug_module_begins_xfer;
  wire              cpu_0_jtag_debug_module_begintransfer;
  wire    [  3: 0] cpu_0_jtag_debug_module_byteenable;
  wire              cpu_0_jtag_debug_module_chipselect;
  wire    [  3: 0] cpu_0_jtag_debug_module_chosen_master_double_vector;
  wire    [  1: 0] cpu_0_jtag_debug_module_chosen_master_rot_left;
  wire              cpu_0_jtag_debug_module_debugaccess;
  wire              cpu_0_jtag_debug_module_end_xfer;
  wire              cpu_0_jtag_debug_module_firsttransfer;
  wire    [  1: 0] cpu_0_jtag_debug_module_grant_vector;
  wire              cpu_0_jtag_debug_module_in_a_read_cycle;
  wire              cpu_0_jtag_debug_module_in_a_write_cycle;
  wire    [  1: 0] cpu_0_jtag_debug_module_master_qreq_vector;
  wire              cpu_0_jtag_debug_module_non_bursting_master_requests;
  wire    [ 31: 0] cpu_0_jtag_debug_module_readdata_from_sa;
  reg               cpu_0_jtag_debug_module_reg_firsttransfer;
  wire              cpu_0_jtag_debug_module_reset_n;
  wire              cpu_0_jtag_debug_module_resetrequest_from_sa;
  reg     [  1: 0] cpu_0_jtag_debug_module_saved_chosen_master_vector;
  reg               cpu_0_jtag_debug_module_slavearbiterlockenable;
  wire              cpu_0_jtag_debug_module_slavearbiterlockenable2;
  wire              cpu_0_jtag_debug_module_unreg_firsttransfer;
  wire              cpu_0_jtag_debug_module_waits_for_read;
  wire              cpu_0_jtag_debug_module_waits_for_write;
  wire              cpu_0_jtag_debug_module_write;
  wire    [ 31: 0] cpu_0_jtag_debug_module_writedata;
  reg               d1_cpu_0_jtag_debug_module_end_xfer;
  reg               d1_reasons_to_wait;
```

```
  reg             enable_nonzero_assertions;
  wire            end_xfer_arb_share_counter_term_cpu_0_jtag_debug_module;
  wire            in_a_read_cycle;
  wire            in_a_write_cycle;
  reg
last_cycle_cpu_0_data_master_granted_slave_cpu_0_jtag_debug_module;
  reg
last_cycle_cpu_0_instruction_master_granted_slave_cpu_0_jtag_debug_module;
  wire      [ 24: 0] shifted_address_to_cpu_0_jtag_debug_module_from_cpu_0_data_master;
  wire                                                    [ 24: 0]
shifted_address_to_cpu_0_jtag_debug_module_from_cpu_0_instruction_master;
  wire            wait_for_cpu_0_jtag_debug_module_counter;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_reasons_to_wait <= 0;
      else
        d1_reasons_to_wait <= ~cpu_0_jtag_debug_module_end_xfer;
    end


  assign      cpu_0_jtag_debug_module_begins_xfer      =      ~d1_reasons_to_wait      &
((cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module                          |
cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module));
  //assign                  cpu_0_jtag_debug_module_readdata_from_sa                  =
cpu_0_jtag_debug_module_readdata so that symbol knows where to group signals which may
go to master only, which is an e_assign
  assign cpu_0_jtag_debug_module_readdata_from_sa = cpu_0_jtag_debug_module_readdata;

  assign            cpu_0_data_master_requests_cpu_0_jtag_debug_module            =
({cpu_0_data_master_address_to_slave[24    :    11]    ,    11'b0}    ==    25'h1000800)    &
(cpu_0_data_master_read | cpu_0_data_master_write);
  //cpu_0_jtag_debug_module_arb_share_counter set values, which is an e_mux
  assign cpu_0_jtag_debug_module_arb_share_set_values = 1;

  //cpu_0_jtag_debug_module_non_bursting_master_requests mux, which is an e_mux
  assign            cpu_0_jtag_debug_module_non_bursting_master_requests            =
cpu_0_data_master_requests_cpu_0_jtag_debug_module |
    cpu_0_instruction_master_requests_cpu_0_jtag_debug_module |
    cpu_0_data_master_requests_cpu_0_jtag_debug_module |
    cpu_0_instruction_master_requests_cpu_0_jtag_debug_module;

  //cpu_0_jtag_debug_module_any_bursting_master_saved_grant mux, which is an e_mux
  assign cpu_0_jtag_debug_module_any_bursting_master_saved_grant = 0;

  //cpu_0_jtag_debug_module_arb_share_counter_next_value    assignment,    which    is    an
e_assign
  assign            cpu_0_jtag_debug_module_arb_share_counter_next_value            =
cpu_0_jtag_debug_module_firsttransfer  ?  (cpu_0_jtag_debug_module_arb_share_set_values  -
1)              :              |cpu_0_jtag_debug_module_arb_share_counter              ?
(cpu_0_jtag_debug_module_arb_share_counter - 1) : 0;

  //cpu_0_jtag_debug_module_allgrants all slave grants, which is an e_mux
  assign cpu_0_jtag_debug_module_allgrants = (|cpu_0_jtag_debug_module_grant_vector) |
    (|cpu_0_jtag_debug_module_grant_vector) |
    (|cpu_0_jtag_debug_module_grant_vector) |
    (|cpu_0_jtag_debug_module_grant_vector);
```

```
  //cpu_0_jtag_debug_module_end_xfer assignment, which is an e_assign
  assign cpu_0_jtag_debug_module_end_xfer = ~(cpu_0_jtag_debug_module_waits_for_read |
cpu_0_jtag_debug_module_waits_for_write);

  //end_xfer_arb_share_counter_term_cpu_0_jtag_debug_module arb share counter enable
term, which is an e_assign
  assign end_xfer_arb_share_counter_term_cpu_0_jtag_debug_module =
cpu_0_jtag_debug_module_end_xfer &
(~cpu_0_jtag_debug_module_any_bursting_master_saved_grant | in_a_read_cycle |
in_a_write_cycle);

  //cpu_0_jtag_debug_module_arb_share_counter arbitration counter enable, which is an
e_assign
  assign cpu_0_jtag_debug_module_arb_counter_enable =
(end_xfer_arb_share_counter_term_cpu_0_jtag_debug_module &
cpu_0_jtag_debug_module_allgrants) |
(end_xfer_arb_share_counter_term_cpu_0_jtag_debug_module &
~cpu_0_jtag_debug_module_non_bursting_master_requests);

  //cpu_0_jtag_debug_module_arb_share_counter counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          cpu_0_jtag_debug_module_arb_share_counter <= 0;
      else if (cpu_0_jtag_debug_module_arb_counter_enable)
          cpu_0_jtag_debug_module_arb_share_counter <=
cpu_0_jtag_debug_module_arb_share_counter_next_value;
    end


  //cpu_0_jtag_debug_module_slavearbiterlockenable slave enables arbiterlock, which is an
e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          cpu_0_jtag_debug_module_slavearbiterlockenable <= 0;
      else if (({|cpu_0_jtag_debug_module_master_qreq_vector &
end_xfer_arb_share_counter_term_cpu_0_jtag_debug_module) |
(end_xfer_arb_share_counter_term_cpu_0_jtag_debug_module &
~cpu_0_jtag_debug_module_non_bursting_master_requests))
          cpu_0_jtag_debug_module_slavearbiterlockenable <=
|cpu_0_jtag_debug_module_arb_share_counter_next_value;
    end


  //cpu_0/data_master cpu_0/jtag_debug_module arbiterlock, which is an e_assign
  assign cpu_0_data_master_arbiterlock = cpu_0_jtag_debug_module_slavearbiterlockenable &
cpu_0_data_master_continuerequest;

  //cpu_0_jtag_debug_module_slavearbiterlockenable2 slave enables arbiterlock2, which is an
e_assign
  assign cpu_0_jtag_debug_module_slavearbiterlockenable2 =
|cpu_0_jtag_debug_module_arb_share_counter_next_value;

  //cpu_0/data_master cpu_0/jtag_debug_module arbiterlock2, which is an e_assign
  assign cpu_0_data_master_arbiterlock2 =
cpu_0_jtag_debug_module_slavearbiterlockenable2 & cpu_0_data_master_continuerequest;
```

```
  //cpu_0/instruction_master cpu_0/jtag_debug_module arbiterlock, which is an e_assign
  assign cpu_0_instruction_master_arbiterlock = cpu_0_jtag_debug_module_slavearbiterlockenable & cpu_0_instruction_master_continuerequest;

  //cpu_0/instruction_master cpu_0/jtag_debug_module arbiterlock2, which is an e_assign
  assign cpu_0_instruction_master_arbiterlock2 = cpu_0_jtag_debug_module_slavearbiterlockenable2 & cpu_0_instruction_master_continuerequest;

  //cpu_0/instruction_master granted cpu_0/jtag_debug_module last time, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          last_cycle_cpu_0_instruction_master_granted_slave_cpu_0_jtag_debug_module <= 0;
      else
        last_cycle_cpu_0_instruction_master_granted_slave_cpu_0_jtag_debug_module <= cpu_0_instruction_master_saved_grant_cpu_0_jtag_debug_module ? 1 : (cpu_0_jtag_debug_module_arbitration_holdoff_internal | ~cpu_0_instruction_master_requests_cpu_0_jtag_debug_module) ? 0 : last_cycle_cpu_0_instruction_master_granted_slave_cpu_0_jtag_debug_module;
    end


  //cpu_0_instruction_master_continuerequest continued request, which is an e_mux
  assign cpu_0_instruction_master_continuerequest = last_cycle_cpu_0_instruction_master_granted_slave_cpu_0_jtag_debug_module & cpu_0_instruction_master_requests_cpu_0_jtag_debug_module;

  //cpu_0_jtag_debug_module_any_continuerequest at least one master continues requesting, which is an e_mux
  assign cpu_0_jtag_debug_module_any_continuerequest = cpu_0_instruction_master_continuerequest |
    cpu_0_data_master_continuerequest;

  assign cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module = cpu_0_data_master_requests_cpu_0_jtag_debug_module & ~(((~cpu_0_data_master_waitrequest) & cpu_0_data_master_write) | cpu_0_instruction_master_arbiterlock);
  //cpu_0_jtag_debug_module_writedata mux, which is an e_mux
  assign cpu_0_jtag_debug_module_writedata = cpu_0_data_master_writedata;

  assign cpu_0_instruction_master_requests_cpu_0_jtag_debug_module = (({cpu_0_instruction_master_address_to_slave[24 : 11] , 11'b0} == 25'h1000800) & (cpu_0_instruction_master_read)) & cpu_0_instruction_master_read;
  //cpu_0/data_master granted cpu_0/jtag_debug_module last time, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          last_cycle_cpu_0_data_master_granted_slave_cpu_0_jtag_debug_module <= 0;
      else
        last_cycle_cpu_0_data_master_granted_slave_cpu_0_jtag_debug_module <= cpu_0_data_master_saved_grant_cpu_0_jtag_debug_module ? 1 : (cpu_0_jtag_debug_module_arbitration_holdoff_internal | ~cpu_0_data_master_requests_cpu_0_jtag_debug_module) ? 0 : last_cycle_cpu_0_data_master_granted_slave_cpu_0_jtag_debug_module;
```

```
    end


  //cpu_0_data_master_continuerequest continued request, which is an e_mux
  assign cpu_0_data_master_continuerequest =
last_cycle_cpu_0_data_master_granted_slave_cpu_0_jtag_debug_module &
cpu_0_data_master_requests_cpu_0_jtag_debug_module;

  assign cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module =
cpu_0_instruction_master_requests_cpu_0_jtag_debug_module &
~(cpu_0_data_master_arbiterlock);
  //allow new arb cycle for cpu_0/jtag_debug_module, which is an e_assign
  assign cpu_0_jtag_debug_module_allow_new_arb_cycle = ~cpu_0_data_master_arbiterlock &
~cpu_0_instruction_master_arbiterlock;

  //cpu_0/instruction_master assignment into master qualified-requests vector for
cpu_0/jtag_debug_module, which is an e_assign
  assign cpu_0_jtag_debug_module_master_qreq_vector[0] =
cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module;

  //cpu_0/instruction_master grant cpu_0/jtag_debug_module, which is an e_assign
  assign cpu_0_instruction_master_granted_cpu_0_jtag_debug_module =
cpu_0_jtag_debug_module_grant_vector[0];

  //cpu_0/instruction_master saved-grant cpu_0/jtag_debug_module, which is an e_assign
  assign cpu_0_instruction_master_saved_grant_cpu_0_jtag_debug_module =
cpu_0_jtag_debug_module_arb_winner[0] &&
cpu_0_instruction_master_requests_cpu_0_jtag_debug_module;

  //cpu_0/data_master assignment into master qualified-requests vector for
cpu_0/jtag_debug_module, which is an e_assign
  assign cpu_0_jtag_debug_module_master_qreq_vector[1] =
cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module;

  //cpu_0/data_master grant cpu_0/jtag_debug_module, which is an e_assign
  assign cpu_0_data_master_granted_cpu_0_jtag_debug_module =
cpu_0_jtag_debug_module_grant_vector[1];

  //cpu_0/data_master saved-grant cpu_0/jtag_debug_module, which is an e_assign
  assign cpu_0_data_master_saved_grant_cpu_0_jtag_debug_module =
cpu_0_jtag_debug_module_arb_winner[1] &&
cpu_0_data_master_requests_cpu_0_jtag_debug_module;

  //cpu_0/jtag_debug_module chosen-master double-vector, which is an e_assign
  assign cpu_0_jtag_debug_module_chosen_master_double_vector =
{cpu_0_jtag_debug_module_master_qreq_vector,
cpu_0_jtag_debug_module_master_qreq_vector} &
({~cpu_0_jtag_debug_module_master_qreq_vector,
~cpu_0_jtag_debug_module_master_qreq_vector} + cpu_0_jtag_debug_module_arb_addend);

  //stable onehot encoding of arb winner
  assign cpu_0_jtag_debug_module_arb_winner =
(cpu_0_jtag_debug_module_allow_new_arb_cycle & |
cpu_0_jtag_debug_module_grant_vector) ? cpu_0_jtag_debug_module_grant_vector :
cpu_0_jtag_debug_module_saved_chosen_master_vector;

  //saved cpu_0_jtag_debug_module_grant_vector, which is an e_register
  always @(posedge clk or negedge reset_n)
```

```
    begin
      if (reset_n == 0)
          cpu_0_jtag_debug_module_saved_chosen_master_vector <= 0;
      else if (cpu_0_jtag_debug_module_allow_new_arb_cycle)
          cpu_0_jtag_debug_module_saved_chosen_master_vector <=
|cpu_0_jtag_debug_module_grant_vector ? cpu_0_jtag_debug_module_grant_vector :
cpu_0_jtag_debug_module_saved_chosen_master_vector;
    end


  //onehot encoding of chosen master
  assign cpu_0_jtag_debug_module_grant_vector =
{(cpu_0_jtag_debug_module_chosen_master_double_vector[1] |
cpu_0_jtag_debug_module_chosen_master_double_vector[3]),
    (cpu_0_jtag_debug_module_chosen_master_double_vector[0] |
cpu_0_jtag_debug_module_chosen_master_double_vector[2])};

  //cpu_0/jtag_debug_module chosen master rotated left, which is an e_assign
  assign cpu_0_jtag_debug_module_chosen_master_rot_left =
(cpu_0_jtag_debug_module_arb_winner << 1) ? (cpu_0_jtag_debug_module_arb_winner << 1) :
1;

  //cpu_0/jtag_debug_module's addend for next-master-grant
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          cpu_0_jtag_debug_module_arb_addend <= 1;
      else if (|cpu_0_jtag_debug_module_grant_vector)
          cpu_0_jtag_debug_module_arb_addend <= cpu_0_jtag_debug_module_end_xfer?
cpu_0_jtag_debug_module_chosen_master_rot_left : cpu_0_jtag_debug_module_grant_vector;
    end


  assign cpu_0_jtag_debug_module_begintransfer = cpu_0_jtag_debug_module_begins_xfer;
  //cpu_0_jtag_debug_module_reset_n assignment, which is an e_assign
  assign cpu_0_jtag_debug_module_reset_n = reset_n;

  //assign cpu_0_jtag_debug_module_resetrequest_from_sa =
cpu_0_jtag_debug_module_resetrequest so that symbol knows where to group signals which
may go to master only, which is an e_assign
  assign cpu_0_jtag_debug_module_resetrequest_from_sa =
cpu_0_jtag_debug_module_resetrequest;

  assign cpu_0_jtag_debug_module_chipselect =
cpu_0_data_master_granted_cpu_0_jtag_debug_module |
cpu_0_instruction_master_granted_cpu_0_jtag_debug_module;
  //cpu_0_jtag_debug_module_firsttransfer first transaction, which is an e_assign
  assign cpu_0_jtag_debug_module_firsttransfer = cpu_0_jtag_debug_module_begins_xfer ?
cpu_0_jtag_debug_module_unreg_firsttransfer : cpu_0_jtag_debug_module_reg_firsttransfer;

  //cpu_0_jtag_debug_module_unreg_firsttransfer first transaction, which is an e_assign
  assign cpu_0_jtag_debug_module_unreg_firsttransfer =
~(cpu_0_jtag_debug_module_slavearbiterlockenable &
cpu_0_jtag_debug_module_any_continuerequest);

  //cpu_0_jtag_debug_module_reg_firsttransfer first transaction, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
```

```
        if (reset_n == 0)
            cpu_0_jtag_debug_module_reg_firsttransfer <= 1'b1;
        else if (cpu_0_jtag_debug_module_begins_xfer)
            cpu_0_jtag_debug_module_reg_firsttransfer <=
cpu_0_jtag_debug_module_unreg_firsttransfer;
      end


  //cpu_0_jtag_debug_module_beginbursttransfer_internal begin burst transfer, which is an
e_assign
  assign cpu_0_jtag_debug_module_beginbursttransfer_internal =
cpu_0_jtag_debug_module_begins_xfer;

  //cpu_0_jtag_debug_module_arbitration_holdoff_internal arbitration_holdoff, which is an
e_assign
  assign cpu_0_jtag_debug_module_arbitration_holdoff_internal =
cpu_0_jtag_debug_module_begins_xfer & cpu_0_jtag_debug_module_firsttransfer;

  //cpu_0_jtag_debug_module_write assignment, which is an e_mux
  assign cpu_0_jtag_debug_module_write =
cpu_0_data_master_granted_cpu_0_jtag_debug_module & cpu_0_data_master_write;

  assign shifted_address_to_cpu_0_jtag_debug_module_from_cpu_0_data_master =
cpu_0_data_master_address_to_slave;
  //cpu_0_jtag_debug_module_address mux, which is an e_mux
  assign cpu_0_jtag_debug_module_address =
(cpu_0_data_master_granted_cpu_0_jtag_debug_module)?
(shifted_address_to_cpu_0_jtag_debug_module_from_cpu_0_data_master >> 2) :
    (shifted_address_to_cpu_0_jtag_debug_module_from_cpu_0_instruction_master >> 2);

  assign shifted_address_to_cpu_0_jtag_debug_module_from_cpu_0_instruction_master =
cpu_0_instruction_master_address_to_slave;
  //d1_cpu_0_jtag_debug_module_end_xfer register, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_cpu_0_jtag_debug_module_end_xfer <= 1;
      else
        d1_cpu_0_jtag_debug_module_end_xfer <= cpu_0_jtag_debug_module_end_xfer;
    end


  //cpu_0_jtag_debug_module_waits_for_read in a cycle, which is an e_mux
  assign cpu_0_jtag_debug_module_waits_for_read =
cpu_0_jtag_debug_module_in_a_read_cycle & cpu_0_jtag_debug_module_begins_xfer;

  //cpu_0_jtag_debug_module_in_a_read_cycle assignment, which is an e_assign
  assign cpu_0_jtag_debug_module_in_a_read_cycle =
(cpu_0_data_master_granted_cpu_0_jtag_debug_module & cpu_0_data_master_read) |
(cpu_0_instruction_master_granted_cpu_0_jtag_debug_module &
cpu_0_instruction_master_read);

  //in_a_read_cycle assignment, which is an e_mux
  assign in_a_read_cycle = cpu_0_jtag_debug_module_in_a_read_cycle;

  //cpu_0_jtag_debug_module_waits_for_write in a cycle, which is an e_mux
  assign cpu_0_jtag_debug_module_waits_for_write =
cpu_0_jtag_debug_module_in_a_write_cycle & 0;
```

```
  //cpu_0_jtag_debug_module_in_a_write_cycle assignment, which is an e_assign
  assign cpu_0_jtag_debug_module_in_a_write_cycle =
cpu_0_data_master_granted_cpu_0_jtag_debug_module & cpu_0_data_master_write;

  //in_a_write_cycle assignment, which is an e_mux
  assign in_a_write_cycle = cpu_0_jtag_debug_module_in_a_write_cycle;

  assign wait_for_cpu_0_jtag_debug_module_counter = 0;
  //cpu_0_jtag_debug_module_byteenable byte enable port mux, which is an e_mux
  assign cpu_0_jtag_debug_module_byteenable =
(cpu_0_data_master_granted_cpu_0_jtag_debug_module)? cpu_0_data_master_byteenable :
    -1;

  //debugaccess mux, which is an e_mux
  assign cpu_0_jtag_debug_module_debugaccess =
(cpu_0_data_master_granted_cpu_0_jtag_debug_module)? cpu_0_data_master_debugaccess :
    0;


//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
  //cpu_0/jtag_debug_module enable non-zero assertions, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          enable_nonzero_assertions <= 0;
      else
        enable_nonzero_assertions <= 1'b1;
    end


  //grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if (cpu_0_data_master_granted_cpu_0_jtag_debug_module +
cpu_0_instruction_master_granted_cpu_0_jtag_debug_module > 1)
        begin
          $write("%0d ns: > 1 of grant signals are active simultaneously", $time);
          $stop;
        end
    end


  //saved_grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if (cpu_0_data_master_saved_grant_cpu_0_jtag_debug_module +
cpu_0_instruction_master_saved_grant_cpu_0_jtag_debug_module > 1)
        begin
          $write("%0d ns: > 1 of saved_grant signals are active simultaneously", $time);
          $stop;
        end
    end


//////////////// END SIMULATION-ONLY CONTENTS
```

```
//synthesis translate_on

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_data_master_arbitrator (
                                      // inputs:

audio_controller_0_avalon_slave_0_readdata_from_sa,
                                      clk,

command_0_avalon_slave_0_readdata_from_sa,
                                      cpu_0_data_master_address,

cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0,

cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0,
                                      cpu_0_data_master_byteenable_sdram_0_s1,

cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0,

cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0,

cpu_0_data_master_granted_audio_controller_0_avalon_slave_0,

cpu_0_data_master_granted_command_0_avalon_slave_0,

cpu_0_data_master_granted_cpu_0_jtag_debug_module,

cpu_0_data_master_granted_interrupt_0_avalon_slave_0,

cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave,

cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0,
                                      cpu_0_data_master_granted_sdram_0_s1,

cpu_0_data_master_granted_sram_mux_0_avalon_slave_0,

cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0,

cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0,

cpu_0_data_master_qualified_request_command_0_avalon_slave_0,

cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module,

cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0,

cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave,
```

```
cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0,

cpu_0_data_master_qualified_request_sdram_0_s1,

cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0,

cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0,
                                                cpu_0_data_master_read,

cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_command_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module,

cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave,

cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_sdram_0_s1,

cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register,

cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0,

cpu_0_data_master_requests_audio_controller_0_avalon_slave_0,

cpu_0_data_master_requests_command_0_avalon_slave_0,

cpu_0_data_master_requests_cpu_0_jtag_debug_module,

cpu_0_data_master_requests_interrupt_0_avalon_slave_0,

cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave,

cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0,
                                                cpu_0_data_master_requests_sdram_0_s1,

cpu_0_data_master_requests_sram_mux_0_avalon_slave_0,

cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0,
                                                cpu_0_data_master_write,
                                                cpu_0_data_master_writedata,
                                                cpu_0_jtag_debug_module_readdata_from_sa,

d1_audio_controller_0_avalon_slave_0_end_xfer,
                                                d1_command_0_avalon_slave_0_end_xfer,
                                                d1_cpu_0_jtag_debug_module_end_xfer,
                                                d1_interrupt_0_avalon_slave_0_end_xfer,
                                                d1_jtag_uart_0_avalon_jtag_slave_end_xfer,
                                                d1_musiccontroller_0_avalon_slave_0_end_xfer,
                                                d1_sdram_0_s1_end_xfer,
                                                d1_sram_mux_0_avalon_slave_0_end_xfer,
```

```
d1_xycoord_interface_0_avalon_slave_0_end_xfer,
                                                    interrupt_0_avalon_slave_0_irq_from_sa,
                                                    interrupt_0_avalon_slave_0_readdata_from_sa,
                                                    jtag_uart_0_avalon_jtag_slave_irq_from_sa,

jtag_uart_0_avalon_jtag_slave_readdata_from_sa,

jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa,
                                                    musiccontroller_0_avalon_slave_0_irq_from_sa,

musiccontroller_0_avalon_slave_0_readdata_from_sa,
                                                    reset_n,
                                                    sdram_0_s1_readdata_from_sa,
                                                    sdram_0_s1_waitrequest_from_sa,

sram_mux_0_avalon_slave_0_readdata_from_sa,

xycoord_interface_0_avalon_slave_0_readdata_from_sa,

                                                   // outputs:
                                                    cpu_0_data_master_address_to_slave,
                                                    cpu_0_data_master_dbs_address,
                                                    cpu_0_data_master_dbs_write_16,
                                                    cpu_0_data_master_irq,

cpu_0_data_master_no_byte_enables_and_last_term,
                                                    cpu_0_data_master_readdata,
                                                    cpu_0_data_master_waitrequest
                                                 )
;

  output  [ 24: 0] cpu_0_data_master_address_to_slave;
  output  [  1: 0] cpu_0_data_master_dbs_address;
  output  [ 15: 0] cpu_0_data_master_dbs_write_16;
  output  [ 31: 0] cpu_0_data_master_irq;
  output           cpu_0_data_master_no_byte_enables_and_last_term;
  output  [ 31: 0] cpu_0_data_master_readdata;
  output           cpu_0_data_master_waitrequest;
  input   [ 15: 0] audio_controller_0_avalon_slave_0_readdata_from_sa;
  input            clk;
  input   [ 31: 0] command_0_avalon_slave_0_readdata_from_sa;
  input   [ 24: 0] cpu_0_data_master_address;
  input   [  1: 0] cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0;
  input   [  1: 0] cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0;
  input   [  1: 0] cpu_0_data_master_byteenable_sdram_0_s1;
  input   [  1: 0] cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0;
  input   [  1: 0] cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0;
  input            cpu_0_data_master_granted_audio_controller_0_avalon_slave_0;
  input            cpu_0_data_master_granted_command_0_avalon_slave_0;
  input            cpu_0_data_master_granted_cpu_0_jtag_debug_module;
  input            cpu_0_data_master_granted_interrupt_0_avalon_slave_0;
  input            cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave;
  input            cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0;
  input            cpu_0_data_master_granted_sdram_0_s1;
  input            cpu_0_data_master_granted_sram_mux_0_avalon_slave_0;
  input            cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0;
  input
```

```
cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0;
  input              cpu_0_data_master_qualified_request_command_0_avalon_slave_0;
  input              cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module;
  input              cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0;
  input              cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave;
  input
cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0;
  input              cpu_0_data_master_qualified_request_sdram_0_s1;
  input              cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0;
  input
cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0;
  input              cpu_0_data_master_read;
  input
cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0;
  input              cpu_0_data_master_read_data_valid_command_0_avalon_slave_0;
  input              cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module;
  input              cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0;
  input              cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave;
  input              cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0;
  input              cpu_0_data_master_read_data_valid_sdram_0_s1;
  input              cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register;
  input              cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0;
  input
cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0;
  input              cpu_0_data_master_requests_audio_controller_0_avalon_slave_0;
  input              cpu_0_data_master_requests_command_0_avalon_slave_0;
  input              cpu_0_data_master_requests_cpu_0_jtag_debug_module;
  input              cpu_0_data_master_requests_interrupt_0_avalon_slave_0;
  input              cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave;
  input              cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0;
  input              cpu_0_data_master_requests_sdram_0_s1;
  input              cpu_0_data_master_requests_sram_mux_0_avalon_slave_0;
  input              cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0;
  input              cpu_0_data_master_write;
  input   [ 31: 0] cpu_0_data_master_writedata;
  input   [ 31: 0] cpu_0_jtag_debug_module_readdata_from_sa;
  input              d1_audio_controller_0_avalon_slave_0_end_xfer;
  input              d1_command_0_avalon_slave_0_end_xfer;
  input              d1_cpu_0_jtag_debug_module_end_xfer;
  input              d1_interrupt_0_avalon_slave_0_end_xfer;
  input              d1_jtag_uart_0_avalon_jtag_slave_end_xfer;
  input              d1_musiccontroller_0_avalon_slave_0_end_xfer;
  input              d1_sdram_0_s1_end_xfer;
  input              d1_sram_mux_0_avalon_slave_0_end_xfer;
  input              d1_xycoord_interface_0_avalon_slave_0_end_xfer;
  input              interrupt_0_avalon_slave_0_irq_from_sa;
  input   [ 31: 0] interrupt_0_avalon_slave_0_readdata_from_sa;
  input              jtag_uart_0_avalon_jtag_slave_irq_from_sa;
  input   [ 31: 0] jtag_uart_0_avalon_jtag_slave_readdata_from_sa;
  input              jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa;
  input              musiccontroller_0_avalon_slave_0_irq_from_sa;
  input   [ 15: 0] musiccontroller_0_avalon_slave_0_readdata_from_sa;
  input              reset_n;
  input   [ 15: 0] sdram_0_s1_readdata_from_sa;
  input              sdram_0_s1_waitrequest_from_sa;
  input   [ 15: 0] sram_mux_0_avalon_slave_0_readdata_from_sa;
  input   [ 15: 0] xycoord_interface_0_avalon_slave_0_readdata_from_sa;
```

```
  wire  [ 24: 0] cpu_0_data_master_address_to_slave;
  reg   [  1: 0] cpu_0_data_master_dbs_address;
  wire  [  1: 0] cpu_0_data_master_dbs_increment;
  wire  [ 15: 0] cpu_0_data_master_dbs_write_16;
  wire  [ 31: 0] cpu_0_data_master_irq;
  reg            cpu_0_data_master_no_byte_enables_and_last_term;
  wire  [ 31: 0] cpu_0_data_master_readdata;
  wire           cpu_0_data_master_run;
  reg            cpu_0_data_master_waitrequest;
  reg   [ 15: 0] dbs_16_reg_segment_0;
  wire           dbs_count_enable;
  wire           dbs_counter_overflow;
  wire           last_dbs_term_and_run;
  wire  [  1: 0] next_dbs_address;
  wire  [ 15: 0] p1_dbs_16_reg_segment_0;
  wire  [ 31: 0] p1_registered_cpu_0_data_master_readdata;
  wire           pre_dbs_count_enable;
  wire           r_0;
  wire           r_1;
  wire           r_2;
  reg   [ 31: 0] registered_cpu_0_data_master_readdata;
  //r_0 master_run cascaded wait assignment, which is an e_assign
  assign r_0 = 1 & (cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0 |
(cpu_0_data_master_write
& !cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0 &
cpu_0_data_master_dbs_address[1]) |
~cpu_0_data_master_requests_audio_controller_0_avalon_slave_0) &
(cpu_0_data_master_granted_audio_controller_0_avalon_slave_0 |
~cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0) &
((~cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0 |
~cpu_0_data_master_read | (1 & 1 & (cpu_0_data_master_dbs_address[1]) &
cpu_0_data_master_read))) &
((~cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0 |
~cpu_0_data_master_write | (1 & (cpu_0_data_master_dbs_address[1]) &
cpu_0_data_master_write))) & 1 &
(cpu_0_data_master_qualified_request_command_0_avalon_slave_0 |
~cpu_0_data_master_requests_command_0_avalon_slave_0) &
(cpu_0_data_master_granted_command_0_avalon_slave_0 |
~cpu_0_data_master_qualified_request_command_0_avalon_slave_0) &
((~cpu_0_data_master_qualified_request_command_0_avalon_slave_0 |
~cpu_0_data_master_read | (1 & 1 & cpu_0_data_master_read))) &
((~cpu_0_data_master_qualified_request_command_0_avalon_slave_0 |
~cpu_0_data_master_write | (1 & cpu_0_data_master_write))) & 1 &
(cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module |
~cpu_0_data_master_requests_cpu_0_jtag_debug_module) &
(cpu_0_data_master_granted_cpu_0_jtag_debug_module |
~cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module) &
((~cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module |
~cpu_0_data_master_read | (1 & 1 & cpu_0_data_master_read))) &
((~cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module |
~cpu_0_data_master_write | (1 & cpu_0_data_master_write))) & 1 &
(cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0 |
~cpu_0_data_master_requests_interrupt_0_avalon_slave_0) &
((~cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0 |
~cpu_0_data_master_read | (1 & 1 & cpu_0_data_master_read))) &
((~cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0 |
~cpu_0_data_master_write | (1 & cpu_0_data_master_write))) & 1;
```

```
  //cascaded wait assignment, which is an e_assign
  assign cpu_0_data_master_run = r_0 & r_1 & r_2;

  //r_1 master_run cascaded wait assignment, which is an e_assign
  assign r_1 = (cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave |
~cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave) &
((~cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave |
~(cpu_0_data_master_read | cpu_0_data_master_write) | (1 &
~jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa & (cpu_0_data_master_read |
cpu_0_data_master_write)))) &
((~cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave |
~(cpu_0_data_master_read | cpu_0_data_master_write) | (1 &
~jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa & (cpu_0_data_master_read |
cpu_0_data_master_write)))) & 1 &
(cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0 |
(cpu_0_data_master_write
& !cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0 &
cpu_0_data_master_dbs_address[1]) |
~cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0) &
((~cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0 |
~cpu_0_data_master_read | (1 & 1 & (cpu_0_data_master_dbs_address[1]) &
cpu_0_data_master_read))) &
((~cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0 |
~cpu_0_data_master_write | (1 & (cpu_0_data_master_dbs_address[1]) &
cpu_0_data_master_write))) & 1 & (cpu_0_data_master_qualified_request_sdram_0_s1 |
(cpu_0_data_master_read_data_valid_sdram_0_s1 & cpu_0_data_master_dbs_address[1]) |
(cpu_0_data_master_write & !cpu_0_data_master_byteenable_sdram_0_s1 &
cpu_0_data_master_dbs_address[1]) | ~cpu_0_data_master_requests_sdram_0_s1) &
(cpu_0_data_master_granted_sdram_0_s1 |
~cpu_0_data_master_qualified_request_sdram_0_s1) &
((~cpu_0_data_master_qualified_request_sdram_0_s1 | ~cpu_0_data_master_read |
(cpu_0_data_master_read_data_valid_sdram_0_s1 & (cpu_0_data_master_dbs_address[1]) &
cpu_0_data_master_read))) & ((~cpu_0_data_master_qualified_request_sdram_0_s1 |
~cpu_0_data_master_write | (1 & ~sdram_0_s1_waitrequest_from_sa &
(cpu_0_data_master_dbs_address[1]) & cpu_0_data_master_write))) & 1 &
(cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0 |
(cpu_0_data_master_write & !cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0 &
cpu_0_data_master_dbs_address[1]) |
~cpu_0_data_master_requests_sram_mux_0_avalon_slave_0) &
(cpu_0_data_master_granted_sram_mux_0_avalon_slave_0 |
~cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0) &
((~cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0 |
~cpu_0_data_master_read | (1 & 1 & (cpu_0_data_master_dbs_address[1]) &
cpu_0_data_master_read))) &
((~cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0 |
~cpu_0_data_master_write | (1 & (cpu_0_data_master_dbs_address[1]) &
cpu_0_data_master_write))) & 1 &
(cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0 |
(cpu_0_data_master_write
& !cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0 &
cpu_0_data_master_dbs_address[1]) |
~cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0) &
(cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0 |
~cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0);

  //r_2 master_run cascaded wait assignment, which is an e_assign
  assign r_2 = ((~cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0 |
~cpu_0_data_master_read | (1 & 1 & (cpu_0_data_master_dbs_address[1]) &
```

```
cpu_0_data_master_read))) &
((~cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0 |
~cpu_0_data_master_write | (1 & (cpu_0_data_master_dbs_address[1]) &
cpu_0_data_master_write)));

  //optimize select-logic by passing only those address bits which matter.
  assign cpu_0_data_master_address_to_slave = cpu_0_data_master_address[24 : 0];

  //no_byte_enables_and_last_term, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          cpu_0_data_master_no_byte_enables_and_last_term <= 0;
      else
        cpu_0_data_master_no_byte_enables_and_last_term <= last_dbs_term_and_run;
    end


  //compute the last dbs term, which is an e_mux
  assign last_dbs_term_and_run =
(cpu_0_data_master_requests_audio_controller_0_avalon_slave_0)?
(((cpu_0_data_master_dbs_address == 2'b10) & cpu_0_data_master_write
& !cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0)) :
    (cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0)?
(((cpu_0_data_master_dbs_address == 2'b10) & cpu_0_data_master_write
& !cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0)) :
    (cpu_0_data_master_requests_sdram_0_s1)? (((cpu_0_data_master_dbs_address == 2'b10)
& cpu_0_data_master_write & !cpu_0_data_master_byteenable_sdram_0_s1)) :
    (cpu_0_data_master_requests_sram_mux_0_avalon_slave_0)?
(((cpu_0_data_master_dbs_address == 2'b10) & cpu_0_data_master_write
& !cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0)) :
    (((cpu_0_data_master_dbs_address == 2'b10) & cpu_0_data_master_write
& !cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0));

  //pre dbs count enable, which is an e_mux
  assign pre_dbs_count_enable = (((~cpu_0_data_master_no_byte_enables_and_last_term) &
cpu_0_data_master_requests_audio_controller_0_avalon_slave_0 & cpu_0_data_master_write
& !cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0)) |
    (cpu_0_data_master_granted_audio_controller_0_avalon_slave_0 &
cpu_0_data_master_read & 1 & 1 & ~d1_audio_controller_0_avalon_slave_0_end_xfer) |
    (cpu_0_data_master_granted_audio_controller_0_avalon_slave_0 &
cpu_0_data_master_write & 1 & 1) |
    (((~cpu_0_data_master_no_byte_enables_and_last_term) &
cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0 & cpu_0_data_master_write
& !cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0)) |
    (cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0 &
cpu_0_data_master_read & 1 & 1 & ~d1_musiccontroller_0_avalon_slave_0_end_xfer) |
    (cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0 &
cpu_0_data_master_write & 1 & 1) |
    (((~cpu_0_data_master_no_byte_enables_and_last_term) &
cpu_0_data_master_requests_sdram_0_s1 & cpu_0_data_master_write
& !cpu_0_data_master_byteenable_sdram_0_s1)) |
    cpu_0_data_master_read_data_valid_sdram_0_s1 |
    (cpu_0_data_master_granted_sdram_0_s1 & cpu_0_data_master_write & 1 & 1 &
~sdram_0_s1_waitrequest_from_sa) |
    (((~cpu_0_data_master_no_byte_enables_and_last_term) &
cpu_0_data_master_requests_sram_mux_0_avalon_slave_0 & cpu_0_data_master_write
& !cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0)) |
```

```
      (cpu_0_data_master_granted_sram_mux_0_avalon_slave_0 & cpu_0_data_master_read &
1 & 1 & ~d1_sram_mux_0_avalon_slave_0_end_xfer) |
      (cpu_0_data_master_granted_sram_mux_0_avalon_slave_0 & cpu_0_data_master_write &
1 & 1) |
      (((~cpu_0_data_master_no_byte_enables_and_last_term)                    &
cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0                  &
cpu_0_data_master_write
& !cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0)) |
      (cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0              &
cpu_0_data_master_read & 1 & 1 & ~d1_xycoord_interface_0_avalon_slave_0_end_xfer) |
      (cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0              &
cpu_0_data_master_write & 1 & 1);

   //input to dbs-16 stored 0, which is an e_mux
   assign                                  p1_dbs_16_reg_segment_0                =
(cpu_0_data_master_requests_audio_controller_0_avalon_slave_0)?
audio_controller_0_avalon_slave_0_readdata_from_sa :
      (cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0)?
musiccontroller_0_avalon_slave_0_readdata_from_sa :
      (cpu_0_data_master_requests_sdram_0_s1)? sdram_0_s1_readdata_from_sa :
      (cpu_0_data_master_requests_sram_mux_0_avalon_slave_0)?
sram_mux_0_avalon_slave_0_readdata_from_sa :
      xycoord_interface_0_avalon_slave_0_readdata_from_sa;

   //dbs register for dbs-16 segment 0, which is an e_register
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               dbs_16_reg_segment_0 <= 0;
         else if (dbs_count_enable & ((cpu_0_data_master_dbs_address[1]) == 0))
               dbs_16_reg_segment_0 <= p1_dbs_16_reg_segment_0;
      end


   //cpu_0/data_master readdata mux, which is an e_mux
   assign                      cpu_0_data_master_readdata                 =                ({32
{~cpu_0_data_master_requests_audio_controller_0_avalon_slave_0}}                          |
{audio_controller_0_avalon_slave_0_readdata_from_sa[15 : 0],
      dbs_16_reg_segment_0}) &
      ({32              {~cpu_0_data_master_requests_command_0_avalon_slave_0}}          |
command_0_avalon_slave_0_readdata_from_sa) &
      ({32               {~cpu_0_data_master_requests_cpu_0_jtag_debug_module}}          |
cpu_0_jtag_debug_module_readdata_from_sa) &
      ({32               {~cpu_0_data_master_requests_interrupt_0_avalon_slave_0}}       |
interrupt_0_avalon_slave_0_readdata_from_sa) &
      ({32             {~cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave}}      |
registered_cpu_0_data_master_readdata) &
      ({32           {~cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0}}     |
{musiccontroller_0_avalon_slave_0_readdata_from_sa[15 : 0],
      dbs_16_reg_segment_0}) &
      ({32                    {~cpu_0_data_master_requests_sdram_0_s1}}                  |
registered_cpu_0_data_master_readdata) &
      ({32               {~cpu_0_data_master_requests_sram_mux_0_avalon_slave_0}}       |
{sram_mux_0_avalon_slave_0_readdata_from_sa[15 : 0],
      dbs_16_reg_segment_0}) &
      ({32          {~cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0}}   |
{xycoord_interface_0_avalon_slave_0_readdata_from_sa[15 : 0],
      dbs_16_reg_segment_0});
```

```
  //mux write dbs 1, which is an e_mux
  assign cpu_0_data_master_dbs_write_16 = (cpu_0_data_master_dbs_address[1])?
cpu_0_data_master_writedata[31 : 16] :
    (~(cpu_0_data_master_dbs_address[1]))? cpu_0_data_master_writedata[15 : 0] :
    (cpu_0_data_master_dbs_address[1])? cpu_0_data_master_writedata[31 : 16] :
    (~(cpu_0_data_master_dbs_address[1]))? cpu_0_data_master_writedata[15 : 0] :
    (cpu_0_data_master_dbs_address[1])? cpu_0_data_master_writedata[31 : 16] :
    (~(cpu_0_data_master_dbs_address[1]))? cpu_0_data_master_writedata[15 : 0] :
    (cpu_0_data_master_dbs_address[1])? cpu_0_data_master_writedata[31 : 16] :
    (~(cpu_0_data_master_dbs_address[1]))? cpu_0_data_master_writedata[15 : 0] :
    (cpu_0_data_master_dbs_address[1])? cpu_0_data_master_writedata[31 : 16] :
    cpu_0_data_master_writedata[15 : 0];

  //actual waitrequest port, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          cpu_0_data_master_waitrequest <= ~0;
      else
        cpu_0_data_master_waitrequest <= ~((~(cpu_0_data_master_read |
cpu_0_data_master_write))? 0: (cpu_0_data_master_run & cpu_0_data_master_waitrequest));
    end


  //dbs count increment, which is an e_mux
  assign cpu_0_data_master_dbs_increment =
(cpu_0_data_master_requests_audio_controller_0_avalon_slave_0)? 2 :
    (cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0)? 2 :
    (cpu_0_data_master_requests_sdram_0_s1)? 2 :
    (cpu_0_data_master_requests_sram_mux_0_avalon_slave_0)? 2 :
    (cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0)? 2 :
    0;

  //dbs counter overflow, which is an e_assign
  assign dbs_counter_overflow = cpu_0_data_master_dbs_address[1]
& !(next_dbs_address[1]);

  //next master address, which is an e_assign
  assign next_dbs_address = cpu_0_data_master_dbs_address +
cpu_0_data_master_dbs_increment;

  //dbs count enable, which is an e_mux
  assign dbs_count_enable = pre_dbs_count_enable &
    (~(cpu_0_data_master_requests_audio_controller_0_avalon_slave_0 &
~cpu_0_data_master_waitrequest & cpu_0_data_master_write)) &
    (~(cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0 &
~cpu_0_data_master_waitrequest & cpu_0_data_master_write)) &
    (~(cpu_0_data_master_requests_sdram_0_s1 & ~cpu_0_data_master_waitrequest)) &
    (~(cpu_0_data_master_requests_sram_mux_0_avalon_slave_0 &
~cpu_0_data_master_waitrequest & cpu_0_data_master_write)) &
    (~(cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0 &
~cpu_0_data_master_waitrequest & cpu_0_data_master_write));

  //dbs counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
```

```
            cpu_0_data_master_dbs_address <= 0;
        else if (dbs_count_enable)
            cpu_0_data_master_dbs_address <= next_dbs_address;
    end


  //irq assign, which is an e_assign
  assign cpu_0_data_master_irq = {1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    1'b0,
    interrupt_0_avalon_slave_0_irq_from_sa,
    musiccontroller_0_avalon_slave_0_irq_from_sa,
    jtag_uart_0_avalon_jtag_slave_irq_from_sa};

  //unpredictable registered wait state incoming data, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          registered_cpu_0_data_master_readdata <= 0;
      else
        registered_cpu_0_data_master_readdata                                    <=
p1_registered_cpu_0_data_master_readdata;
    end


  //registered readdata mux, which is an e_mux
  assign                p1_registered_cpu_0_data_master_readdata              =              ({32
{~cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave}}                                  |
jtag_uart_0_avalon_jtag_slave_readdata_from_sa) &
    ({32 {~cpu_0_data_master_requests_sdram_0_s1}} | {sdram_0_s1_readdata_from_sa[15 :
0],
    dbs_16_reg_segment_0});
```

```
endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_instruction_master_arbitrator (
                                             // inputs:

audio_controller_0_avalon_slave_0_readdata_from_sa,
                                              clk,

command_0_avalon_slave_0_readdata_from_sa,
                                              cpu_0_instruction_master_address,

cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0,

cpu_0_instruction_master_granted_command_0_avalon_slave_0,

cpu_0_instruction_master_granted_cpu_0_jtag_debug_module,

cpu_0_instruction_master_granted_sdram_0_s1,

cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0,

cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0,

cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0,

cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0,

cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module,

cpu_0_instruction_master_qualified_request_sdram_0_s1,

cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0,

cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0,
                                              cpu_0_instruction_master_read,

cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0,

cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0,

cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module,

cpu_0_instruction_master_read_data_valid_sdram_0_s1,

cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register,

cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0,
```

```
cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0,

cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0,

cpu_0_instruction_master_requests_command_0_avalon_slave_0,

cpu_0_instruction_master_requests_cpu_0_jtag_debug_module,

cpu_0_instruction_master_requests_sdram_0_s1,

cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0,

cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0,

cpu_0_jtag_debug_module_readdata_from_sa,

d1_audio_controller_0_avalon_slave_0_end_xfer,

d1_command_0_avalon_slave_0_end_xfer,

d1_cpu_0_jtag_debug_module_end_xfer,
                                                    d1_sdram_0_s1_end_xfer,

d1_sram_mux_0_avalon_slave_0_end_xfer,

d1_xycoord_interface_0_avalon_slave_0_end_xfer,
                                                    reset_n,
                                                    sdram_0_s1_readdata_from_sa,
                                                    sdram_0_s1_waitrequest_from_sa,

sram_mux_0_avalon_slave_0_readdata_from_sa,

xycoord_interface_0_avalon_slave_0_readdata_from_sa,

                                                  // outputs:

cpu_0_instruction_master_address_to_slave,

cpu_0_instruction_master_dbs_address,
                                                    cpu_0_instruction_master_readdata,
                                                    cpu_0_instruction_master_waitrequest
                                                )
;

  output  [ 24: 0] cpu_0_instruction_master_address_to_slave;
  output  [  1: 0] cpu_0_instruction_master_dbs_address;
  output  [ 31: 0] cpu_0_instruction_master_readdata;
  output           cpu_0_instruction_master_waitrequest;
  input   [ 15: 0] audio_controller_0_avalon_slave_0_readdata_from_sa;
  input            clk;
  input   [ 31: 0] command_0_avalon_slave_0_readdata_from_sa;
  input   [ 24: 0] cpu_0_instruction_master_address;
  input            cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0;
  input            cpu_0_instruction_master_granted_command_0_avalon_slave_0;
  input            cpu_0_instruction_master_granted_cpu_0_jtag_debug_module;
  input            cpu_0_instruction_master_granted_sdram_0_s1;
  input            cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0;
```

```
  input              cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0;
  input
cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0;
  input
cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0;
  input              cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module;
  input              cpu_0_instruction_master_qualified_request_sdram_0_s1;
  input
cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0;
  input
cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0;
  input              cpu_0_instruction_master_read;
  input
cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0;
  input              cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0;
  input              cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module;
  input              cpu_0_instruction_master_read_data_valid_sdram_0_s1;
  input              cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register;
  input
cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0;
  input
cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0;
  input              cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0;
  input              cpu_0_instruction_master_requests_command_0_avalon_slave_0;
  input              cpu_0_instruction_master_requests_cpu_0_jtag_debug_module;
  input              cpu_0_instruction_master_requests_sdram_0_s1;
  input              cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0;
  input
cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0;
  input    [ 31: 0] cpu_0_jtag_debug_module_readdata_from_sa;
  input              d1_audio_controller_0_avalon_slave_0_end_xfer;
  input              d1_command_0_avalon_slave_0_end_xfer;
  input              d1_cpu_0_jtag_debug_module_end_xfer;
  input              d1_sdram_0_s1_end_xfer;
  input              d1_sram_mux_0_avalon_slave_0_end_xfer;
  input              d1_xycoord_interface_0_avalon_slave_0_end_xfer;
  input              reset_n;
  input    [ 15: 0] sdram_0_s1_readdata_from_sa;
  input              sdram_0_s1_waitrequest_from_sa;
  input    [ 15: 0] sram_mux_0_avalon_slave_0_readdata_from_sa;
  input    [ 15: 0] xycoord_interface_0_avalon_slave_0_readdata_from_sa;

  reg                active_and_waiting_last_time;
  reg      [ 24: 0] cpu_0_instruction_master_address_last_time;
  wire     [ 24: 0] cpu_0_instruction_master_address_to_slave;
  reg      [  1: 0] cpu_0_instruction_master_dbs_address;
  wire     [  1: 0] cpu_0_instruction_master_dbs_increment;
  reg                cpu_0_instruction_master_read_last_time;
  wire     [ 31: 0] cpu_0_instruction_master_readdata;
  wire               cpu_0_instruction_master_run;
  wire               cpu_0_instruction_master_waitrequest;
  reg      [ 15: 0] dbs_16_reg_segment_0;
  wire               dbs_count_enable;
  wire               dbs_counter_overflow;
  wire     [  1: 0] next_dbs_address;
  wire     [ 15: 0] p1_dbs_16_reg_segment_0;
  wire               pre_dbs_count_enable;
  wire               r_0;
```

```
  wire        r_1;
  wire        r_2;
  //r_0 master_run cascaded wait assignment, which is an e_assign
  assign r_0 = 1 &
(cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0 |
~cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0) &
(cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0 |
~cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0) &
((~cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0 |
~cpu_0_instruction_master_read | (1 & ~d1_audio_controller_0_avalon_slave_0_end_xfer &
(cpu_0_instruction_master_dbs_address[1]) & cpu_0_instruction_master_read))) & 1 &
(cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0 |
~cpu_0_instruction_master_requests_command_0_avalon_slave_0) &
(cpu_0_instruction_master_granted_command_0_avalon_slave_0 |
~cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0) &
((~cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0 |
~cpu_0_instruction_master_read | (1 & ~d1_command_0_avalon_slave_0_end_xfer &
cpu_0_instruction_master_read))) & 1 &
(cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module |
~cpu_0_instruction_master_requests_cpu_0_jtag_debug_module) &
(cpu_0_instruction_master_granted_cpu_0_jtag_debug_module |
~cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module) &
((~cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module |
~cpu_0_instruction_master_read | (1 & ~d1_cpu_0_jtag_debug_module_end_xfer &
cpu_0_instruction_master_read)));

  //cascaded wait assignment, which is an e_assign
  assign cpu_0_instruction_master_run = r_0 & r_1 & r_2;

  //r_1 master_run cascaded wait assignment, which is an e_assign
  assign r_1 = 1 & (cpu_0_instruction_master_qualified_request_sdram_0_s1 |
(cpu_0_instruction_master_read_data_valid_sdram_0_s1 &
cpu_0_instruction_master_dbs_address[1]) | ~cpu_0_instruction_master_requests_sdram_0_s1)
& (cpu_0_instruction_master_granted_sdram_0_s1 |
~cpu_0_instruction_master_qualified_request_sdram_0_s1) &
((~cpu_0_instruction_master_qualified_request_sdram_0_s1 | ~cpu_0_instruction_master_read
| (cpu_0_instruction_master_read_data_valid_sdram_0_s1 &
(cpu_0_instruction_master_dbs_address[1]) & cpu_0_instruction_master_read))) & 1 &
(cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0 |
~cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0) &
(cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0 |
~cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0) &
((~cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0 |
~cpu_0_instruction_master_read | (1 & ~d1_sram_mux_0_avalon_slave_0_end_xfer &
(cpu_0_instruction_master_dbs_address[1]) & cpu_0_instruction_master_read))) & 1 &
(cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0 |
~cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0);

  //r_2 master_run cascaded wait assignment, which is an e_assign
  assign r_2 = (cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0 |
~cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0) &
((~cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0 |
~cpu_0_instruction_master_read | (1 & ~d1_xycoord_interface_0_avalon_slave_0_end_xfer &
(cpu_0_instruction_master_dbs_address[1]) & cpu_0_instruction_master_read)));

  //optimize select-logic by passing only those address bits which matter.
  assign cpu_0_instruction_master_address_to_slave = cpu_0_instruction_master_address[24 :
0];
```

```
  //input to dbs-16 stored 0, which is an e_mux
  assign p1_dbs_16_reg_segment_0 =
(cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0)?
audio_controller_0_avalon_slave_0_readdata_from_sa :
    (cpu_0_instruction_master_requests_sdram_0_s1)? sdram_0_s1_readdata_from_sa :
    (cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0)?
sram_mux_0_avalon_slave_0_readdata_from_sa :
    xycoord_interface_0_avalon_slave_0_readdata_from_sa;

  //dbs register for dbs-16 segment 0, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          dbs_16_reg_segment_0 <= 0;
      else if (dbs_count_enable & ((cpu_0_instruction_master_dbs_address[1]) == 0))
          dbs_16_reg_segment_0 <= p1_dbs_16_reg_segment_0;
    end


  //cpu_0/instruction_master readdata mux, which is an e_mux
  assign cpu_0_instruction_master_readdata = ({32
{~cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0}} |
{audio_controller_0_avalon_slave_0_readdata_from_sa[15 : 0],
    dbs_16_reg_segment_0}) &
    ({32 {~cpu_0_instruction_master_requests_command_0_avalon_slave_0}} |
command_0_avalon_slave_0_readdata_from_sa) &
    ({32 {~cpu_0_instruction_master_requests_cpu_0_jtag_debug_module}} |
cpu_0_jtag_debug_module_readdata_from_sa) &
    ({32 {~cpu_0_instruction_master_requests_sdram_0_s1}} |
{sdram_0_s1_readdata_from_sa[15 : 0],
    dbs_16_reg_segment_0}) &
    ({32 {~cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0}} |
{sram_mux_0_avalon_slave_0_readdata_from_sa[15 : 0],
    dbs_16_reg_segment_0}) &
    ({32 {~cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0}} |
{xycoord_interface_0_avalon_slave_0_readdata_from_sa[15 : 0],
    dbs_16_reg_segment_0});

  //actual waitrequest port, which is an e_assign
  assign cpu_0_instruction_master_waitrequest = ~cpu_0_instruction_master_run;

  //dbs count increment, which is an e_mux
  assign cpu_0_instruction_master_dbs_increment =
(cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0)? 2 :
    (cpu_0_instruction_master_requests_sdram_0_s1)? 2 :
    (cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0)? 2 :
    (cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0)? 2 :
    0;

  //dbs counter overflow, which is an e_assign
  assign dbs_counter_overflow = cpu_0_instruction_master_dbs_address[1]
& !(next_dbs_address[1]);

  //next master address, which is an e_assign
  assign next_dbs_address = cpu_0_instruction_master_dbs_address +
cpu_0_instruction_master_dbs_increment;
```

```
  //dbs count enable, which is an e_mux
  assign dbs_count_enable = pre_dbs_count_enable;

  //dbs counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          cpu_0_instruction_master_dbs_address <= 0;
      else if (dbs_count_enable)
          cpu_0_instruction_master_dbs_address <= next_dbs_address;
    end


  //pre dbs count enable, which is an e_mux
  assign pre_dbs_count_enable =
(cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0 &
cpu_0_instruction_master_read & 1 & 1 & ~d1_audio_controller_0_avalon_slave_0_end_xfer) |
    cpu_0_instruction_master_read_data_valid_sdram_0_s1 |
    (cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0 &
cpu_0_instruction_master_read & 1 & 1 & ~d1_sram_mux_0_avalon_slave_0_end_xfer) |
    (cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0 &
cpu_0_instruction_master_read & 1 & 1 & ~d1_xycoord_interface_0_avalon_slave_0_end_xfer);


//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
  //cpu_0_instruction_master_address check against wait, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          cpu_0_instruction_master_address_last_time <= 0;
      else
        cpu_0_instruction_master_address_last_time <= cpu_0_instruction_master_address;
    end


  //cpu_0/instruction_master waited last time, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          active_and_waiting_last_time <= 0;
      else
        active_and_waiting_last_time <= cpu_0_instruction_master_waitrequest &
(cpu_0_instruction_master_read);
    end


  //cpu_0_instruction_master_address matches last port_name, which is an e_process
  always @(posedge clk)
    begin
      if (active_and_waiting_last_time & (cpu_0_instruction_master_address !=
cpu_0_instruction_master_address_last_time))
        begin
          $write("%0d ns: cpu_0_instruction_master_address did not heed wait!!!", $time);
          $stop;
        end
    end
```

```
  //cpu_0_instruction_master_read check against wait, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          cpu_0_instruction_master_read_last_time <= 0;
      else
        cpu_0_instruction_master_read_last_time <= cpu_0_instruction_master_read;
    end


  //cpu_0_instruction_master_read matches last port_name, which is an e_process
  always @(posedge clk)
    begin
      if (active_and_waiting_last_time & (cpu_0_instruction_master_read !=
cpu_0_instruction_master_read_last_time))
        begin
          $write("%0d ns: cpu_0_instruction_master_read did not heed wait!!!", $time);
          $stop;
        end
    end



//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module interrupt_0_avalon_slave_0_arbitrator (
                                               // inputs:
                                                 clk,

cpu_0_data_master_address_to_slave,
                                                 cpu_0_data_master_read,
                                                 cpu_0_data_master_waitrequest,
                                                 cpu_0_data_master_write,
                                                 cpu_0_data_master_writedata,
                                                 interrupt_0_avalon_slave_0_irq,

interrupt_0_avalon_slave_0_readdata,
                                                 reset_n,

                                               // outputs:

cpu_0_data_master_granted_interrupt_0_avalon_slave_0,

cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0,
```

```
cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0,

cpu_0_data_master_requests_interrupt_0_avalon_slave_0,

d1_interrupt_0_avalon_slave_0_end_xfer,
                                                        interrupt_0_avalon_slave_0_address,

interrupt_0_avalon_slave_0_chipselect,

interrupt_0_avalon_slave_0_irq_from_sa,
                                                        interrupt_0_avalon_slave_0_read,

interrupt_0_avalon_slave_0_readdata_from_sa,
                                                        interrupt_0_avalon_slave_0_reset_n,
                                                        interrupt_0_avalon_slave_0_write,

interrupt_0_avalon_slave_0_writedata
                                                    )
;

  output           cpu_0_data_master_granted_interrupt_0_avalon_slave_0;
  output           cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0;
  output           cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0;
  output           cpu_0_data_master_requests_interrupt_0_avalon_slave_0;
  output           d1_interrupt_0_avalon_slave_0_end_xfer;
  output   [  4: 0] interrupt_0_avalon_slave_0_address;
  output           interrupt_0_avalon_slave_0_chipselect;
  output           interrupt_0_avalon_slave_0_irq_from_sa;
  output           interrupt_0_avalon_slave_0_read;
  output   [ 31: 0] interrupt_0_avalon_slave_0_readdata_from_sa;
  output           interrupt_0_avalon_slave_0_reset_n;
  output           interrupt_0_avalon_slave_0_write;
  output   [ 31: 0] interrupt_0_avalon_slave_0_writedata;
  input            clk;
  input    [ 24: 0] cpu_0_data_master_address_to_slave;
  input            cpu_0_data_master_read;
  input            cpu_0_data_master_waitrequest;
  input            cpu_0_data_master_write;
  input    [ 31: 0] cpu_0_data_master_writedata;
  input            interrupt_0_avalon_slave_0_irq;
  input    [ 31: 0] interrupt_0_avalon_slave_0_readdata;
  input            reset_n;

  wire             cpu_0_data_master_arbiterlock;
  wire             cpu_0_data_master_arbiterlock2;
  wire             cpu_0_data_master_continuerequest;
  wire             cpu_0_data_master_granted_interrupt_0_avalon_slave_0;
  wire             cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0;
  wire             cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0;
  wire             cpu_0_data_master_requests_interrupt_0_avalon_slave_0;
  wire             cpu_0_data_master_saved_grant_interrupt_0_avalon_slave_0;
  reg              d1_interrupt_0_avalon_slave_0_end_xfer;
  reg              d1_reasons_to_wait;
  reg              enable_nonzero_assertions;
  wire             end_xfer_arb_share_counter_term_interrupt_0_avalon_slave_0;
  wire             in_a_read_cycle;
  wire             in_a_write_cycle;
```

```
  wire       [  4: 0] interrupt_0_avalon_slave_0_address;
  wire                interrupt_0_avalon_slave_0_allgrants;
  wire                interrupt_0_avalon_slave_0_allow_new_arb_cycle;
  wire                interrupt_0_avalon_slave_0_any_bursting_master_saved_grant;
  wire                interrupt_0_avalon_slave_0_any_continuerequest;
  wire                interrupt_0_avalon_slave_0_arb_counter_enable;
  reg        [  1: 0] interrupt_0_avalon_slave_0_arb_share_counter;
  wire       [  1: 0] interrupt_0_avalon_slave_0_arb_share_counter_next_value;
  wire       [  1: 0] interrupt_0_avalon_slave_0_arb_share_set_values;
  wire                interrupt_0_avalon_slave_0_beginbursttransfer_internal;
  wire                interrupt_0_avalon_slave_0_begins_xfer;
  wire                interrupt_0_avalon_slave_0_chipselect;
  wire                interrupt_0_avalon_slave_0_end_xfer;
  wire                interrupt_0_avalon_slave_0_firsttransfer;
  wire                interrupt_0_avalon_slave_0_grant_vector;
  wire                interrupt_0_avalon_slave_0_in_a_read_cycle;
  wire                interrupt_0_avalon_slave_0_in_a_write_cycle;
  wire                interrupt_0_avalon_slave_0_irq_from_sa;
  wire                interrupt_0_avalon_slave_0_master_qreq_vector;
  wire                interrupt_0_avalon_slave_0_non_bursting_master_requests;
  wire                interrupt_0_avalon_slave_0_read;
  wire       [ 31: 0] interrupt_0_avalon_slave_0_readdata_from_sa;
  reg                 interrupt_0_avalon_slave_0_reg_firsttransfer;
  wire                interrupt_0_avalon_slave_0_reset_n;
  reg                 interrupt_0_avalon_slave_0_slavearbiterlockenable;
  wire                interrupt_0_avalon_slave_0_slavearbiterlockenable2;
  wire                interrupt_0_avalon_slave_0_unreg_firsttransfer;
  wire                interrupt_0_avalon_slave_0_waits_for_read;
  wire                interrupt_0_avalon_slave_0_waits_for_write;
  wire                interrupt_0_avalon_slave_0_write;
  wire       [ 31: 0] interrupt_0_avalon_slave_0_writedata;
  wire       [ 24: 0] shifted_address_to_interrupt_0_avalon_slave_0_from_cpu_0_data_master;
  wire                wait_for_interrupt_0_avalon_slave_0_counter;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_reasons_to_wait <= 0;
      else
        d1_reasons_to_wait <= ~interrupt_0_avalon_slave_0_end_xfer;
    end


  assign interrupt_0_avalon_slave_0_begins_xfer = ~d1_reasons_to_wait &
((cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0));
  //assign interrupt_0_avalon_slave_0_readdata_from_sa =
interrupt_0_avalon_slave_0_readdata so that symbol knows where to group signals which may
go to master only, which is an e_assign
  assign interrupt_0_avalon_slave_0_readdata_from_sa =
interrupt_0_avalon_slave_0_readdata;

  assign cpu_0_data_master_requests_interrupt_0_avalon_slave_0 =
({cpu_0_data_master_address_to_slave[24 : 7] , 7'b0} == 25'h0) & (cpu_0_data_master_read |
cpu_0_data_master_write);
  //interrupt_0_avalon_slave_0_arb_share_counter set values, which is an e_mux
  assign interrupt_0_avalon_slave_0_arb_share_set_values = 1;

  //interrupt_0_avalon_slave_0_non_bursting_master_requests mux, which is an e_mux
  assign interrupt_0_avalon_slave_0_non_bursting_master_requests =
```

```
cpu_0_data_master_requests_interrupt_0_avalon_slave_0;

  //interrupt_0_avalon_slave_0_any_bursting_master_saved_grant mux, which is an e_mux
  assign interrupt_0_avalon_slave_0_any_bursting_master_saved_grant = 0;

  //interrupt_0_avalon_slave_0_arb_share_counter_next_value assignment, which is an
e_assign
  assign interrupt_0_avalon_slave_0_arb_share_counter_next_value =
interrupt_0_avalon_slave_0_firsttransfer ? (interrupt_0_avalon_slave_0_arb_share_set_values -
1) : |interrupt_0_avalon_slave_0_arb_share_counter ?
(interrupt_0_avalon_slave_0_arb_share_counter - 1) : 0;

  //interrupt_0_avalon_slave_0_allgrants all slave grants, which is an e_mux
  assign interrupt_0_avalon_slave_0_allgrants = |interrupt_0_avalon_slave_0_grant_vector;

  //interrupt_0_avalon_slave_0_end_xfer assignment, which is an e_assign
  assign interrupt_0_avalon_slave_0_end_xfer = ~(interrupt_0_avalon_slave_0_waits_for_read
| interrupt_0_avalon_slave_0_waits_for_write);

  //end_xfer_arb_share_counter_term_interrupt_0_avalon_slave_0 arb share counter enable
term, which is an e_assign
  assign end_xfer_arb_share_counter_term_interrupt_0_avalon_slave_0 =
interrupt_0_avalon_slave_0_end_xfer &
(~interrupt_0_avalon_slave_0_any_bursting_master_saved_grant | in_a_read_cycle |
in_a_write_cycle);

  //interrupt_0_avalon_slave_0_arb_share_counter arbitration counter enable, which is an
e_assign
  assign interrupt_0_avalon_slave_0_arb_counter_enable =
(end_xfer_arb_share_counter_term_interrupt_0_avalon_slave_0 &
interrupt_0_avalon_slave_0_allgrants) |
(end_xfer_arb_share_counter_term_interrupt_0_avalon_slave_0 &
~interrupt_0_avalon_slave_0_non_bursting_master_requests);

  //interrupt_0_avalon_slave_0_arb_share_counter counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          interrupt_0_avalon_slave_0_arb_share_counter <= 0;
      else if (interrupt_0_avalon_slave_0_arb_counter_enable)
          interrupt_0_avalon_slave_0_arb_share_counter <=
interrupt_0_avalon_slave_0_arb_share_counter_next_value;
    end


  //interrupt_0_avalon_slave_0_slavearbiterlockenable slave enables arbiterlock, which is an
e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          interrupt_0_avalon_slave_0_slavearbiterlockenable <= 0;
      else if ((|interrupt_0_avalon_slave_0_master_qreq_vector &
end_xfer_arb_share_counter_term_interrupt_0_avalon_slave_0) |
(end_xfer_arb_share_counter_term_interrupt_0_avalon_slave_0 &
~interrupt_0_avalon_slave_0_non_bursting_master_requests))
          interrupt_0_avalon_slave_0_slavearbiterlockenable <=
|interrupt_0_avalon_slave_0_arb_share_counter_next_value;
    end
```

```
  //cpu_0/data_master interrupt_0/avalon_slave_0 arbiterlock, which is an e_assign
  assign cpu_0_data_master_arbiterlock = interrupt_0_avalon_slave_0_slavearbiterlockenable
& cpu_0_data_master_continuerequest;

  //interrupt_0_avalon_slave_0_slavearbiterlockenable2 slave enables arbiterlock2, which is an
e_assign
  assign interrupt_0_avalon_slave_0_slavearbiterlockenable2 =
|interrupt_0_avalon_slave_0_arb_share_counter_next_value;

  //cpu_0/data_master interrupt_0/avalon_slave_0 arbiterlock2, which is an e_assign
  assign cpu_0_data_master_arbiterlock2 =
interrupt_0_avalon_slave_0_slavearbiterlockenable2 & cpu_0_data_master_continuerequest;

  //interrupt_0_avalon_slave_0_any_continuerequest at least one master continues requesting,
which is an e_assign
  assign interrupt_0_avalon_slave_0_any_continuerequest = 1;

  //cpu_0_data_master_continuerequest continued request, which is an e_assign
  assign cpu_0_data_master_continuerequest = 1;

  assign cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0 =
cpu_0_data_master_requests_interrupt_0_avalon_slave_0 &
~(((~cpu_0_data_master_waitrequest) & cpu_0_data_master_write));
  //interrupt_0_avalon_slave_0_writedata mux, which is an e_mux
  assign interrupt_0_avalon_slave_0_writedata = cpu_0_data_master_writedata;

  //master is always granted when requested
  assign cpu_0_data_master_granted_interrupt_0_avalon_slave_0 =
cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0;

  //cpu_0/data_master saved-grant interrupt_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_saved_grant_interrupt_0_avalon_slave_0 =
cpu_0_data_master_requests_interrupt_0_avalon_slave_0;

  //allow new arb cycle for interrupt_0/avalon_slave_0, which is an e_assign
  assign interrupt_0_avalon_slave_0_allow_new_arb_cycle = 1;

  //placeholder chosen master
  assign interrupt_0_avalon_slave_0_grant_vector = 1;

  //placeholder vector of master qualified-requests
  assign interrupt_0_avalon_slave_0_master_qreq_vector = 1;

  //interrupt_0_avalon_slave_0_reset_n assignment, which is an e_assign
  assign interrupt_0_avalon_slave_0_reset_n = reset_n;

  assign interrupt_0_avalon_slave_0_chipselect =
cpu_0_data_master_granted_interrupt_0_avalon_slave_0;
  //interrupt_0_avalon_slave_0_firsttransfer first transaction, which is an e_assign
  assign interrupt_0_avalon_slave_0_firsttransfer = interrupt_0_avalon_slave_0_begins_xfer ?
interrupt_0_avalon_slave_0_unreg_firsttransfer : interrupt_0_avalon_slave_0_reg_firsttransfer;

  //interrupt_0_avalon_slave_0_unreg_firsttransfer first transaction, which is an e_assign
  assign interrupt_0_avalon_slave_0_unreg_firsttransfer =
~(interrupt_0_avalon_slave_0_slavearbiterlockenable &
interrupt_0_avalon_slave_0_any_continuerequest);
```

```
  //interrupt_0_avalon_slave_0_reg_firsttransfer first transaction, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          interrupt_0_avalon_slave_0_reg_firsttransfer <= 1'b1;
      else if (interrupt_0_avalon_slave_0_begins_xfer)
          interrupt_0_avalon_slave_0_reg_firsttransfer <=
interrupt_0_avalon_slave_0_unreg_firsttransfer;
    end


  //interrupt_0_avalon_slave_0_beginbursttransfer_internal begin burst transfer, which is an
e_assign
  assign interrupt_0_avalon_slave_0_beginbursttransfer_internal =
interrupt_0_avalon_slave_0_begins_xfer;

  //interrupt_0_avalon_slave_0_read assignment, which is an e_mux
  assign interrupt_0_avalon_slave_0_read =
cpu_0_data_master_granted_interrupt_0_avalon_slave_0 & cpu_0_data_master_read;

  //interrupt_0_avalon_slave_0_write assignment, which is an e_mux
  assign interrupt_0_avalon_slave_0_write =
cpu_0_data_master_granted_interrupt_0_avalon_slave_0 & cpu_0_data_master_write;

  assign shifted_address_to_interrupt_0_avalon_slave_0_from_cpu_0_data_master =
cpu_0_data_master_address_to_slave;
  //interrupt_0_avalon_slave_0_address mux, which is an e_mux
  assign interrupt_0_avalon_slave_0_address =
shifted_address_to_interrupt_0_avalon_slave_0_from_cpu_0_data_master >> 2;

  //d1_interrupt_0_avalon_slave_0_end_xfer register, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_interrupt_0_avalon_slave_0_end_xfer <= 1;
      else
        d1_interrupt_0_avalon_slave_0_end_xfer <= interrupt_0_avalon_slave_0_end_xfer;
    end


  //interrupt_0_avalon_slave_0_waits_for_read in a cycle, which is an e_mux
  assign interrupt_0_avalon_slave_0_waits_for_read =
interrupt_0_avalon_slave_0_in_a_read_cycle & interrupt_0_avalon_slave_0_begins_xfer;

  //interrupt_0_avalon_slave_0_in_a_read_cycle assignment, which is an e_assign
  assign interrupt_0_avalon_slave_0_in_a_read_cycle =
cpu_0_data_master_granted_interrupt_0_avalon_slave_0 & cpu_0_data_master_read;

  //in_a_read_cycle assignment, which is an e_mux
  assign in_a_read_cycle = interrupt_0_avalon_slave_0_in_a_read_cycle;

  //interrupt_0_avalon_slave_0_waits_for_write in a cycle, which is an e_mux
  assign interrupt_0_avalon_slave_0_waits_for_write =
interrupt_0_avalon_slave_0_in_a_write_cycle & 0;

  //interrupt_0_avalon_slave_0_in_a_write_cycle assignment, which is an e_assign
  assign interrupt_0_avalon_slave_0_in_a_write_cycle =
```

```
cpu_0_data_master_granted_interrupt_0_avalon_slave_0 & cpu_0_data_master_write;

  //in_a_write_cycle assignment, which is an e_mux
  assign in_a_write_cycle = interrupt_0_avalon_slave_0_in_a_write_cycle;

  assign wait_for_interrupt_0_avalon_slave_0_counter = 0;
  //assign interrupt_0_avalon_slave_0_irq_from_sa = interrupt_0_avalon_slave_0_irq so that symbol knows where to group signals which may go to master only, which is an e_assign
  assign interrupt_0_avalon_slave_0_irq_from_sa = interrupt_0_avalon_slave_0_irq;


//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
  //interrupt_0/avalon_slave_0 enable non-zero assertions, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          enable_nonzero_assertions <= 0;
      else
        enable_nonzero_assertions <= 1'b1;
    end



//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module jtag_uart_0_avalon_jtag_slave_arbitrator (
                                                  // inputs:
                                                    clk,

cpu_0_data_master_address_to_slave,
                                                    cpu_0_data_master_read,
                                                    cpu_0_data_master_waitrequest,
                                                    cpu_0_data_master_write,
                                                    cpu_0_data_master_writedata,

jtag_uart_0_avalon_jtag_slave_dataavailable,

jtag_uart_0_avalon_jtag_slave_irq,

jtag_uart_0_avalon_jtag_slave_readdata,

jtag_uart_0_avalon_jtag_slave_readyfordata,

jtag_uart_0_avalon_jtag_slave_waitrequest,
```

```
                                                                reset_n,

                                                               // outputs:

cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave,

cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave,

cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave,

cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave,

d1_jtag_uart_0_avalon_jtag_slave_end_xfer,

jtag_uart_0_avalon_jtag_slave_address,

jtag_uart_0_avalon_jtag_slave_chipselect,

jtag_uart_0_avalon_jtag_slave_dataavailable_from_sa,

jtag_uart_0_avalon_jtag_slave_irq_from_sa,

jtag_uart_0_avalon_jtag_slave_read_n,

jtag_uart_0_avalon_jtag_slave_readdata_from_sa,

jtag_uart_0_avalon_jtag_slave_readyfordata_from_sa,

jtag_uart_0_avalon_jtag_slave_reset_n,

jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa,

jtag_uart_0_avalon_jtag_slave_write_n,

jtag_uart_0_avalon_jtag_slave_writedata
                                                                )
;

  output           cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave;
  output           cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave;
  output           cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave;
  output           cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave;
  output           d1_jtag_uart_0_avalon_jtag_slave_end_xfer;
  output           jtag_uart_0_avalon_jtag_slave_address;
  output           jtag_uart_0_avalon_jtag_slave_chipselect;
  output           jtag_uart_0_avalon_jtag_slave_dataavailable_from_sa;
  output           jtag_uart_0_avalon_jtag_slave_irq_from_sa;
  output           jtag_uart_0_avalon_jtag_slave_read_n;
  output   [ 31: 0] jtag_uart_0_avalon_jtag_slave_readdata_from_sa;
  output           jtag_uart_0_avalon_jtag_slave_readyfordata_from_sa;
  output           jtag_uart_0_avalon_jtag_slave_reset_n;
  output           jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa;
  output           jtag_uart_0_avalon_jtag_slave_write_n;
  output   [ 31: 0] jtag_uart_0_avalon_jtag_slave_writedata;
  input            clk;
  input    [ 24: 0] cpu_0_data_master_address_to_slave;
  input            cpu_0_data_master_read;
  input            cpu_0_data_master_waitrequest;
```

```
input              cpu_0_data_master_write;
input  [ 31: 0] cpu_0_data_master_writedata;
input              jtag_uart_0_avalon_jtag_slave_dataavailable;
input              jtag_uart_0_avalon_jtag_slave_irq;
input  [ 31: 0] jtag_uart_0_avalon_jtag_slave_readdata;
input              jtag_uart_0_avalon_jtag_slave_readyfordata;
input              jtag_uart_0_avalon_jtag_slave_waitrequest;
input              reset_n;

wire               cpu_0_data_master_arbiterlock;
wire               cpu_0_data_master_arbiterlock2;
wire               cpu_0_data_master_continuerequest;
wire               cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave;
wire               cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave;
wire               cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave;
wire               cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave;
wire               cpu_0_data_master_saved_grant_jtag_uart_0_avalon_jtag_slave;
reg                d1_jtag_uart_0_avalon_jtag_slave_end_xfer;
reg                d1_reasons_to_wait;
reg                enable_nonzero_assertions;
wire               end_xfer_arb_share_counter_term_jtag_uart_0_avalon_jtag_slave;
wire               in_a_read_cycle;
wire               in_a_write_cycle;
wire               jtag_uart_0_avalon_jtag_slave_address;
wire               jtag_uart_0_avalon_jtag_slave_allgrants;
wire               jtag_uart_0_avalon_jtag_slave_allow_new_arb_cycle;
wire               jtag_uart_0_avalon_jtag_slave_any_bursting_master_saved_grant;
wire               jtag_uart_0_avalon_jtag_slave_any_continuerequest;
wire               jtag_uart_0_avalon_jtag_slave_arb_counter_enable;
reg    [  1: 0] jtag_uart_0_avalon_jtag_slave_arb_share_counter;
wire   [  1: 0] jtag_uart_0_avalon_jtag_slave_arb_share_counter_next_value;
wire   [  1: 0] jtag_uart_0_avalon_jtag_slave_arb_share_set_values;
wire               jtag_uart_0_avalon_jtag_slave_beginbursttransfer_internal;
wire               jtag_uart_0_avalon_jtag_slave_begins_xfer;
wire               jtag_uart_0_avalon_jtag_slave_chipselect;
wire               jtag_uart_0_avalon_jtag_slave_dataavailable_from_sa;
wire               jtag_uart_0_avalon_jtag_slave_end_xfer;
wire               jtag_uart_0_avalon_jtag_slave_firsttransfer;
wire               jtag_uart_0_avalon_jtag_slave_grant_vector;
wire               jtag_uart_0_avalon_jtag_slave_in_a_read_cycle;
wire               jtag_uart_0_avalon_jtag_slave_in_a_write_cycle;
wire               jtag_uart_0_avalon_jtag_slave_irq_from_sa;
wire               jtag_uart_0_avalon_jtag_slave_master_qreq_vector;
wire               jtag_uart_0_avalon_jtag_slave_non_bursting_master_requests;
wire               jtag_uart_0_avalon_jtag_slave_read_n;
wire   [ 31: 0] jtag_uart_0_avalon_jtag_slave_readdata_from_sa;
wire               jtag_uart_0_avalon_jtag_slave_readyfordata_from_sa;
reg                jtag_uart_0_avalon_jtag_slave_reg_firsttransfer;
wire               jtag_uart_0_avalon_jtag_slave_reset_n;
reg                jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable;
wire               jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable2;
wire               jtag_uart_0_avalon_jtag_slave_unreg_firsttransfer;
wire               jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa;
wire               jtag_uart_0_avalon_jtag_slave_waits_for_read;
wire               jtag_uart_0_avalon_jtag_slave_waits_for_write;
wire               jtag_uart_0_avalon_jtag_slave_write_n;
wire   [ 31: 0] jtag_uart_0_avalon_jtag_slave_writedata;
wire                                                    [  24:  0]
```

```
shifted_address_to_jtag_uart_0_avalon_jtag_slave_from_cpu_0_data_master;
  wire                    wait_for_jtag_uart_0_avalon_jtag_slave_counter;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_reasons_to_wait <= 0;
      else
        d1_reasons_to_wait <= ~jtag_uart_0_avalon_jtag_slave_end_xfer;
    end


  assign jtag_uart_0_avalon_jtag_slave_begins_xfer = ~d1_reasons_to_wait &
((cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave));
  //assign                    jtag_uart_0_avalon_jtag_slave_readdata_from_sa                    =
jtag_uart_0_avalon_jtag_slave_readdata so that symbol knows where to group signals which
may go to master only, which is an e_assign
  assign                    jtag_uart_0_avalon_jtag_slave_readdata_from_sa                    =
jtag_uart_0_avalon_jtag_slave_readdata;

  assign                cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave                =
({cpu_0_data_master_address_to_slave[24    :    3]    ,    3'b0}    ==    25'h1001180)    &
(cpu_0_data_master_read | cpu_0_data_master_write);
  //assign                    jtag_uart_0_avalon_jtag_slave_dataavailable_from_sa                    =
jtag_uart_0_avalon_jtag_slave_dataavailable so that symbol knows where to group signals which
may go to master only, which is an e_assign
  assign                    jtag_uart_0_avalon_jtag_slave_dataavailable_from_sa                    =
jtag_uart_0_avalon_jtag_slave_dataavailable;

  //assign                    jtag_uart_0_avalon_jtag_slave_readyfordata_from_sa                    =
jtag_uart_0_avalon_jtag_slave_readyfordata so that symbol knows where to group signals which
may go to master only, which is an e_assign
  assign                    jtag_uart_0_avalon_jtag_slave_readyfordata_from_sa                    =
jtag_uart_0_avalon_jtag_slave_readyfordata;

  //assign                    jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa                    =
jtag_uart_0_avalon_jtag_slave_waitrequest so that symbol knows where to group signals which
may go to master only, which is an e_assign
  assign                    jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa                    =
jtag_uart_0_avalon_jtag_slave_waitrequest;

  //jtag_uart_0_avalon_jtag_slave_arb_share_counter set values, which is an e_mux
  assign jtag_uart_0_avalon_jtag_slave_arb_share_set_values = 1;

  //jtag_uart_0_avalon_jtag_slave_non_bursting_master_requests mux, which is an e_mux
  assign                jtag_uart_0_avalon_jtag_slave_non_bursting_master_requests                =
cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave;

  //jtag_uart_0_avalon_jtag_slave_any_bursting_master_saved_grant mux, which is an e_mux
  assign jtag_uart_0_avalon_jtag_slave_any_bursting_master_saved_grant = 0;

  //jtag_uart_0_avalon_jtag_slave_arb_share_counter_next_value assignment, which is an
e_assign
  assign                jtag_uart_0_avalon_jtag_slave_arb_share_counter_next_value                =
jtag_uart_0_avalon_jtag_slave_firsttransfer                                                     ?
(jtag_uart_0_avalon_jtag_slave_arb_share_set_values                -                1)          :
|jtag_uart_0_avalon_jtag_slave_arb_share_counter                                               ?
(jtag_uart_0_avalon_jtag_slave_arb_share_counter - 1) : 0;
```

```
  //jtag_uart_0_avalon_jtag_slave_allgrants all slave grants, which is an e_mux
  assign jtag_uart_0_avalon_jtag_slave_allgrants =
|jtag_uart_0_avalon_jtag_slave_grant_vector;

  //jtag_uart_0_avalon_jtag_slave_end_xfer assignment, which is an e_assign
  assign jtag_uart_0_avalon_jtag_slave_end_xfer =
~(jtag_uart_0_avalon_jtag_slave_waits_for_read |
jtag_uart_0_avalon_jtag_slave_waits_for_write);

  //end_xfer_arb_share_counter_term_jtag_uart_0_avalon_jtag_slave arb share counter
enable term, which is an e_assign
  assign end_xfer_arb_share_counter_term_jtag_uart_0_avalon_jtag_slave =
jtag_uart_0_avalon_jtag_slave_end_xfer &
(~jtag_uart_0_avalon_jtag_slave_any_bursting_master_saved_grant | in_a_read_cycle |
in_a_write_cycle);

  //jtag_uart_0_avalon_jtag_slave_arb_share_counter arbitration counter enable, which is an
e_assign
  assign jtag_uart_0_avalon_jtag_slave_arb_counter_enable =
(end_xfer_arb_share_counter_term_jtag_uart_0_avalon_jtag_slave &
jtag_uart_0_avalon_jtag_slave_allgrants) |
(end_xfer_arb_share_counter_term_jtag_uart_0_avalon_jtag_slave &
~jtag_uart_0_avalon_jtag_slave_non_bursting_master_requests);

  //jtag_uart_0_avalon_jtag_slave_arb_share_counter counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          jtag_uart_0_avalon_jtag_slave_arb_share_counter <= 0;
      else if (jtag_uart_0_avalon_jtag_slave_arb_counter_enable)
          jtag_uart_0_avalon_jtag_slave_arb_share_counter <=
jtag_uart_0_avalon_jtag_slave_arb_share_counter_next_value;
    end


  //jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable slave enables arbiterlock, which is an
e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable <= 0;
      else if (((|jtag_uart_0_avalon_jtag_slave_master_qreq_vector &
end_xfer_arb_share_counter_term_jtag_uart_0_avalon_jtag_slave) |
(end_xfer_arb_share_counter_term_jtag_uart_0_avalon_jtag_slave &
~jtag_uart_0_avalon_jtag_slave_non_bursting_master_requests))
          jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable <=
|jtag_uart_0_avalon_jtag_slave_arb_share_counter_next_value;
    end


  //cpu_0/data_master jtag_uart_0/avalon_jtag_slave arbiterlock, which is an e_assign
  assign cpu_0_data_master_arbiterlock =
jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable & cpu_0_data_master_continuerequest;

  //jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable2 slave enables arbiterlock2, which is
an e_assign
  assign jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable2 =
|jtag_uart_0_avalon_jtag_slave_arb_share_counter_next_value;
```

```
  //cpu_0/data_master jtag_uart_0/avalon_jtag_slave arbiterlock2, which is an e_assign
  assign cpu_0_data_master_arbiterlock2 = jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable2 & cpu_0_data_master_continuerequest;

  //jtag_uart_0_avalon_jtag_slave_any_continuerequest at least one master continues requesting, which is an e_assign
  assign jtag_uart_0_avalon_jtag_slave_any_continuerequest = 1;

  //cpu_0_data_master_continuerequest continued request, which is an e_assign
  assign cpu_0_data_master_continuerequest = 1;

  assign cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave = cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave & ~((cpu_0_data_master_read & (~cpu_0_data_master_waitrequest)) | ((~cpu_0_data_master_waitrequest) & cpu_0_data_master_write));
  //jtag_uart_0_avalon_jtag_slave_writedata mux, which is an e_mux
  assign jtag_uart_0_avalon_jtag_slave_writedata = cpu_0_data_master_writedata;

  //master is always granted when requested
  assign cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave = cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave;

  //cpu_0/data_master saved-grant jtag_uart_0/avalon_jtag_slave, which is an e_assign
  assign cpu_0_data_master_saved_grant_jtag_uart_0_avalon_jtag_slave = cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave;

  //allow new arb cycle for jtag_uart_0/avalon_jtag_slave, which is an e_assign
  assign jtag_uart_0_avalon_jtag_slave_allow_new_arb_cycle = 1;

  //placeholder chosen master
  assign jtag_uart_0_avalon_jtag_slave_grant_vector = 1;

  //placeholder vector of master qualified-requests
  assign jtag_uart_0_avalon_jtag_slave_master_qreq_vector = 1;

  //jtag_uart_0_avalon_jtag_slave_reset_n assignment, which is an e_assign
  assign jtag_uart_0_avalon_jtag_slave_reset_n = reset_n;

  assign jtag_uart_0_avalon_jtag_slave_chipselect = cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave;
  //jtag_uart_0_avalon_jtag_slave_firsttransfer first transaction, which is an e_assign
  assign jtag_uart_0_avalon_jtag_slave_firsttransfer = jtag_uart_0_avalon_jtag_slave_begins_xfer ? jtag_uart_0_avalon_jtag_slave_unreg_firsttransfer : jtag_uart_0_avalon_jtag_slave_reg_firsttransfer;

  //jtag_uart_0_avalon_jtag_slave_unreg_firsttransfer first transaction, which is an e_assign
  assign jtag_uart_0_avalon_jtag_slave_unreg_firsttransfer = ~(jtag_uart_0_avalon_jtag_slave_slavearbiterlockenable & jtag_uart_0_avalon_jtag_slave_any_continuerequest);

  //jtag_uart_0_avalon_jtag_slave_reg_firsttransfer first transaction, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          jtag_uart_0_avalon_jtag_slave_reg_firsttransfer <= 1'b1;
      else if (jtag_uart_0_avalon_jtag_slave_begins_xfer)
          jtag_uart_0_avalon_jtag_slave_reg_firsttransfer <=
```

```
jtag_uart_0_avalon_jtag_slave_unreg_firsttransfer;
      end


   //jtag_uart_0_avalon_jtag_slave_beginbursttransfer_internal begin burst transfer, which is an
e_assign
   assign                jtag_uart_0_avalon_jtag_slave_beginbursttransfer_internal                =
jtag_uart_0_avalon_jtag_slave_begins_xfer;

   //~jtag_uart_0_avalon_jtag_slave_read_n assignment, which is an e_mux
   assign                                jtag_uart_0_avalon_jtag_slave_read_n                     =
~(cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave & cpu_0_data_master_read);

   //~jtag_uart_0_avalon_jtag_slave_write_n assignment, which is an e_mux
   assign                                jtag_uart_0_avalon_jtag_slave_write_n                    =
~(cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave & cpu_0_data_master_write);

   assign      shifted_address_to_jtag_uart_0_avalon_jtag_slave_from_cpu_0_data_master      =
cpu_0_data_master_address_to_slave;
   //jtag_uart_0_avalon_jtag_slave_address mux, which is an e_mux
   assign                                jtag_uart_0_avalon_jtag_slave_address                    =
shifted_address_to_jtag_uart_0_avalon_jtag_slave_from_cpu_0_data_master >> 2;

   //d1_jtag_uart_0_avalon_jtag_slave_end_xfer register, which is an e_register
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               d1_jtag_uart_0_avalon_jtag_slave_end_xfer <= 1;
         else
            d1_jtag_uart_0_avalon_jtag_slave_end_xfer                                          <=
jtag_uart_0_avalon_jtag_slave_end_xfer;
      end


   //jtag_uart_0_avalon_jtag_slave_waits_for_read in a cycle, which is an e_mux
   assign                                jtag_uart_0_avalon_jtag_slave_waits_for_read             =
jtag_uart_0_avalon_jtag_slave_in_a_read_cycle                                                      &
jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa;

   //jtag_uart_0_avalon_jtag_slave_in_a_read_cycle assignment, which is an e_assign
   assign                                jtag_uart_0_avalon_jtag_slave_in_a_read_cycle             =
cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave & cpu_0_data_master_read;

   //in_a_read_cycle assignment, which is an e_mux
   assign in_a_read_cycle = jtag_uart_0_avalon_jtag_slave_in_a_read_cycle;

   //jtag_uart_0_avalon_jtag_slave_waits_for_write in a cycle, which is an e_mux
   assign                                jtag_uart_0_avalon_jtag_slave_waits_for_write            =
jtag_uart_0_avalon_jtag_slave_in_a_write_cycle                                                     &
jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa;

   //jtag_uart_0_avalon_jtag_slave_in_a_write_cycle assignment, which is an e_assign
   assign                                jtag_uart_0_avalon_jtag_slave_in_a_write_cycle            =
cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave & cpu_0_data_master_write;

   //in_a_write_cycle assignment, which is an e_mux
   assign in_a_write_cycle = jtag_uart_0_avalon_jtag_slave_in_a_write_cycle;
```

```
  assign wait_for_jtag_uart_0_avalon_jtag_slave_counter = 0;
  //assign  jtag_uart_0_avalon_jtag_slave_irq_from_sa  =  jtag_uart_0_avalon_jtag_slave_irq  so
that symbol knows where to group signals which may go to master only, which is an e_assign
  assign jtag_uart_0_avalon_jtag_slave_irq_from_sa = jtag_uart_0_avalon_jtag_slave_irq;


//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
  //jtag_uart_0/avalon_jtag_slave enable non-zero assertions, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          enable_nonzero_assertions <= 0;
      else
        enable_nonzero_assertions <= 1'b1;
    end



//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module musiccontroller_0_avalon_slave_0_arbitrator (
                                                     // inputs:
                                                      clk,

cpu_0_data_master_address_to_slave,

cpu_0_data_master_byteenable,

cpu_0_data_master_dbs_address,

cpu_0_data_master_dbs_write_16,

cpu_0_data_master_no_byte_enables_and_last_term,
                                                      cpu_0_data_master_read,

cpu_0_data_master_waitrequest,
                                                      cpu_0_data_master_write,

musiccontroller_0_avalon_slave_0_irq,

musiccontroller_0_avalon_slave_0_readdata,
                                                      reset_n,

                                                     // outputs:
```

```
cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0,

cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0,

cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0,

cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0,

d1_musiccontroller_0_avalon_slave_0_end_xfer,

musiccontroller_0_avalon_slave_0_address,

musiccontroller_0_avalon_slave_0_chipselect,

musiccontroller_0_avalon_slave_0_irq_from_sa,

musiccontroller_0_avalon_slave_0_read,

musiccontroller_0_avalon_slave_0_readdata_from_sa,

musiccontroller_0_avalon_slave_0_reset_n,

musiccontroller_0_avalon_slave_0_write,

musiccontroller_0_avalon_slave_0_writedata
                                                                )
;

   output    [    1: 0] cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0;
   output                cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0;
   output
cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0;
   output                cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0;
   output                cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0;
   output                d1_musiccontroller_0_avalon_slave_0_end_xfer;
   output    [ 16: 0] musiccontroller_0_avalon_slave_0_address;
   output                musiccontroller_0_avalon_slave_0_chipselect;
   output                musiccontroller_0_avalon_slave_0_irq_from_sa;
   output                musiccontroller_0_avalon_slave_0_read;
   output    [ 15: 0] musiccontroller_0_avalon_slave_0_readdata_from_sa;
   output                musiccontroller_0_avalon_slave_0_reset_n;
   output                musiccontroller_0_avalon_slave_0_write;
   output    [ 15: 0] musiccontroller_0_avalon_slave_0_writedata;
   input                 clk;
   input     [ 24: 0] cpu_0_data_master_address_to_slave;
   input     [    3: 0] cpu_0_data_master_byteenable;
   input     [    1: 0] cpu_0_data_master_dbs_address;
   input     [ 15: 0] cpu_0_data_master_dbs_write_16;
   input                 cpu_0_data_master_no_byte_enables_and_last_term;
   input                 cpu_0_data_master_read;
   input                 cpu_0_data_master_waitrequest;
   input                 cpu_0_data_master_write;
   input                 musiccontroller_0_avalon_slave_0_irq;
   input     [ 15: 0] musiccontroller_0_avalon_slave_0_readdata;
   input                 reset_n;
```

```
  wire                cpu_0_data_master_arbiterlock;
  wire                cpu_0_data_master_arbiterlock2;
  wire    [  1: 0] cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0;
  wire                                                [                1:                0]
cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0_segment_0;
  wire                                                [                1:                0]
cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0_segment_1;
  wire                cpu_0_data_master_continuerequest;
  wire                cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0;
  wire
cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0;
  wire                cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0;
  wire                cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0;
  wire                cpu_0_data_master_saved_grant_musiccontroller_0_avalon_slave_0;
  reg                 d1_musiccontroller_0_avalon_slave_0_end_xfer;
  reg                 d1_reasons_to_wait;
  reg                 enable_nonzero_assertions;
  wire                end_xfer_arb_share_counter_term_musiccontroller_0_avalon_slave_0;
  wire                in_a_read_cycle;
  wire                in_a_write_cycle;
  wire    [ 16: 0] musiccontroller_0_avalon_slave_0_address;
  wire                musiccontroller_0_avalon_slave_0_allgrants;
  wire                musiccontroller_0_avalon_slave_0_allow_new_arb_cycle;
  wire                musiccontroller_0_avalon_slave_0_any_bursting_master_saved_grant;
  wire                musiccontroller_0_avalon_slave_0_any_continuerequest;
  wire                musiccontroller_0_avalon_slave_0_arb_counter_enable;
  reg     [  1: 0] musiccontroller_0_avalon_slave_0_arb_share_counter;
  wire    [  1: 0] musiccontroller_0_avalon_slave_0_arb_share_counter_next_value;
  wire    [  1: 0] musiccontroller_0_avalon_slave_0_arb_share_set_values;
  wire                musiccontroller_0_avalon_slave_0_beginbursttransfer_internal;
  wire                musiccontroller_0_avalon_slave_0_begins_xfer;
  wire                musiccontroller_0_avalon_slave_0_chipselect;
  wire                musiccontroller_0_avalon_slave_0_end_xfer;
  wire                musiccontroller_0_avalon_slave_0_firsttransfer;
  wire                musiccontroller_0_avalon_slave_0_grant_vector;
  wire                musiccontroller_0_avalon_slave_0_in_a_read_cycle;
  wire                musiccontroller_0_avalon_slave_0_in_a_write_cycle;
  wire                musiccontroller_0_avalon_slave_0_irq_from_sa;
  wire                musiccontroller_0_avalon_slave_0_master_qreq_vector;
  wire                musiccontroller_0_avalon_slave_0_non_bursting_master_requests;
  wire                musiccontroller_0_avalon_slave_0_read;
  wire    [ 15: 0] musiccontroller_0_avalon_slave_0_readdata_from_sa;
  reg                 musiccontroller_0_avalon_slave_0_reg_firsttransfer;
  wire                musiccontroller_0_avalon_slave_0_reset_n;
  reg                 musiccontroller_0_avalon_slave_0_slavearbiterlockenable;
  wire                musiccontroller_0_avalon_slave_0_slavearbiterlockenable2;
  wire                musiccontroller_0_avalon_slave_0_unreg_firsttransfer;
  wire                musiccontroller_0_avalon_slave_0_waits_for_read;
  wire                musiccontroller_0_avalon_slave_0_waits_for_write;
  wire                musiccontroller_0_avalon_slave_0_write;
  wire    [ 15: 0] musiccontroller_0_avalon_slave_0_writedata;
  wire                                                [               24:                0]
shifted_address_to_musiccontroller_0_avalon_slave_0_from_cpu_0_data_master;
  wire                wait_for_musiccontroller_0_avalon_slave_0_counter;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
```

```
            d1_reasons_to_wait <= 0;
        else
          d1_reasons_to_wait <= ~musiccontroller_0_avalon_slave_0_end_xfer;
      end


  assign musiccontroller_0_avalon_slave_0_begins_xfer = ~d1_reasons_to_wait & ((cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0));
  //assign musiccontroller_0_avalon_slave_0_readdata_from_sa = musiccontroller_0_avalon_slave_0_readdata so that symbol knows where to group signals which may go to master only, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_readdata_from_sa = musiccontroller_0_avalon_slave_0_readdata;

  assign cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0 = ({cpu_0_data_master_address_to_slave[24 : 18] , 18'b0} == 25'h40000) & (cpu_0_data_master_read | cpu_0_data_master_write);
  //musiccontroller_0_avalon_slave_0_arb_share_counter set values, which is an e_mux
  assign musiccontroller_0_avalon_slave_0_arb_share_set_values = (cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0)? 2 :
    1;

  //musiccontroller_0_avalon_slave_0_non_bursting_master_requests mux, which is an e_mux
  assign musiccontroller_0_avalon_slave_0_non_bursting_master_requests = cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0;

  //musiccontroller_0_avalon_slave_0_any_bursting_master_saved_grant mux, which is an e_mux
  assign musiccontroller_0_avalon_slave_0_any_bursting_master_saved_grant = 0;

  //musiccontroller_0_avalon_slave_0_arb_share_counter_next_value assignment, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_arb_share_counter_next_value = musiccontroller_0_avalon_slave_0_firsttransfer ? (musiccontroller_0_avalon_slave_0_arb_share_set_values - 1) : |musiccontroller_0_avalon_slave_0_arb_share_counter ? (musiccontroller_0_avalon_slave_0_arb_share_counter - 1) : 0;

  //musiccontroller_0_avalon_slave_0_allgrants all slave grants, which is an e_mux
  assign musiccontroller_0_avalon_slave_0_allgrants = |musiccontroller_0_avalon_slave_0_grant_vector;

  //musiccontroller_0_avalon_slave_0_end_xfer assignment, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_end_xfer = ~(musiccontroller_0_avalon_slave_0_waits_for_read | musiccontroller_0_avalon_slave_0_waits_for_write);

  //end_xfer_arb_share_counter_term_musiccontroller_0_avalon_slave_0 arb share counter enable term, which is an e_assign
  assign end_xfer_arb_share_counter_term_musiccontroller_0_avalon_slave_0 = musiccontroller_0_avalon_slave_0_end_xfer & (~musiccontroller_0_avalon_slave_0_any_bursting_master_saved_grant | in_a_read_cycle | in_a_write_cycle);

  //musiccontroller_0_avalon_slave_0_arb_share_counter arbitration counter enable, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_arb_counter_enable = (end_xfer_arb_share_counter_term_musiccontroller_0_avalon_slave_0 &
```

```
musiccontroller_0_avalon_slave_0_allgrants)                                    |
(end_xfer_arb_share_counter_term_musiccontroller_0_avalon_slave_0              &
~musiccontroller_0_avalon_slave_0_non_bursting_master_requests);

  //musiccontroller_0_avalon_slave_0_arb_share_counter counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          musiccontroller_0_avalon_slave_0_arb_share_counter <= 0;
      else if (musiccontroller_0_avalon_slave_0_arb_counter_enable)
          musiccontroller_0_avalon_slave_0_arb_share_counter                   <=
musiccontroller_0_avalon_slave_0_arb_share_counter_next_value;
    end


  //musiccontroller_0_avalon_slave_0_slavearbiterlockenable slave enables arbiterlock, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          musiccontroller_0_avalon_slave_0_slavearbiterlockenable <= 0;
      else if (({|musiccontroller_0_avalon_slave_0_master_qreq_vector &
end_xfer_arb_share_counter_term_musiccontroller_0_avalon_slave_0)              |
(end_xfer_arb_share_counter_term_musiccontroller_0_avalon_slave_0              &
~musiccontroller_0_avalon_slave_0_non_bursting_master_requests))
          musiccontroller_0_avalon_slave_0_slavearbiterlockenable              <=
|musiccontroller_0_avalon_slave_0_arb_share_counter_next_value;
    end


  //cpu_0/data_master musiccontroller_0/avalon_slave_0 arbiterlock, which is an e_assign
  assign                                cpu_0_data_master_arbiterlock          =
musiccontroller_0_avalon_slave_0_slavearbiterlockenable                        &
cpu_0_data_master_continuerequest;

  //musiccontroller_0_avalon_slave_0_slavearbiterlockenable2 slave enables arbiterlock2, which is an e_assign
  assign                musiccontroller_0_avalon_slave_0_slavearbiterlockenable2 =
|musiccontroller_0_avalon_slave_0_arb_share_counter_next_value;

  //cpu_0/data_master musiccontroller_0/avalon_slave_0 arbiterlock2, which is an e_assign
  assign                               cpu_0_data_master_arbiterlock2          =
musiccontroller_0_avalon_slave_0_slavearbiterlockenable2                       &
cpu_0_data_master_continuerequest;

  //musiccontroller_0_avalon_slave_0_any_continuerequest at least one master continues requesting, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_any_continuerequest = 1;

  //cpu_0_data_master_continuerequest continued request, which is an e_assign
  assign cpu_0_data_master_continuerequest = 1;

  assign        cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0 =
cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0                    &
~(((~cpu_0_data_master_waitrequest   |   cpu_0_data_master_no_byte_enables_and_last_term
|               !cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0)          &
cpu_0_data_master_write));
  //musiccontroller_0_avalon_slave_0_writedata mux, which is an e_mux
```

```
  assign musiccontroller_0_avalon_slave_0_writedata = cpu_0_data_master_dbs_write_16;

  //master is always granted when requested
  assign cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0 = cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0;

  //cpu_0/data_master saved-grant musiccontroller_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_saved_grant_musiccontroller_0_avalon_slave_0 = cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0;

  //allow new arb cycle for musiccontroller_0/avalon_slave_0, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_allow_new_arb_cycle = 1;

  //placeholder chosen master
  assign musiccontroller_0_avalon_slave_0_grant_vector = 1;

  //placeholder vector of master qualified-requests
  assign musiccontroller_0_avalon_slave_0_master_qreq_vector = 1;

  //musiccontroller_0_avalon_slave_0_reset_n assignment, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_reset_n = reset_n;

  assign musiccontroller_0_avalon_slave_0_chipselect = cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0;
  //musiccontroller_0_avalon_slave_0_firsttransfer first transaction, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_firsttransfer = musiccontroller_0_avalon_slave_0_begins_xfer ? musiccontroller_0_avalon_slave_0_unreg_firsttransfer : musiccontroller_0_avalon_slave_0_reg_firsttransfer;

  //musiccontroller_0_avalon_slave_0_unreg_firsttransfer first transaction, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_unreg_firsttransfer = ~(musiccontroller_0_avalon_slave_0_slavearbiterlockenable & musiccontroller_0_avalon_slave_0_any_continuerequest);

  //musiccontroller_0_avalon_slave_0_reg_firsttransfer first transaction, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          musiccontroller_0_avalon_slave_0_reg_firsttransfer <= 1'b1;
      else if (musiccontroller_0_avalon_slave_0_begins_xfer)
          musiccontroller_0_avalon_slave_0_reg_firsttransfer <= musiccontroller_0_avalon_slave_0_unreg_firsttransfer;
    end


  //musiccontroller_0_avalon_slave_0_beginbursttransfer_internal begin burst transfer, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_beginbursttransfer_internal = musiccontroller_0_avalon_slave_0_begins_xfer;

  //musiccontroller_0_avalon_slave_0_read assignment, which is an e_mux
  assign musiccontroller_0_avalon_slave_0_read = cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0 & cpu_0_data_master_read;

  //musiccontroller_0_avalon_slave_0_write assignment, which is an e_mux
  assign musiccontroller_0_avalon_slave_0_write = cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0 & cpu_0_data_master_write;
```

```
  assign shifted_address_to_musiccontroller_0_avalon_slave_0_from_cpu_0_data_master =
{cpu_0_data_master_address_to_slave >> 2,
    cpu_0_data_master_dbs_address[1],
    {1 {1'b0}}};

  //musiccontroller_0_avalon_slave_0_address mux, which is an e_mux
  assign musiccontroller_0_avalon_slave_0_address =
shifted_address_to_musiccontroller_0_avalon_slave_0_from_cpu_0_data_master >> 1;

  //d1_musiccontroller_0_avalon_slave_0_end_xfer register, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_musiccontroller_0_avalon_slave_0_end_xfer <= 1;
      else
        d1_musiccontroller_0_avalon_slave_0_end_xfer <=
musiccontroller_0_avalon_slave_0_end_xfer;
    end


  //musiccontroller_0_avalon_slave_0_waits_for_read in a cycle, which is an e_mux
  assign musiccontroller_0_avalon_slave_0_waits_for_read =
musiccontroller_0_avalon_slave_0_in_a_read_cycle &
musiccontroller_0_avalon_slave_0_begins_xfer;

  //musiccontroller_0_avalon_slave_0_in_a_read_cycle assignment, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_in_a_read_cycle =
cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0 & cpu_0_data_master_read;

  //in_a_read_cycle assignment, which is an e_mux
  assign in_a_read_cycle = musiccontroller_0_avalon_slave_0_in_a_read_cycle;

  //musiccontroller_0_avalon_slave_0_waits_for_write in a cycle, which is an e_mux
  assign musiccontroller_0_avalon_slave_0_waits_for_write =
musiccontroller_0_avalon_slave_0_in_a_write_cycle & 0;

  //musiccontroller_0_avalon_slave_0_in_a_write_cycle assignment, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_in_a_write_cycle =
cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0 & cpu_0_data_master_write;

  //in_a_write_cycle assignment, which is an e_mux
  assign in_a_write_cycle = musiccontroller_0_avalon_slave_0_in_a_write_cycle;

  assign wait_for_musiccontroller_0_avalon_slave_0_counter = 0;
  assign {cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0_segment_1,
cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0_segment_0} =
cpu_0_data_master_byteenable;
  assign cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0 =
((cpu_0_data_master_dbs_address[1] == 0))?
cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0_segment_0 :
    cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0_segment_1;

  //assign musiccontroller_0_avalon_slave_0_irq_from_sa =
musiccontroller_0_avalon_slave_0_irq so that symbol knows where to group signals which may
go to master only, which is an e_assign
  assign musiccontroller_0_avalon_slave_0_irq_from_sa =
musiccontroller_0_avalon_slave_0_irq;
```

```
//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
   //musiccontroller_0/avalon_slave_0 enable non-zero assertions, which is an e_register
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               enable_nonzero_assertions <= 0;
         else
            enable_nonzero_assertions <= 1'b1;
      end



//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module rdv_fifo_for_cpu_0_data_master_to_sdram_0_s1_module (
                                                             // inputs:
                                                               clear_fifo,
                                                               clk,
                                                               data_in,
                                                               read,
                                                               reset_n,
                                                               sync_reset,
                                                               write,

                                                             // outputs:
                                                               data_out,
                                                               empty,

fifo_contains_ones_n,
                                                               full
                                                            )
;

   output           data_out;
   output           empty;
   output           fifo_contains_ones_n;
   output           full;
   input            clear_fifo;
   input            clk;
   input            data_in;
   input            read;
   input            reset_n;
```

```
input                   sync_reset;
input                   write;

wire                    data_out;
wire                    empty;
reg                     fifo_contains_ones_n;
wire                    full;
reg                     full_0;
reg                     full_1;
reg                     full_2;
reg                     full_3;
reg                     full_4;
reg                     full_5;
reg                     full_6;
wire                    full_7;
reg       [   3: 0] how_many_ones;
wire      [   3: 0] one_count_minus_one;
wire      [   3: 0] one_count_plus_one;
wire                    p0_full_0;
wire                    p0_stage_0;
wire                    p1_full_1;
wire                    p1_stage_1;
wire                    p2_full_2;
wire                    p2_stage_2;
wire                    p3_full_3;
wire                    p3_stage_3;
wire                    p4_full_4;
wire                    p4_stage_4;
wire                    p5_full_5;
wire                    p5_stage_5;
wire                    p6_full_6;
wire                    p6_stage_6;
reg                     stage_0;
reg                     stage_1;
reg                     stage_2;
reg                     stage_3;
reg                     stage_4;
reg                     stage_5;
reg                     stage_6;
wire      [   3: 0] updated_one_count;
assign data_out = stage_0;
assign full = full_6;
assign empty = !full_0;
assign full_7 = 0;
//data_6, which is an e_mux
assign p6_stage_6 = ((full_7 & ~clear_fifo) == 0)? data_in :
   data_in;

//data_reg_6, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            stage_6 <= 0;
      else if (clear_fifo | sync_reset | read | (write & !full_6))
            if (sync_reset & full_6 & !((full_7 == 0) & read & write))
                  stage_6 <= 0;
            else
               stage_6 <= p6_stage_6;
```

```
    end


//control_6, which is an e_mux
assign p6_full_6 = ((read & !write) == 0)? full_5 :
    0;

//control_reg_6, which is an e_register
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            full_6 <= 0;
        else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
                full_6 <= 0;
            else
                full_6 <= p6_full_6;
    end


//data_5, which is an e_mux
assign p5_stage_5 = ((full_6 & ~clear_fifo) == 0)? data_in :
    stage_6;

//data_reg_5, which is an e_register
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            stage_5 <= 0;
        else if (clear_fifo | sync_reset | read | (write & !full_5))
            if (sync_reset & full_5 & !((full_6 == 0) & read & write))
                stage_5 <= 0;
            else
                stage_5 <= p5_stage_5;
    end


//control_5, which is an e_mux
assign p5_full_5 = ((read & !write) == 0)? full_4 :
    full_6;

//control_reg_5, which is an e_register
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            full_5 <= 0;
        else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
                full_5 <= 0;
            else
                full_5 <= p5_full_5;
    end


//data_4, which is an e_mux
assign p4_stage_4 = ((full_5 & ~clear_fifo) == 0)? data_in :
    stage_5;
```

```
//data_reg_4, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            stage_4 <= 0;
      else if (clear_fifo | sync_reset | read | (write & !full_4))
            if (sync_reset & full_4 & !((full_5 == 0) & read & write))
                  stage_4 <= 0;
            else
               stage_4 <= p4_stage_4;
   end


//control_4, which is an e_mux
assign p4_full_4 = ((read & !write) == 0)? full_3 :
   full_5;

//control_reg_4, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            full_4 <= 0;
      else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
                  full_4 <= 0;
            else
               full_4 <= p4_full_4;
   end


//data_3, which is an e_mux
assign p3_stage_3 = ((full_4 & ~clear_fifo) == 0)? data_in :
   stage_4;

//data_reg_3, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            stage_3 <= 0;
      else if (clear_fifo | sync_reset | read | (write & !full_3))
            if (sync_reset & full_3 & !((full_4 == 0) & read & write))
                  stage_3 <= 0;
            else
               stage_3 <= p3_stage_3;
   end


//control_3, which is an e_mux
assign p3_full_3 = ((read & !write) == 0)? full_2 :
   full_4;

//control_reg_3, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            full_3 <= 0;
      else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
```

```
                full_3 <= 0;
            else
                full_3 <= p3_full_3;
    end


//data_2, which is an e_mux
assign p2_stage_2 = ((full_3 & ~clear_fifo) == 0)? data_in :
    stage_3;

//data_reg_2, which is an e_register
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            stage_2 <= 0;
        else if (clear_fifo | sync_reset | read | (write & !full_2))
            if (sync_reset & full_2 & !((full_3 == 0) & read & write))
                stage_2 <= 0;
            else
                stage_2 <= p2_stage_2;
    end


//control_2, which is an e_mux
assign p2_full_2 = ((read & !write) == 0)? full_1 :
    full_3;

//control_reg_2, which is an e_register
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            full_2 <= 0;
        else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
                full_2 <= 0;
            else
                full_2 <= p2_full_2;
    end


//data_1, which is an e_mux
assign p1_stage_1 = ((full_2 & ~clear_fifo) == 0)? data_in :
    stage_2;

//data_reg_1, which is an e_register
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            stage_1 <= 0;
        else if (clear_fifo | sync_reset | read | (write & !full_1))
            if (sync_reset & full_1 & !((full_2 == 0) & read & write))
                stage_1 <= 0;
            else
                stage_1 <= p1_stage_1;
    end


//control_1, which is an e_mux
```

```
assign p1_full_1 = ((read & !write) == 0)? full_0 :
   full_2;

//control_reg_1, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            full_1 <= 0;
      else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
                  full_1 <= 0;
            else
               full_1 <= p1_full_1;
   end


//data_0, which is an e_mux
assign p0_stage_0 = ((full_1 & ~clear_fifo) == 0)? data_in :
   stage_1;

//data_reg_0, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            stage_0 <= 0;
      else if (clear_fifo | sync_reset | read | (write & !full_0))
            if (sync_reset & full_0 & !((full_1 == 0) & read & write))
                  stage_0 <= 0;
            else
               stage_0 <= p0_stage_0;
   end


//control_0, which is an e_mux
assign p0_full_0 = ((read & !write) == 0)? 1 :
   full_1;

//control_reg_0, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            full_0 <= 0;
      else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo & ~write)
                  full_0 <= 0;
            else
               full_0 <= p0_full_0;
   end


assign one_count_plus_one = how_many_ones + 1;
assign one_count_minus_one = how_many_ones - 1;
//updated_one_count, which is an e_mux
assign updated_one_count = ((((clear_fifo | sync_reset) & !write)))? 0 :
   ((((clear_fifo | sync_reset) & write)))? |data_in :
   ((read & (|data_in) & write & (|stage_0)))? how_many_ones :
   ((write & (|data_in)))? one_count_plus_one :
   ((read & (|stage_0)))? one_count_minus_one :
```

```
    how_many_ones;

  //counts how many ones in the data pipeline, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          how_many_ones <= 0;
      else if (clear_fifo | sync_reset | read | write)
          how_many_ones <= updated_one_count;
    end


  //this fifo contains ones in the data pipeline, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          fifo_contains_ones_n <= 1;
      else if (clear_fifo | sync_reset | read | write)
          fifo_contains_ones_n <= ~(|updated_one_count);
    end



endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module rdv_fifo_for_cpu_0_instruction_master_to_sdram_0_s1_module (
                                                                    // inputs:
                                                                      clear_fifo,
                                                                      clk,
                                                                      data_in,
                                                                      read,
                                                                      reset_n,
                                                                      sync_reset,
                                                                      write,

                                                                    // outputs:
                                                                      data_out,
                                                                      empty,

fifo_contains_ones_n,

                                                                      full
                                                                   )
;

  output           data_out;
  output           empty;
  output           fifo_contains_ones_n;
  output           full;
  input            clear_fifo;
```

```
input                   clk;
input                   data_in;
input                   read;
input                   reset_n;
input                   sync_reset;
input                   write;

wire                    data_out;
wire                    empty;
reg                     fifo_contains_ones_n;
wire                    full;
reg                     full_0;
reg                     full_1;
reg                     full_2;
reg                     full_3;
reg                     full_4;
reg                     full_5;
reg                     full_6;
wire                    full_7;
reg         [   3: 0] how_many_ones;
wire        [   3: 0] one_count_minus_one;
wire        [   3: 0] one_count_plus_one;
wire                    p0_full_0;
wire                    p0_stage_0;
wire                    p1_full_1;
wire                    p1_stage_1;
wire                    p2_full_2;
wire                    p2_stage_2;
wire                    p3_full_3;
wire                    p3_stage_3;
wire                    p4_full_4;
wire                    p4_stage_4;
wire                    p5_full_5;
wire                    p5_stage_5;
wire                    p6_full_6;
wire                    p6_stage_6;
reg                     stage_0;
reg                     stage_1;
reg                     stage_2;
reg                     stage_3;
reg                     stage_4;
reg                     stage_5;
reg                     stage_6;
wire        [   3: 0] updated_one_count;
assign data_out = stage_0;
assign full = full_6;
assign empty = !full_0;
assign full_7 = 0;
//data_6, which is an e_mux
assign p6_stage_6 = ((full_7 & ~clear_fifo) == 0)? data_in :
  data_in;

//data_reg_6, which is an e_register
always @(posedge clk or negedge reset_n)
  begin
    if (reset_n == 0)
        stage_6 <= 0;
    else if (clear_fifo | sync_reset | read | (write & !full_6))
```

```
            if (sync_reset & full_6 & !((full_7 == 0) & read & write))
                stage_6 <= 0;
            else
              stage_6 <= p6_stage_6;
   end


//control_6, which is an e_mux
assign p6_full_6 = ((read & !write) == 0)? full_5 :
   0;

//control_reg_6, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            full_6 <= 0;
      else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
                full_6 <= 0;
            else
              full_6 <= p6_full_6;
   end


//data_5, which is an e_mux
assign p5_stage_5 = ((full_6 & ~clear_fifo) == 0)? data_in :
   stage_6;

//data_reg_5, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            stage_5 <= 0;
      else if (clear_fifo | sync_reset | read | (write & !full_5))
            if (sync_reset & full_5 & !((full_6 == 0) & read & write))
                stage_5 <= 0;
            else
              stage_5 <= p5_stage_5;
   end


//control_5, which is an e_mux
assign p5_full_5 = ((read & !write) == 0)? full_4 :
   full_6;

//control_reg_5, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            full_5 <= 0;
      else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
                full_5 <= 0;
            else
              full_5 <= p5_full_5;
   end
```

```
//data_4, which is an e_mux
assign p4_stage_4 = ((full_5 & ~clear_fifo) == 0)? data_in :
   stage_5;

//data_reg_4, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            stage_4 <= 0;
      else if (clear_fifo | sync_reset | read | (write & !full_4))
            if (sync_reset & full_4 & !((full_5 == 0) & read & write))
                  stage_4 <= 0;
            else
               stage_4 <= p4_stage_4;
   end


//control_4, which is an e_mux
assign p4_full_4 = ((read & !write) == 0)? full_3 :
   full_5;

//control_reg_4, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            full_4 <= 0;
      else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
                  full_4 <= 0;
            else
               full_4 <= p4_full_4;
   end


//data_3, which is an e_mux
assign p3_stage_3 = ((full_4 & ~clear_fifo) == 0)? data_in :
   stage_4;

//data_reg_3, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            stage_3 <= 0;
      else if (clear_fifo | sync_reset | read | (write & !full_3))
            if (sync_reset & full_3 & !((full_4 == 0) & read & write))
                  stage_3 <= 0;
            else
               stage_3 <= p3_stage_3;
   end


//control_3, which is an e_mux
assign p3_full_3 = ((read & !write) == 0)? full_2 :
   full_4;

//control_reg_3, which is an e_register
always @(posedge clk or negedge reset_n)
   begin
```

```
    if (reset_n == 0)
        full_3 <= 0;
    else if (clear_fifo | (read ^ write) | (write & !full_0))
        if (clear_fifo)
            full_3 <= 0;
        else
          full_3 <= p3_full_3;
  end


//data_2, which is an e_mux
assign p2_stage_2 = ((full_3 & ~clear_fifo) == 0)? data_in :
  stage_3;

//data_reg_2, which is an e_register
always @(posedge clk or negedge reset_n)
  begin
    if (reset_n == 0)
        stage_2 <= 0;
    else if (clear_fifo | sync_reset | read | (write & !full_2))
        if (sync_reset & full_2 & !((full_3 == 0) & read & write))
            stage_2 <= 0;
        else
          stage_2 <= p2_stage_2;
  end


//control_2, which is an e_mux
assign p2_full_2 = ((read & !write) == 0)? full_1 :
  full_3;

//control_reg_2, which is an e_register
always @(posedge clk or negedge reset_n)
  begin
    if (reset_n == 0)
        full_2 <= 0;
    else if (clear_fifo | (read ^ write) | (write & !full_0))
        if (clear_fifo)
            full_2 <= 0;
        else
          full_2 <= p2_full_2;
  end


//data_1, which is an e_mux
assign p1_stage_1 = ((full_2 & ~clear_fifo) == 0)? data_in :
  stage_2;

//data_reg_1, which is an e_register
always @(posedge clk or negedge reset_n)
  begin
    if (reset_n == 0)
        stage_1 <= 0;
    else if (clear_fifo | sync_reset | read | (write & !full_1))
        if (sync_reset & full_1 & !((full_2 == 0) & read & write))
            stage_1 <= 0;
        else
          stage_1 <= p1_stage_1;
```

```
    end


//control_1, which is an e_mux
assign p1_full_1 = ((read & !write) == 0)? full_0 :
    full_2;

//control_reg_1, which is an e_register
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            full_1 <= 0;
        else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo)
                full_1 <= 0;
            else
                full_1 <= p1_full_1;
    end


//data_0, which is an e_mux
assign p0_stage_0 = ((full_1 & ~clear_fifo) == 0)? data_in :
    stage_1;

//data_reg_0, which is an e_register
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            stage_0 <= 0;
        else if (clear_fifo | sync_reset | read | (write & !full_0))
            if (sync_reset & full_0 & !((full_1 == 0) & read & write))
                stage_0 <= 0;
            else
                stage_0 <= p0_stage_0;
    end


//control_0, which is an e_mux
assign p0_full_0 = ((read & !write) == 0)? 1 :
    full_1;

//control_reg_0, which is an e_register
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            full_0 <= 0;
        else if (clear_fifo | (read ^ write) | (write & !full_0))
            if (clear_fifo & ~write)
                full_0 <= 0;
            else
                full_0 <= p0_full_0;
    end


assign one_count_plus_one = how_many_ones + 1;
assign one_count_minus_one = how_many_ones - 1;
//updated_one_count, which is an e_mux
assign updated_one_count = ((((clear_fifo | sync_reset) & !write)))? 0 :
```

```
    ((((clear_fifo | sync_reset) & write)))? |data_in :
    ((read & (|data_in) & write & (|stage_0)))? how_many_ones :
    ((write & (|data_in)))? one_count_plus_one :
    ((read & (|stage_0)))? one_count_minus_one :
    how_many_ones;

  //counts how many ones in the data pipeline, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          how_many_ones <= 0;
      else if (clear_fifo | sync_reset | read | write)
          how_many_ones <= updated_one_count;
    end


  //this fifo contains ones in the data pipeline, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          fifo_contains_ones_n <= 1;
      else if (clear_fifo | sync_reset | read | write)
          fifo_contains_ones_n <= ~(|updated_one_count);
    end



endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module sdram_0_s1_arbitrator (
                               // inputs:
                                 clk,
                                 cpu_0_data_master_address_to_slave,
                                 cpu_0_data_master_byteenable,
                                 cpu_0_data_master_dbs_address,
                                 cpu_0_data_master_dbs_write_16,
                                 cpu_0_data_master_no_byte_enables_and_last_term,
                                 cpu_0_data_master_read,
                                 cpu_0_data_master_waitrequest,
                                 cpu_0_data_master_write,
                                 cpu_0_instruction_master_address_to_slave,
                                 cpu_0_instruction_master_dbs_address,
                                 cpu_0_instruction_master_read,
                                 reset_n,
                                 sdram_0_s1_readdata,
                                 sdram_0_s1_readdatavalid,
                                 sdram_0_s1_waitrequest,

                               // outputs:
```

```
                                        cpu_0_data_master_byteenable_sdram_0_s1,
                                        cpu_0_data_master_granted_sdram_0_s1,
                                        cpu_0_data_master_qualified_request_sdram_0_s1,
                                        cpu_0_data_master_read_data_valid_sdram_0_s1,

cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register,
                                        cpu_0_data_master_requests_sdram_0_s1,
                                        cpu_0_instruction_master_granted_sdram_0_s1,

cpu_0_instruction_master_qualified_request_sdram_0_s1,
                                        cpu_0_instruction_master_read_data_valid_sdram_0_s1,

cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register,
                                        cpu_0_instruction_master_requests_sdram_0_s1,
                                        d1_sdram_0_s1_end_xfer,
                                        sdram_0_s1_address,
                                        sdram_0_s1_byteenable_n,
                                        sdram_0_s1_chipselect,
                                        sdram_0_s1_read_n,
                                        sdram_0_s1_readdata_from_sa,
                                        sdram_0_s1_reset_n,
                                        sdram_0_s1_waitrequest_from_sa,
                                        sdram_0_s1_write_n,
                                        sdram_0_s1_writedata
                                    )
;

  output  [  1: 0] cpu_0_data_master_byteenable_sdram_0_s1;
  output           cpu_0_data_master_granted_sdram_0_s1;
  output           cpu_0_data_master_qualified_request_sdram_0_s1;
  output           cpu_0_data_master_read_data_valid_sdram_0_s1;
  output           cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register;
  output           cpu_0_data_master_requests_sdram_0_s1;
  output           cpu_0_instruction_master_granted_sdram_0_s1;
  output           cpu_0_instruction_master_qualified_request_sdram_0_s1;
  output           cpu_0_instruction_master_read_data_valid_sdram_0_s1;
  output           cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register;
  output           cpu_0_instruction_master_requests_sdram_0_s1;
  output           d1_sdram_0_s1_end_xfer;
  output  [ 21: 0] sdram_0_s1_address;
  output  [  1: 0] sdram_0_s1_byteenable_n;
  output           sdram_0_s1_chipselect;
  output           sdram_0_s1_read_n;
  output  [ 15: 0] sdram_0_s1_readdata_from_sa;
  output           sdram_0_s1_reset_n;
  output           sdram_0_s1_waitrequest_from_sa;
  output           sdram_0_s1_write_n;
  output  [ 15: 0] sdram_0_s1_writedata;
  input            clk;
  input   [ 24: 0] cpu_0_data_master_address_to_slave;
  input   [  3: 0] cpu_0_data_master_byteenable;
  input   [  1: 0] cpu_0_data_master_dbs_address;
  input   [ 15: 0] cpu_0_data_master_dbs_write_16;
  input            cpu_0_data_master_no_byte_enables_and_last_term;
  input            cpu_0_data_master_read;
  input            cpu_0_data_master_waitrequest;
  input            cpu_0_data_master_write;
  input   [ 24: 0] cpu_0_instruction_master_address_to_slave;
```

```
input     [  1: 0] cpu_0_instruction_master_dbs_address;
input              cpu_0_instruction_master_read;
input              reset_n;
input     [ 15: 0] sdram_0_s1_readdata;
input              sdram_0_s1_readdatavalid;
input              sdram_0_s1_waitrequest;

wire               cpu_0_data_master_arbiterlock;
wire               cpu_0_data_master_arbiterlock2;
wire      [  1: 0] cpu_0_data_master_byteenable_sdram_0_s1;
wire      [  1: 0] cpu_0_data_master_byteenable_sdram_0_s1_segment_0;
wire      [  1: 0] cpu_0_data_master_byteenable_sdram_0_s1_segment_1;
wire               cpu_0_data_master_continuerequest;
wire               cpu_0_data_master_granted_sdram_0_s1;
wire               cpu_0_data_master_qualified_request_sdram_0_s1;
wire               cpu_0_data_master_rdv_fifo_empty_sdram_0_s1;
wire               cpu_0_data_master_rdv_fifo_output_from_sdram_0_s1;
wire               cpu_0_data_master_read_data_valid_sdram_0_s1;
wire               cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register;
wire               cpu_0_data_master_requests_sdram_0_s1;
wire               cpu_0_data_master_saved_grant_sdram_0_s1;
wire               cpu_0_instruction_master_arbiterlock;
wire               cpu_0_instruction_master_arbiterlock2;
wire               cpu_0_instruction_master_continuerequest;
wire               cpu_0_instruction_master_granted_sdram_0_s1;
wire               cpu_0_instruction_master_qualified_request_sdram_0_s1;
wire               cpu_0_instruction_master_rdv_fifo_empty_sdram_0_s1;
wire               cpu_0_instruction_master_rdv_fifo_output_from_sdram_0_s1;
wire               cpu_0_instruction_master_read_data_valid_sdram_0_s1;
wire               cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register;
wire               cpu_0_instruction_master_requests_sdram_0_s1;
wire               cpu_0_instruction_master_saved_grant_sdram_0_s1;
reg                d1_reasons_to_wait;
reg                d1_sdram_0_s1_end_xfer;
reg                enable_nonzero_assertions;
wire               end_xfer_arb_share_counter_term_sdram_0_s1;
wire               in_a_read_cycle;
wire               in_a_write_cycle;
reg                last_cycle_cpu_0_data_master_granted_slave_sdram_0_s1;
reg                last_cycle_cpu_0_instruction_master_granted_slave_sdram_0_s1;
wire      [ 21: 0] sdram_0_s1_address;
wire               sdram_0_s1_allgrants;
wire               sdram_0_s1_allow_new_arb_cycle;
wire               sdram_0_s1_any_bursting_master_saved_grant;
wire               sdram_0_s1_any_continuerequest;
reg       [  1: 0] sdram_0_s1_arb_addend;
wire               sdram_0_s1_arb_counter_enable;
reg       [  1: 0] sdram_0_s1_arb_share_counter;
wire      [  1: 0] sdram_0_s1_arb_share_counter_next_value;
wire      [  1: 0] sdram_0_s1_arb_share_set_values;
wire      [  1: 0] sdram_0_s1_arb_winner;
wire               sdram_0_s1_arbitration_holdoff_internal;
wire               sdram_0_s1_beginbursttransfer_internal;
wire               sdram_0_s1_begins_xfer;
wire      [  1: 0] sdram_0_s1_byteenable_n;
wire               sdram_0_s1_chipselect;
wire      [  3: 0] sdram_0_s1_chosen_master_double_vector;
wire      [  1: 0] sdram_0_s1_chosen_master_rot_left;
```

```
  wire                    sdram_0_s1_end_xfer;
  wire                    sdram_0_s1_firsttransfer;
  wire      [  1: 0] sdram_0_s1_grant_vector;
  wire                    sdram_0_s1_in_a_read_cycle;
  wire                    sdram_0_s1_in_a_write_cycle;
  wire      [  1: 0] sdram_0_s1_master_qreq_vector;
  wire                    sdram_0_s1_move_on_to_next_transaction;
  wire                    sdram_0_s1_non_bursting_master_requests;
  wire                    sdram_0_s1_read_n;
  wire      [ 15: 0] sdram_0_s1_readdata_from_sa;
  wire                    sdram_0_s1_readdatavalid_from_sa;
  reg                     sdram_0_s1_reg_firsttransfer;
  wire                    sdram_0_s1_reset_n;
  reg       [  1: 0] sdram_0_s1_saved_chosen_master_vector;
  reg                     sdram_0_s1_slavearbiterlockenable;
  wire                    sdram_0_s1_slavearbiterlockenable2;
  wire                    sdram_0_s1_unreg_firsttransfer;
  wire                    sdram_0_s1_waitrequest_from_sa;
  wire                    sdram_0_s1_waits_for_read;
  wire                    sdram_0_s1_waits_for_write;
  wire                    sdram_0_s1_write_n;
  wire      [ 15: 0] sdram_0_s1_writedata;
  wire      [ 24: 0] shifted_address_to_sdram_0_s1_from_cpu_0_data_master;
  wire      [ 24: 0] shifted_address_to_sdram_0_s1_from_cpu_0_instruction_master;
  wire                    wait_for_sdram_0_s1_counter;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_reasons_to_wait <= 0;
      else
        d1_reasons_to_wait <= ~sdram_0_s1_end_xfer;
    end


  assign sdram_0_s1_begins_xfer = ~d1_reasons_to_wait &
((cpu_0_data_master_qualified_request_sdram_0_s1 |
cpu_0_instruction_master_qualified_request_sdram_0_s1));
  //assign sdram_0_s1_readdata_from_sa = sdram_0_s1_readdata so that symbol knows where
to group signals which may go to master only, which is an e_assign
  assign sdram_0_s1_readdata_from_sa = sdram_0_s1_readdata;

  assign cpu_0_data_master_requests_sdram_0_s1 =
({cpu_0_data_master_address_to_slave[24 : 23] , 23'b0} == 25'h800000) &
(cpu_0_data_master_read | cpu_0_data_master_write);
  //assign sdram_0_s1_waitrequest_from_sa = sdram_0_s1_waitrequest so that symbol knows
where to group signals which may go to master only, which is an e_assign
  assign sdram_0_s1_waitrequest_from_sa = sdram_0_s1_waitrequest;

  //assign sdram_0_s1_readdatavalid_from_sa = sdram_0_s1_readdatavalid so that symbol
knows where to group signals which may go to master only, which is an e_assign
  assign sdram_0_s1_readdatavalid_from_sa = sdram_0_s1_readdatavalid;

  //sdram_0_s1_arb_share_counter set values, which is an e_mux
  assign sdram_0_s1_arb_share_set_values = (cpu_0_data_master_granted_sdram_0_s1)? 2 :
    (cpu_0_instruction_master_granted_sdram_0_s1)? 2 :
    (cpu_0_data_master_granted_sdram_0_s1)? 2 :
    (cpu_0_instruction_master_granted_sdram_0_s1)? 2 :
    1;
```

```
  //sdram_0_s1_non_bursting_master_requests mux, which is an e_mux
  assign sdram_0_s1_non_bursting_master_requests =
cpu_0_data_master_requests_sdram_0_s1 |
    cpu_0_instruction_master_requests_sdram_0_s1 |
    cpu_0_data_master_requests_sdram_0_s1 |
    cpu_0_instruction_master_requests_sdram_0_s1;

  //sdram_0_s1_any_bursting_master_saved_grant mux, which is an e_mux
  assign sdram_0_s1_any_bursting_master_saved_grant = 0;

  //sdram_0_s1_arb_share_counter_next_value assignment, which is an e_assign
  assign sdram_0_s1_arb_share_counter_next_value = sdram_0_s1_firsttransfer ?
(sdram_0_s1_arb_share_set_values - 1) : |sdram_0_s1_arb_share_counter ?
(sdram_0_s1_arb_share_counter - 1) : 0;

  //sdram_0_s1_allgrants all slave grants, which is an e_mux
  assign sdram_0_s1_allgrants = (|sdram_0_s1_grant_vector) |
    (|sdram_0_s1_grant_vector) |
    (|sdram_0_s1_grant_vector) |
    (|sdram_0_s1_grant_vector);

  //sdram_0_s1_end_xfer assignment, which is an e_assign
  assign sdram_0_s1_end_xfer = ~(sdram_0_s1_waits_for_read | sdram_0_s1_waits_for_write);

  //end_xfer_arb_share_counter_term_sdram_0_s1 arb share counter enable term, which is an
e_assign
  assign end_xfer_arb_share_counter_term_sdram_0_s1 = sdram_0_s1_end_xfer &
(~sdram_0_s1_any_bursting_master_saved_grant | in_a_read_cycle | in_a_write_cycle);

  //sdram_0_s1_arb_share_counter arbitration counter enable, which is an e_assign
  assign sdram_0_s1_arb_counter_enable = (end_xfer_arb_share_counter_term_sdram_0_s1 &
sdram_0_s1_allgrants) | (end_xfer_arb_share_counter_term_sdram_0_s1 &
~sdram_0_s1_non_bursting_master_requests);

  //sdram_0_s1_arb_share_counter counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          sdram_0_s1_arb_share_counter <= 0;
      else if (sdram_0_s1_arb_counter_enable)
          sdram_0_s1_arb_share_counter <= sdram_0_s1_arb_share_counter_next_value;
    end


  //sdram_0_s1_slavearbiterlockenable slave enables arbiterlock, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          sdram_0_s1_slavearbiterlockenable <= 0;
      else if ((|sdram_0_s1_master_qreq_vector &
end_xfer_arb_share_counter_term_sdram_0_s1) |
(end_xfer_arb_share_counter_term_sdram_0_s1 &
~sdram_0_s1_non_bursting_master_requests))
          sdram_0_s1_slavearbiterlockenable <=
|sdram_0_s1_arb_share_counter_next_value;
    end
```

```
  //cpu_0/data_master sdram_0/s1 arbiterlock, which is an e_assign
  assign cpu_0_data_master_arbiterlock = sdram_0_s1_slavearbiterlockenable & cpu_0_data_master_continuerequest;

  //sdram_0_s1_slavearbiterlockenable2 slave enables arbiterlock2, which is an e_assign
  assign sdram_0_s1_slavearbiterlockenable2 = |sdram_0_s1_arb_share_counter_next_value;

  //cpu_0/data_master sdram_0/s1 arbiterlock2, which is an e_assign
  assign cpu_0_data_master_arbiterlock2 = sdram_0_s1_slavearbiterlockenable2 & cpu_0_data_master_continuerequest;

  //cpu_0/instruction_master sdram_0/s1 arbiterlock, which is an e_assign
  assign cpu_0_instruction_master_arbiterlock = sdram_0_s1_slavearbiterlockenable & cpu_0_instruction_master_continuerequest;

  //cpu_0/instruction_master sdram_0/s1 arbiterlock2, which is an e_assign
  assign cpu_0_instruction_master_arbiterlock2 = sdram_0_s1_slavearbiterlockenable2 & cpu_0_instruction_master_continuerequest;

  //cpu_0/instruction_master granted sdram_0/s1 last time, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          last_cycle_cpu_0_instruction_master_granted_slave_sdram_0_s1 <= 0;
      else
        last_cycle_cpu_0_instruction_master_granted_slave_sdram_0_s1 <= cpu_0_instruction_master_saved_grant_sdram_0_s1 ? 1 : (sdram_0_s1_arbitration_holdoff_internal | ~cpu_0_instruction_master_requests_sdram_0_s1) ? 0 : last_cycle_cpu_0_instruction_master_granted_slave_sdram_0_s1;
    end


  //cpu_0_instruction_master_continuerequest continued request, which is an e_mux
  assign cpu_0_instruction_master_continuerequest = last_cycle_cpu_0_instruction_master_granted_slave_sdram_0_s1 & cpu_0_instruction_master_requests_sdram_0_s1;

  //sdram_0_s1_any_continuerequest at least one master continues requesting, which is an e_mux
  assign sdram_0_s1_any_continuerequest = cpu_0_instruction_master_continuerequest |
    cpu_0_data_master_continuerequest;

  assign cpu_0_data_master_qualified_request_sdram_0_s1 = cpu_0_data_master_requests_sdram_0_s1 & ~((cpu_0_data_master_read & (~cpu_0_data_master_waitrequest | (|cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register))) | ((~cpu_0_data_master_waitrequest | cpu_0_data_master_no_byte_enables_and_last_term | !cpu_0_data_master_byteenable_sdram_0_s1) & cpu_0_data_master_write) | cpu_0_instruction_master_arbiterlock);
  //unique name for sdram_0_s1_move_on_to_next_transaction, which is an e_assign
  assign sdram_0_s1_move_on_to_next_transaction = sdram_0_s1_readdatavalid_from_sa;

  //rdv_fifo_for_cpu_0_data_master_to_sdram_0_s1, which is an e_fifo_with_registered_outputs
  rdv_fifo_for_cpu_0_data_master_to_sdram_0_s1_module rdv_fifo_for_cpu_0_data_master_to_sdram_0_s1
    (
```

```
        .clear_fifo               (1'b0),
        .clk                      (clk),
        .data_in                  (cpu_0_data_master_granted_sdram_0_s1),
        .data_out                 (cpu_0_data_master_rdv_fifo_output_from_sdram_0_s1),
        .empty                    (),
        .fifo_contains_ones_n (cpu_0_data_master_rdv_fifo_empty_sdram_0_s1),
        .full                     (),
        .read                     (sdram_0_s1_move_on_to_next_transaction),
        .reset_n                  (reset_n),
        .sync_reset               (1'b0),
        .write                    (in_a_read_cycle & ~sdram_0_s1_waits_for_read)
      );

  assign cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register =
~cpu_0_data_master_rdv_fifo_empty_sdram_0_s1;
  //local readdatavalid cpu_0_data_master_read_data_valid_sdram_0_s1, which is an e_mux
  assign cpu_0_data_master_read_data_valid_sdram_0_s1 =
(sdram_0_s1_readdatavalid_from_sa & cpu_0_data_master_rdv_fifo_output_from_sdram_0_s1)
& ~ cpu_0_data_master_rdv_fifo_empty_sdram_0_s1;

  //sdram_0_s1_writedata mux, which is an e_mux
  assign sdram_0_s1_writedata = cpu_0_data_master_dbs_write_16;

  assign cpu_0_instruction_master_requests_sdram_0_s1 =
(({cpu_0_instruction_master_address_to_slave[24 : 23] , 23'b0} == 25'h800000) &
(cpu_0_instruction_master_read)) & cpu_0_instruction_master_read;
  //cpu_0/data_master granted sdram_0/s1 last time, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          last_cycle_cpu_0_data_master_granted_slave_sdram_0_s1 <= 0;
      else
        last_cycle_cpu_0_data_master_granted_slave_sdram_0_s1 <=
cpu_0_data_master_saved_grant_sdram_0_s1 ? 1 : (sdram_0_s1_arbitration_holdoff_internal |
~cpu_0_data_master_requests_sdram_0_s1) ? 0 :
last_cycle_cpu_0_data_master_granted_slave_sdram_0_s1;
    end


  //cpu_0_data_master_continuerequest continued request, which is an e_mux
  assign cpu_0_data_master_continuerequest =
last_cycle_cpu_0_data_master_granted_slave_sdram_0_s1 &
cpu_0_data_master_requests_sdram_0_s1;

  assign cpu_0_instruction_master_qualified_request_sdram_0_s1 =
cpu_0_instruction_master_requests_sdram_0_s1 & ~((cpu_0_instruction_master_read &
((|cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register))) |
cpu_0_data_master_arbiterlock);
  //rdv_fifo_for_cpu_0_instruction_master_to_sdram_0_s1, which is an
e_fifo_with_registered_outputs
  rdv_fifo_for_cpu_0_instruction_master_to_sdram_0_s1_module
rdv_fifo_for_cpu_0_instruction_master_to_sdram_0_s1
    (
      .clear_fifo               (1'b0),
      .clk                      (clk),
      .data_in                  (cpu_0_instruction_master_granted_sdram_0_s1),
      .data_out
(cpu_0_instruction_master_rdv_fifo_output_from_sdram_0_s1),
```

```
        .empty                  (),
        .fifo_contains_ones_n (cpu_0_instruction_master_rdv_fifo_empty_sdram_0_s1),
        .full                   (),
        .read                   (sdram_0_s1_move_on_to_next_transaction),
        .reset_n                (reset_n),
        .sync_reset             (1'b0),
        .write                  (in_a_read_cycle & ~sdram_0_s1_waits_for_read)
      );

  assign cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register = ~cpu_0_instruction_master_rdv_fifo_empty_sdram_0_s1;
  //local readdatavalid cpu_0_instruction_master_read_data_valid_sdram_0_s1, which is an e_mux
  assign cpu_0_instruction_master_read_data_valid_sdram_0_s1 = (sdram_0_s1_readdatavalid_from_sa & cpu_0_instruction_master_rdv_fifo_output_from_sdram_0_s1) & ~ cpu_0_instruction_master_rdv_fifo_empty_sdram_0_s1;

  //allow new arb cycle for sdram_0/s1, which is an e_assign
  assign sdram_0_s1_allow_new_arb_cycle = ~cpu_0_data_master_arbiterlock & ~cpu_0_instruction_master_arbiterlock;

  //cpu_0/instruction_master assignment into master qualified-requests vector for sdram_0/s1, which is an e_assign
  assign sdram_0_s1_master_qreq_vector[0] = cpu_0_instruction_master_qualified_request_sdram_0_s1;

  //cpu_0/instruction_master grant sdram_0/s1, which is an e_assign
  assign cpu_0_instruction_master_granted_sdram_0_s1 = sdram_0_s1_grant_vector[0];

  //cpu_0/instruction_master saved-grant sdram_0/s1, which is an e_assign
  assign cpu_0_instruction_master_saved_grant_sdram_0_s1 = sdram_0_s1_arb_winner[0] && cpu_0_instruction_master_requests_sdram_0_s1;

  //cpu_0/data_master assignment into master qualified-requests vector for sdram_0/s1, which is an e_assign
  assign sdram_0_s1_master_qreq_vector[1] = cpu_0_data_master_qualified_request_sdram_0_s1;

  //cpu_0/data_master grant sdram_0/s1, which is an e_assign
  assign cpu_0_data_master_granted_sdram_0_s1 = sdram_0_s1_grant_vector[1];

  //cpu_0/data_master saved-grant sdram_0/s1, which is an e_assign
  assign cpu_0_data_master_saved_grant_sdram_0_s1 = sdram_0_s1_arb_winner[1] && cpu_0_data_master_requests_sdram_0_s1;

  //sdram_0/s1 chosen-master double-vector, which is an e_assign
  assign sdram_0_s1_chosen_master_double_vector = {sdram_0_s1_master_qreq_vector, sdram_0_s1_master_qreq_vector} & ({~sdram_0_s1_master_qreq_vector, ~sdram_0_s1_master_qreq_vector} + sdram_0_s1_arb_addend);

  //stable onehot encoding of arb winner
  assign sdram_0_s1_arb_winner = (sdram_0_s1_allow_new_arb_cycle & | sdram_0_s1_grant_vector) ? sdram_0_s1_grant_vector : sdram_0_s1_saved_chosen_master_vector;

  //saved sdram_0_s1_grant_vector, which is an e_register
  always @(posedge clk or negedge reset_n)
```

```
      begin
        if (reset_n == 0)
            sdram_0_s1_saved_chosen_master_vector <= 0;
        else if (sdram_0_s1_allow_new_arb_cycle)
            sdram_0_s1_saved_chosen_master_vector <= |sdram_0_s1_grant_vector ?
sdram_0_s1_grant_vector : sdram_0_s1_saved_chosen_master_vector;
      end


   //onehot encoding of chosen master
   assign sdram_0_s1_grant_vector = {(sdram_0_s1_chosen_master_double_vector[1] |
sdram_0_s1_chosen_master_double_vector[3]),
      (sdram_0_s1_chosen_master_double_vector[0] |
sdram_0_s1_chosen_master_double_vector[2])};

   //sdram_0/s1 chosen master rotated left, which is an e_assign
   assign sdram_0_s1_chosen_master_rot_left = (sdram_0_s1_arb_winner << 1) ?
(sdram_0_s1_arb_winner << 1) : 1;

   //sdram_0/s1's addend for next-master-grant
   always @(posedge clk or negedge reset_n)
      begin
        if (reset_n == 0)
            sdram_0_s1_arb_addend <= 1;
        else if (|sdram_0_s1_grant_vector)
            sdram_0_s1_arb_addend <= sdram_0_s1_end_xfer?
sdram_0_s1_chosen_master_rot_left : sdram_0_s1_grant_vector;
      end


   //sdram_0_s1_reset_n assignment, which is an e_assign
   assign sdram_0_s1_reset_n = reset_n;

   assign sdram_0_s1_chipselect = cpu_0_data_master_granted_sdram_0_s1 |
cpu_0_instruction_master_granted_sdram_0_s1;
   //sdram_0_s1_firsttransfer first transaction, which is an e_assign
   assign sdram_0_s1_firsttransfer = sdram_0_s1_begins_xfer ? sdram_0_s1_unreg_firsttransfer :
sdram_0_s1_reg_firsttransfer;

   //sdram_0_s1_unreg_firsttransfer first transaction, which is an e_assign
   assign sdram_0_s1_unreg_firsttransfer = ~(sdram_0_s1_slavearbiterlockenable &
sdram_0_s1_any_continuerequest);

   //sdram_0_s1_reg_firsttransfer first transaction, which is an e_register
   always @(posedge clk or negedge reset_n)
      begin
        if (reset_n == 0)
            sdram_0_s1_reg_firsttransfer <= 1'b1;
        else if (sdram_0_s1_begins_xfer)
            sdram_0_s1_reg_firsttransfer <= sdram_0_s1_unreg_firsttransfer;
      end


   //sdram_0_s1_beginbursttransfer_internal begin burst transfer, which is an e_assign
   assign sdram_0_s1_beginbursttransfer_internal = sdram_0_s1_begins_xfer;

   //sdram_0_s1_arbitration_holdoff_internal arbitration_holdoff, which is an e_assign
   assign sdram_0_s1_arbitration_holdoff_internal = sdram_0_s1_begins_xfer &
```

```
sdram_0_s1_firsttransfer;

  //~sdram_0_s1_read_n assignment, which is an e_mux
  assign sdram_0_s1_read_n = ~((cpu_0_data_master_granted_sdram_0_s1 & cpu_0_data_master_read) | (cpu_0_instruction_master_granted_sdram_0_s1 & cpu_0_instruction_master_read));

  //~sdram_0_s1_write_n assignment, which is an e_mux
  assign sdram_0_s1_write_n = ~(cpu_0_data_master_granted_sdram_0_s1 & cpu_0_data_master_write);

  assign shifted_address_to_sdram_0_s1_from_cpu_0_data_master = {cpu_0_data_master_address_to_slave >> 2,
    cpu_0_data_master_dbs_address[1],
    {1 {1'b0}}};

  //sdram_0_s1_address mux, which is an e_mux
  assign sdram_0_s1_address = (cpu_0_data_master_granted_sdram_0_s1)? (shifted_address_to_sdram_0_s1_from_cpu_0_data_master >> 1) :
    (shifted_address_to_sdram_0_s1_from_cpu_0_instruction_master >> 1);

  assign shifted_address_to_sdram_0_s1_from_cpu_0_instruction_master = {cpu_0_instruction_master_address_to_slave >> 2,
    cpu_0_instruction_master_dbs_address[1],
    {1 {1'b0}}};

  //d1_sdram_0_s1_end_xfer register, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_sdram_0_s1_end_xfer <= 1;
      else
        d1_sdram_0_s1_end_xfer <= sdram_0_s1_end_xfer;
    end


  //sdram_0_s1_waits_for_read in a cycle, which is an e_mux
  assign sdram_0_s1_waits_for_read = sdram_0_s1_in_a_read_cycle & sdram_0_s1_waitrequest_from_sa;

  //sdram_0_s1_in_a_read_cycle assignment, which is an e_assign
  assign sdram_0_s1_in_a_read_cycle = (cpu_0_data_master_granted_sdram_0_s1 & cpu_0_data_master_read) | (cpu_0_instruction_master_granted_sdram_0_s1 & cpu_0_instruction_master_read);

  //in_a_read_cycle assignment, which is an e_mux
  assign in_a_read_cycle = sdram_0_s1_in_a_read_cycle;

  //sdram_0_s1_waits_for_write in a cycle, which is an e_mux
  assign sdram_0_s1_waits_for_write = sdram_0_s1_in_a_write_cycle & sdram_0_s1_waitrequest_from_sa;

  //sdram_0_s1_in_a_write_cycle assignment, which is an e_assign
  assign sdram_0_s1_in_a_write_cycle = cpu_0_data_master_granted_sdram_0_s1 & cpu_0_data_master_write;

  //in_a_write_cycle assignment, which is an e_mux
  assign in_a_write_cycle = sdram_0_s1_in_a_write_cycle;
```

```
  assign wait_for_sdram_0_s1_counter = 0;
  //~sdram_0_s1_byteenable_n byte enable port mux, which is an e_mux
  assign sdram_0_s1_byteenable_n = ~((cpu_0_data_master_granted_sdram_0_s1)?
cpu_0_data_master_byteenable_sdram_0_s1 :
    -1);

  assign {cpu_0_data_master_byteenable_sdram_0_s1_segment_1,
cpu_0_data_master_byteenable_sdram_0_s1_segment_0} = cpu_0_data_master_byteenable;
  assign cpu_0_data_master_byteenable_sdram_0_s1 = ((cpu_0_data_master_dbs_address[1]
== 0))? cpu_0_data_master_byteenable_sdram_0_s1_segment_0 :
    cpu_0_data_master_byteenable_sdram_0_s1_segment_1;


//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
  //sdram_0/s1 enable non-zero assertions, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          enable_nonzero_assertions <= 0;
      else
        enable_nonzero_assertions <= 1'b1;
    end


  //grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if (cpu_0_data_master_granted_sdram_0_s1 +
cpu_0_instruction_master_granted_sdram_0_s1 > 1)
        begin
          $write("%0d ns: > 1 of grant signals are active simultaneously", $time);
          $stop;
        end
    end


  //saved_grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if (cpu_0_data_master_saved_grant_sdram_0_s1 +
cpu_0_instruction_master_saved_grant_sdram_0_s1 > 1)
        begin
          $write("%0d ns: > 1 of saved_grant signals are active simultaneously", $time);
          $stop;
        end
    end



//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule
```

```
// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module sram_mux_0_avalon_slave_0_arbitrator (
                                              // inputs:
                                               clk,
                                               cpu_0_data_master_address_to_slave,
                                               cpu_0_data_master_byteenable,
                                               cpu_0_data_master_dbs_address,
                                               cpu_0_data_master_dbs_write_16,

cpu_0_data_master_no_byte_enables_and_last_term,
                                               cpu_0_data_master_read,
                                               cpu_0_data_master_waitrequest,
                                               cpu_0_data_master_write,

cpu_0_instruction_master_address_to_slave,

cpu_0_instruction_master_dbs_address,
                                               cpu_0_instruction_master_read,
                                               reset_n,

sram_mux_0_avalon_slave_0_readdata,

                                              // outputs:

cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0,

cpu_0_data_master_granted_sram_mux_0_avalon_slave_0,

cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0,

cpu_0_data_master_requests_sram_mux_0_avalon_slave_0,

cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0,

cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0,

cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0,

cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0,

d1_sram_mux_0_avalon_slave_0_end_xfer,
                                               sram_mux_0_avalon_slave_0_address,
                                               sram_mux_0_avalon_slave_0_read,

sram_mux_0_avalon_slave_0_readdata_from_sa,
                                               sram_mux_0_avalon_slave_0_write,

sram_mux_0_avalon_slave_0_writedata
                                             )
```

```
;

   output    [  1: 0] cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0;
   output             cpu_0_data_master_granted_sram_mux_0_avalon_slave_0;
   output             cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0;
   output             cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0;
   output             cpu_0_data_master_requests_sram_mux_0_avalon_slave_0;
   output             cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0;
   output
cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0;
   output
cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0;
   output             cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0;
   output             d1_sram_mux_0_avalon_slave_0_end_xfer;
   output   [ 17: 0] sram_mux_0_avalon_slave_0_address;
   output             sram_mux_0_avalon_slave_0_read;
   output   [ 15: 0] sram_mux_0_avalon_slave_0_readdata_from_sa;
   output             sram_mux_0_avalon_slave_0_write;
   output   [ 15: 0] sram_mux_0_avalon_slave_0_writedata;
   input              clk;
   input    [ 24: 0] cpu_0_data_master_address_to_slave;
   input    [  3: 0] cpu_0_data_master_byteenable;
   input    [  1: 0] cpu_0_data_master_dbs_address;
   input    [ 15: 0] cpu_0_data_master_dbs_write_16;
   input             cpu_0_data_master_no_byte_enables_and_last_term;
   input             cpu_0_data_master_read;
   input             cpu_0_data_master_waitrequest;
   input             cpu_0_data_master_write;
   input    [ 24: 0] cpu_0_instruction_master_address_to_slave;
   input    [  1: 0] cpu_0_instruction_master_dbs_address;
   input             cpu_0_instruction_master_read;
   input             reset_n;
   input    [ 15: 0] sram_mux_0_avalon_slave_0_readdata;

   wire              cpu_0_data_master_arbiterlock;
   wire              cpu_0_data_master_arbiterlock2;
   wire     [  1: 0] cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0;
   wire     [  1: 0] cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0_segment_0;
   wire     [  1: 0] cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0_segment_1;
   wire              cpu_0_data_master_continuerequest;
   wire              cpu_0_data_master_granted_sram_mux_0_avalon_slave_0;
   wire              cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0;
   wire              cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0;
   wire              cpu_0_data_master_requests_sram_mux_0_avalon_slave_0;
   wire              cpu_0_data_master_saved_grant_sram_mux_0_avalon_slave_0;
   wire              cpu_0_instruction_master_arbiterlock;
   wire              cpu_0_instruction_master_arbiterlock2;
   wire              cpu_0_instruction_master_continuerequest;
   wire              cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0;
   wire
cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0;
   wire
cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0;
   wire              cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0;
   wire              cpu_0_instruction_master_saved_grant_sram_mux_0_avalon_slave_0;
   reg               d1_reasons_to_wait;
   reg               d1_sram_mux_0_avalon_slave_0_end_xfer;
   reg               enable_nonzero_assertions;
```

```
  wire        end_xfer_arb_share_counter_term_sram_mux_0_avalon_slave_0;
  wire        in_a_read_cycle;
  wire        in_a_write_cycle;
  reg
last_cycle_cpu_0_data_master_granted_slave_sram_mux_0_avalon_slave_0;
  reg
last_cycle_cpu_0_instruction_master_granted_slave_sram_mux_0_avalon_slave_0;
  wire                                                [ 24: 0]
shifted_address_to_sram_mux_0_avalon_slave_0_from_cpu_0_data_master;
  wire                                                [ 24: 0]
shifted_address_to_sram_mux_0_avalon_slave_0_from_cpu_0_instruction_master;
  wire      [ 17: 0] sram_mux_0_avalon_slave_0_address;
  wire        sram_mux_0_avalon_slave_0_allgrants;
  wire        sram_mux_0_avalon_slave_0_allow_new_arb_cycle;
  wire        sram_mux_0_avalon_slave_0_any_bursting_master_saved_grant;
  wire        sram_mux_0_avalon_slave_0_any_continuerequest;
  reg       [  1: 0] sram_mux_0_avalon_slave_0_arb_addend;
  wire        sram_mux_0_avalon_slave_0_arb_counter_enable;
  reg       [  1: 0] sram_mux_0_avalon_slave_0_arb_share_counter;
  wire      [  1: 0] sram_mux_0_avalon_slave_0_arb_share_counter_next_value;
  wire      [  1: 0] sram_mux_0_avalon_slave_0_arb_share_set_values;
  wire      [  1: 0] sram_mux_0_avalon_slave_0_arb_winner;
  wire        sram_mux_0_avalon_slave_0_arbitration_holdoff_internal;
  wire        sram_mux_0_avalon_slave_0_beginbursttransfer_internal;
  wire        sram_mux_0_avalon_slave_0_begins_xfer;
  wire      [  3: 0] sram_mux_0_avalon_slave_0_chosen_master_double_vector;
  wire      [  1: 0] sram_mux_0_avalon_slave_0_chosen_master_rot_left;
  wire        sram_mux_0_avalon_slave_0_end_xfer;
  wire        sram_mux_0_avalon_slave_0_firsttransfer;
  wire      [  1: 0] sram_mux_0_avalon_slave_0_grant_vector;
  wire        sram_mux_0_avalon_slave_0_in_a_read_cycle;
  wire        sram_mux_0_avalon_slave_0_in_a_write_cycle;
  wire      [  1: 0] sram_mux_0_avalon_slave_0_master_qreq_vector;
  wire        sram_mux_0_avalon_slave_0_non_bursting_master_requests;
  wire        sram_mux_0_avalon_slave_0_read;
  wire      [ 15: 0] sram_mux_0_avalon_slave_0_readdata_from_sa;
  reg         sram_mux_0_avalon_slave_0_reg_firsttransfer;
  reg       [  1: 0] sram_mux_0_avalon_slave_0_saved_chosen_master_vector;
  reg         sram_mux_0_avalon_slave_0_slavearbiterlockenable;
  wire        sram_mux_0_avalon_slave_0_slavearbiterlockenable2;
  wire        sram_mux_0_avalon_slave_0_unreg_firsttransfer;
  wire        sram_mux_0_avalon_slave_0_waits_for_read;
  wire        sram_mux_0_avalon_slave_0_waits_for_write;
  wire        sram_mux_0_avalon_slave_0_write;
  wire      [ 15: 0] sram_mux_0_avalon_slave_0_writedata;
  wire        wait_for_sram_mux_0_avalon_slave_0_counter;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_reasons_to_wait <= 0;
      else
        d1_reasons_to_wait <= ~sram_mux_0_avalon_slave_0_end_xfer;
    end


  assign      sram_mux_0_avalon_slave_0_begins_xfer      =      ~d1_reasons_to_wait      &
((cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0                          |
cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0));
```

```
  //assign sram_mux_0_avalon_slave_0_readdata_from_sa = sram_mux_0_avalon_slave_0_readdata so that symbol knows where to group signals which may go to master only, which is an e_assign
  assign sram_mux_0_avalon_slave_0_readdata_from_sa = sram_mux_0_avalon_slave_0_readdata;

  assign cpu_0_data_master_requests_sram_mux_0_avalon_slave_0 = ({cpu_0_data_master_address_to_slave[24 : 19] , 19'b0} == 25'h100000) & (cpu_0_data_master_read | cpu_0_data_master_write);
  //sram_mux_0_avalon_slave_0_arb_share_counter set values, which is an e_mux
  assign sram_mux_0_avalon_slave_0_arb_share_set_values = (cpu_0_data_master_granted_sram_mux_0_avalon_slave_0)? 2 :
    (cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0)? 2 :
    (cpu_0_data_master_granted_sram_mux_0_avalon_slave_0)? 2 :
    (cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0)? 2 :
    1;

  //sram_mux_0_avalon_slave_0_non_bursting_master_requests mux, which is an e_mux
  assign sram_mux_0_avalon_slave_0_non_bursting_master_requests = cpu_0_data_master_requests_sram_mux_0_avalon_slave_0 |
    cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0 |
    cpu_0_data_master_requests_sram_mux_0_avalon_slave_0 |
    cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0;

  //sram_mux_0_avalon_slave_0_any_bursting_master_saved_grant mux, which is an e_mux
  assign sram_mux_0_avalon_slave_0_any_bursting_master_saved_grant = 0;

  //sram_mux_0_avalon_slave_0_arb_share_counter_next_value assignment, which is an e_assign
  assign sram_mux_0_avalon_slave_0_arb_share_counter_next_value = sram_mux_0_avalon_slave_0_firsttransfer ? (sram_mux_0_avalon_slave_0_arb_share_set_values - 1) : |sram_mux_0_avalon_slave_0_arb_share_counter ? (sram_mux_0_avalon_slave_0_arb_share_counter - 1) : 0;

  //sram_mux_0_avalon_slave_0_allgrants all slave grants, which is an e_mux
  assign sram_mux_0_avalon_slave_0_allgrants = (|sram_mux_0_avalon_slave_0_grant_vector) |
    (|sram_mux_0_avalon_slave_0_grant_vector) |
    (|sram_mux_0_avalon_slave_0_grant_vector) |
    (|sram_mux_0_avalon_slave_0_grant_vector);

  //sram_mux_0_avalon_slave_0_end_xfer assignment, which is an e_assign
  assign sram_mux_0_avalon_slave_0_end_xfer = ~(sram_mux_0_avalon_slave_0_waits_for_read | sram_mux_0_avalon_slave_0_waits_for_write);

  //end_xfer_arb_share_counter_term_sram_mux_0_avalon_slave_0 arb share counter enable term, which is an e_assign
  assign end_xfer_arb_share_counter_term_sram_mux_0_avalon_slave_0 = sram_mux_0_avalon_slave_0_end_xfer & (~sram_mux_0_avalon_slave_0_any_bursting_master_saved_grant | in_a_read_cycle | in_a_write_cycle);

  //sram_mux_0_avalon_slave_0_arb_share_counter arbitration counter enable, which is an e_assign
  assign sram_mux_0_avalon_slave_0_arb_counter_enable = (end_xfer_arb_share_counter_term_sram_mux_0_avalon_slave_0 &
```

```
sram_mux_0_avalon_slave_0_allgrants)                                                    |
(end_xfer_arb_share_counter_term_sram_mux_0_avalon_slave_0                              &
~sram_mux_0_avalon_slave_0_non_bursting_master_requests);

  //sram_mux_0_avalon_slave_0_arb_share_counter counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          sram_mux_0_avalon_slave_0_arb_share_counter <= 0;
      else if (sram_mux_0_avalon_slave_0_arb_counter_enable)
          sram_mux_0_avalon_slave_0_arb_share_counter                                   <=
sram_mux_0_avalon_slave_0_arb_share_counter_next_value;
    end


  //sram_mux_0_avalon_slave_0_slavearbiterlockenable slave enables arbiterlock, which is an
e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          sram_mux_0_avalon_slave_0_slavearbiterlockenable <= 0;
      else if ((|sram_mux_0_avalon_slave_0_master_qreq_vector                           &
end_xfer_arb_share_counter_term_sram_mux_0_avalon_slave_0)                              |
(end_xfer_arb_share_counter_term_sram_mux_0_avalon_slave_0                              &
~sram_mux_0_avalon_slave_0_non_bursting_master_requests))
          sram_mux_0_avalon_slave_0_slavearbiterlockenable                              <=
|sram_mux_0_avalon_slave_0_arb_share_counter_next_value;
    end


  //cpu_0/data_master sram_mux_0/avalon_slave_0 arbiterlock, which is an e_assign
  assign cpu_0_data_master_arbiterlock = sram_mux_0_avalon_slave_0_slavearbiterlockenable
& cpu_0_data_master_continuerequest;

  //sram_mux_0_avalon_slave_0_slavearbiterlockenable2 slave enables arbiterlock2, which is
an e_assign
  assign                    sram_mux_0_avalon_slave_0_slavearbiterlockenable2            =
|sram_mux_0_avalon_slave_0_arb_share_counter_next_value;

  //cpu_0/data_master sram_mux_0/avalon_slave_0 arbiterlock2, which is an e_assign
  assign                              cpu_0_data_master_arbiterlock2                    =
sram_mux_0_avalon_slave_0_slavearbiterlockenable2 & cpu_0_data_master_continuerequest;

  //cpu_0/instruction_master sram_mux_0/avalon_slave_0 arbiterlock, which is an e_assign
  assign                            cpu_0_instruction_master_arbiterlock              =
sram_mux_0_avalon_slave_0_slavearbiterlockenable                                        &
cpu_0_instruction_master_continuerequest;

  //cpu_0/instruction_master sram_mux_0/avalon_slave_0 arbiterlock2, which is an e_assign
  assign                            cpu_0_instruction_master_arbiterlock2             =
sram_mux_0_avalon_slave_0_slavearbiterlockenable2                                       &
cpu_0_instruction_master_continuerequest;

  //cpu_0/instruction_master granted sram_mux_0/avalon_slave_0 last time, which is an
e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
```

```
            last_cycle_cpu_0_instruction_master_granted_slave_sram_mux_0_avalon_slave_0
<= 0;
        else
          last_cycle_cpu_0_instruction_master_granted_slave_sram_mux_0_avalon_slave_0 <=
cpu_0_instruction_master_saved_grant_sram_mux_0_avalon_slave_0 ? 1 :
(sram_mux_0_avalon_slave_0_arbitration_holdoff_internal |
~cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0) ? 0 :
last_cycle_cpu_0_instruction_master_granted_slave_sram_mux_0_avalon_slave_0;
      end


  //cpu_0_instruction_master_continuerequest continued request, which is an e_mux
  assign cpu_0_instruction_master_continuerequest =
last_cycle_cpu_0_instruction_master_granted_slave_sram_mux_0_avalon_slave_0 &
cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0;

  //sram_mux_0_avalon_slave_0_any_continuerequest at least one master continues
requesting, which is an e_mux
  assign sram_mux_0_avalon_slave_0_any_continuerequest =
cpu_0_instruction_master_continuerequest |
      cpu_0_data_master_continuerequest;

  assign cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0 =
cpu_0_data_master_requests_sram_mux_0_avalon_slave_0 &
~(((~cpu_0_data_master_waitrequest | cpu_0_data_master_no_byte_enables_and_last_term
| !cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0) & cpu_0_data_master_write)
| cpu_0_instruction_master_arbiterlock);
  //sram_mux_0_avalon_slave_0_writedata mux, which is an e_mux
  assign sram_mux_0_avalon_slave_0_writedata = cpu_0_data_master_dbs_write_16;

  assign cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0 =
(({cpu_0_instruction_master_address_to_slave[24 : 19] , 19'b0} == 25'h100000) &
(cpu_0_instruction_master_read)) & cpu_0_instruction_master_read;
  //cpu_0/data_master granted sram_mux_0/avalon_slave_0 last time, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          last_cycle_cpu_0_data_master_granted_slave_sram_mux_0_avalon_slave_0 <= 0;
      else
        last_cycle_cpu_0_data_master_granted_slave_sram_mux_0_avalon_slave_0 <=
cpu_0_data_master_saved_grant_sram_mux_0_avalon_slave_0 ? 1 :
(sram_mux_0_avalon_slave_0_arbitration_holdoff_internal |
~cpu_0_data_master_requests_sram_mux_0_avalon_slave_0) ? 0 :
last_cycle_cpu_0_data_master_granted_slave_sram_mux_0_avalon_slave_0;
      end


  //cpu_0_data_master_continuerequest continued request, which is an e_mux
  assign cpu_0_data_master_continuerequest =
last_cycle_cpu_0_data_master_granted_slave_sram_mux_0_avalon_slave_0 &
cpu_0_data_master_requests_sram_mux_0_avalon_slave_0;

  assign cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0 =
cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0 &
~(cpu_0_data_master_arbiterlock);
  //allow new arb cycle for sram_mux_0/avalon_slave_0, which is an e_assign
  assign sram_mux_0_avalon_slave_0_allow_new_arb_cycle = ~cpu_0_data_master_arbiterlock
& ~cpu_0_instruction_master_arbiterlock;
```

```
  //cpu_0/instruction_master assignment into master qualified-requests vector for sram_mux_0/avalon_slave_0, which is an e_assign
  assign sram_mux_0_avalon_slave_0_master_qreq_vector[0] = cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0;

  //cpu_0/instruction_master grant sram_mux_0/avalon_slave_0, which is an e_assign
  assign cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0 = sram_mux_0_avalon_slave_0_grant_vector[0];

  //cpu_0/instruction_master saved-grant sram_mux_0/avalon_slave_0, which is an e_assign
  assign cpu_0_instruction_master_saved_grant_sram_mux_0_avalon_slave_0 = sram_mux_0_avalon_slave_0_arb_winner[0] && cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0;

  //cpu_0/data_master assignment into master qualified-requests vector for sram_mux_0/avalon_slave_0, which is an e_assign
  assign sram_mux_0_avalon_slave_0_master_qreq_vector[1] = cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0;

  //cpu_0/data_master grant sram_mux_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_granted_sram_mux_0_avalon_slave_0 = sram_mux_0_avalon_slave_0_grant_vector[1];

  //cpu_0/data_master saved-grant sram_mux_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_saved_grant_sram_mux_0_avalon_slave_0 = sram_mux_0_avalon_slave_0_arb_winner[1] && cpu_0_data_master_requests_sram_mux_0_avalon_slave_0;

  //sram_mux_0/avalon_slave_0 chosen-master double-vector, which is an e_assign
  assign sram_mux_0_avalon_slave_0_chosen_master_double_vector = {sram_mux_0_avalon_slave_0_master_qreq_vector,
sram_mux_0_avalon_slave_0_master_qreq_vector} & ({~sram_mux_0_avalon_slave_0_master_qreq_vector,
~sram_mux_0_avalon_slave_0_master_qreq_vector} + sram_mux_0_avalon_slave_0_arb_addend);

  //stable onehot encoding of arb winner
  assign sram_mux_0_avalon_slave_0_arb_winner = (sram_mux_0_avalon_slave_0_allow_new_arb_cycle & | sram_mux_0_avalon_slave_0_grant_vector) ? sram_mux_0_avalon_slave_0_grant_vector : sram_mux_0_avalon_slave_0_saved_chosen_master_vector;

  //saved sram_mux_0_avalon_slave_0_grant_vector, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          sram_mux_0_avalon_slave_0_saved_chosen_master_vector <= 0;
      else if (sram_mux_0_avalon_slave_0_allow_new_arb_cycle)
          sram_mux_0_avalon_slave_0_saved_chosen_master_vector <= |sram_mux_0_avalon_slave_0_grant_vector ? sram_mux_0_avalon_slave_0_grant_vector : sram_mux_0_avalon_slave_0_saved_chosen_master_vector;
    end


  //onehot encoding of chosen master
  assign sram_mux_0_avalon_slave_0_grant_vector = {(sram_mux_0_avalon_slave_0_chosen_master_double_vector[1] |
```

```
sram_mux_0_avalon_slave_0_chosen_master_double_vector[3]),
      (sram_mux_0_avalon_slave_0_chosen_master_double_vector[0]                    |
sram_mux_0_avalon_slave_0_chosen_master_double_vector[2])};

   //sram_mux_0/avalon_slave_0 chosen master rotated left, which is an e_assign
   assign                  sram_mux_0_avalon_slave_0_chosen_master_rot_left                =
(sram_mux_0_avalon_slave_0_arb_winner << 1) ? (sram_mux_0_avalon_slave_0_arb_winner <<
1) : 1;

   //sram_mux_0/avalon_slave_0's addend for next-master-grant
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               sram_mux_0_avalon_slave_0_arb_addend <= 1;
         else if (|sram_mux_0_avalon_slave_0_grant_vector)
               sram_mux_0_avalon_slave_0_arb_addend                                    <=
sram_mux_0_avalon_slave_0_end_xfer? sram_mux_0_avalon_slave_0_chosen_master_rot_left :
sram_mux_0_avalon_slave_0_grant_vector;
      end


   //sram_mux_0_avalon_slave_0_firsttransfer first transaction, which is an e_assign
   assign sram_mux_0_avalon_slave_0_firsttransfer = sram_mux_0_avalon_slave_0_begins_xfer ?
sram_mux_0_avalon_slave_0_unreg_firsttransfer                                          :
sram_mux_0_avalon_slave_0_reg_firsttransfer;

   //sram_mux_0_avalon_slave_0_unreg_firsttransfer first transaction, which is an e_assign
   assign                  sram_mux_0_avalon_slave_0_unreg_firsttransfer                =
~(sram_mux_0_avalon_slave_0_slavearbiterlockenable                                     &
sram_mux_0_avalon_slave_0_any_continuerequest);

   //sram_mux_0_avalon_slave_0_reg_firsttransfer first transaction, which is an e_register
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               sram_mux_0_avalon_slave_0_reg_firsttransfer <= 1'b1;
         else if (sram_mux_0_avalon_slave_0_begins_xfer)
               sram_mux_0_avalon_slave_0_reg_firsttransfer                             <=
sram_mux_0_avalon_slave_0_unreg_firsttransfer;
      end


   //sram_mux_0_avalon_slave_0_beginbursttransfer_internal begin burst transfer, which is an
e_assign
   assign                  sram_mux_0_avalon_slave_0_beginbursttransfer_internal          =
sram_mux_0_avalon_slave_0_begins_xfer;

   //sram_mux_0_avalon_slave_0_arbitration_holdoff_internal arbitration_holdoff, which is an
e_assign
   assign                  sram_mux_0_avalon_slave_0_arbitration_holdoff_internal         =
sram_mux_0_avalon_slave_0_begins_xfer & sram_mux_0_avalon_slave_0_firsttransfer;

   //sram_mux_0_avalon_slave_0_read assignment, which is an e_mux
   assign                  sram_mux_0_avalon_slave_0_read                                 =
(cpu_0_data_master_granted_sram_mux_0_avalon_slave_0   &   cpu_0_data_master_read)   |
(cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0                            &
cpu_0_instruction_master_read);
```

```
  //sram_mux_0_avalon_slave_0_write assignment, which is an e_mux
  assign sram_mux_0_avalon_slave_0_write = cpu_0_data_master_granted_sram_mux_0_avalon_slave_0 & cpu_0_data_master_write;

  assign shifted_address_to_sram_mux_0_avalon_slave_0_from_cpu_0_data_master = {cpu_0_data_master_address_to_slave >> 2,
      cpu_0_data_master_dbs_address[1],
      {1 {1'b0}}};

  //sram_mux_0_avalon_slave_0_address mux, which is an e_mux
  assign sram_mux_0_avalon_slave_0_address = (cpu_0_data_master_granted_sram_mux_0_avalon_slave_0)?
(shifted_address_to_sram_mux_0_avalon_slave_0_from_cpu_0_data_master >> 1) :
      (shifted_address_to_sram_mux_0_avalon_slave_0_from_cpu_0_instruction_master >> 1);

  assign shifted_address_to_sram_mux_0_avalon_slave_0_from_cpu_0_instruction_master = {cpu_0_instruction_master_address_to_slave >> 2,
      cpu_0_instruction_master_dbs_address[1],
      {1 {1'b0}}};

  //d1_sram_mux_0_avalon_slave_0_end_xfer register, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_sram_mux_0_avalon_slave_0_end_xfer <= 1;
      else
        d1_sram_mux_0_avalon_slave_0_end_xfer <= sram_mux_0_avalon_slave_0_end_xfer;
    end


  //sram_mux_0_avalon_slave_0_waits_for_read in a cycle, which is an e_mux
  assign sram_mux_0_avalon_slave_0_waits_for_read = sram_mux_0_avalon_slave_0_in_a_read_cycle & sram_mux_0_avalon_slave_0_begins_xfer;

  //sram_mux_0_avalon_slave_0_in_a_read_cycle assignment, which is an e_assign
  assign sram_mux_0_avalon_slave_0_in_a_read_cycle = (cpu_0_data_master_granted_sram_mux_0_avalon_slave_0 & cpu_0_data_master_read) |
(cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0 & cpu_0_instruction_master_read);

  //in_a_read_cycle assignment, which is an e_mux
  assign in_a_read_cycle = sram_mux_0_avalon_slave_0_in_a_read_cycle;

  //sram_mux_0_avalon_slave_0_waits_for_write in a cycle, which is an e_mux
  assign sram_mux_0_avalon_slave_0_waits_for_write = sram_mux_0_avalon_slave_0_in_a_write_cycle & 0;

  //sram_mux_0_avalon_slave_0_in_a_write_cycle assignment, which is an e_assign
  assign sram_mux_0_avalon_slave_0_in_a_write_cycle = cpu_0_data_master_granted_sram_mux_0_avalon_slave_0 & cpu_0_data_master_write;

  //in_a_write_cycle assignment, which is an e_mux
  assign in_a_write_cycle = sram_mux_0_avalon_slave_0_in_a_write_cycle;

  assign wait_for_sram_mux_0_avalon_slave_0_counter = 0;
  assign {cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0_segment_1,
cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0_segment_0} = cpu_0_data_master_byteenable;
```

```
  assign cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0 =
((cpu_0_data_master_dbs_address[1] == 0))?
cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0_segment_0 :
    cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0_segment_1;


//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
  //sram_mux_0/avalon_slave_0 enable non-zero assertions, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          enable_nonzero_assertions <= 0;
      else
        enable_nonzero_assertions <= 1'b1;
    end


  //grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if (cpu_0_data_master_granted_sram_mux_0_avalon_slave_0 +
cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0 > 1)
        begin
          $write("%0d ns: > 1 of grant signals are active simultaneously", $time);
          $stop;
        end
    end


  //saved_grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if (cpu_0_data_master_saved_grant_sram_mux_0_avalon_slave_0 +
cpu_0_instruction_master_saved_grant_sram_mux_0_avalon_slave_0 > 1)
        begin
          $write("%0d ns: > 1 of saved_grant signals are active simultaneously", $time);
          $stop;
        end
    end



//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030
```

```
module xycoord_interface_0_avalon_slave_0_arbitrator (
                                                        // inputs:
                                                          clk,

cpu_0_data_master_address_to_slave,

cpu_0_data_master_byteenable,

cpu_0_data_master_dbs_address,

cpu_0_data_master_dbs_write_16,

cpu_0_data_master_no_byte_enables_and_last_term,
                                                          cpu_0_data_master_read,

cpu_0_data_master_waitrequest,
                                                          cpu_0_data_master_write,

cpu_0_instruction_master_address_to_slave,

cpu_0_instruction_master_dbs_address,

cpu_0_instruction_master_read,
                                                          reset_n,

xycoord_interface_0_avalon_slave_0_readdata,

                                                        // outputs:

cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0,

cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0,

cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0,

cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0,

cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0,

cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0,

cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0,

cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0,

cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0,

d1_xycoord_interface_0_avalon_slave_0_end_xfer,

xycoord_interface_0_avalon_slave_0_address,

xycoord_interface_0_avalon_slave_0_chipselect,

xycoord_interface_0_avalon_slave_0_read,

xycoord_interface_0_avalon_slave_0_readdata_from_sa,

xycoord_interface_0_avalon_slave_0_reset_n,
```

```
xycoord_interface_0_avalon_slave_0_write,

xycoord_interface_0_avalon_slave_0_writedata
                                                                )
;

   output    [    1: 0] cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0;
   output                cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0;
   output
cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0;
   output
cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0;
   output                cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0;
   output
cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0;
   output
cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0;
   output
cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0;
   output
cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0;
   output                d1_xycoord_interface_0_avalon_slave_0_end_xfer;
   output    [    4: 0] xycoord_interface_0_avalon_slave_0_address;
   output                xycoord_interface_0_avalon_slave_0_chipselect;
   output                xycoord_interface_0_avalon_slave_0_read;
   output    [ 15: 0] xycoord_interface_0_avalon_slave_0_readdata_from_sa;
   output                xycoord_interface_0_avalon_slave_0_reset_n;
   output                xycoord_interface_0_avalon_slave_0_write;
   output    [ 15: 0] xycoord_interface_0_avalon_slave_0_writedata;
   input                 clk;
   input     [ 24: 0] cpu_0_data_master_address_to_slave;
   input     [    3: 0] cpu_0_data_master_byteenable;
   input     [    1: 0] cpu_0_data_master_dbs_address;
   input     [ 15: 0] cpu_0_data_master_dbs_write_16;
   input                 cpu_0_data_master_no_byte_enables_and_last_term;
   input                 cpu_0_data_master_read;
   input                 cpu_0_data_master_waitrequest;
   input                 cpu_0_data_master_write;
   input     [ 24: 0] cpu_0_instruction_master_address_to_slave;
   input     [    1: 0] cpu_0_instruction_master_dbs_address;
   input                 cpu_0_instruction_master_read;
   input                 reset_n;
   input     [ 15: 0] xycoord_interface_0_avalon_slave_0_readdata;

   wire                  cpu_0_data_master_arbiterlock;
   wire                  cpu_0_data_master_arbiterlock2;
   wire      [    1: 0] cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0;
   wire                                                  [                    1:                0]
cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0_segment_0;
   wire                                                  [                    1:                0]
cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0_segment_1;
   wire                  cpu_0_data_master_continuerequest;
   wire                  cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0;
   wire
cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0;
   wire
cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0;
```

```
  wire            cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0;
  wire            cpu_0_data_master_saved_grant_xycoord_interface_0_avalon_slave_0;
  wire            cpu_0_instruction_master_arbiterlock;
  wire            cpu_0_instruction_master_arbiterlock2;
  wire            cpu_0_instruction_master_continuerequest;
  wire
cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0;
  wire
cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0;
  wire
cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0;
  wire
cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0;
  wire
cpu_0_instruction_master_saved_grant_xycoord_interface_0_avalon_slave_0;
  reg             d1_reasons_to_wait;
  reg             d1_xycoord_interface_0_avalon_slave_0_end_xfer;
  reg             enable_nonzero_assertions;
  wire
end_xfer_arb_share_counter_term_xycoord_interface_0_avalon_slave_0;
  wire            in_a_read_cycle;
  wire            in_a_write_cycle;
  reg
last_cycle_cpu_0_data_master_granted_slave_xycoord_interface_0_avalon_slave_0;
  reg
last_cycle_cpu_0_instruction_master_granted_slave_xycoord_interface_0_avalon_slave_0;
  wire                                                [ 24: 0]
shifted_address_to_xycoord_interface_0_avalon_slave_0_from_cpu_0_data_master;
  wire                                                [ 24: 0]
shifted_address_to_xycoord_interface_0_avalon_slave_0_from_cpu_0_instruction_master;
  wire            wait_for_xycoord_interface_0_avalon_slave_0_counter;
  wire    [  4: 0] xycoord_interface_0_avalon_slave_0_address;
  wire            xycoord_interface_0_avalon_slave_0_allgrants;
  wire            xycoord_interface_0_avalon_slave_0_allow_new_arb_cycle;
  wire            xycoord_interface_0_avalon_slave_0_any_bursting_master_saved_grant;
  wire            xycoord_interface_0_avalon_slave_0_any_continuerequest;
  reg     [  1: 0] xycoord_interface_0_avalon_slave_0_arb_addend;
  wire            xycoord_interface_0_avalon_slave_0_arb_counter_enable;
  reg     [  1: 0] xycoord_interface_0_avalon_slave_0_arb_share_counter;
  wire    [  1: 0] xycoord_interface_0_avalon_slave_0_arb_share_counter_next_value;
  wire    [  1: 0] xycoord_interface_0_avalon_slave_0_arb_share_set_values;
  wire    [  1: 0] xycoord_interface_0_avalon_slave_0_arb_winner;
  wire            xycoord_interface_0_avalon_slave_0_arbitration_holdoff_internal;
  wire            xycoord_interface_0_avalon_slave_0_beginbursttransfer_internal;
  wire            xycoord_interface_0_avalon_slave_0_begins_xfer;
  wire            xycoord_interface_0_avalon_slave_0_chipselect;
  wire    [  3: 0] xycoord_interface_0_avalon_slave_0_chosen_master_double_vector;
  wire    [  1: 0] xycoord_interface_0_avalon_slave_0_chosen_master_rot_left;
  wire            xycoord_interface_0_avalon_slave_0_end_xfer;
  wire            xycoord_interface_0_avalon_slave_0_firsttransfer;
  wire    [  1: 0] xycoord_interface_0_avalon_slave_0_grant_vector;
  wire            xycoord_interface_0_avalon_slave_0_in_a_read_cycle;
  wire            xycoord_interface_0_avalon_slave_0_in_a_write_cycle;
  wire    [  1: 0] xycoord_interface_0_avalon_slave_0_master_qreq_vector;
  wire            xycoord_interface_0_avalon_slave_0_non_bursting_master_requests;
  wire            xycoord_interface_0_avalon_slave_0_read;
  wire    [ 15: 0] xycoord_interface_0_avalon_slave_0_readdata_from_sa;
  reg             xycoord_interface_0_avalon_slave_0_reg_firsttransfer;
```

```
  wire                    xycoord_interface_0_avalon_slave_0_reset_n;
  reg       [  1: 0] xycoord_interface_0_avalon_slave_0_saved_chosen_master_vector;
  reg                     xycoord_interface_0_avalon_slave_0_slavearbiterlockenable;
  wire                    xycoord_interface_0_avalon_slave_0_slavearbiterlockenable2;
  wire                    xycoord_interface_0_avalon_slave_0_unreg_firsttransfer;
  wire                    xycoord_interface_0_avalon_slave_0_waits_for_read;
  wire                    xycoord_interface_0_avalon_slave_0_waits_for_write;
  wire                    xycoord_interface_0_avalon_slave_0_write;
  wire      [ 15: 0] xycoord_interface_0_avalon_slave_0_writedata;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_reasons_to_wait <= 0;
      else
        d1_reasons_to_wait <= ~xycoord_interface_0_avalon_slave_0_end_xfer;
    end


  assign xycoord_interface_0_avalon_slave_0_begins_xfer = ~d1_reasons_to_wait &
((cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0 |
cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0));
  //assign xycoord_interface_0_avalon_slave_0_readdata_from_sa =
xycoord_interface_0_avalon_slave_0_readdata so that symbol knows where to group signals
which may go to master only, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_readdata_from_sa =
xycoord_interface_0_avalon_slave_0_readdata;

  assign cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0 =
({cpu_0_data_master_address_to_slave[24 : 6] , 6'b0} == 25'h80) & (cpu_0_data_master_read |
cpu_0_data_master_write);
  //xycoord_interface_0_avalon_slave_0_arb_share_counter set values, which is an e_mux
  assign xycoord_interface_0_avalon_slave_0_arb_share_set_values =
(cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0)? 2 :
    (cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0)? 2 :
    (cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0)? 2 :
    (cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0)? 2 :
    1;

  //xycoord_interface_0_avalon_slave_0_non_bursting_master_requests mux, which is an
e_mux
  assign xycoord_interface_0_avalon_slave_0_non_bursting_master_requests =
cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0 |
    cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0 |
    cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0 |
    cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0;

  //xycoord_interface_0_avalon_slave_0_any_bursting_master_saved_grant mux, which is an
e_mux
  assign xycoord_interface_0_avalon_slave_0_any_bursting_master_saved_grant = 0;

  //xycoord_interface_0_avalon_slave_0_arb_share_counter_next_value assignment, which is
an e_assign
  assign xycoord_interface_0_avalon_slave_0_arb_share_counter_next_value =
xycoord_interface_0_avalon_slave_0_firsttransfer ?
(xycoord_interface_0_avalon_slave_0_arb_share_set_values - 1) :
|xycoord_interface_0_avalon_slave_0_arb_share_counter ?
(xycoord_interface_0_avalon_slave_0_arb_share_counter - 1) : 0;
```

```
  //xycoord_interface_0_avalon_slave_0_allgrants all slave grants, which is an e_mux
  assign xycoord_interface_0_avalon_slave_0_allgrants =
(|xycoord_interface_0_avalon_slave_0_grant_vector) |
    (|xycoord_interface_0_avalon_slave_0_grant_vector) |
    (|xycoord_interface_0_avalon_slave_0_grant_vector) |
    (|xycoord_interface_0_avalon_slave_0_grant_vector);

  //xycoord_interface_0_avalon_slave_0_end_xfer assignment, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_end_xfer =
~(xycoord_interface_0_avalon_slave_0_waits_for_read |
xycoord_interface_0_avalon_slave_0_waits_for_write);

  //end_xfer_arb_share_counter_term_xycoord_interface_0_avalon_slave_0 arb share counter
enable term, which is an e_assign
  assign end_xfer_arb_share_counter_term_xycoord_interface_0_avalon_slave_0 =
xycoord_interface_0_avalon_slave_0_end_xfer &
(~xycoord_interface_0_avalon_slave_0_any_bursting_master_saved_grant | in_a_read_cycle |
in_a_write_cycle);

  //xycoord_interface_0_avalon_slave_0_arb_share_counter arbitration counter enable, which
is an e_assign
  assign xycoord_interface_0_avalon_slave_0_arb_counter_enable =
(end_xfer_arb_share_counter_term_xycoord_interface_0_avalon_slave_0 &
xycoord_interface_0_avalon_slave_0_allgrants) |
(end_xfer_arb_share_counter_term_xycoord_interface_0_avalon_slave_0 &
~xycoord_interface_0_avalon_slave_0_non_bursting_master_requests);

  //xycoord_interface_0_avalon_slave_0_arb_share_counter counter, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          xycoord_interface_0_avalon_slave_0_arb_share_counter <= 0;
      else if (xycoord_interface_0_avalon_slave_0_arb_counter_enable)
          xycoord_interface_0_avalon_slave_0_arb_share_counter <=
xycoord_interface_0_avalon_slave_0_arb_share_counter_next_value;
    end


  //xycoord_interface_0_avalon_slave_0_slavearbiterlockenable slave enables arbiterlock,
which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          xycoord_interface_0_avalon_slave_0_slavearbiterlockenable <= 0;
      else if ((|xycoord_interface_0_avalon_slave_0_master_qreq_vector &
end_xfer_arb_share_counter_term_xycoord_interface_0_avalon_slave_0) |
(end_xfer_arb_share_counter_term_xycoord_interface_0_avalon_slave_0 &
~xycoord_interface_0_avalon_slave_0_non_bursting_master_requests))
          xycoord_interface_0_avalon_slave_0_slavearbiterlockenable <=
|xycoord_interface_0_avalon_slave_0_arb_share_counter_next_value;
    end


  //cpu_0/data_master xycoord_interface_0/avalon_slave_0 arbiterlock, which is an e_assign
  assign cpu_0_data_master_arbiterlock =
xycoord_interface_0_avalon_slave_0_slavearbiterlockenable &
cpu_0_data_master_continuerequest;
```

```
  //xycoord_interface_0_avalon_slave_0_slavearbiterlockenable2 slave enables arbiterlock2, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_slavearbiterlockenable2 = |xycoord_interface_0_avalon_slave_0_arb_share_counter_next_value;

  //cpu_0/data_master xycoord_interface_0/avalon_slave_0 arbiterlock2, which is an e_assign
  assign cpu_0_data_master_arbiterlock2 = xycoord_interface_0_avalon_slave_0_slavearbiterlockenable2 & cpu_0_data_master_continuerequest;

  //cpu_0/instruction_master xycoord_interface_0/avalon_slave_0 arbiterlock, which is an e_assign
  assign cpu_0_instruction_master_arbiterlock = xycoord_interface_0_avalon_slave_0_slavearbiterlockenable & cpu_0_instruction_master_continuerequest;

  //cpu_0/instruction_master xycoord_interface_0/avalon_slave_0 arbiterlock2, which is an e_assign
  assign cpu_0_instruction_master_arbiterlock2 = xycoord_interface_0_avalon_slave_0_slavearbiterlockenable2 & cpu_0_instruction_master_continuerequest;

  //cpu_0/instruction_master granted xycoord_interface_0/avalon_slave_0 last time, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)

last_cycle_cpu_0_instruction_master_granted_slave_xycoord_interface_0_avalon_slave_0 <= 0;
      else

last_cycle_cpu_0_instruction_master_granted_slave_xycoord_interface_0_avalon_slave_0 <= cpu_0_instruction_master_saved_grant_xycoord_interface_0_avalon_slave_0 ? 1 : (xycoord_interface_0_avalon_slave_0_arbitration_holdoff_internal | ~cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0) ? 0 : last_cycle_cpu_0_instruction_master_granted_slave_xycoord_interface_0_avalon_slave_0;
    end


  //cpu_0_instruction_master_continuerequest continued request, which is an e_mux
  assign cpu_0_instruction_master_continuerequest = last_cycle_cpu_0_instruction_master_granted_slave_xycoord_interface_0_avalon_slave_0 & cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0;

  //xycoord_interface_0_avalon_slave_0_any_continuerequest at least one master continues requesting, which is an e_mux
  assign xycoord_interface_0_avalon_slave_0_any_continuerequest = cpu_0_instruction_master_continuerequest |
    cpu_0_data_master_continuerequest;

  assign cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0 = cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0 & ~(((~cpu_0_data_master_waitrequest | cpu_0_data_master_no_byte_enables_and_last_term | !cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0) & cpu_0_data_master_write) | cpu_0_instruction_master_arbiterlock);
  //xycoord_interface_0_avalon_slave_0_writedata mux, which is an e_mux
  assign xycoord_interface_0_avalon_slave_0_writedata = cpu_0_data_master_dbs_write_16;
```

```
  assign cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0 = (({cpu_0_instruction_master_address_to_slave[24 : 6] , 6'b0} == 25'h80) & (cpu_0_instruction_master_read)) & cpu_0_instruction_master_read;
  //cpu_0/data_master granted xycoord_interface_0/avalon_slave_0 last time, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          last_cycle_cpu_0_data_master_granted_slave_xycoord_interface_0_avalon_slave_0 <= 0;
      else
        last_cycle_cpu_0_data_master_granted_slave_xycoord_interface_0_avalon_slave_0 <= cpu_0_data_master_saved_grant_xycoord_interface_0_avalon_slave_0 ? 1 : (xycoord_interface_0_avalon_slave_0_arbitration_holdoff_internal | ~cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0) ? 0 : last_cycle_cpu_0_data_master_granted_slave_xycoord_interface_0_avalon_slave_0;
    end


  //cpu_0_data_master_continuerequest continued request, which is an e_mux
  assign cpu_0_data_master_continuerequest = last_cycle_cpu_0_data_master_granted_slave_xycoord_interface_0_avalon_slave_0 & cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0;

  assign cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0 = cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0 & ~(cpu_0_data_master_arbiterlock);
  //allow new arb cycle for xycoord_interface_0/avalon_slave_0, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_allow_new_arb_cycle = ~cpu_0_data_master_arbiterlock & ~cpu_0_instruction_master_arbiterlock;

  //cpu_0/instruction_master assignment into master qualified-requests vector for xycoord_interface_0/avalon_slave_0, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_master_qreq_vector[0] = cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0;

  //cpu_0/instruction_master grant xycoord_interface_0/avalon_slave_0, which is an e_assign
  assign cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0 = xycoord_interface_0_avalon_slave_0_grant_vector[0];

  //cpu_0/instruction_master saved-grant xycoord_interface_0/avalon_slave_0, which is an e_assign
  assign cpu_0_instruction_master_saved_grant_xycoord_interface_0_avalon_slave_0 = xycoord_interface_0_avalon_slave_0_arb_winner[0] && cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0;

  //cpu_0/data_master assignment into master qualified-requests vector for xycoord_interface_0/avalon_slave_0, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_master_qreq_vector[1] = cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0;

  //cpu_0/data_master grant xycoord_interface_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0 = xycoord_interface_0_avalon_slave_0_grant_vector[1];

  //cpu_0/data_master saved-grant xycoord_interface_0/avalon_slave_0, which is an e_assign
  assign cpu_0_data_master_saved_grant_xycoord_interface_0_avalon_slave_0 = xycoord_interface_0_avalon_slave_0_arb_winner[1] &&
```

```
cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0;

  //xycoord_interface_0/avalon_slave_0 chosen-master double-vector, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_chosen_master_double_vector =
{xycoord_interface_0_avalon_slave_0_master_qreq_vector,
xycoord_interface_0_avalon_slave_0_master_qreq_vector} &
({~xycoord_interface_0_avalon_slave_0_master_qreq_vector,
~xycoord_interface_0_avalon_slave_0_master_qreq_vector} +
xycoord_interface_0_avalon_slave_0_arb_addend);

  //stable onehot encoding of arb winner
  assign xycoord_interface_0_avalon_slave_0_arb_winner =
(xycoord_interface_0_avalon_slave_0_allow_new_arb_cycle & |
xycoord_interface_0_avalon_slave_0_grant_vector) ?
xycoord_interface_0_avalon_slave_0_grant_vector :
xycoord_interface_0_avalon_slave_0_saved_chosen_master_vector;

  //saved xycoord_interface_0_avalon_slave_0_grant_vector, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          xycoord_interface_0_avalon_slave_0_saved_chosen_master_vector <= 0;
      else if (xycoord_interface_0_avalon_slave_0_allow_new_arb_cycle)
          xycoord_interface_0_avalon_slave_0_saved_chosen_master_vector <=
|xycoord_interface_0_avalon_slave_0_grant_vector ?
xycoord_interface_0_avalon_slave_0_grant_vector :
xycoord_interface_0_avalon_slave_0_saved_chosen_master_vector;
    end


  //onehot encoding of chosen master
  assign xycoord_interface_0_avalon_slave_0_grant_vector =
{(xycoord_interface_0_avalon_slave_0_chosen_master_double_vector[1] |
xycoord_interface_0_avalon_slave_0_chosen_master_double_vector[3]),
    (xycoord_interface_0_avalon_slave_0_chosen_master_double_vector[0] |
xycoord_interface_0_avalon_slave_0_chosen_master_double_vector[2])};

  //xycoord_interface_0/avalon_slave_0 chosen master rotated left, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_chosen_master_rot_left =
(xycoord_interface_0_avalon_slave_0_arb_winner << 1) ?
(xycoord_interface_0_avalon_slave_0_arb_winner << 1) : 1;

  //xycoord_interface_0/avalon_slave_0's addend for next-master-grant
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          xycoord_interface_0_avalon_slave_0_arb_addend <= 1;
      else if (|xycoord_interface_0_avalon_slave_0_grant_vector)
          xycoord_interface_0_avalon_slave_0_arb_addend <=
xycoord_interface_0_avalon_slave_0_end_xfer?
xycoord_interface_0_avalon_slave_0_chosen_master_rot_left :
xycoord_interface_0_avalon_slave_0_grant_vector;
    end


  //xycoord_interface_0_avalon_slave_0_reset_n assignment, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_reset_n = reset_n;
```

```
  assign xycoord_interface_0_avalon_slave_0_chipselect =
cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0 |
cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0;
  //xycoord_interface_0_avalon_slave_0_firsttransfer first transaction, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_firsttransfer =
xycoord_interface_0_avalon_slave_0_begins_xfer ?
xycoord_interface_0_avalon_slave_0_unreg_firsttransfer :
xycoord_interface_0_avalon_slave_0_reg_firsttransfer;

  //xycoord_interface_0_avalon_slave_0_unreg_firsttransfer first transaction, which is an
e_assign
  assign xycoord_interface_0_avalon_slave_0_unreg_firsttransfer =
~(xycoord_interface_0_avalon_slave_0_slavearbiterlockenable &
xycoord_interface_0_avalon_slave_0_any_continuerequest);

  //xycoord_interface_0_avalon_slave_0_reg_firsttransfer first transaction, which is an
e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          xycoord_interface_0_avalon_slave_0_reg_firsttransfer <= 1'b1;
      else if (xycoord_interface_0_avalon_slave_0_begins_xfer)
          xycoord_interface_0_avalon_slave_0_reg_firsttransfer <=
xycoord_interface_0_avalon_slave_0_unreg_firsttransfer;
    end


  //xycoord_interface_0_avalon_slave_0_beginbursttransfer_internal begin burst transfer,
which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_beginbursttransfer_internal =
xycoord_interface_0_avalon_slave_0_begins_xfer;

  //xycoord_interface_0_avalon_slave_0_arbitration_holdoff_internal arbitration_holdoff,
which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_arbitration_holdoff_internal =
xycoord_interface_0_avalon_slave_0_begins_xfer &
xycoord_interface_0_avalon_slave_0_firsttransfer;

  //xycoord_interface_0_avalon_slave_0_read assignment, which is an e_mux
  assign xycoord_interface_0_avalon_slave_0_read =
(cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0 & cpu_0_data_master_read)
|  (cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0 &
cpu_0_instruction_master_read);

  //xycoord_interface_0_avalon_slave_0_write assignment, which is an e_mux
  assign xycoord_interface_0_avalon_slave_0_write =
cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0 &
cpu_0_data_master_write;

  assign shifted_address_to_xycoord_interface_0_avalon_slave_0_from_cpu_0_data_master =
{cpu_0_data_master_address_to_slave >> 2,
    cpu_0_data_master_dbs_address[1],
    {1 {1'b0}}};

  //xycoord_interface_0_avalon_slave_0_address mux, which is an e_mux
  assign xycoord_interface_0_avalon_slave_0_address =
(cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0)?
(shifted_address_to_xycoord_interface_0_avalon_slave_0_from_cpu_0_data_master >> 1) :
```

```
(shifted_address_to_xycoord_interface_0_avalon_slave_0_from_cpu_0_instruction_master >>
1);

  assign
shifted_address_to_xycoord_interface_0_avalon_slave_0_from_cpu_0_instruction_master =
{cpu_0_instruction_master_address_to_slave >> 2,
    cpu_0_instruction_master_dbs_address[1],
    {1 {1'b0}}};

  //d1_xycoord_interface_0_avalon_slave_0_end_xfer register, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          d1_xycoord_interface_0_avalon_slave_0_end_xfer <= 1;
      else
        d1_xycoord_interface_0_avalon_slave_0_end_xfer <=
xycoord_interface_0_avalon_slave_0_end_xfer;
    end


  //xycoord_interface_0_avalon_slave_0_waits_for_read in a cycle, which is an e_mux
  assign xycoord_interface_0_avalon_slave_0_waits_for_read =
xycoord_interface_0_avalon_slave_0_in_a_read_cycle &
xycoord_interface_0_avalon_slave_0_begins_xfer;

  //xycoord_interface_0_avalon_slave_0_in_a_read_cycle assignment, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_in_a_read_cycle =
(cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0 & cpu_0_data_master_read)
|       (cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0 &
cpu_0_instruction_master_read);

  //in_a_read_cycle assignment, which is an e_mux
  assign in_a_read_cycle = xycoord_interface_0_avalon_slave_0_in_a_read_cycle;

  //xycoord_interface_0_avalon_slave_0_waits_for_write in a cycle, which is an e_mux
  assign xycoord_interface_0_avalon_slave_0_waits_for_write =
xycoord_interface_0_avalon_slave_0_in_a_write_cycle & 0;

  //xycoord_interface_0_avalon_slave_0_in_a_write_cycle assignment, which is an e_assign
  assign xycoord_interface_0_avalon_slave_0_in_a_write_cycle =
cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0 &
cpu_0_data_master_write;

  //in_a_write_cycle assignment, which is an e_mux
  assign in_a_write_cycle = xycoord_interface_0_avalon_slave_0_in_a_write_cycle;

  assign wait_for_xycoord_interface_0_avalon_slave_0_counter = 0;
  assign {cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0_segment_1,
cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0_segment_0} =
cpu_0_data_master_byteenable;
  assign cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0 =
((cpu_0_data_master_dbs_address[1] == 0))?
cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0_segment_0 :
    cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0_segment_1;


//synthesis translate_off
```

```
//////////////// SIMULATION-ONLY CONTENTS
  //xycoord_interface_0/avalon_slave_0 enable non-zero assertions, which is an e_register
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          enable_nonzero_assertions <= 0;
      else
        enable_nonzero_assertions <= 1'b1;
    end


  //grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if          (cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0          +
cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0 > 1)
        begin
          $write("%0d ns: > 1 of grant signals are active simultaneously", $time);
          $stop;
        end
    end


  //saved_grant signals are active simultaneously, which is an e_process
  always @(posedge clk)
    begin
      if        (cpu_0_data_master_saved_grant_xycoord_interface_0_avalon_slave_0        +
cpu_0_instruction_master_saved_grant_xycoord_interface_0_avalon_slave_0 > 1)
        begin
          $write("%0d ns: > 1 of saved_grant signals are active simultaneously", $time);
          $stop;
        end
    end



//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module nios_system_reset_clk_0_domain_synch_module (
                                                     // inputs:
                                                      clk,
                                                      data_in,
                                                      reset_n,

                                                     // outputs:
```

```
                                                                    data_out
                                                                )
;

   output               data_out;
   input                clk;
   input                data_in;
   input                reset_n;

   reg                  data_in_d1 /* synthesis ALTERA_ATTRIBUTE = "{-from \"*\"} CUT=ON ;
PRESERVE_REGISTER=ON ; SUPPRESS_DA_RULE_INTERNAL=R101"    */;
   reg                  data_out /* synthesis ALTERA_ATTRIBUTE = "PRESERVE_REGISTER=ON ;
SUPPRESS_DA_RULE_INTERNAL=R101"    */;
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               data_in_d1 <= 0;
         else
            data_in_d1 <= data_in;
      end


   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               data_out <= 0;
         else
            data_out <= data_in_d1;
      end



endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module nios_system (
                              // 1) global signals:
                                clk_0,
                                reset_n,

                              // the_command_0
                                oMOLECTRL_from_the_command_0,
                                oPANELCTRL_from_the_command_0,

                              // the_interrupt_0
                                irq_adc_to_the_interrupt_0,

                              // the_musiccontroller_0
                                AUD_ADCDAT_to_the_musiccontroller_0,
                                AUD_ADCLRCK_from_the_musiccontroller_0,
```

```
                                AUD_BCLK_to_and_from_the_musiccontroller_0,
                                AUD_DACDAT_from_the_musiccontroller_0,
                                AUD_DACLRCK_from_the_musiccontroller_0,
                                AUD_XCK_from_the_musiccontroller_0,
                                clk_18_to_the_musiccontroller_0,

                               // the_sdram_0
                                zs_addr_from_the_sdram_0,
                                zs_ba_from_the_sdram_0,
                                zs_cas_n_from_the_sdram_0,
                                zs_cke_from_the_sdram_0,
                                zs_cs_n_from_the_sdram_0,
                                zs_dq_to_and_from_the_sdram_0,
                                zs_dqm_from_the_sdram_0,
                                zs_ras_n_from_the_sdram_0,
                                zs_we_n_from_the_sdram_0,

                               // the_sram_mux_0
                                SRAM_ADDR_from_the_sram_mux_0,
                                SRAM_CE_N_from_the_sram_mux_0,
                                SRAM_DQ_to_and_from_the_sram_mux_0,
                                SRAM_LB_N_from_the_sram_mux_0,
                                SRAM_OE_N_from_the_sram_mux_0,
                                SRAM_UB_N_from_the_sram_mux_0,
                                SRAM_WE_N_from_the_sram_mux_0,
                                VGA_ADDR_to_the_sram_mux_0,
                                VGA_DQ_from_the_sram_mux_0,
                                VGA_OE_N_to_the_sram_mux_0,
                                VGA_WE_N_to_the_sram_mux_0,

                               // the_xycoord_interface_0
                                iX_COORD_to_the_xycoord_interface_0,
                                iY_COORD_to_the_xycoord_interface_0
                             )
;

  output              AUD_ADCLRCK_from_the_musiccontroller_0;
  inout               AUD_BCLK_to_and_from_the_musiccontroller_0;
  output              AUD_DACDAT_from_the_musiccontroller_0;
  output              AUD_DACLRCK_from_the_musiccontroller_0;
  output              AUD_XCK_from_the_musiccontroller_0;
  output  [ 17: 0] SRAM_ADDR_from_the_sram_mux_0;
  output              SRAM_CE_N_from_the_sram_mux_0;
  inout   [ 15: 0] SRAM_DQ_to_and_from_the_sram_mux_0;
  output              SRAM_LB_N_from_the_sram_mux_0;
  output              SRAM_OE_N_from_the_sram_mux_0;
  output              SRAM_UB_N_from_the_sram_mux_0;
  output              SRAM_WE_N_from_the_sram_mux_0;
  output  [ 15: 0] VGA_DQ_from_the_sram_mux_0;
  output  [ 31: 0] oMOLECTRL_from_the_command_0;
  output  [ 31: 0] oPANELCTRL_from_the_command_0;
  output  [ 11: 0] zs_addr_from_the_sdram_0;
  output  [  1: 0] zs_ba_from_the_sdram_0;
  output              zs_cas_n_from_the_sdram_0;
  output              zs_cke_from_the_sdram_0;
  output              zs_cs_n_from_the_sdram_0;
  inout   [ 15: 0] zs_dq_to_and_from_the_sdram_0;
  output  [  1: 0] zs_dqm_from_the_sdram_0;
```

```
output              zs_ras_n_from_the_sdram_0;
output              zs_we_n_from_the_sdram_0;
input               AUD_ADCDAT_to_the_musiccontroller_0;
input     [ 17: 0] VGA_ADDR_to_the_sram_mux_0;
input               VGA_OE_N_to_the_sram_mux_0;
input               VGA_WE_N_to_the_sram_mux_0;
input               clk_0;
input               clk_18_to_the_musiccontroller_0;
input     [ 11: 0] iX_COORD_to_the_xycoord_interface_0;
input     [ 11: 0] iY_COORD_to_the_xycoord_interface_0;
input               irq_adc_to_the_interrupt_0;
input               reset_n;

wire                AUD_ADCLRCK_from_the_musiccontroller_0;
wire                AUD_BCLK_to_and_from_the_musiccontroller_0;
wire                AUD_DACDAT_from_the_musiccontroller_0;
wire                AUD_DACLRCK_from_the_musiccontroller_0;
wire                AUD_XCK_from_the_musiccontroller_0;
wire      [ 17: 0] SRAM_ADDR_from_the_sram_mux_0;
wire                SRAM_CE_N_from_the_sram_mux_0;
wire      [ 15: 0] SRAM_DQ_to_and_from_the_sram_mux_0;
wire                SRAM_LB_N_from_the_sram_mux_0;
wire                SRAM_OE_N_from_the_sram_mux_0;
wire                SRAM_UB_N_from_the_sram_mux_0;
wire                SRAM_WE_N_from_the_sram_mux_0;
wire      [ 15: 0] VGA_DQ_from_the_sram_mux_0;
wire      [  4: 0] audio_controller_0_avalon_slave_0_address;
wire                audio_controller_0_avalon_slave_0_chipselect;
wire                audio_controller_0_avalon_slave_0_read;
wire      [ 15: 0] audio_controller_0_avalon_slave_0_readdata;
wire      [ 15: 0] audio_controller_0_avalon_slave_0_readdata_from_sa;
wire                audio_controller_0_avalon_slave_0_reset_n;
wire                audio_controller_0_avalon_slave_0_write;
wire      [ 15: 0] audio_controller_0_avalon_slave_0_writedata;
wire                clk_0_reset_n;
wire      [  4: 0] command_0_avalon_slave_0_address;
wire                command_0_avalon_slave_0_chipselect;
wire                command_0_avalon_slave_0_read;
wire      [ 31: 0] command_0_avalon_slave_0_readdata;
wire      [ 31: 0] command_0_avalon_slave_0_readdata_from_sa;
wire                command_0_avalon_slave_0_reset_n;
wire                command_0_avalon_slave_0_write;
wire      [ 31: 0] command_0_avalon_slave_0_writedata;
wire      [ 24: 0] cpu_0_data_master_address;
wire      [ 24: 0] cpu_0_data_master_address_to_slave;
wire      [  3: 0] cpu_0_data_master_byteenable;
wire      [  1: 0] cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0;
wire      [  1: 0] cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0;
wire      [  1: 0] cpu_0_data_master_byteenable_sdram_0_s1;
wire      [  1: 0] cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0;
wire      [  1: 0] cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0;
wire      [  1: 0] cpu_0_data_master_dbs_address;
wire      [ 15: 0] cpu_0_data_master_dbs_write_16;
wire                cpu_0_data_master_debugaccess;
wire                cpu_0_data_master_granted_audio_controller_0_avalon_slave_0;
wire                cpu_0_data_master_granted_command_0_avalon_slave_0;
wire                cpu_0_data_master_granted_cpu_0_jtag_debug_module;
wire                cpu_0_data_master_granted_interrupt_0_avalon_slave_0;
```

```
  wire                    cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave;
  wire                    cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0;
  wire                    cpu_0_data_master_granted_sdram_0_s1;
  wire                    cpu_0_data_master_granted_sram_mux_0_avalon_slave_0;
  wire                    cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0;
  wire      [ 31: 0] cpu_0_data_master_irq;
  wire                    cpu_0_data_master_no_byte_enables_and_last_term;
  wire
cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0;
  wire                    cpu_0_data_master_qualified_request_command_0_avalon_slave_0;
  wire                    cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module;
  wire                    cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0;
  wire                    cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave;
  wire
cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0;
  wire                    cpu_0_data_master_qualified_request_sdram_0_s1;
  wire                    cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0;
  wire
cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0;
  wire                    cpu_0_data_master_read;
  wire
cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0;
  wire                    cpu_0_data_master_read_data_valid_command_0_avalon_slave_0;
  wire                    cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module;
  wire                    cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0;
  wire                    cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave;
  wire                    cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0;
  wire                    cpu_0_data_master_read_data_valid_sdram_0_s1;
  wire                    cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register;
  wire                    cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0;
  wire
cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0;
  wire      [ 31: 0] cpu_0_data_master_readdata;
  wire                    cpu_0_data_master_requests_audio_controller_0_avalon_slave_0;
  wire                    cpu_0_data_master_requests_command_0_avalon_slave_0;
  wire                    cpu_0_data_master_requests_cpu_0_jtag_debug_module;
  wire                    cpu_0_data_master_requests_interrupt_0_avalon_slave_0;
  wire                    cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave;
  wire                    cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0;
  wire                    cpu_0_data_master_requests_sdram_0_s1;
  wire                    cpu_0_data_master_requests_sram_mux_0_avalon_slave_0;
  wire                    cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0;
  wire                    cpu_0_data_master_waitrequest;
  wire                    cpu_0_data_master_write;
  wire      [ 31: 0] cpu_0_data_master_writedata;
  wire      [ 24: 0] cpu_0_instruction_master_address;
  wire      [ 24: 0] cpu_0_instruction_master_address_to_slave;
  wire      [  1: 0] cpu_0_instruction_master_dbs_address;
  wire                    cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0;
  wire                    cpu_0_instruction_master_granted_command_0_avalon_slave_0;
  wire                    cpu_0_instruction_master_granted_cpu_0_jtag_debug_module;
  wire                    cpu_0_instruction_master_granted_sdram_0_s1;
  wire                    cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0;
  wire
cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0;
  wire
cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0;
  wire
```

```
cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0;
   wire
cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module;
   wire                    cpu_0_instruction_master_qualified_request_sdram_0_s1;
   wire
cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0;
   wire
cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0;
   wire                    cpu_0_instruction_master_read;
   wire
cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0;
   wire
cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0;
   wire                    cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module;
   wire                    cpu_0_instruction_master_read_data_valid_sdram_0_s1;
   wire                    cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register;
   wire
cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0;
   wire
cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0;
   wire      [ 31: 0] cpu_0_instruction_master_readdata;
   wire                    cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0;
   wire                    cpu_0_instruction_master_requests_command_0_avalon_slave_0;
   wire                    cpu_0_instruction_master_requests_cpu_0_jtag_debug_module;
   wire                    cpu_0_instruction_master_requests_sdram_0_s1;
   wire                    cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0;
   wire
cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0;
   wire                    cpu_0_instruction_master_waitrequest;
   wire      [  8: 0] cpu_0_jtag_debug_module_address;
   wire                    cpu_0_jtag_debug_module_begintransfer;
   wire      [  3: 0] cpu_0_jtag_debug_module_byteenable;
   wire                    cpu_0_jtag_debug_module_chipselect;
   wire                    cpu_0_jtag_debug_module_debugaccess;
   wire      [ 31: 0] cpu_0_jtag_debug_module_readdata;
   wire      [ 31: 0] cpu_0_jtag_debug_module_readdata_from_sa;
   wire                    cpu_0_jtag_debug_module_reset_n;
   wire                    cpu_0_jtag_debug_module_resetrequest;
   wire                    cpu_0_jtag_debug_module_resetrequest_from_sa;
   wire                    cpu_0_jtag_debug_module_write;
   wire      [ 31: 0] cpu_0_jtag_debug_module_writedata;
   wire                    d1_audio_controller_0_avalon_slave_0_end_xfer;
   wire                    d1_command_0_avalon_slave_0_end_xfer;
   wire                    d1_cpu_0_jtag_debug_module_end_xfer;
   wire                    d1_interrupt_0_avalon_slave_0_end_xfer;
   wire                    d1_jtag_uart_0_avalon_jtag_slave_end_xfer;
   wire                    d1_musiccontroller_0_avalon_slave_0_end_xfer;
   wire                    d1_sdram_0_s1_end_xfer;
   wire                    d1_sram_mux_0_avalon_slave_0_end_xfer;
   wire                    d1_xycoord_interface_0_avalon_slave_0_end_xfer;
   wire      [  4: 0] interrupt_0_avalon_slave_0_address;
   wire                    interrupt_0_avalon_slave_0_chipselect;
   wire                    interrupt_0_avalon_slave_0_irq;
   wire                    interrupt_0_avalon_slave_0_irq_from_sa;
   wire                    interrupt_0_avalon_slave_0_read;
   wire      [ 31: 0] interrupt_0_avalon_slave_0_readdata;
   wire      [ 31: 0] interrupt_0_avalon_slave_0_readdata_from_sa;
   wire                    interrupt_0_avalon_slave_0_reset_n;
```

```
wire            interrupt_0_avalon_slave_0_write;
wire    [ 31: 0] interrupt_0_avalon_slave_0_writedata;
wire            jtag_uart_0_avalon_jtag_slave_address;
wire            jtag_uart_0_avalon_jtag_slave_chipselect;
wire            jtag_uart_0_avalon_jtag_slave_dataavailable;
wire            jtag_uart_0_avalon_jtag_slave_dataavailable_from_sa;
wire            jtag_uart_0_avalon_jtag_slave_irq;
wire            jtag_uart_0_avalon_jtag_slave_irq_from_sa;
wire            jtag_uart_0_avalon_jtag_slave_read_n;
wire    [ 31: 0] jtag_uart_0_avalon_jtag_slave_readdata;
wire    [ 31: 0] jtag_uart_0_avalon_jtag_slave_readdata_from_sa;
wire            jtag_uart_0_avalon_jtag_slave_readyfordata;
wire            jtag_uart_0_avalon_jtag_slave_readyfordata_from_sa;
wire            jtag_uart_0_avalon_jtag_slave_reset_n;
wire            jtag_uart_0_avalon_jtag_slave_waitrequest;
wire            jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa;
wire            jtag_uart_0_avalon_jtag_slave_write_n;
wire    [ 31: 0] jtag_uart_0_avalon_jtag_slave_writedata;
wire    [ 16: 0] musiccontroller_0_avalon_slave_0_address;
wire            musiccontroller_0_avalon_slave_0_chipselect;
wire            musiccontroller_0_avalon_slave_0_irq;
wire            musiccontroller_0_avalon_slave_0_irq_from_sa;
wire            musiccontroller_0_avalon_slave_0_read;
wire    [ 15: 0] musiccontroller_0_avalon_slave_0_readdata;
wire    [ 15: 0] musiccontroller_0_avalon_slave_0_readdata_from_sa;
wire            musiccontroller_0_avalon_slave_0_reset_n;
wire            musiccontroller_0_avalon_slave_0_write;
wire    [ 15: 0] musiccontroller_0_avalon_slave_0_writedata;
wire    [ 31: 0] oMOLECTRL_from_the_command_0;
wire    [ 31: 0] oPANELCTRL_from_the_command_0;
wire            reset_n_sources;
wire    [ 21: 0] sdram_0_s1_address;
wire    [  1: 0] sdram_0_s1_byteenable_n;
wire            sdram_0_s1_chipselect;
wire            sdram_0_s1_read_n;
wire    [ 15: 0] sdram_0_s1_readdata;
wire    [ 15: 0] sdram_0_s1_readdata_from_sa;
wire            sdram_0_s1_readdatavalid;
wire            sdram_0_s1_reset_n;
wire            sdram_0_s1_waitrequest;
wire            sdram_0_s1_waitrequest_from_sa;
wire            sdram_0_s1_write_n;
wire    [ 15: 0] sdram_0_s1_writedata;
wire    [ 17: 0] sram_mux_0_avalon_slave_0_address;
wire            sram_mux_0_avalon_slave_0_read;
wire    [ 15: 0] sram_mux_0_avalon_slave_0_readdata;
wire    [ 15: 0] sram_mux_0_avalon_slave_0_readdata_from_sa;
wire            sram_mux_0_avalon_slave_0_write;
wire    [ 15: 0] sram_mux_0_avalon_slave_0_writedata;
wire    [  4: 0] xycoord_interface_0_avalon_slave_0_address;
wire            xycoord_interface_0_avalon_slave_0_chipselect;
wire            xycoord_interface_0_avalon_slave_0_read;
wire    [ 15: 0] xycoord_interface_0_avalon_slave_0_readdata;
wire    [ 15: 0] xycoord_interface_0_avalon_slave_0_readdata_from_sa;
wire            xycoord_interface_0_avalon_slave_0_reset_n;
wire            xycoord_interface_0_avalon_slave_0_write;
wire    [ 15: 0] xycoord_interface_0_avalon_slave_0_writedata;
wire    [ 11: 0] zs_addr_from_the_sdram_0;
```

```
  wire       [  1: 0] zs_ba_from_the_sdram_0;
  wire                zs_cas_n_from_the_sdram_0;
  wire                zs_cke_from_the_sdram_0;
  wire                zs_cs_n_from_the_sdram_0;
  wire       [ 15: 0] zs_dq_to_and_from_the_sdram_0;
  wire       [  1: 0] zs_dqm_from_the_sdram_0;
  wire                zs_ras_n_from_the_sdram_0;
  wire                zs_we_n_from_the_sdram_0;
  audio_controller_0_avalon_slave_0_arbitrator the_audio_controller_0_avalon_slave_0
    (
      .audio_controller_0_avalon_slave_0_address
(audio_controller_0_avalon_slave_0_address),
      .audio_controller_0_avalon_slave_0_chipselect
(audio_controller_0_avalon_slave_0_chipselect),
      .audio_controller_0_avalon_slave_0_read
(audio_controller_0_avalon_slave_0_read),
      .audio_controller_0_avalon_slave_0_readdata
(audio_controller_0_avalon_slave_0_readdata),
      .audio_controller_0_avalon_slave_0_readdata_from_sa
(audio_controller_0_avalon_slave_0_readdata_from_sa),
      .audio_controller_0_avalon_slave_0_reset_n
(audio_controller_0_avalon_slave_0_reset_n),
      .audio_controller_0_avalon_slave_0_write
(audio_controller_0_avalon_slave_0_write),
      .audio_controller_0_avalon_slave_0_writedata
(audio_controller_0_avalon_slave_0_writedata),
      .clk
(clk_0),
      .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
      .cpu_0_data_master_byteenable
(cpu_0_data_master_byteenable),
      .cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0
(cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0),
      .cpu_0_data_master_dbs_address
(cpu_0_data_master_dbs_address),
      .cpu_0_data_master_dbs_write_16
(cpu_0_data_master_dbs_write_16),
      .cpu_0_data_master_granted_audio_controller_0_avalon_slave_0
(cpu_0_data_master_granted_audio_controller_0_avalon_slave_0),
      .cpu_0_data_master_no_byte_enables_and_last_term
(cpu_0_data_master_no_byte_enables_and_last_term),
      .cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0
(cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0),
      .cpu_0_data_master_read
(cpu_0_data_master_read),
      .cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0),
      .cpu_0_data_master_requests_audio_controller_0_avalon_slave_0
(cpu_0_data_master_requests_audio_controller_0_avalon_slave_0),
      .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
      .cpu_0_data_master_write
(cpu_0_data_master_write),
      .cpu_0_instruction_master_address_to_slave
(cpu_0_instruction_master_address_to_slave),
      .cpu_0_instruction_master_dbs_address
(cpu_0_instruction_master_dbs_address),
```

```
        .cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0
(cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0),
        .cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0
(cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0),
        .cpu_0_instruction_master_read
(cpu_0_instruction_master_read),
        .cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0
(cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0),
        .cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0
(cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0),
        .d1_audio_controller_0_avalon_slave_0_end_xfer
(d1_audio_controller_0_avalon_slave_0_end_xfer),
        .reset_n
(clk_0_reset_n)
      );

   audio_controller_0 the_audio_controller_0
      (
        .address       (audio_controller_0_avalon_slave_0_address),
        .chipselect (audio_controller_0_avalon_slave_0_chipselect),
        .clk           (clk_0),
        .read          (audio_controller_0_avalon_slave_0_read),
        .readdata      (audio_controller_0_avalon_slave_0_readdata),
        .reset_n       (audio_controller_0_avalon_slave_0_reset_n),
        .write         (audio_controller_0_avalon_slave_0_write),
        .writedata     (audio_controller_0_avalon_slave_0_writedata)
      );

   command_0_avalon_slave_0_arbitrator the_command_0_avalon_slave_0
      (
        .clk
(clk_0),
        .command_0_avalon_slave_0_address
(command_0_avalon_slave_0_address),
        .command_0_avalon_slave_0_chipselect
(command_0_avalon_slave_0_chipselect),
        .command_0_avalon_slave_0_read
(command_0_avalon_slave_0_read),
        .command_0_avalon_slave_0_readdata
(command_0_avalon_slave_0_readdata),
        .command_0_avalon_slave_0_readdata_from_sa
(command_0_avalon_slave_0_readdata_from_sa),
        .command_0_avalon_slave_0_reset_n
(command_0_avalon_slave_0_reset_n),
        .command_0_avalon_slave_0_write
(command_0_avalon_slave_0_write),
        .command_0_avalon_slave_0_writedata
(command_0_avalon_slave_0_writedata),
        .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
        .cpu_0_data_master_granted_command_0_avalon_slave_0
(cpu_0_data_master_granted_command_0_avalon_slave_0),
        .cpu_0_data_master_qualified_request_command_0_avalon_slave_0
(cpu_0_data_master_qualified_request_command_0_avalon_slave_0),
        .cpu_0_data_master_read
(cpu_0_data_master_read),
        .cpu_0_data_master_read_data_valid_command_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_command_0_avalon_slave_0),
```

```
        .cpu_0_data_master_requests_command_0_avalon_slave_0
(cpu_0_data_master_requests_command_0_avalon_slave_0),
        .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
        .cpu_0_data_master_write
(cpu_0_data_master_write),
        .cpu_0_data_master_writedata
(cpu_0_data_master_writedata),
        .cpu_0_instruction_master_address_to_slave
(cpu_0_instruction_master_address_to_slave),
        .cpu_0_instruction_master_granted_command_0_avalon_slave_0
(cpu_0_instruction_master_granted_command_0_avalon_slave_0),
        .cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0
(cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0),
        .cpu_0_instruction_master_read
(cpu_0_instruction_master_read),
        .cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0
(cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0),
        .cpu_0_instruction_master_requests_command_0_avalon_slave_0
(cpu_0_instruction_master_requests_command_0_avalon_slave_0),
        .d1_command_0_avalon_slave_0_end_xfer
(d1_command_0_avalon_slave_0_end_xfer),
        .reset_n
(clk_0_reset_n)
      );

   command_0 the_command_0
      (
        .address       (command_0_avalon_slave_0_address),
        .chipselect (command_0_avalon_slave_0_chipselect),
        .clk             (clk_0),
        .oMOLECTRL    (oMOLECTRL_from_the_command_0),
        .oPANELCTRL (oPANELCTRL_from_the_command_0),
        .read            (command_0_avalon_slave_0_read),
        .readdata     (command_0_avalon_slave_0_readdata),
        .reset_n       (command_0_avalon_slave_0_reset_n),
        .write          (command_0_avalon_slave_0_write),
        .writedata    (command_0_avalon_slave_0_writedata)
      );

   cpu_0_jtag_debug_module_arbitrator the_cpu_0_jtag_debug_module
      (
        .clk                                                                    (clk_0),
        .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
        .cpu_0_data_master_byteenable
(cpu_0_data_master_byteenable),
        .cpu_0_data_master_debugaccess
(cpu_0_data_master_debugaccess),
        .cpu_0_data_master_granted_cpu_0_jtag_debug_module
(cpu_0_data_master_granted_cpu_0_jtag_debug_module),
        .cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module
(cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module),
        .cpu_0_data_master_read
(cpu_0_data_master_read),
        .cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module
(cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module),
        .cpu_0_data_master_requests_cpu_0_jtag_debug_module
```

```
(cpu_0_data_master_requests_cpu_0_jtag_debug_module),
        .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
        .cpu_0_data_master_write
(cpu_0_data_master_write),
        .cpu_0_data_master_writedata
(cpu_0_data_master_writedata),
        .cpu_0_instruction_master_address_to_slave
(cpu_0_instruction_master_address_to_slave),
        .cpu_0_instruction_master_granted_cpu_0_jtag_debug_module
(cpu_0_instruction_master_granted_cpu_0_jtag_debug_module),
        .cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module
(cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module),
        .cpu_0_instruction_master_read
(cpu_0_instruction_master_read),
        .cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module
(cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module),
        .cpu_0_instruction_master_requests_cpu_0_jtag_debug_module
(cpu_0_instruction_master_requests_cpu_0_jtag_debug_module),
        .cpu_0_jtag_debug_module_address
(cpu_0_jtag_debug_module_address),
        .cpu_0_jtag_debug_module_begintransfer
(cpu_0_jtag_debug_module_begintransfer),
        .cpu_0_jtag_debug_module_byteenable
(cpu_0_jtag_debug_module_byteenable),
        .cpu_0_jtag_debug_module_chipselect
(cpu_0_jtag_debug_module_chipselect),
        .cpu_0_jtag_debug_module_debugaccess
(cpu_0_jtag_debug_module_debugaccess),
        .cpu_0_jtag_debug_module_readdata
(cpu_0_jtag_debug_module_readdata),
        .cpu_0_jtag_debug_module_readdata_from_sa
(cpu_0_jtag_debug_module_readdata_from_sa),
        .cpu_0_jtag_debug_module_reset_n
(cpu_0_jtag_debug_module_reset_n),
        .cpu_0_jtag_debug_module_resetrequest
(cpu_0_jtag_debug_module_resetrequest),
        .cpu_0_jtag_debug_module_resetrequest_from_sa
(cpu_0_jtag_debug_module_resetrequest_from_sa),
        .cpu_0_jtag_debug_module_write
(cpu_0_jtag_debug_module_write),
        .cpu_0_jtag_debug_module_writedata
(cpu_0_jtag_debug_module_writedata),
        .d1_cpu_0_jtag_debug_module_end_xfer
(d1_cpu_0_jtag_debug_module_end_xfer),
        .reset_n
(clk_0_reset_n)
      );

   cpu_0_data_master_arbitrator the_cpu_0_data_master
      (
        .audio_controller_0_avalon_slave_0_readdata_from_sa
(audio_controller_0_avalon_slave_0_readdata_from_sa),
        .clk
(clk_0),
        .command_0_avalon_slave_0_readdata_from_sa
(command_0_avalon_slave_0_readdata_from_sa),
        .cpu_0_data_master_address
```

```
(cpu_0_data_master_address),
        .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
        .cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0
(cpu_0_data_master_byteenable_audio_controller_0_avalon_slave_0),
        .cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0),
        .cpu_0_data_master_byteenable_sdram_0_s1
(cpu_0_data_master_byteenable_sdram_0_s1),
        .cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0
(cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0),
        .cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0),
        .cpu_0_data_master_dbs_address
(cpu_0_data_master_dbs_address),
        .cpu_0_data_master_dbs_write_16
(cpu_0_data_master_dbs_write_16),
        .cpu_0_data_master_granted_audio_controller_0_avalon_slave_0
(cpu_0_data_master_granted_audio_controller_0_avalon_slave_0),
        .cpu_0_data_master_granted_command_0_avalon_slave_0
(cpu_0_data_master_granted_command_0_avalon_slave_0),
        .cpu_0_data_master_granted_cpu_0_jtag_debug_module
(cpu_0_data_master_granted_cpu_0_jtag_debug_module),
        .cpu_0_data_master_granted_interrupt_0_avalon_slave_0
(cpu_0_data_master_granted_interrupt_0_avalon_slave_0),
        .cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave
(cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave),
        .cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0),
        .cpu_0_data_master_granted_sdram_0_s1
(cpu_0_data_master_granted_sdram_0_s1),
        .cpu_0_data_master_granted_sram_mux_0_avalon_slave_0
(cpu_0_data_master_granted_sram_mux_0_avalon_slave_0),
        .cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0),
        .cpu_0_data_master_irq
(cpu_0_data_master_irq),
        .cpu_0_data_master_no_byte_enables_and_last_term
(cpu_0_data_master_no_byte_enables_and_last_term),
        .cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0
(cpu_0_data_master_qualified_request_audio_controller_0_avalon_slave_0),
        .cpu_0_data_master_qualified_request_command_0_avalon_slave_0
(cpu_0_data_master_qualified_request_command_0_avalon_slave_0),
        .cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module
(cpu_0_data_master_qualified_request_cpu_0_jtag_debug_module),
        .cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0
(cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0),
        .cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave
(cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave),
        .cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0),
        .cpu_0_data_master_qualified_request_sdram_0_s1
(cpu_0_data_master_qualified_request_sdram_0_s1),
        .cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0
(cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0),
        .cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0),
        .cpu_0_data_master_read
```

```
(cpu_0_data_master_read),
        .cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_audio_controller_0_avalon_slave_0),
        .cpu_0_data_master_read_data_valid_command_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_command_0_avalon_slave_0),
        .cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module
(cpu_0_data_master_read_data_valid_cpu_0_jtag_debug_module),
        .cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0),
        .cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave
(cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave),
        .cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0),
        .cpu_0_data_master_read_data_valid_sdram_0_s1
(cpu_0_data_master_read_data_valid_sdram_0_s1),
        .cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register
(cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register),
        .cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0),
        .cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0),
        .cpu_0_data_master_readdata
(cpu_0_data_master_readdata),
        .cpu_0_data_master_requests_audio_controller_0_avalon_slave_0
(cpu_0_data_master_requests_audio_controller_0_avalon_slave_0),
        .cpu_0_data_master_requests_command_0_avalon_slave_0
(cpu_0_data_master_requests_command_0_avalon_slave_0),
        .cpu_0_data_master_requests_cpu_0_jtag_debug_module
(cpu_0_data_master_requests_cpu_0_jtag_debug_module),
        .cpu_0_data_master_requests_interrupt_0_avalon_slave_0
(cpu_0_data_master_requests_interrupt_0_avalon_slave_0),
        .cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave
(cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave),
        .cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0),
        .cpu_0_data_master_requests_sdram_0_s1
(cpu_0_data_master_requests_sdram_0_s1),
        .cpu_0_data_master_requests_sram_mux_0_avalon_slave_0
(cpu_0_data_master_requests_sram_mux_0_avalon_slave_0),
        .cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0),
        .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
        .cpu_0_data_master_write
(cpu_0_data_master_write),
        .cpu_0_data_master_writedata
(cpu_0_data_master_writedata),
        .cpu_0_jtag_debug_module_readdata_from_sa
(cpu_0_jtag_debug_module_readdata_from_sa),
        .d1_audio_controller_0_avalon_slave_0_end_xfer
(d1_audio_controller_0_avalon_slave_0_end_xfer),
        .d1_command_0_avalon_slave_0_end_xfer
(d1_command_0_avalon_slave_0_end_xfer),
        .d1_cpu_0_jtag_debug_module_end_xfer
(d1_cpu_0_jtag_debug_module_end_xfer),
        .d1_interrupt_0_avalon_slave_0_end_xfer
(d1_interrupt_0_avalon_slave_0_end_xfer),
        .d1_jtag_uart_0_avalon_jtag_slave_end_xfer
```

```
(d1_jtag_uart_0_avalon_jtag_slave_end_xfer),
        .d1_musiccontroller_0_avalon_slave_0_end_xfer
(d1_musiccontroller_0_avalon_slave_0_end_xfer),
        .d1_sdram_0_s1_end_xfer
(d1_sdram_0_s1_end_xfer),
        .d1_sram_mux_0_avalon_slave_0_end_xfer
(d1_sram_mux_0_avalon_slave_0_end_xfer),
        .d1_xycoord_interface_0_avalon_slave_0_end_xfer
(d1_xycoord_interface_0_avalon_slave_0_end_xfer),
        .interrupt_0_avalon_slave_0_irq_from_sa
(interrupt_0_avalon_slave_0_irq_from_sa),
        .interrupt_0_avalon_slave_0_readdata_from_sa
(interrupt_0_avalon_slave_0_readdata_from_sa),
        .jtag_uart_0_avalon_jtag_slave_irq_from_sa
(jtag_uart_0_avalon_jtag_slave_irq_from_sa),
        .jtag_uart_0_avalon_jtag_slave_readdata_from_sa
(jtag_uart_0_avalon_jtag_slave_readdata_from_sa),
        .jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa
(jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa),
        .musiccontroller_0_avalon_slave_0_irq_from_sa
(musiccontroller_0_avalon_slave_0_irq_from_sa),
        .musiccontroller_0_avalon_slave_0_readdata_from_sa
(musiccontroller_0_avalon_slave_0_readdata_from_sa),
        .reset_n
(clk_0_reset_n),
        .sdram_0_s1_readdata_from_sa
(sdram_0_s1_readdata_from_sa),
        .sdram_0_s1_waitrequest_from_sa
(sdram_0_s1_waitrequest_from_sa),
        .sram_mux_0_avalon_slave_0_readdata_from_sa
(sram_mux_0_avalon_slave_0_readdata_from_sa),
        .xycoord_interface_0_avalon_slave_0_readdata_from_sa
(xycoord_interface_0_avalon_slave_0_readdata_from_sa)
      );

   cpu_0_instruction_master_arbitrator the_cpu_0_instruction_master
      (
        .audio_controller_0_avalon_slave_0_readdata_from_sa
(audio_controller_0_avalon_slave_0_readdata_from_sa),
        .clk
(clk_0),
        .command_0_avalon_slave_0_readdata_from_sa
(command_0_avalon_slave_0_readdata_from_sa),
        .cpu_0_instruction_master_address
(cpu_0_instruction_master_address),
        .cpu_0_instruction_master_address_to_slave
(cpu_0_instruction_master_address_to_slave),
        .cpu_0_instruction_master_dbs_address
(cpu_0_instruction_master_dbs_address),
        .cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0
(cpu_0_instruction_master_granted_audio_controller_0_avalon_slave_0),
        .cpu_0_instruction_master_granted_command_0_avalon_slave_0
(cpu_0_instruction_master_granted_command_0_avalon_slave_0),
        .cpu_0_instruction_master_granted_cpu_0_jtag_debug_module
(cpu_0_instruction_master_granted_cpu_0_jtag_debug_module),
        .cpu_0_instruction_master_granted_sdram_0_s1
(cpu_0_instruction_master_granted_sdram_0_s1),
        .cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0
```

```
(cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0),
        .cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0
(cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0),
        .cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0
(cpu_0_instruction_master_qualified_request_audio_controller_0_avalon_slave_0),
        .cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0
(cpu_0_instruction_master_qualified_request_command_0_avalon_slave_0),
        .cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module
(cpu_0_instruction_master_qualified_request_cpu_0_jtag_debug_module),
        .cpu_0_instruction_master_qualified_request_sdram_0_s1
(cpu_0_instruction_master_qualified_request_sdram_0_s1),
        .cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0
(cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0),
        .cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0
(cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0),
        .cpu_0_instruction_master_read
(cpu_0_instruction_master_read),
        .cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0
(cpu_0_instruction_master_read_data_valid_audio_controller_0_avalon_slave_0),
        .cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0
(cpu_0_instruction_master_read_data_valid_command_0_avalon_slave_0),
        .cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module
(cpu_0_instruction_master_read_data_valid_cpu_0_jtag_debug_module),
        .cpu_0_instruction_master_read_data_valid_sdram_0_s1
(cpu_0_instruction_master_read_data_valid_sdram_0_s1),
        .cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register
(cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register),
        .cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0
(cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0),
        .cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0
(cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0),
        .cpu_0_instruction_master_readdata
(cpu_0_instruction_master_readdata),
        .cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0
(cpu_0_instruction_master_requests_audio_controller_0_avalon_slave_0),
        .cpu_0_instruction_master_requests_command_0_avalon_slave_0
(cpu_0_instruction_master_requests_command_0_avalon_slave_0),
        .cpu_0_instruction_master_requests_cpu_0_jtag_debug_module
(cpu_0_instruction_master_requests_cpu_0_jtag_debug_module),
        .cpu_0_instruction_master_requests_sdram_0_s1
(cpu_0_instruction_master_requests_sdram_0_s1),
        .cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0
(cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0),
        .cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0
(cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0),
        .cpu_0_instruction_master_waitrequest
(cpu_0_instruction_master_waitrequest),
        .cpu_0_jtag_debug_module_readdata_from_sa
(cpu_0_jtag_debug_module_readdata_from_sa),
        .d1_audio_controller_0_avalon_slave_0_end_xfer
(d1_audio_controller_0_avalon_slave_0_end_xfer),
        .d1_command_0_avalon_slave_0_end_xfer
(d1_command_0_avalon_slave_0_end_xfer),
        .d1_cpu_0_jtag_debug_module_end_xfer
(d1_cpu_0_jtag_debug_module_end_xfer),
        .d1_sdram_0_s1_end_xfer
(d1_sdram_0_s1_end_xfer),
        .d1_sram_mux_0_avalon_slave_0_end_xfer
```

```
(d1_sram_mux_0_avalon_slave_0_end_xfer),
        .d1_xycoord_interface_0_avalon_slave_0_end_xfer
(d1_xycoord_interface_0_avalon_slave_0_end_xfer),
        .reset_n
(clk_0_reset_n),
        .sdram_0_s1_readdata_from_sa
(sdram_0_s1_readdata_from_sa),
        .sdram_0_s1_waitrequest_from_sa
(sdram_0_s1_waitrequest_from_sa),
        .sram_mux_0_avalon_slave_0_readdata_from_sa
(sram_mux_0_avalon_slave_0_readdata_from_sa),
        .xycoord_interface_0_avalon_slave_0_readdata_from_sa
(xycoord_interface_0_avalon_slave_0_readdata_from_sa)
      );

   cpu_0 the_cpu_0
      (
        .clk                                   (clk_0),
        .d_address                             (cpu_0_data_master_address),
        .d_byteenable                          (cpu_0_data_master_byteenable),
        .d_irq                                 (cpu_0_data_master_irq),
        .d_read                                (cpu_0_data_master_read),
        .d_readdata                            (cpu_0_data_master_readdata),
        .d_waitrequest                         (cpu_0_data_master_waitrequest),
        .d_write                               (cpu_0_data_master_write),
        .d_writedata                           (cpu_0_data_master_writedata),
        .i_address                             (cpu_0_instruction_master_address),
        .i_read                                (cpu_0_instruction_master_read),
        .i_readdata                            (cpu_0_instruction_master_readdata),
        .i_waitrequest                         (cpu_0_instruction_master_waitrequest),
        .jtag_debug_module_address             (cpu_0_jtag_debug_module_address),
        .jtag_debug_module_begintransfer       (cpu_0_jtag_debug_module_begintransfer),
        .jtag_debug_module_byteenable          (cpu_0_jtag_debug_module_byteenable),
        .jtag_debug_module_debugaccess         (cpu_0_jtag_debug_module_debugaccess),
        .jtag_debug_module_debugaccess_to_roms (cpu_0_data_master_debugaccess),
        .jtag_debug_module_readdata            (cpu_0_jtag_debug_module_readdata),
        .jtag_debug_module_resetrequest        (cpu_0_jtag_debug_module_resetrequest),
        .jtag_debug_module_select              (cpu_0_jtag_debug_module_chipselect),
        .jtag_debug_module_write               (cpu_0_jtag_debug_module_write),
        .jtag_debug_module_writedata           (cpu_0_jtag_debug_module_writedata),
        .reset_n                               (cpu_0_jtag_debug_module_reset_n)
      );

   interrupt_0_avalon_slave_0_arbitrator the_interrupt_0_avalon_slave_0
      (
        .clk                                                                    (clk_0),
        .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
        .cpu_0_data_master_granted_interrupt_0_avalon_slave_0
(cpu_0_data_master_granted_interrupt_0_avalon_slave_0),
        .cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0
(cpu_0_data_master_qualified_request_interrupt_0_avalon_slave_0),
        .cpu_0_data_master_read
(cpu_0_data_master_read),
        .cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_interrupt_0_avalon_slave_0),
        .cpu_0_data_master_requests_interrupt_0_avalon_slave_0
(cpu_0_data_master_requests_interrupt_0_avalon_slave_0),
```

```
        .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
        .cpu_0_data_master_write
(cpu_0_data_master_write),
        .cpu_0_data_master_writedata
(cpu_0_data_master_writedata),
        .d1_interrupt_0_avalon_slave_0_end_xfer
(d1_interrupt_0_avalon_slave_0_end_xfer),
        .interrupt_0_avalon_slave_0_address
(interrupt_0_avalon_slave_0_address),
        .interrupt_0_avalon_slave_0_chipselect
(interrupt_0_avalon_slave_0_chipselect),
        .interrupt_0_avalon_slave_0_irq
(interrupt_0_avalon_slave_0_irq),
        .interrupt_0_avalon_slave_0_irq_from_sa
(interrupt_0_avalon_slave_0_irq_from_sa),
        .interrupt_0_avalon_slave_0_read
(interrupt_0_avalon_slave_0_read),
        .interrupt_0_avalon_slave_0_readdata
(interrupt_0_avalon_slave_0_readdata),
        .interrupt_0_avalon_slave_0_readdata_from_sa
(interrupt_0_avalon_slave_0_readdata_from_sa),
        .interrupt_0_avalon_slave_0_reset_n
(interrupt_0_avalon_slave_0_reset_n),
        .interrupt_0_avalon_slave_0_write
(interrupt_0_avalon_slave_0_write),
        .interrupt_0_avalon_slave_0_writedata
(interrupt_0_avalon_slave_0_writedata),
        .reset_n
(clk_0_reset_n)
      );

   interrupt_0 the_interrupt_0
      (
        .address       (interrupt_0_avalon_slave_0_address),
        .chipselect (interrupt_0_avalon_slave_0_chipselect),
        .clk              (clk_0),
        .irq              (interrupt_0_avalon_slave_0_irq),
        .irq_adc       (irq_adc_to_the_interrupt_0),
        .read             (interrupt_0_avalon_slave_0_read),
        .readdata      (interrupt_0_avalon_slave_0_readdata),
        .reset_n       (interrupt_0_avalon_slave_0_reset_n),
        .write            (interrupt_0_avalon_slave_0_write),
        .writedata    (interrupt_0_avalon_slave_0_writedata)
      );

   jtag_uart_0_avalon_jtag_slave_arbitrator the_jtag_uart_0_avalon_jtag_slave
      (
        .clk                                                                                    (clk_0),
        .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
        .cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave
(cpu_0_data_master_granted_jtag_uart_0_avalon_jtag_slave),
        .cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave
(cpu_0_data_master_qualified_request_jtag_uart_0_avalon_jtag_slave),
        .cpu_0_data_master_read
(cpu_0_data_master_read),
        .cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave
```

```
(cpu_0_data_master_read_data_valid_jtag_uart_0_avalon_jtag_slave),
        .cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave
(cpu_0_data_master_requests_jtag_uart_0_avalon_jtag_slave),
        .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
        .cpu_0_data_master_write
(cpu_0_data_master_write),
        .cpu_0_data_master_writedata
(cpu_0_data_master_writedata),
        .d1_jtag_uart_0_avalon_jtag_slave_end_xfer
(d1_jtag_uart_0_avalon_jtag_slave_end_xfer),
        .jtag_uart_0_avalon_jtag_slave_address
(jtag_uart_0_avalon_jtag_slave_address),
        .jtag_uart_0_avalon_jtag_slave_chipselect
(jtag_uart_0_avalon_jtag_slave_chipselect),
        .jtag_uart_0_avalon_jtag_slave_dataavailable
(jtag_uart_0_avalon_jtag_slave_dataavailable),
        .jtag_uart_0_avalon_jtag_slave_dataavailable_from_sa
(jtag_uart_0_avalon_jtag_slave_dataavailable_from_sa),
        .jtag_uart_0_avalon_jtag_slave_irq
(jtag_uart_0_avalon_jtag_slave_irq),
        .jtag_uart_0_avalon_jtag_slave_irq_from_sa
(jtag_uart_0_avalon_jtag_slave_irq_from_sa),
        .jtag_uart_0_avalon_jtag_slave_read_n
(jtag_uart_0_avalon_jtag_slave_read_n),
        .jtag_uart_0_avalon_jtag_slave_readdata
(jtag_uart_0_avalon_jtag_slave_readdata),
        .jtag_uart_0_avalon_jtag_slave_readdata_from_sa
(jtag_uart_0_avalon_jtag_slave_readdata_from_sa),
        .jtag_uart_0_avalon_jtag_slave_readyfordata
(jtag_uart_0_avalon_jtag_slave_readyfordata),
        .jtag_uart_0_avalon_jtag_slave_readyfordata_from_sa
(jtag_uart_0_avalon_jtag_slave_readyfordata_from_sa),
        .jtag_uart_0_avalon_jtag_slave_reset_n
(jtag_uart_0_avalon_jtag_slave_reset_n),
        .jtag_uart_0_avalon_jtag_slave_waitrequest
(jtag_uart_0_avalon_jtag_slave_waitrequest),
        .jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa
(jtag_uart_0_avalon_jtag_slave_waitrequest_from_sa),
        .jtag_uart_0_avalon_jtag_slave_write_n
(jtag_uart_0_avalon_jtag_slave_write_n),
        .jtag_uart_0_avalon_jtag_slave_writedata
(jtag_uart_0_avalon_jtag_slave_writedata),
        .reset_n
(clk_0_reset_n)
      );

   jtag_uart_0 the_jtag_uart_0
     (
       .av_address         (jtag_uart_0_avalon_jtag_slave_address),
       .av_chipselect    (jtag_uart_0_avalon_jtag_slave_chipselect),
       .av_irq               (jtag_uart_0_avalon_jtag_slave_irq),
       .av_read_n          (jtag_uart_0_avalon_jtag_slave_read_n),
       .av_readdata       (jtag_uart_0_avalon_jtag_slave_readdata),
       .av_waitrequest (jtag_uart_0_avalon_jtag_slave_waitrequest),
       .av_write_n         (jtag_uart_0_avalon_jtag_slave_write_n),
       .av_writedata      (jtag_uart_0_avalon_jtag_slave_writedata),
       .clk                   (clk_0),
```

```
        .dataavailable    (jtag_uart_0_avalon_jtag_slave_dataavailable),
        .readyfordata     (jtag_uart_0_avalon_jtag_slave_readyfordata),
        .rst_n            (jtag_uart_0_avalon_jtag_slave_reset_n)
    );

  musiccontroller_0_avalon_slave_0_arbitrator the_musiccontroller_0_avalon_slave_0
    (
      .clk
(clk_0),
      .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
      .cpu_0_data_master_byteenable
(cpu_0_data_master_byteenable),
      .cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_byteenable_musiccontroller_0_avalon_slave_0),
      .cpu_0_data_master_dbs_address
(cpu_0_data_master_dbs_address),
      .cpu_0_data_master_dbs_write_16
(cpu_0_data_master_dbs_write_16),
      .cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_granted_musiccontroller_0_avalon_slave_0),
      .cpu_0_data_master_no_byte_enables_and_last_term
(cpu_0_data_master_no_byte_enables_and_last_term),
      .cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_qualified_request_musiccontroller_0_avalon_slave_0),
      .cpu_0_data_master_read
(cpu_0_data_master_read),
      .cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_musiccontroller_0_avalon_slave_0),
      .cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0
(cpu_0_data_master_requests_musiccontroller_0_avalon_slave_0),
      .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
      .cpu_0_data_master_write
(cpu_0_data_master_write),
      .d1_musiccontroller_0_avalon_slave_0_end_xfer
(d1_musiccontroller_0_avalon_slave_0_end_xfer),
      .musiccontroller_0_avalon_slave_0_address
(musiccontroller_0_avalon_slave_0_address),
      .musiccontroller_0_avalon_slave_0_chipselect
(musiccontroller_0_avalon_slave_0_chipselect),
      .musiccontroller_0_avalon_slave_0_irq
(musiccontroller_0_avalon_slave_0_irq),
      .musiccontroller_0_avalon_slave_0_irq_from_sa
(musiccontroller_0_avalon_slave_0_irq_from_sa),
      .musiccontroller_0_avalon_slave_0_read
(musiccontroller_0_avalon_slave_0_read),
      .musiccontroller_0_avalon_slave_0_readdata
(musiccontroller_0_avalon_slave_0_readdata),
      .musiccontroller_0_avalon_slave_0_readdata_from_sa
(musiccontroller_0_avalon_slave_0_readdata_from_sa),
      .musiccontroller_0_avalon_slave_0_reset_n
(musiccontroller_0_avalon_slave_0_reset_n),
      .musiccontroller_0_avalon_slave_0_write
(musiccontroller_0_avalon_slave_0_write),
      .musiccontroller_0_avalon_slave_0_writedata
(musiccontroller_0_avalon_slave_0_writedata),
      .reset_n
```

```
(clk_0_reset_n)
      );

   musiccontroller_0 the_musiccontroller_0
      (
        .AUD_ADCDAT    (AUD_ADCDAT_to_the_musiccontroller_0),
        .AUD_ADCLRCK (AUD_ADCLRCK_from_the_musiccontroller_0),
        .AUD_BCLK       (AUD_BCLK_to_and_from_the_musiccontroller_0),
        .AUD_DACDAT    (AUD_DACDAT_from_the_musiccontroller_0),
        .AUD_DACLRCK (AUD_DACLRCK_from_the_musiccontroller_0),
        .AUD_XCK         (AUD_XCK_from_the_musiccontroller_0),
        .address         (musiccontroller_0_avalon_slave_0_address),
        .chipselect    (musiccontroller_0_avalon_slave_0_chipselect),
        .clk               (clk_0),
        .clk_18          (clk_18_to_the_musiccontroller_0),
        .irq               (musiccontroller_0_avalon_slave_0_irq),
        .read              (musiccontroller_0_avalon_slave_0_read),
        .readdata       (musiccontroller_0_avalon_slave_0_readdata),
        .reset_n         (musiccontroller_0_avalon_slave_0_reset_n),
        .write             (musiccontroller_0_avalon_slave_0_write),
        .writedata      (musiccontroller_0_avalon_slave_0_writedata)
      );

   sdram_0_s1_arbitrator the_sdram_0_s1
      (
        .clk                                                                    (clk_0),
        .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
        .cpu_0_data_master_byteenable
(cpu_0_data_master_byteenable),
        .cpu_0_data_master_byteenable_sdram_0_s1
(cpu_0_data_master_byteenable_sdram_0_s1),
        .cpu_0_data_master_dbs_address
(cpu_0_data_master_dbs_address),
        .cpu_0_data_master_dbs_write_16
(cpu_0_data_master_dbs_write_16),
        .cpu_0_data_master_granted_sdram_0_s1
(cpu_0_data_master_granted_sdram_0_s1),
        .cpu_0_data_master_no_byte_enables_and_last_term
(cpu_0_data_master_no_byte_enables_and_last_term),
        .cpu_0_data_master_qualified_request_sdram_0_s1
(cpu_0_data_master_qualified_request_sdram_0_s1),
        .cpu_0_data_master_read
(cpu_0_data_master_read),
        .cpu_0_data_master_read_data_valid_sdram_0_s1
(cpu_0_data_master_read_data_valid_sdram_0_s1),
        .cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register
(cpu_0_data_master_read_data_valid_sdram_0_s1_shift_register),
        .cpu_0_data_master_requests_sdram_0_s1
(cpu_0_data_master_requests_sdram_0_s1),
        .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
        .cpu_0_data_master_write
(cpu_0_data_master_write),
        .cpu_0_instruction_master_address_to_slave
(cpu_0_instruction_master_address_to_slave),
        .cpu_0_instruction_master_dbs_address
(cpu_0_instruction_master_dbs_address),
```

```
        .cpu_0_instruction_master_granted_sdram_0_s1
(cpu_0_instruction_master_granted_sdram_0_s1),
        .cpu_0_instruction_master_qualified_request_sdram_0_s1
(cpu_0_instruction_master_qualified_request_sdram_0_s1),
        .cpu_0_instruction_master_read
(cpu_0_instruction_master_read),
        .cpu_0_instruction_master_read_data_valid_sdram_0_s1
(cpu_0_instruction_master_read_data_valid_sdram_0_s1),
        .cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register
(cpu_0_instruction_master_read_data_valid_sdram_0_s1_shift_register),
        .cpu_0_instruction_master_requests_sdram_0_s1
(cpu_0_instruction_master_requests_sdram_0_s1),
        .d1_sdram_0_s1_end_xfer
(d1_sdram_0_s1_end_xfer),
        .reset_n
(clk_0_reset_n),
        .sdram_0_s1_address
(sdram_0_s1_address),
        .sdram_0_s1_byteenable_n
(sdram_0_s1_byteenable_n),
        .sdram_0_s1_chipselect
(sdram_0_s1_chipselect),
        .sdram_0_s1_read_n
(sdram_0_s1_read_n),
        .sdram_0_s1_readdata
(sdram_0_s1_readdata),
        .sdram_0_s1_readdata_from_sa
(sdram_0_s1_readdata_from_sa),
        .sdram_0_s1_readdatavalid
(sdram_0_s1_readdatavalid),
        .sdram_0_s1_reset_n
(sdram_0_s1_reset_n),
        .sdram_0_s1_waitrequest
(sdram_0_s1_waitrequest),
        .sdram_0_s1_waitrequest_from_sa
(sdram_0_s1_waitrequest_from_sa),
        .sdram_0_s1_write_n
(sdram_0_s1_write_n),
        .sdram_0_s1_writedata
(sdram_0_s1_writedata)
      );

   sdram_0 the_sdram_0
     (
       .az_addr             (sdram_0_s1_address),
       .az_be_n             (sdram_0_s1_byteenable_n),
       .az_cs               (sdram_0_s1_chipselect),
       .az_data             (sdram_0_s1_writedata),
       .az_rd_n             (sdram_0_s1_read_n),
       .az_wr_n             (sdram_0_s1_write_n),
       .clk                 (clk_0),
       .reset_n             (sdram_0_s1_reset_n),
       .za_data             (sdram_0_s1_readdata),
       .za_valid            (sdram_0_s1_readdatavalid),
       .za_waitrequest (sdram_0_s1_waitrequest),
       .zs_addr             (zs_addr_from_the_sdram_0),
       .zs_ba               (zs_ba_from_the_sdram_0),
       .zs_cas_n            (zs_cas_n_from_the_sdram_0),
```

```
        .zs_cke               (zs_cke_from_the_sdram_0),
        .zs_cs_n              (zs_cs_n_from_the_sdram_0),
        .zs_dq                (zs_dq_to_and_from_the_sdram_0),
        .zs_dqm               (zs_dqm_from_the_sdram_0),
        .zs_ras_n             (zs_ras_n_from_the_sdram_0),
        .zs_we_n              (zs_we_n_from_the_sdram_0)
    );

  sram_mux_0_avalon_slave_0_arbitrator the_sram_mux_0_avalon_slave_0
    (
      .clk
(clk_0),
      .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
      .cpu_0_data_master_byteenable
(cpu_0_data_master_byteenable),
      .cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0
(cpu_0_data_master_byteenable_sram_mux_0_avalon_slave_0),
      .cpu_0_data_master_dbs_address
(cpu_0_data_master_dbs_address),
      .cpu_0_data_master_dbs_write_16
(cpu_0_data_master_dbs_write_16),
      .cpu_0_data_master_granted_sram_mux_0_avalon_slave_0
(cpu_0_data_master_granted_sram_mux_0_avalon_slave_0),
      .cpu_0_data_master_no_byte_enables_and_last_term
(cpu_0_data_master_no_byte_enables_and_last_term),
      .cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0
(cpu_0_data_master_qualified_request_sram_mux_0_avalon_slave_0),
      .cpu_0_data_master_read
(cpu_0_data_master_read),
      .cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_sram_mux_0_avalon_slave_0),
      .cpu_0_data_master_requests_sram_mux_0_avalon_slave_0
(cpu_0_data_master_requests_sram_mux_0_avalon_slave_0),
      .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
      .cpu_0_data_master_write
(cpu_0_data_master_write),
      .cpu_0_instruction_master_address_to_slave
(cpu_0_instruction_master_address_to_slave),
      .cpu_0_instruction_master_dbs_address
(cpu_0_instruction_master_dbs_address),
      .cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0
(cpu_0_instruction_master_granted_sram_mux_0_avalon_slave_0),
      .cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0
(cpu_0_instruction_master_qualified_request_sram_mux_0_avalon_slave_0),
      .cpu_0_instruction_master_read
(cpu_0_instruction_master_read),
      .cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0
(cpu_0_instruction_master_read_data_valid_sram_mux_0_avalon_slave_0),
      .cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0
(cpu_0_instruction_master_requests_sram_mux_0_avalon_slave_0),
      .d1_sram_mux_0_avalon_slave_0_end_xfer
(d1_sram_mux_0_avalon_slave_0_end_xfer),
      .reset_n
(clk_0_reset_n),
      .sram_mux_0_avalon_slave_0_address
(sram_mux_0_avalon_slave_0_address),
```

```
        .sram_mux_0_avalon_slave_0_read
(sram_mux_0_avalon_slave_0_read),
        .sram_mux_0_avalon_slave_0_readdata
(sram_mux_0_avalon_slave_0_readdata),
        .sram_mux_0_avalon_slave_0_readdata_from_sa
(sram_mux_0_avalon_slave_0_readdata_from_sa),
        .sram_mux_0_avalon_slave_0_write
(sram_mux_0_avalon_slave_0_write),
        .sram_mux_0_avalon_slave_0_writedata
(sram_mux_0_avalon_slave_0_writedata)
      );

   sram_mux_0 the_sram_mux_0
      (
        .SRAM_ADDR (SRAM_ADDR_from_the_sram_mux_0),
        .SRAM_CE_N (SRAM_CE_N_from_the_sram_mux_0),
        .SRAM_DQ     (SRAM_DQ_to_and_from_the_sram_mux_0),
        .SRAM_LB_N (SRAM_LB_N_from_the_sram_mux_0),
        .SRAM_OE_N (SRAM_OE_N_from_the_sram_mux_0),
        .SRAM_UB_N (SRAM_UB_N_from_the_sram_mux_0),
        .SRAM_WE_N (SRAM_WE_N_from_the_sram_mux_0),
        .VGA_ADDR    (VGA_ADDR_to_the_sram_mux_0),
        .VGA_DQ      (VGA_DQ_from_the_sram_mux_0),
        .VGA_OE_N    (VGA_OE_N_to_the_sram_mux_0),
        .VGA_WE_N    (VGA_WE_N_to_the_sram_mux_0),
        .address     (sram_mux_0_avalon_slave_0_address),
        .read        (sram_mux_0_avalon_slave_0_read),
        .readdata    (sram_mux_0_avalon_slave_0_readdata),
        .write       (sram_mux_0_avalon_slave_0_write),
        .writedata (sram_mux_0_avalon_slave_0_writedata)
      );

   xycoord_interface_0_avalon_slave_0_arbitrator the_xycoord_interface_0_avalon_slave_0
      (
        .clk
(clk_0),
        .cpu_0_data_master_address_to_slave
(cpu_0_data_master_address_to_slave),
        .cpu_0_data_master_byteenable
(cpu_0_data_master_byteenable),
        .cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_byteenable_xycoord_interface_0_avalon_slave_0),
        .cpu_0_data_master_dbs_address
(cpu_0_data_master_dbs_address),
        .cpu_0_data_master_dbs_write_16
(cpu_0_data_master_dbs_write_16),
        .cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_granted_xycoord_interface_0_avalon_slave_0),
        .cpu_0_data_master_no_byte_enables_and_last_term
(cpu_0_data_master_no_byte_enables_and_last_term),
        .cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_qualified_request_xycoord_interface_0_avalon_slave_0),
        .cpu_0_data_master_read
(cpu_0_data_master_read),
        .cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_read_data_valid_xycoord_interface_0_avalon_slave_0),
        .cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0
(cpu_0_data_master_requests_xycoord_interface_0_avalon_slave_0),
```

```
        .cpu_0_data_master_waitrequest
(cpu_0_data_master_waitrequest),
        .cpu_0_data_master_write
(cpu_0_data_master_write),
        .cpu_0_instruction_master_address_to_slave
(cpu_0_instruction_master_address_to_slave),
        .cpu_0_instruction_master_dbs_address
(cpu_0_instruction_master_dbs_address),
        .cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0
(cpu_0_instruction_master_granted_xycoord_interface_0_avalon_slave_0),
        .cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0
(cpu_0_instruction_master_qualified_request_xycoord_interface_0_avalon_slave_0),
        .cpu_0_instruction_master_read
(cpu_0_instruction_master_read),
        .cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0
(cpu_0_instruction_master_read_data_valid_xycoord_interface_0_avalon_slave_0),
        .cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0
(cpu_0_instruction_master_requests_xycoord_interface_0_avalon_slave_0),
        .d1_xycoord_interface_0_avalon_slave_0_end_xfer
(d1_xycoord_interface_0_avalon_slave_0_end_xfer),
        .reset_n
(clk_0_reset_n),
        .xycoord_interface_0_avalon_slave_0_address
(xycoord_interface_0_avalon_slave_0_address),
        .xycoord_interface_0_avalon_slave_0_chipselect
(xycoord_interface_0_avalon_slave_0_chipselect),
        .xycoord_interface_0_avalon_slave_0_read
(xycoord_interface_0_avalon_slave_0_read),
        .xycoord_interface_0_avalon_slave_0_readdata
(xycoord_interface_0_avalon_slave_0_readdata),
        .xycoord_interface_0_avalon_slave_0_readdata_from_sa
(xycoord_interface_0_avalon_slave_0_readdata_from_sa),
        .xycoord_interface_0_avalon_slave_0_reset_n
(xycoord_interface_0_avalon_slave_0_reset_n),
        .xycoord_interface_0_avalon_slave_0_write
(xycoord_interface_0_avalon_slave_0_write),
        .xycoord_interface_0_avalon_slave_0_writedata
(xycoord_interface_0_avalon_slave_0_writedata)
      );

   xycoord_interface_0 the_xycoord_interface_0
      (
        .address       (xycoord_interface_0_avalon_slave_0_address),
        .chipselect (xycoord_interface_0_avalon_slave_0_chipselect),
        .clk              (clk_0),
        .iX_COORD      (iX_COORD_to_the_xycoord_interface_0),
        .iY_COORD      (iY_COORD_to_the_xycoord_interface_0),
        .read             (xycoord_interface_0_avalon_slave_0_read),
        .readdata      (xycoord_interface_0_avalon_slave_0_readdata),
        .reset_n       (xycoord_interface_0_avalon_slave_0_reset_n),
        .write           (xycoord_interface_0_avalon_slave_0_write),
        .writedata    (xycoord_interface_0_avalon_slave_0_writedata)
      );

   //reset is asserted asynchronously and deasserted synchronously
   nios_system_reset_clk_0_domain_synch_module nios_system_reset_clk_0_domain_synch
      (
        .clk          (clk_0),
```

```
          .data_in    (1'b1),
          .data_out (clk_0_reset_n),
          .reset_n    (reset_n_sources)
       );

   //reset sources mux, which is an e_mux
   assign reset_n_sources = ~(~reset_n |
      0 |
      cpu_0_jtag_debug_module_resetrequest_from_sa |
      cpu_0_jtag_debug_module_resetrequest_from_sa);


endmodule


//synthesis translate_off



// <ALTERA_NOTE> CODE INSERTED BETWEEN HERE

// AND HERE WILL BE PRESERVED </ALTERA_NOTE>


// If user logic components use Altsync_Ram with convert_hex2ver.dll,
// set USE_convert_hex2ver in the user comments section above

// `ifdef USE_convert_hex2ver
// `else
// `define NO_PLI 1
// `endif

`include "/opt/altera/altera12.1/quartus/eda/sim_lib/altera_mf.v"
`include "/opt/altera/altera12.1/quartus/eda/sim_lib/220model.v"
`include "/opt/altera/altera12.1/quartus/eda/sim_lib/sgate.v"
`include "interrupt.v"
`include "command.v"
`include "command_0.v"
`include "xycoord_interface.v"
`include "xycoord_interface_0.v"
`include "sram_mux.v"
`include "sram_mux_0.v"
// musiccontroller.vhd
// musiccontroller_0.vhd
`include "audio_controller.v"
`include "audio_controller_0.v"
`include "sdram_0.v"
`include "cpu_0_test_bench.v"
`include "cpu_0_oci_test_bench.v"
`include "cpu_0_jtag_debug_module_tck.v"
`include "cpu_0_jtag_debug_module_sysclk.v"
`include "cpu_0_jtag_debug_module_wrapper.v"
`include "cpu_0.v"
`include "jtag_uart_0.v"

`timescale 1ns / 1ps

module test_bench
```

```
;


   wire              AUD_ADCDAT_to_the_musiccontroller_0;
   wire              AUD_ADCLRCK_from_the_musiccontroller_0;
   wire              AUD_BCLK_to_and_from_the_musiccontroller_0;
   wire              AUD_DACDAT_from_the_musiccontroller_0;
   wire              AUD_DACLRCK_from_the_musiccontroller_0;
   wire              AUD_XCK_from_the_musiccontroller_0;
   wire      [ 17: 0] SRAM_ADDR_from_the_sram_mux_0;
   wire              SRAM_CE_N_from_the_sram_mux_0;
   wire      [ 15: 0] SRAM_DQ_to_and_from_the_sram_mux_0;
   wire              SRAM_LB_N_from_the_sram_mux_0;
   wire              SRAM_OE_N_from_the_sram_mux_0;
   wire              SRAM_UB_N_from_the_sram_mux_0;
   wire              SRAM_WE_N_from_the_sram_mux_0;
   wire      [ 17: 0] VGA_ADDR_to_the_sram_mux_0;
   wire      [ 15: 0] VGA_DQ_from_the_sram_mux_0;
   wire              VGA_OE_N_to_the_sram_mux_0;
   wire              VGA_WE_N_to_the_sram_mux_0;
   wire              clk;
   reg               clk_0;
   wire              clk_18_to_the_musiccontroller_0;
   wire      [ 11: 0] iX_COORD_to_the_xycoord_interface_0;
   wire      [ 11: 0] iY_COORD_to_the_xycoord_interface_0;
   wire              irq_adc_to_the_interrupt_0;
   wire              jtag_uart_0_avalon_jtag_slave_dataavailable_from_sa;
   wire              jtag_uart_0_avalon_jtag_slave_readyfordata_from_sa;
   wire      [ 31: 0] oMOLECTRL_from_the_command_0;
   wire      [ 31: 0] oPANELCTRL_from_the_command_0;
   reg               reset_n;
   wire      [ 11: 0] zs_addr_from_the_sdram_0;
   wire      [  1: 0] zs_ba_from_the_sdram_0;
   wire              zs_cas_n_from_the_sdram_0;
   wire              zs_cke_from_the_sdram_0;
   wire              zs_cs_n_from_the_sdram_0;
   wire      [ 15: 0] zs_dq_to_and_from_the_sdram_0;
   wire      [  1: 0] zs_dqm_from_the_sdram_0;
   wire              zs_ras_n_from_the_sdram_0;
   wire              zs_we_n_from_the_sdram_0;


// <ALTERA_NOTE> CODE INSERTED BETWEEN HERE
//    add your signals and additional architecture here
// AND HERE WILL BE PRESERVED </ALTERA_NOTE>

   //Set us up the Dut
   nios_system DUT
      (
        .AUD_ADCDAT_to_the_musiccontroller_0
(AUD_ADCDAT_to_the_musiccontroller_0),
        .AUD_ADCLRCK_from_the_musiccontroller_0
(AUD_ADCLRCK_from_the_musiccontroller_0),
        .AUD_BCLK_to_and_from_the_musiccontroller_0
(AUD_BCLK_to_and_from_the_musiccontroller_0),
        .AUD_DACDAT_from_the_musiccontroller_0
(AUD_DACDAT_from_the_musiccontroller_0),
        .AUD_DACLRCK_from_the_musiccontroller_0
```

```
(AUD_DACLRCK_from_the_musiccontroller_0),
        .AUD_XCK_from_the_musiccontroller_0
(AUD_XCK_from_the_musiccontroller_0),
        .SRAM_ADDR_from_the_sram_mux_0
(SRAM_ADDR_from_the_sram_mux_0),
        .SRAM_CE_N_from_the_sram_mux_0
(SRAM_CE_N_from_the_sram_mux_0),
        .SRAM_DQ_to_and_from_the_sram_mux_0
(SRAM_DQ_to_and_from_the_sram_mux_0),
        .SRAM_LB_N_from_the_sram_mux_0
(SRAM_LB_N_from_the_sram_mux_0),
        .SRAM_OE_N_from_the_sram_mux_0
(SRAM_OE_N_from_the_sram_mux_0),
        .SRAM_UB_N_from_the_sram_mux_0
(SRAM_UB_N_from_the_sram_mux_0),
        .SRAM_WE_N_from_the_sram_mux_0
(SRAM_WE_N_from_the_sram_mux_0),
        .VGA_ADDR_to_the_sram_mux_0
(VGA_ADDR_to_the_sram_mux_0),
        .VGA_DQ_from_the_sram_mux_0
(VGA_DQ_from_the_sram_mux_0),
        .VGA_OE_N_to_the_sram_mux_0
(VGA_OE_N_to_the_sram_mux_0),
        .VGA_WE_N_to_the_sram_mux_0
(VGA_WE_N_to_the_sram_mux_0),
        .clk_0                                  (clk_0),
        .clk_18_to_the_musiccontroller_0        (clk_18_to_the_musiccontroller_0),
        .iX_COORD_to_the_xycoord_interface_0
(iX_COORD_to_the_xycoord_interface_0),
        .iY_COORD_to_the_xycoord_interface_0
(iY_COORD_to_the_xycoord_interface_0),
        .irq_adc_to_the_interrupt_0             (irq_adc_to_the_interrupt_0),
        .oMOLECTRL_from_the_command_0
(oMOLECTRL_from_the_command_0),
        .oPANELCTRL_from_the_command_0
(oPANELCTRL_from_the_command_0),
        .reset_n                                (reset_n),
        .zs_addr_from_the_sdram_0               (zs_addr_from_the_sdram_0),
        .zs_ba_from_the_sdram_0                 (zs_ba_from_the_sdram_0),
        .zs_cas_n_from_the_sdram_0              (zs_cas_n_from_the_sdram_0),
        .zs_cke_from_the_sdram_0                (zs_cke_from_the_sdram_0),
        .zs_cs_n_from_the_sdram_0               (zs_cs_n_from_the_sdram_0),
        .zs_dq_to_and_from_the_sdram_0          (zs_dq_to_and_from_the_sdram_0),
        .zs_dqm_from_the_sdram_0                (zs_dqm_from_the_sdram_0),
        .zs_ras_n_from_the_sdram_0              (zs_ras_n_from_the_sdram_0),
        .zs_we_n_from_the_sdram_0               (zs_we_n_from_the_sdram_0)
    );

  initial
    clk_0 = 1'b0;
  always
    #10 clk_0 <= ~clk_0;

  initial
    begin
      reset_n <= 0;
      #200 reset_n <= 1;
    end
```

```
endmodule

//synthesis translate_on
```

Adc_spi_controller.v

```
module adc_spi_controller    (
                        iCLK,

                        iRST_n,
                        oADC_DIN,
                        oADC_DCLK,
                        oADC_CS,
                        iADC_DOUT,
                        iADC_BUSY,
                        iADC_PENIRQ_n,
                        oTOUCH_IRQ,
                        oX_COORD,
                        oY_COORD,
                        oNEW_COORD,
                        ////////////////////
                        );


//=============================================================================
// PARAMETER declarations
//=============================================================================

parameter SYSCLK_FRQ  = 50000000;
parameter ADC_DCLK_FRQ    = 1000;
```

```
parameter ADC_DCLK_CNT    = SYSCLK_FRQ/(ADC_DCLK_FRQ*2);


//=============================================================================
// PORT declarations
//=============================================================================
input                iCLK;
input                iRST_n;
input                iADC_DOUT;
input                iADC_PENIRQ_n;
input                iADC_BUSY;
output               oADC_DIN;
output               oADC_DCLK;
output               oADC_CS;
output               oTOUCH_IRQ;
output    [11:0]     oX_COORD;
output    [11:0]     oY_COORD;
output               oNEW_COORD;
//=============================================================================
// REG/WIRE declarations
//=============================================================================
reg                  d1_PENIRQ_n;
reg                  d2_PENIRQ_n;
wire            touch_irq;
reg       [15:0]     dclk_cnt;
wire            dclk;
reg                  transmit_en;
reg       [6:0] spi_ctrl_cnt;
wire            oADC_CS;
reg                  mcs;
```

```
reg                     mdclk;
wire [7:0] x_config_reg;
wire [7:0] y_config_reg;
wire [7:0] ctrl_reg;
reg      [7:0] mdata_in;
reg                     y_coordinate_config;
wire            eof_transmition;
reg      [5:0] bit_cnt;
reg                     madc_out;
reg      [11:0]     mx_coordinate;
reg      [11:0]     my_coordinate;
reg      [11:0]     oX_COORD;
reg      [11:0]     oY_COORD;
wire            rd_coord_strob;
reg                     oNEW_COORD;
reg      [5:0] irq_cnt;
reg      [15:0]     clk_cnt;
//=============================================================================
// Structural coding
//=============================================================================
assign     x_config_reg = 8'h92;
assign     y_config_reg = 8'hd2;

always@(posedge iCLK or negedge iRST_n)
    begin
        if (!iRST_n)
            madc_out <= 0;
        else
            madc_out <= iADC_DOUT;
```

```
    end

///////////////     pen irq detect    ////////
always@(posedge iCLK or negedge iRST_n)
    begin
        if (!iRST_n)
            begin
                d1_PENIRQ_n <= 0;
                d2_PENIRQ_n <= 0;
            end
        else
            begin
                d1_PENIRQ_n <= iADC_PENIRQ_n;
                d2_PENIRQ_n <= d1_PENIRQ_n;
            end
    end

// if iADC_PENIRQ_n form high to low , touch_irq goes high
assign          touch_irq = d2_PENIRQ_n & ~d1_PENIRQ_n;
assign          oTOUCH_IRQ = touch_irq;
// if touch_irq goes high , starting transmit procedure ,transmit_en goes high
// if end of transmition and no penirq , transmit procedure stop.

always@(posedge iCLK or negedge iRST_n)
    begin
        if (!iRST_n)
            transmit_en <= 0;
        else if (eof_transmition&&iADC_PENIRQ_n)
            transmit_en <= 0;
```

```
        else if (touch_irq)
            transmit_en <= 1;
    end

always@(posedge iCLK or negedge iRST_n)
    begin
        if (!iRST_n)
            dclk_cnt <= 0;
        else if (transmit_en)
            begin
                if (dclk_cnt == ADC_DCLK_CNT)
                    dclk_cnt <= 0;
                else
                    dclk_cnt <= dclk_cnt + 1;
            end
        else
            dclk_cnt <= 0;
    end

assign  dclk =  (dclk_cnt == ADC_DCLK_CNT)? 1 : 0;

always@(posedge iCLK or negedge iRST_n)
    begin
        if (!iRST_n)
            spi_ctrl_cnt <= 0;
        else if (dclk)
            begin
                if (spi_ctrl_cnt == 65)
                    spi_ctrl_cnt <= 0;
```

```
                else
                    spi_ctrl_cnt <= spi_ctrl_cnt + 1;
            end
    end

always@(posedge iCLK or negedge iRST_n)
    begin
        if (!iRST_n)
            begin
                mcs         <= 1;
                mdclk       <= 0;
                mdata_in <= 0;
                y_coordinate_config <= 0;
                mx_coordinate <= 0;
                my_coordinate <= 0;
            end
        else if (transmit_en)
            begin
                if (dclk)
                    begin
                        if (spi_ctrl_cnt == 0)
                            begin
                                mcs         <= 0;
                                mdata_in <= ctrl_reg;
                            end
                        else if (spi_ctrl_cnt == 49)
                            begin
                                mdclk       <= 0;
                                y_coordinate_config <= ~y_coordinate_config;
```

```
                        if (y_coordinate_config)
                            mcs     <= 1;
                        else
                            mcs     <= 0;
                    end
                else if (spi_ctrl_cnt != 0)
                    mdclk   <= ~mdclk;
                if (mdclk)
                    mdata_in <= {mdata_in[6:0],1'b0};
                if (!mdclk)
                    begin
                        if(rd_coord_strob)
                            begin
                                if(y_coordinate_config)
                                    my_coordinate                   <=
{my_coordinate[10:0],madc_out};
                                else
                                    mx_coordinate                   <=
{mx_coordinate[10:0],madc_out};
                            end
                    end
            end
        end
    end

assign  oADC_CS    = mcs;
assign  oADC_DIN =    mdata_in[7];
assign  oADC_DCLK = mdclk;
assign  ctrl_reg = y_coordinate_config ? y_config_reg : x_config_reg;
```

```
assign      eof_transmition = (y_coordinate_config & (spi_ctrl_cnt == 49) & dclk);

assign      rd_coord_strob = ((spi_ctrl_cnt>=19)&&(spi_ctrl_cnt<=41)) ? 1 : 0;

always@(posedge iCLK or negedge iRST_n)
      begin
            if (!iRST_n)
                  begin
                        oX_COORD <= 0;
                        oY_COORD <= 0;
                  end
            else if (eof_transmition&&(my_coordinate!=0))
                  begin
                        oX_COORD <= mx_coordinate;
                        oY_COORD <= my_coordinate;
                  end
      end

always@(posedge iCLK or negedge iRST_n)
      begin
            if (!iRST_n)
                  oNEW_COORD <= 0;
            else if (eof_transmition&&(my_coordinate!=0))
                  oNEW_COORD <= 1;
            else
                  oNEW_COORD <= 0;
      end
```

```
endmodule
```

Audio_controller.v

```
module audio_controller         (
                                clk,                    // LCD display clock
                                reset_n,                // systen reset
                                read,
                                write,
                                chipselect,
                                address,
                                readdata,
                                writedata
                                );


input           clk;
input           reset_n;
input           read;
input           write;
input           chipselect;
input [4:0]address;
output [15:0]readdata;
input [15:0]writedata;

endmodule
```

Command.v

```
module command(
                clk,
                reset_n,
                chipselect,
                read,
                write,
                address,
                readdata,
                writedata,
                oMOLECTRL,
                oPANELCTRL
                );

//================================
//IO PORT
//================================
input clk;
input reset_n;
input chipselect;
input read;
input write;
input [4:0] address;
output [31:0] readdata;
input [31:0] writedata;
```

```
output [31:0]oMOLECTRL;
output [31:0]oPANELCTRL;

//====================================
//REG and WIRE
//====================================
reg [31:0] mole;
reg [31:0] panel;



//====================================
//STRUCTURE LOGIC
//====================================
always@(posedge clk or negedge reset_n)
begin
	if(!reset_n)
		begin
			mole <= 0;
			panel <= 0;
		end
	else if (chipselect)
		begin
			if(write)
			begin
				if(address == 5'b00000)
					panel <= writedata;
				else if(address == 5'b00001)
					mole <= writedata;
			end
		end
end

assign oMOLECTRL = mole;
assign oPANELCTRL = panel;

endmodule
```

Control_panel.v

```
module control_panel(
xcount,
ycount,

)

endmodule
```

Cpu_0.v

```
//Legal Notice: (C)2013 Altera Corporation. All rights reserved.  Your
//use of Altera Corporation's design tools, logic functions and other
//software and tools, and its AMPP partner logic functions, and any
//output files any of the foregoing (including device programming or
//simulation files), and any associated documentation or information are
```

```
//expressly subject to the terms and conditions of the Altera Program
//License Subscription Agreement or other applicable license agreement,
//including, without limitation, that your use is for the sole purpose
//of programming logic devices manufactured by Altera and sold by Altera
//or its authorized distributors.  Please refer to the applicable
//agreement for further details.

// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_register_bank_a_module (
                                      // inputs:
                                       clock,
                                       data,
                                       rdaddress,
                                       wraddress,
                                       wren,

                                      // outputs:
                                       q
                                    )
;

  parameter lpm_file = "UNUSED";


  output  [ 31: 0] q;
  input            clock;
  input   [ 31: 0] data;
  input   [  4: 0] rdaddress;
  input   [  4: 0] wraddress;
  input            wren;

  wire    [ 31: 0] q;
  wire    [ 31: 0] ram_q;
  assign q = ram_q;
  altsyncram the_altsyncram
    (
      .address_a (wraddress),
      .address_b (rdaddress),
      .clock0 (clock),
      .data_a (data),
      .q_b (ram_q),
      .wren_a (wren)
    );

  defparam the_altsyncram.address_reg_b = "CLOCK0",
           the_altsyncram.init_file = lpm_file,
           the_altsyncram.maximum_depth = 0,
           the_altsyncram.numwords_a = 32,
           the_altsyncram.numwords_b = 32,
           the_altsyncram.operation_mode = "DUAL_PORT",
           the_altsyncram.outdata_reg_b = "UNREGISTERED",
```

```
                the_altsyncram.ram_block_type = "AUTO",
                the_altsyncram.rdcontrol_reg_b = "CLOCK0",
                the_altsyncram.read_during_write_mode_mixed_ports = "DONT_CARE",
                the_altsyncram.width_a = 32,
                the_altsyncram.width_b = 32,
                the_altsyncram.widthad_a = 5,
                the_altsyncram.widthad_b = 5;


endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_register_bank_b_module (
                                                    // inputs:
                                                      clock,
                                                      data,
                                                      rdaddress,
                                                      wraddress,
                                                      wren,

                                                    // outputs:
                                                      q
                                                   )
;

   parameter lpm_file = "UNUSED";


   output    [ 31: 0] q;
   input              clock;
   input     [ 31: 0] data;
   input     [  4: 0] rdaddress;
   input     [  4: 0] wraddress;
   input              wren;

   wire      [ 31: 0] q;
   wire      [ 31: 0] ram_q;
   assign q = ram_q;
   altsyncram the_altsyncram
     (
       .address_a (wraddress),
       .address_b (rdaddress),
       .clock0 (clock),
       .data_a (data),
       .q_b (ram_q),
       .wren_a (wren)
     );

   defparam the_altsyncram.address_reg_b = "CLOCK0",
                the_altsyncram.init_file = lpm_file,
```

```
                the_altsyncram.maximum_depth = 0,
                the_altsyncram.numwords_a = 32,
                the_altsyncram.numwords_b = 32,
                the_altsyncram.operation_mode = "DUAL_PORT",
                the_altsyncram.outdata_reg_b = "UNREGISTERED",
                the_altsyncram.ram_block_type = "AUTO",
                the_altsyncram.rdcontrol_reg_b = "CLOCK0",
                the_altsyncram.read_during_write_mode_mixed_ports = "DONT_CARE",
                the_altsyncram.width_a = 32,
                the_altsyncram.width_b = 32,
                the_altsyncram.widthad_a = 5,
                the_altsyncram.widthad_b = 5;


endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_debug (
                                        // inputs:
                                          clk,
                                          dbrk_break,
                                          debugreq,
                                          hbreak_enabled,
                                          jdo,
                                          jrst_n,
                                          ocireg_ers,
                                          ocireg_mrs,
                                          reset,
                                          st_ready_test_idle,
                                          take_action_ocimem_a,
                                          take_action_ocireg,
                                          xbrk_break,

                                        // outputs:
                                          debugack,
                                          monitor_error,
                                          monitor_go,
                                          monitor_ready,
                                          oci_hbreak_req,
                                          resetlatch,
                                          resetrequest
                                       )
;

  output           debugack;
  output           monitor_error;
  output           monitor_go;
  output           monitor_ready;
  output           oci_hbreak_req;
  output           resetlatch;
```

```
  output              resetrequest;
  input               clk;
  input               dbrk_break;
  input               debugreq;
  input               hbreak_enabled;
  input    [ 37: 0] jdo;
  input               jrst_n;
  input               ocireg_ers;
  input               ocireg_mrs;
  input               reset;
  input               st_ready_test_idle;
  input               take_action_ocimem_a;
  input               take_action_ocireg;
  input               xbrk_break;

  wire                debugack;
  reg                                 jtag_break   /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
  reg                               monitor_error  /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=D101" */;
  reg                                 monitor_go   /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=D101" */;
  reg                               monitor_ready  /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=D101" */;
  wire                oci_hbreak_req;
  reg                                 probepresent /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
  reg                                   resetlatch /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
  reg                                 resetrequest /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
  always @(posedge clk or negedge jrst_n)
    begin
      if (jrst_n == 0)
        begin
          probepresent <= 1'b0;
          resetrequest <= 1'b0;
          jtag_break <= 1'b0;
        end
      else if (take_action_ocimem_a)
        begin
          resetrequest <= jdo[22];
          jtag_break <= jdo[21]       ? 1
                      : jdo[20]    ? 0
                      : jtag_break;

          probepresent <= jdo[19]       ? 1
                      : jdo[18]    ? 0
                      :    probepresent;

          resetlatch <= jdo[24] ? 0 : resetlatch;
        end
      else if (reset)
        begin
          jtag_break <= probepresent;
          resetlatch <= 1;
        end
      else if (~debugack & debugreq & probepresent)
```

```
                jtag_break <= 1'b1;
        end


    always @(posedge clk or negedge jrst_n)
        begin
            if (jrst_n == 0)
                begin
                    monitor_ready <= 1'b0;
                    monitor_error <= 1'b0;
                    monitor_go <= 1'b0;
                end
            else
                begin
                    if (take_action_ocimem_a && jdo[25])
                            monitor_ready <= 1'b0;
                    else if (take_action_ocireg && ocireg_mrs)
                            monitor_ready <= 1'b1;
                    if (take_action_ocimem_a && jdo[25])
                            monitor_error <= 1'b0;
                    else if (take_action_ocireg && ocireg_ers)
                            monitor_error <= 1'b1;
                    if (take_action_ocimem_a && jdo[23])
                            monitor_go <= 1'b1;
                    else if (st_ready_test_idle)
                            monitor_go <= 1'b0;
                end
        end


    assign oci_hbreak_req = jtag_break | dbrk_break | xbrk_break | debugreq;
    assign debugack = ~hbreak_enabled;

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_ociram_lpm_dram_bdp_component_module (
                                                                        // inputs:
                                                                          address_a,
                                                                          address_b,
                                                                          byteena_a,
                                                                          clock0,
                                                                          clock1,
                                                                          clocken0,
                                                                          clocken1,
                                                                          data_a,
                                                                          data_b,
                                                                          wren_a,
                                                                          wren_b,
```

```
                                                                    // outputs:
                                                                      q_a,
                                                                      q_b
                                                                   )
;

   parameter lpm_file = "UNUSED";


   output    [ 31: 0] q_a;
   output    [ 31: 0] q_b;
   input     [    7: 0] address_a;
   input     [    7: 0] address_b;
   input     [    3: 0] byteena_a;
   input                clock0;
   input                clock1;
   input                clocken0;
   input                clocken1;
   input     [ 31: 0] data_a;
   input     [ 31: 0] data_b;
   input                wren_a;
   input                wren_b;

   wire      [ 31: 0] q_a;
   wire      [ 31: 0] q_b;
   altsyncram the_altsyncram
     (
       .address_a (address_a),
       .address_b (address_b),
       .byteena_a (byteena_a),
       .clock0 (clock0),
       .clock1 (clock1),
       .clocken0 (clocken0),
       .clocken1 (clocken1),
       .data_a (data_a),
       .data_b (data_b),
       .q_a (q_a),
       .q_b (q_b),
       .wren_a (wren_a),
       .wren_b (wren_b)
     );

   defparam the_altsyncram.address_aclr_a = "NONE",
            the_altsyncram.address_aclr_b = "NONE",
            the_altsyncram.address_reg_b = "CLOCK1",
            the_altsyncram.indata_aclr_a = "NONE",
            the_altsyncram.indata_aclr_b = "NONE",
            the_altsyncram.init_file = lpm_file,
            the_altsyncram.intended_device_family = "CYCLONEII",
            the_altsyncram.lpm_type = "altsyncram",
            the_altsyncram.numwords_a = 256,
            the_altsyncram.numwords_b = 256,
            the_altsyncram.operation_mode = "BIDIR_DUAL_PORT",
            the_altsyncram.outdata_aclr_a = "NONE",
            the_altsyncram.outdata_aclr_b = "NONE",
            the_altsyncram.outdata_reg_a = "UNREGISTERED",
            the_altsyncram.outdata_reg_b = "UNREGISTERED",
            the_altsyncram.ram_block_type = "AUTO",
```

```
                the_altsyncram.read_during_write_mode_mixed_ports = "OLD_DATA",
                the_altsyncram.width_a = 32,
                the_altsyncram.width_b = 32,
                the_altsyncram.width_byteena_a = 4,
                the_altsyncram.widthad_a = 8,
                the_altsyncram.widthad_b = 8,
                the_altsyncram.wrcontrol_aclr_a = "NONE",
                the_altsyncram.wrcontrol_aclr_b = "NONE";


endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_ocimem (
                                    // inputs:
                                      address,
                                      begintransfer,
                                      byteenable,
                                      chipselect,
                                      clk,
                                      debugaccess,
                                      jdo,
                                      jrst_n,
                                      resetrequest,
                                      take_action_ocimem_a,
                                      take_action_ocimem_b,
                                      take_no_action_ocimem_a,
                                      write,
                                      writedata,

                                    // outputs:
                                      MonDReg,
                                      oci_ram_readdata
                                   )
;

  output  [ 31: 0] MonDReg;
  output  [ 31: 0] oci_ram_readdata;
  input   [  8: 0] address;
  input            begintransfer;
  input   [  3: 0] byteenable;
  input            chipselect;
  input            clk;
  input            debugaccess;
  input   [ 37: 0] jdo;
  input            jrst_n;
  input            resetrequest;
  input            take_action_ocimem_a;
  input            take_action_ocimem_b;
  input            take_no_action_ocimem_a;
```

```
  input                  write;
  input     [ 31: 0] writedata;

  reg                  [  10:  0] MonAReg /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,D103,R101\"" */;
  reg                  [  31:  0] MonDReg /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,D103,R101\"" */;
  reg                                     MonRd /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,D103,R101\"" */;
  reg                                    MonRd1 /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,D103,R101\"" */;
  reg                                     MonWr /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,D103,R101\"" */;
  wire                   avalon;
  wire      [ 31: 0] cfgdout;
  wire      [ 31: 0] oci_ram_readdata;
  wire      [ 31: 0] sramdout;
  assign avalon = begintransfer & ~resetrequest;
  always @(posedge clk or negedge jrst_n)
    begin
      if (jrst_n == 0)
        begin
          MonWr <= 1'b0;
          MonRd <= 1'b0;
          MonRd1 <= 1'b0;
          MonAReg <= 0;
          MonDReg <= 0;
        end
      else
        begin
          if (take_no_action_ocimem_a)
            begin
              MonAReg[10 : 2] <= MonAReg[10 : 2]+1;
              MonRd <= 1'b1;
            end
          else if (take_action_ocimem_a)
            begin
              MonAReg[10 : 2] <= { jdo[17],
                                   jdo[33 : 26] };

              MonRd <= 1'b1;
            end
          else if (take_action_ocimem_b)
            begin
              MonAReg[10 : 2] <= MonAReg[10 : 2]+1;
              MonDReg <= jdo[34 : 3];
              MonWr <= 1'b1;
            end
          else
            begin
              if (~avalon)
                begin
                  MonWr <= 0;
                  MonRd <= 0;
                end
              if (MonRd1)
                  MonDReg <= MonAReg[10] ? cfgdout : sramdout;
            end
```

```
            MonRd1 <= MonRd;
          end
      end


//cpu_0_ociram_lpm_dram_bdp_component, which is an nios_tdp_ram
cpu_0_ociram_lpm_dram_bdp_component_module cpu_0_ociram_lpm_dram_bdp_component
   (
      .address_a (address[7 : 0]),
      .address_b (MonAReg[9 : 2]),
      .byteena_a (byteenable),
      .clock0    (clk),
      .clock1    (clk),
      .clocken0  (1'b1),
      .clocken1  (1'b1),
      .data_a    (writedata),
      .data_b    (MonDReg[31 : 0]),
      .q_a       (oci_ram_readdata),
      .q_b       (sramdout),
      .wren_a    (chipselect & write & debugaccess &
                              ~address[8]
                              ),
      .wren_b    (MonWr)
   );

//synthesis translate_off
`ifdef NO_PLI
defparam                cpu_0_ociram_lpm_dram_bdp_component.lpm_file                =
"cpu_0_ociram_default_contents.dat";
`else
defparam                cpu_0_ociram_lpm_dram_bdp_component.lpm_file                =
"cpu_0_ociram_default_contents.hex";
`endif
//synthesis translate_on
//synthesis read_comments_as_HDL on
//defparam                cpu_0_ociram_lpm_dram_bdp_component.lpm_file                =
"cpu_0_ociram_default_contents.mif";
//synthesis read_comments_as_HDL off
   assign cfgdout = (MonAReg[4 : 2] == 3'd0)? 32'h00800020 :
      (MonAReg[4 : 2] == 3'd1)? 32'h00001919 :
      (MonAReg[4 : 2] == 3'd2)? 32'h00040000 :
      (MonAReg[4 : 2] == 3'd3)? 32'h00000000 :
      (MonAReg[4 : 2] == 3'd4)? 32'h20000000 :
      (MonAReg[4 : 2] == 3'd5)? 32'h00800000 :
      (MonAReg[4 : 2] == 3'd6)? 32'h00000000 :
      32'h00000000;


endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030
```

```
module cpu_0_nios2_avalon_reg (
                                        // inputs:
                                          address,
                                          chipselect,
                                          clk,
                                          debugaccess,
                                          monitor_error,
                                          monitor_go,
                                          monitor_ready,
                                          reset_n,
                                          write,
                                          writedata,

                                        // outputs:
                                          oci_ienable,
                                          oci_reg_readdata,
                                          oci_single_step_mode,
                                          ocireg_ers,
                                          ocireg_mrs,
                                          take_action_ocireg
                                       )
;

  output  [ 31: 0] oci_ienable;
  output  [ 31: 0] oci_reg_readdata;
  output           oci_single_step_mode;
  output           ocireg_ers;
  output           ocireg_mrs;
  output           take_action_ocireg;
  input   [  8: 0] address;
  input            chipselect;
  input            clk;
  input            debugaccess;
  input            monitor_error;
  input            monitor_go;
  input            monitor_ready;
  input            reset_n;
  input            write;
  input   [ 31: 0] writedata;

  reg     [ 31: 0] oci_ienable;
  wire             oci_reg_00_addressed;
  wire             oci_reg_01_addressed;
  wire    [ 31: 0] oci_reg_readdata;
  reg              oci_single_step_mode;
  wire             ocireg_ers;
  wire             ocireg_mrs;
  wire             ocireg_sstep;
  wire             take_action_oci_intr_mask_reg;
  wire             take_action_ocireg;
  wire             write_strobe;
  assign oci_reg_00_addressed = address == 9'h100;
  assign oci_reg_01_addressed = address == 9'h101;
  assign write_strobe = chipselect & write & debugaccess;
  assign take_action_ocireg = write_strobe & oci_reg_00_addressed;
  assign take_action_oci_intr_mask_reg = write_strobe & oci_reg_01_addressed;
  assign ocireg_ers = writedata[1];
```

```
  assign ocireg_mrs = writedata[0];
  assign ocireg_sstep = writedata[3];
  assign oci_reg_readdata = oci_reg_00_addressed ? {28'b0, oci_single_step_mode, monitor_go,
      monitor_ready, monitor_error} :
      oci_reg_01_addressed ?  oci_ienable :
      32'b0;

  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          oci_single_step_mode <= 1'b0;
      else if (take_action_ocireg)
          oci_single_step_mode <= ocireg_sstep;
    end


  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          oci_ienable <= 32'b00000000000000000000000000000111;
      else if (take_action_oci_intr_mask_reg)
          oci_ienable <= writedata | ~(32'b00000000000000000000000000000111);
    end



endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_break (
                               // inputs:
                                 clk,
                                 dbrk_break,
                                 dbrk_goto0,
                                 dbrk_goto1,
                                 jdo,
                                 jrst_n,
                                 reset_n,
                                 take_action_break_a,
                                 take_action_break_b,
                                 take_action_break_c,
                                 take_no_action_break_a,
                                 take_no_action_break_b,
                                 take_no_action_break_c,
                                 xbrk_goto0,
                                 xbrk_goto1,

                               // outputs:
                                 break_readreg,
```

```
                                        dbrk_hit0_latch,
                                        dbrk_hit1_latch,
                                        dbrk_hit2_latch,
                                        dbrk_hit3_latch,
                                        trigbrktype,
                                        trigger_state_0,
                                        trigger_state_1,
                                        xbrk_ctrl0,
                                        xbrk_ctrl1,
                                        xbrk_ctrl2,
                                        xbrk_ctrl3
                                    )
;

   output    [ 31: 0] break_readreg;
   output             dbrk_hit0_latch;
   output             dbrk_hit1_latch;
   output             dbrk_hit2_latch;
   output             dbrk_hit3_latch;
   output             trigbrktype;
   output             trigger_state_0;
   output             trigger_state_1;
   output    [  7: 0] xbrk_ctrl0;
   output    [  7: 0] xbrk_ctrl1;
   output    [  7: 0] xbrk_ctrl2;
   output    [  7: 0] xbrk_ctrl3;
   input              clk;
   input              dbrk_break;
   input              dbrk_goto0;
   input              dbrk_goto1;
   input     [ 37: 0] jdo;
   input              jrst_n;
   input              reset_n;
   input              take_action_break_a;
   input              take_action_break_b;
   input              take_action_break_c;
   input              take_no_action_break_a;
   input              take_no_action_break_b;
   input              take_no_action_break_c;
   input              xbrk_goto0;
   input              xbrk_goto1;

   wire      [  3: 0] break_a_wpr;
   wire      [  1: 0] break_a_wpr_high_bits;
   wire      [  1: 0] break_a_wpr_low_bits;
   wire      [  1: 0] break_b_rr;
   wire      [  1: 0] break_c_rr;
   reg       [ 31: 0] break_readreg /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
   wire               dbrk0_high_value;
   wire               dbrk0_low_value;
   wire               dbrk1_high_value;
   wire               dbrk1_low_value;
   wire               dbrk2_high_value;
   wire               dbrk2_low_value;
   wire               dbrk3_high_value;
   wire               dbrk3_low_value;
   wire               dbrk_hit0_latch;
```

```
  wire              dbrk_hit1_latch;
  wire              dbrk_hit2_latch;
  wire              dbrk_hit3_latch;
  wire              take_action_any_break;
  reg                                   trigbrktype /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
  reg               trigger_state;
  wire              trigger_state_0;
  wire              trigger_state_1;
  wire    [ 31: 0] xbrk0_value;
  wire    [ 31: 0] xbrk1_value;
  wire    [ 31: 0] xbrk2_value;
  wire    [ 31: 0] xbrk3_value;
  reg               [    7:    0]   xbrk_ctrl0   /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
  reg               [    7:    0]   xbrk_ctrl1   /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
  reg               [    7:    0]   xbrk_ctrl2   /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
  reg               [    7:    0]   xbrk_ctrl3   /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,R101\"" */;
  assign break_a_wpr = jdo[35 : 32];
  assign break_a_wpr_high_bits = break_a_wpr[3 : 2];
  assign break_a_wpr_low_bits = break_a_wpr[1 : 0];
  assign break_b_rr = jdo[33 : 32];
  assign break_c_rr = jdo[33 : 32];
  assign take_action_any_break = take_action_break_a | take_action_break_b |
take_action_break_c;
  always @(posedge clk or negedge jrst_n)
    begin
      if (jrst_n == 0)
        begin
          xbrk_ctrl0 <= 0;
          xbrk_ctrl1 <= 0;
          xbrk_ctrl2 <= 0;
          xbrk_ctrl3 <= 0;
          trigbrktype <= 0;
        end
      else
        begin
          if (take_action_any_break)
              trigbrktype <= 0;
          else if (dbrk_break)
              trigbrktype <= 1;
          if (take_action_break_b)
            begin
              if ((break_b_rr == 2'b00) && (0 >= 1))
                begin
                  xbrk_ctrl0[0] <= jdo[27];
                  xbrk_ctrl0[1] <= jdo[28];
                  xbrk_ctrl0[2] <= jdo[29];
                  xbrk_ctrl0[3] <= jdo[30];
                  xbrk_ctrl0[4] <= jdo[21];
                  xbrk_ctrl0[5] <= jdo[20];
                  xbrk_ctrl0[6] <= jdo[19];
                  xbrk_ctrl0[7] <= jdo[18];
                end
              if ((break_b_rr == 2'b01) && (0 >= 2))
```

```
                    begin
                        xbrk_ctrl1[0] <= jdo[27];
                        xbrk_ctrl1[1] <= jdo[28];
                        xbrk_ctrl1[2] <= jdo[29];
                        xbrk_ctrl1[3] <= jdo[30];
                        xbrk_ctrl1[4] <= jdo[21];
                        xbrk_ctrl1[5] <= jdo[20];
                        xbrk_ctrl1[6] <= jdo[19];
                        xbrk_ctrl1[7] <= jdo[18];
                    end
                if ((break_b_rr == 2'b10) && (0 >= 3))
                    begin
                        xbrk_ctrl2[0] <= jdo[27];
                        xbrk_ctrl2[1] <= jdo[28];
                        xbrk_ctrl2[2] <= jdo[29];
                        xbrk_ctrl2[3] <= jdo[30];
                        xbrk_ctrl2[4] <= jdo[21];
                        xbrk_ctrl2[5] <= jdo[20];
                        xbrk_ctrl2[6] <= jdo[19];
                        xbrk_ctrl2[7] <= jdo[18];
                    end
                if ((break_b_rr == 2'b11) && (0 >= 4))
                    begin
                        xbrk_ctrl3[0] <= jdo[27];
                        xbrk_ctrl3[1] <= jdo[28];
                        xbrk_ctrl3[2] <= jdo[29];
                        xbrk_ctrl3[3] <= jdo[30];
                        xbrk_ctrl3[4] <= jdo[21];
                        xbrk_ctrl3[5] <= jdo[20];
                        xbrk_ctrl3[6] <= jdo[19];
                        xbrk_ctrl3[7] <= jdo[18];
                    end
            end
        end
    end


assign dbrk_hit0_latch = 1'b0;
assign dbrk0_low_value = 0;
assign dbrk0_high_value = 0;
assign dbrk_hit1_latch = 1'b0;
assign dbrk1_low_value = 0;
assign dbrk1_high_value = 0;
assign dbrk_hit2_latch = 1'b0;
assign dbrk2_low_value = 0;
assign dbrk2_high_value = 0;
assign dbrk_hit3_latch = 1'b0;
assign dbrk3_low_value = 0;
assign dbrk3_high_value = 0;
assign xbrk0_value = 32'b0;
assign xbrk1_value = 32'b0;
assign xbrk2_value = 32'b0;
assign xbrk3_value = 32'b0;
always @(posedge clk or negedge jrst_n)
    begin
        if (jrst_n == 0)
            break_readreg <= 32'b0;
        else if (take_action_any_break)
```

```
        break_readreg <= jdo[31 : 0];
    else if (take_no_action_break_a)
        case (break_a_wpr_high_bits)

            2'd0: begin
                case (break_a_wpr_low_bits) // synthesis full_case

                    2'd0: begin
                        break_readreg <= xbrk0_value;
                    end // 2'd0

                    2'd1: begin
                        break_readreg <= xbrk1_value;
                    end // 2'd1

                    2'd2: begin
                        break_readreg <= xbrk2_value;
                    end // 2'd2

                    2'd3: begin
                        break_readreg <= xbrk3_value;
                    end // 2'd3

                endcase // break_a_wpr_low_bits
            end // 2'd0

            2'd1: begin
                break_readreg <= 32'b0;
            end // 2'd1

            2'd2: begin
                case (break_a_wpr_low_bits) // synthesis full_case

                    2'd0: begin
                        break_readreg <= dbrk0_low_value;
                    end // 2'd0

                    2'd1: begin
                        break_readreg <= dbrk1_low_value;
                    end // 2'd1

                    2'd2: begin
                        break_readreg <= dbrk2_low_value;
                    end // 2'd2

                    2'd3: begin
                        break_readreg <= dbrk3_low_value;
                    end // 2'd3

                endcase // break_a_wpr_low_bits
            end // 2'd2

            2'd3: begin
                case (break_a_wpr_low_bits) // synthesis full_case

                    2'd0: begin
                        break_readreg <= dbrk0_high_value;
                    end // 2'd0
```

```
                                2'd1: begin
                                      break_readreg <= dbrk1_high_value;
                                end // 2'd1

                                2'd2: begin
                                      break_readreg <= dbrk2_high_value;
                                end // 2'd2

                                2'd3: begin
                                      break_readreg <= dbrk3_high_value;
                                end // 2'd3

                          endcase // break_a_wpr_low_bits
                    end // 2'd3

              endcase // break_a_wpr_high_bits
        else if (take_no_action_break_b)
              break_readreg <= jdo[31 : 0];
        else if (take_no_action_break_c)
              break_readreg <= jdo[31 : 0];
      end


   always @(posedge clk or negedge reset_n)
      begin
        if (reset_n == 0)
              trigger_state <= 0;
        else if (trigger_state_1 & (xbrk_goto0 | dbrk_goto0))
              trigger_state <= 0;
        else if (trigger_state_0 & (xbrk_goto1 | dbrk_goto1))
              trigger_state <= -1;
      end


   assign trigger_state_0 = ~trigger_state;
   assign trigger_state_1 = trigger_state;

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_xbrk (
                                          // inputs:
                                            D_valid,
                                            E_valid,
                                            F_pc,
                                            clk,
                                            reset_n,
                                            trigger_state_0,
                                            trigger_state_1,
```

```
                                    xbrk_ctrl0,
                                    xbrk_ctrl1,
                                    xbrk_ctrl2,
                                    xbrk_ctrl3,

                                   // outputs:
                                    xbrk_break,
                                    xbrk_goto0,
                                    xbrk_goto1,
                                    xbrk_traceoff,
                                    xbrk_traceon,
                                    xbrk_trigout
                                  )
;

  output            xbrk_break;
  output            xbrk_goto0;
  output            xbrk_goto1;
  output            xbrk_traceoff;
  output            xbrk_traceon;
  output            xbrk_trigout;
  input             D_valid;
  input             E_valid;
  input     [ 22: 0] F_pc;
  input             clk;
  input             reset_n;
  input             trigger_state_0;
  input             trigger_state_1;
  input     [  7: 0] xbrk_ctrl0;
  input     [  7: 0] xbrk_ctrl1;
  input     [  7: 0] xbrk_ctrl2;
  input     [  7: 0] xbrk_ctrl3;

  wire              D_cpu_addr_en;
  wire              E_cpu_addr_en;
  reg               E_xbrk_goto0;
  reg               E_xbrk_goto1;
  reg               E_xbrk_traceoff;
  reg               E_xbrk_traceon;
  reg               E_xbrk_trigout;
  wire      [ 24: 0] cpu_i_address;
  wire              xbrk0_armed;
  wire              xbrk0_break_hit;
  wire              xbrk0_goto0_hit;
  wire              xbrk0_goto1_hit;
  wire              xbrk0_toff_hit;
  wire              xbrk0_ton_hit;
  wire              xbrk0_tout_hit;
  wire              xbrk1_armed;
  wire              xbrk1_break_hit;
  wire              xbrk1_goto0_hit;
  wire              xbrk1_goto1_hit;
  wire              xbrk1_toff_hit;
  wire              xbrk1_ton_hit;
  wire              xbrk1_tout_hit;
  wire              xbrk2_armed;
  wire              xbrk2_break_hit;
  wire              xbrk2_goto0_hit;
```

```
  wire            xbrk2_goto1_hit;
  wire            xbrk2_toff_hit;
  wire            xbrk2_ton_hit;
  wire            xbrk2_tout_hit;
  wire            xbrk3_armed;
  wire            xbrk3_break_hit;
  wire            xbrk3_goto0_hit;
  wire            xbrk3_goto1_hit;
  wire            xbrk3_toff_hit;
  wire            xbrk3_ton_hit;
  wire            xbrk3_tout_hit;
  reg             xbrk_break;
  wire            xbrk_break_hit;
  wire            xbrk_goto0;
  wire            xbrk_goto0_hit;
  wire            xbrk_goto1;
  wire            xbrk_goto1_hit;
  wire            xbrk_toff_hit;
  wire            xbrk_ton_hit;
  wire            xbrk_tout_hit;
  wire            xbrk_traceoff;
  wire            xbrk_traceon;
  wire            xbrk_trigout;
  assign cpu_i_address = {F_pc, 2'b00};
  assign D_cpu_addr_en = D_valid;
  assign E_cpu_addr_en = E_valid;
  assign xbrk0_break_hit = 0;
  assign xbrk0_ton_hit = 0;
  assign xbrk0_toff_hit = 0;
  assign xbrk0_tout_hit = 0;
  assign xbrk0_goto0_hit = 0;
  assign xbrk0_goto1_hit = 0;
  assign xbrk1_break_hit = 0;
  assign xbrk1_ton_hit = 0;
  assign xbrk1_toff_hit = 0;
  assign xbrk1_tout_hit = 0;
  assign xbrk1_goto0_hit = 0;
  assign xbrk1_goto1_hit = 0;
  assign xbrk2_break_hit = 0;
  assign xbrk2_ton_hit = 0;
  assign xbrk2_toff_hit = 0;
  assign xbrk2_tout_hit = 0;
  assign xbrk2_goto0_hit = 0;
  assign xbrk2_goto1_hit = 0;
  assign xbrk3_break_hit = 0;
  assign xbrk3_ton_hit = 0;
  assign xbrk3_toff_hit = 0;
  assign xbrk3_tout_hit = 0;
  assign xbrk3_goto0_hit = 0;
  assign xbrk3_goto1_hit = 0;
  assign xbrk_break_hit = (xbrk0_break_hit) | (xbrk1_break_hit) | (xbrk2_break_hit) |
(xbrk3_break_hit);
  assign xbrk_ton_hit = (xbrk0_ton_hit) | (xbrk1_ton_hit) | (xbrk2_ton_hit) | (xbrk3_ton_hit);
  assign xbrk_toff_hit = (xbrk0_toff_hit) | (xbrk1_toff_hit) | (xbrk2_toff_hit) | (xbrk3_toff_hit);
  assign xbrk_tout_hit = (xbrk0_tout_hit) | (xbrk1_tout_hit) | (xbrk2_tout_hit) |
(xbrk3_tout_hit);
  assign xbrk_goto0_hit = (xbrk0_goto0_hit) | (xbrk1_goto0_hit) | (xbrk2_goto0_hit) |
(xbrk3_goto0_hit);
```

```
  assign xbrk_goto1_hit = (xbrk0_goto1_hit) | (xbrk1_goto1_hit) | (xbrk2_goto1_hit) |
(xbrk3_goto1_hit);
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            xbrk_break <= 0;
      else if (E_cpu_addr_en)
            xbrk_break <= xbrk_break_hit;
    end


  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            E_xbrk_traceon <= 0;
      else if (E_cpu_addr_en)
            E_xbrk_traceon <= xbrk_ton_hit;
    end


  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            E_xbrk_traceoff <= 0;
      else if (E_cpu_addr_en)
            E_xbrk_traceoff <= xbrk_toff_hit;
    end


  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            E_xbrk_trigout <= 0;
      else if (E_cpu_addr_en)
            E_xbrk_trigout <= xbrk_tout_hit;
    end


  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            E_xbrk_goto0 <= 0;
      else if (E_cpu_addr_en)
            E_xbrk_goto0 <= xbrk_goto0_hit;
    end


  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            E_xbrk_goto1 <= 0;
      else if (E_cpu_addr_en)
            E_xbrk_goto1 <= xbrk_goto1_hit;
    end


  assign xbrk_traceon = 1'b0;
  assign xbrk_traceoff = 1'b0;
```

```
  assign xbrk_trigout = 1'b0;
  assign xbrk_goto0 = 1'b0;
  assign xbrk_goto1 = 1'b0;
  assign xbrk0_armed = (xbrk_ctrl0[4] & trigger_state_0) ||
    (xbrk_ctrl0[5] & trigger_state_1);

  assign xbrk1_armed = (xbrk_ctrl1[4] & trigger_state_0) ||
    (xbrk_ctrl1[5] & trigger_state_1);

  assign xbrk2_armed = (xbrk_ctrl2[4] & trigger_state_0) ||
    (xbrk_ctrl2[5] & trigger_state_1);

  assign xbrk3_armed = (xbrk_ctrl3[4] & trigger_state_0) ||
    (xbrk_ctrl3[5] & trigger_state_1);


endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_dbrk (
                              // inputs:
                                E_st_data,
                                av_ld_data_aligned_filtered,
                                clk,
                                d_address,
                                d_read,
                                d_waitrequest,
                                d_write,
                                debugack,
                                reset_n,

                              // outputs:
                                cpu_d_address,
                                cpu_d_read,
                                cpu_d_readdata,
                                cpu_d_wait,
                                cpu_d_write,
                                cpu_d_writedata,
                                dbrk_break,
                                dbrk_goto0,
                                dbrk_goto1,
                                dbrk_traceme,
                                dbrk_traceoff,
                                dbrk_traceon,
                                dbrk_trigout
                             )
;

  output  [ 24: 0] cpu_d_address;
  output           cpu_d_read;
```

```
output   [ 31: 0] cpu_d_readdata;
output            cpu_d_wait;
output            cpu_d_write;
output   [ 31: 0] cpu_d_writedata;
output            dbrk_break;
output            dbrk_goto0;
output            dbrk_goto1;
output            dbrk_traceme;
output            dbrk_traceoff;
output            dbrk_traceon;
output            dbrk_trigout;
input    [ 31: 0] E_st_data;
input    [ 31: 0] av_ld_data_aligned_filtered;
input             clk;
input    [ 24: 0] d_address;
input             d_read;
input             d_waitrequest;
input             d_write;
input             debugack;
input             reset_n;

wire     [ 24: 0] cpu_d_address;
wire              cpu_d_read;
wire     [ 31: 0] cpu_d_readdata;
wire              cpu_d_wait;
wire              cpu_d_write;
wire     [ 31: 0] cpu_d_writedata;
wire              dbrk0_armed;
wire              dbrk0_break_pulse;
wire              dbrk0_goto0;
wire              dbrk0_goto1;
wire              dbrk0_traceme;
wire              dbrk0_traceoff;
wire              dbrk0_traceon;
wire              dbrk0_trigout;
wire              dbrk1_armed;
wire              dbrk1_break_pulse;
wire              dbrk1_goto0;
wire              dbrk1_goto1;
wire              dbrk1_traceme;
wire              dbrk1_traceoff;
wire              dbrk1_traceon;
wire              dbrk1_trigout;
wire              dbrk2_armed;
wire              dbrk2_break_pulse;
wire              dbrk2_goto0;
wire              dbrk2_goto1;
wire              dbrk2_traceme;
wire              dbrk2_traceoff;
wire              dbrk2_traceon;
wire              dbrk2_trigout;
wire              dbrk3_armed;
wire              dbrk3_break_pulse;
wire              dbrk3_goto0;
wire              dbrk3_goto1;
wire              dbrk3_traceme;
wire              dbrk3_traceoff;
wire              dbrk3_traceon;
```

```
wire                    dbrk3_trigout;
reg                     dbrk_break;
reg                     dbrk_break_pulse;
wire       [ 31: 0] dbrk_data;
reg                     dbrk_goto0;
reg                     dbrk_goto1;
reg                     dbrk_traceme;
reg                     dbrk_traceoff;
reg                     dbrk_traceon;
reg                     dbrk_trigout;
assign cpu_d_address = d_address;
assign cpu_d_readdata = av_ld_data_aligned_filtered;
assign cpu_d_read = d_read;
assign cpu_d_writedata = E_st_data;
assign cpu_d_write = d_write;
assign cpu_d_wait = d_waitrequest;
assign dbrk_data = cpu_d_write ? cpu_d_writedata : cpu_d_readdata;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
           dbrk_break <= 0;
      else
         dbrk_break <= dbrk_break      ? ~debugack
                     : dbrk_break_pulse;

   end


assign dbrk0_armed = 1'b0;
assign dbrk0_trigout = 1'b0;
assign dbrk0_break_pulse = 1'b0;
assign dbrk0_traceoff = 1'b0;
assign dbrk0_traceon = 1'b0;
assign dbrk0_traceme = 1'b0;
assign dbrk0_goto0 = 1'b0;
assign dbrk0_goto1 = 1'b0;
assign dbrk1_armed = 1'b0;
assign dbrk1_trigout = 1'b0;
assign dbrk1_break_pulse = 1'b0;
assign dbrk1_traceoff = 1'b0;
assign dbrk1_traceon = 1'b0;
assign dbrk1_traceme = 1'b0;
assign dbrk1_goto0 = 1'b0;
assign dbrk1_goto1 = 1'b0;
assign dbrk2_armed = 1'b0;
assign dbrk2_trigout = 1'b0;
assign dbrk2_break_pulse = 1'b0;
assign dbrk2_traceoff = 1'b0;
assign dbrk2_traceon = 1'b0;
assign dbrk2_traceme = 1'b0;
assign dbrk2_goto0 = 1'b0;
assign dbrk2_goto1 = 1'b0;
assign dbrk3_armed = 1'b0;
assign dbrk3_trigout = 1'b0;
assign dbrk3_break_pulse = 1'b0;
assign dbrk3_traceoff = 1'b0;
assign dbrk3_traceon = 1'b0;
assign dbrk3_traceme = 1'b0;
```

```
  assign dbrk3_goto0 = 1'b0;
  assign dbrk3_goto1 = 1'b0;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
        begin
          dbrk_trigout <= 0;
          dbrk_break_pulse <= 0;
          dbrk_traceoff <= 0;
          dbrk_traceon <= 0;
          dbrk_traceme <= 0;
          dbrk_goto0 <= 0;
          dbrk_goto1 <= 0;
        end
      else
        begin
          dbrk_trigout <= dbrk0_trigout | dbrk1_trigout | dbrk2_trigout | dbrk3_trigout;
          dbrk_break_pulse <= dbrk0_break_pulse | dbrk1_break_pulse | dbrk2_break_pulse
| dbrk3_break_pulse;
          dbrk_traceoff <= dbrk0_traceoff | dbrk1_traceoff | dbrk2_traceoff | dbrk3_traceoff;
          dbrk_traceon <= dbrk0_traceon | dbrk1_traceon | dbrk2_traceon | dbrk3_traceon;
          dbrk_traceme <= dbrk0_traceme | dbrk1_traceme | dbrk2_traceme |
dbrk3_traceme;
          dbrk_goto0 <= dbrk0_goto0 | dbrk1_goto0 | dbrk2_goto0 | dbrk3_goto0;
          dbrk_goto1 <= dbrk0_goto1 | dbrk1_goto1 | dbrk2_goto1 | dbrk3_goto1;
        end
    end



endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_itrace (
                                        // inputs:
                                          clk,
                                          dbrk_traceoff,
                                          dbrk_traceon,
                                          jdo,
                                          jrst_n,
                                          take_action_tracectrl,
                                          trc_enb,
                                          xbrk_traceoff,
                                          xbrk_traceon,
                                          xbrk_wrap_traceoff,

                                        // outputs:
                                          dct_buffer,
                                          dct_count,
                                          itm,
```

```
                                        trc_ctrl,
                                        trc_on
                                    )
;

   output   [ 29: 0] dct_buffer;
   output   [  3: 0] dct_count;
   output   [ 35: 0] itm;
   output   [ 15: 0] trc_ctrl;
   output            trc_on;
   input             clk;
   input             dbrk_traceoff;
   input             dbrk_traceon;
   input    [ 15: 0] jdo;
   input             jrst_n;
   input             take_action_tracectrl;
   input             trc_enb;
   input             xbrk_traceoff;
   input             xbrk_traceon;
   input             xbrk_wrap_traceoff;

   wire              curr_pid;
   reg               [ 29: 0] dct_buffer /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   wire     [  1: 0] dct_code;
   reg               [  3: 0] dct_count /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   wire              dct_is_taken;
   wire     [ 31: 0] excaddr;
   wire              instr_retired;
   wire              is_advanced_exception;
   wire              is_cond_dct;
   wire              is_dct;
   wire              is_exception_no_break;
   wire              is_fast_tlb_miss_exception;
   wire              is_idct;
   reg               [ 35: 0] itm /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   wire              not_in_debug_mode;
   reg               pending_curr_pid /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   reg               [ 31: 0] pending_excaddr /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   reg               pending_exctype /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   reg               [  3: 0] pending_frametype /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   reg               pending_prev_pid /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   reg               prev_pid /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   reg               prev_pid_valid /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   wire              record_dct_outcome_in_sync;
   wire              record_itrace;
   wire     [ 31: 0] retired_pcb;
   reg               snapped_curr_pid /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
```

```
  reg                                   snapped_pid  /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101"  */;
  reg                                snapped_prev_pid  /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101"  */;
  wire      [  1: 0] sync_code;
  wire      [  6: 0] sync_interval;
  wire               sync_pending;
  reg               [      6:   0]  sync_timer  /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101"  */;
  wire      [  6: 0] sync_timer_next;
  reg                                  trc_clear  /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=D101"  */;
  wire      [ 15: 0] trc_ctrl;
  reg               [   10:   0]  trc_ctrl_reg  /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,D103,R101\""  */;
  wire               trc_on;
  assign is_cond_dct = 1'b0;
  assign is_dct = 1'b0;
  assign dct_is_taken = 1'b0;
  assign is_idct = 1'b0;
  assign retired_pcb = 32'b0;
  assign not_in_debug_mode = 1'b0;
  assign instr_retired = 1'b0;
  assign is_advanced_exception = 1'b0;
  assign is_exception_no_break = 1'b0;
  assign is_fast_tlb_miss_exception = 1'b0;
  assign curr_pid = 1'b0;
  assign excaddr = 32'b0;
  assign sync_code = trc_ctrl[3 : 2];
  assign sync_interval = { sync_code[1] & sync_code[0], 1'b0, sync_code[1] & ~sync_code[0],
1'b0, ~sync_code[1] & sync_code[0], 2'b00 };
  assign sync_pending = sync_timer == 0;
  assign record_dct_outcome_in_sync = dct_is_taken & sync_pending;
  assign sync_timer_next = sync_pending ? sync_timer : (sync_timer - 1);
  assign record_itrace = trc_on & trc_ctrl[4];
  assign dct_code = {is_cond_dct, dct_is_taken};
  always @(posedge clk or negedge jrst_n)
    begin
      if (jrst_n == 0)
          trc_clear <= 0;
      else
        trc_clear <= ~trc_enb &
                  take_action_tracectrl & jdo[4];

    end


  always @(posedge clk or negedge jrst_n)
    begin
      if (jrst_n == 0)
        begin
          itm <= 0;
          dct_buffer <= 0;
          dct_count <= 0;
          sync_timer <= 0;
          pending_frametype <= 4'b0000;
          pending_exctype <= 1'b0;
          pending_excaddr <= 0;
```

```
        prev_pid <= 0;
        prev_pid_valid <= 0;
        snapped_pid <= 0;
        snapped_curr_pid <= 0;
        snapped_prev_pid <= 0;
        pending_curr_pid <= 0;
        pending_prev_pid <= 0;
    end
else if (trc_clear || (!0 && !0))
    begin
        itm <= 0;
        dct_buffer <= 0;
        dct_count <= 0;
        sync_timer <= 0;
        pending_frametype <= 4'b0000;
        pending_exctype <= 1'b0;
        pending_excaddr <= 0;
        prev_pid <= 0;
        prev_pid_valid <= 0;
        snapped_pid <= 0;
        snapped_curr_pid <= 0;
        snapped_prev_pid <= 0;
        pending_curr_pid <= 0;
        pending_prev_pid <= 0;
    end
else
    begin
        if (!prev_pid_valid)
            begin
                prev_pid <= curr_pid;
                prev_pid_valid <= 1;
            end
        if ((curr_pid != prev_pid) & prev_pid_valid & !snapped_pid)
            begin
                snapped_pid <= 1;
                snapped_curr_pid <= curr_pid;
                snapped_prev_pid <= prev_pid;
                prev_pid <= curr_pid;
                prev_pid_valid <= 1;
            end
        if (instr_retired | is_advanced_exception)
            begin
                if (~record_itrace)
                        pending_frametype <= 4'b1010;
                else if (is_exception_no_break)
                    begin
                        pending_frametype <= 4'b0010;
                        pending_excaddr <= excaddr;
                        if (is_fast_tlb_miss_exception)
                                pending_exctype <= 1'b1;
                        else
                            pending_exctype <= 1'b0;
                    end
                else if (is_idct)
                        pending_frametype <= 4'b1001;
                else if (record_dct_outcome_in_sync)
                        pending_frametype <= 4'b1000;
                else if (!is_dct & snapped_pid)
```

```
                    begin
                        pending_frametype <= 4'b0011;
                        pending_curr_pid <= snapped_curr_pid;
                        pending_prev_pid <= snapped_prev_pid;
                        snapped_pid <= 0;
                    end
                else
                    pending_frametype <= 4'b0000;
                if ((dct_count != 0) &
            (~record_itrace |
             is_exception_no_break |
             is_idct |
             record_dct_outcome_in_sync |
             (!is_dct & snapped_pid)))
                    begin
                        itm <= {4'b0001, dct_buffer, 2'b00};
                        dct_buffer <= 0;
                        dct_count <= 0;
                        sync_timer <= sync_timer_next;
                    end
                else
                    begin
                        if (record_itrace & (is_dct & (dct_count != 4'd15)) &
~record_dct_outcome_in_sync & ~is_advanced_exception)
                            begin
                                dct_buffer <= {dct_code, dct_buffer[29 : 2]};
                                dct_count <= dct_count + 1;
                            end
                        if (record_itrace & (pending_frametype == 4'b0010))
                                itm <= {4'b0010, pending_excaddr[31 : 1], pending_exctype};
                        else if (record_itrace & (
                    (pending_frametype == 4'b1000) |
                    (pending_frametype == 4'b1010) |
                    (pending_frametype == 4'b1001)))
                            begin
                                itm <= {pending_frametype, retired_pcb};
                                sync_timer <= sync_interval;
                                if (0 &
                            ((pending_frametype == 4'b1000) | (pending_frametype == 4'b1010)) &
                            !snapped_pid & prev_pid_valid)
                                    begin
                                        snapped_pid <= 1;
                                        snapped_curr_pid <= curr_pid;
                                        snapped_prev_pid <= prev_pid;
                                    end
                            end
                        else if (record_itrace &
                    0 & (pending_frametype == 4'b0011))
                                itm <= {4'b0011, 2'b00, pending_prev_pid, 2'b00, pending_curr_pid};
                        else if (record_itrace & is_dct)
                            begin
                                if (dct_count == 4'd15)
                                    begin
                                        itm <= {4'b0001, dct_code, dct_buffer};
                                        dct_buffer <= 0;
                                        dct_count <= 0;
                                        sync_timer <= sync_timer_next;
                                    end
```

```
                        else
                            itm <= 4'b0000;
                        end
                    else
                        itm <= {4'b0000, 32'b0};
                end
            end
        else
            itm <= {4'b0000, 32'b0};
        end
    end


  always @(posedge clk or negedge jrst_n)
    begin
      if (jrst_n == 0)
        begin
          trc_ctrl_reg[0] <= 1'b0;
          trc_ctrl_reg[1] <= 1'b0;
          trc_ctrl_reg[3 : 2] <= 2'b00;
          trc_ctrl_reg[4] <= 1'b0;
          trc_ctrl_reg[7 : 5] <= 3'b000;
          trc_ctrl_reg[8] <= 0;
          trc_ctrl_reg[9] <= 1'b0;
          trc_ctrl_reg[10] <= 1'b0;
        end
      else if (take_action_tracectrl)
        begin
          trc_ctrl_reg[0] <= jdo[5];
          trc_ctrl_reg[1] <= jdo[6];
          trc_ctrl_reg[3 : 2] <= jdo[8 : 7];
          trc_ctrl_reg[4] <= jdo[9];
          trc_ctrl_reg[9] <= jdo[14];
          trc_ctrl_reg[10] <= jdo[2];
          if (0)
              trc_ctrl_reg[7 : 5] <= jdo[12 : 10];
          if (0 & 0)
              trc_ctrl_reg[8] <= jdo[13];
        end
      else if (xbrk_wrap_traceoff)
        begin
          trc_ctrl_reg[1] <= 0;
          trc_ctrl_reg[0] <= 0;
        end
      else if (dbrk_traceoff | xbrk_traceoff)
          trc_ctrl_reg[1] <= 0;
      else if (trc_ctrl_reg[0] &
                                          (dbrk_traceon | xbrk_traceon))
          trc_ctrl_reg[1] <= 1;
    end


  assign trc_ctrl = (0 || 0) ? {6'b000000, trc_ctrl_reg} : 0;
  assign trc_on = trc_ctrl[1] & (trc_ctrl[9] | not_in_debug_mode);

endmodule
```

```
// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_td_mode (
                                                // inputs:
                                                  ctrl,

                                                // outputs:
                                                  td_mode
                                               )
;

   output    [    3: 0] td_mode;
   input     [    8: 0] ctrl;

   wire      [    2: 0] ctrl_bits_for_mux;
   reg       [    3: 0] td_mode;
   assign ctrl_bits_for_mux = ctrl[7 : 5];
   always @(ctrl_bits_for_mux)
      begin
         case (ctrl_bits_for_mux)

               3'b000: begin
                     td_mode = 4'b0000;
               end // 3'b000

               3'b001: begin
                     td_mode = 4'b1000;
               end // 3'b001

               3'b010: begin
                     td_mode = 4'b0100;
               end // 3'b010

               3'b011: begin
                     td_mode = 4'b1100;
               end // 3'b011

               3'b100: begin
                     td_mode = 4'b0010;
               end // 3'b100

               3'b101: begin
                     td_mode = 4'b1010;
               end // 3'b101

               3'b110: begin
                     td_mode = 4'b0101;
               end // 3'b110

               3'b111: begin
                     td_mode = 4'b1111;
               end // 3'b111
```

```
          endcase // ctrl_bits_for_mux
        end



endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_dtrace (
                                                // inputs:
                                                  clk,
                                                  cpu_d_address,
                                                  cpu_d_read,
                                                  cpu_d_readdata,
                                                  cpu_d_wait,
                                                  cpu_d_write,
                                                  cpu_d_writedata,
                                                  jrst_n,
                                                  trc_ctrl,

                                                // outputs:
                                                  atm,
                                                  dtm
                                               )
;

   output    [ 35: 0] atm;
   output    [ 35: 0] dtm;
   input              clk;
   input     [ 24: 0] cpu_d_address;
   input              cpu_d_read;
   input     [ 31: 0] cpu_d_readdata;
   input              cpu_d_wait;
   input              cpu_d_write;
   input     [ 31: 0] cpu_d_writedata;
   input              jrst_n;
   input     [ 15: 0] trc_ctrl;

   reg                [ 35: 0] atm /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   wire      [ 31: 0] cpu_d_address_0_padded;
   wire      [ 31: 0] cpu_d_readdata_0_padded;
   wire      [ 31: 0] cpu_d_writedata_0_padded;
   reg                [ 35: 0] dtm /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   wire               record_load_addr;
   wire               record_load_data;
   wire               record_store_addr;
   wire               record_store_data;
```

```
  wire       [  3: 0] td_mode_trc_ctrl;
  assign cpu_d_writedata_0_padded = cpu_d_writedata | 32'b0;
  assign cpu_d_readdata_0_padded = cpu_d_readdata | 32'b0;
  assign cpu_d_address_0_padded = cpu_d_address | 32'b0;
  //cpu_0_nios2_oci_trc_ctrl_td_mode, which is an e_instance
  cpu_0_nios2_oci_td_mode cpu_0_nios2_oci_trc_ctrl_td_mode
    (
      .ctrl       (trc_ctrl[8 : 0]),
      .td_mode (td_mode_trc_ctrl)
    );

  assign {record_load_addr, record_store_addr,
            record_load_data, record_store_data} = td_mode_trc_ctrl;
  always @(posedge clk or negedge jrst_n)
    begin
      if (jrst_n == 0)
        begin
          atm <= 0;
          dtm <= 0;
        end
      else if (0)
        begin
          if (cpu_d_write & ~cpu_d_wait & record_store_addr)
                atm <= {4'b0101, cpu_d_address_0_padded};
          else if (cpu_d_read & ~cpu_d_wait & record_load_addr)
                atm <= {4'b0100, cpu_d_address_0_padded};
          else
             atm <= {4'b0000, cpu_d_address_0_padded};
          if (cpu_d_write & ~cpu_d_wait & record_store_data)
                dtm <= {4'b0111, cpu_d_writedata_0_padded};
          else if (cpu_d_read & ~cpu_d_wait & record_load_data)
                dtm <= {4'b0110, cpu_d_readdata_0_padded};
          else
             dtm <= {4'b0000, cpu_d_readdata_0_padded};
        end
      else
        begin
          atm <= 0;
          dtm <= 0;
        end
    end



endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_compute_tm_count (
                                                        // inputs:
                                                          atm_valid,
```

```
                                                    dtm_valid,
                                                    itm_valid,

                                                  // outputs:
                                                    compute_tm_count
                                                )
;

   output    [    1: 0] compute_tm_count;
   input                atm_valid;
   input                dtm_valid;
   input                itm_valid;

   reg       [    1: 0] compute_tm_count;
   wire      [    2: 0] switch_for_mux;
   assign switch_for_mux = {itm_valid, atm_valid, dtm_valid};
   always @(switch_for_mux)
     begin
       case (switch_for_mux)

            3'b000: begin
                compute_tm_count = 0;
            end // 3'b000

            3'b001: begin
                compute_tm_count = 1;
            end // 3'b001

            3'b010: begin
                compute_tm_count = 1;
            end // 3'b010

            3'b011: begin
                compute_tm_count = 2;
            end // 3'b011

            3'b100: begin
                compute_tm_count = 1;
            end // 3'b100

            3'b101: begin
                compute_tm_count = 2;
            end // 3'b101

            3'b110: begin
                compute_tm_count = 2;
            end // 3'b110

            3'b111: begin
                compute_tm_count = 3;
            end // 3'b111

       endcase // switch_for_mux
     end


endmodule
```

```
// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_fifowp_inc (
                                    // inputs:
                                      free2,
                                      free3,
                                      tm_count,

                                    // outputs:
                                      fifowp_inc
                                   )
;

   output    [  3: 0] fifowp_inc;
   input              free2;
   input              free3;
   input     [  1: 0] tm_count;

   reg       [  3: 0] fifowp_inc;
   always @(free2 or free3 or tm_count)
      begin
         if (free3 & (tm_count == 3))
               fifowp_inc = 3;
         else if (free2 & (tm_count >= 2))
               fifowp_inc = 2;
         else if (tm_count >= 1)
               fifowp_inc = 1;
         else
            fifowp_inc = 0;
      end



endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_fifocount_inc (
                                       // inputs:
                                         empty,
                                         free2,
                                         free3,
                                         tm_count,
```

```
                                                    // outputs:
                                                      fifocount_inc
                                                   )
;

   output    [    4: 0] fifocount_inc;
   input                empty;
   input                free2;
   input                free3;
   input     [    1: 0] tm_count;

   reg       [    4: 0] fifocount_inc;
   always @(empty or free2 or free3 or tm_count)
      begin
         if (empty)
               fifocount_inc = tm_count[1 : 0];
         else if (free3 & (tm_count == 3))
               fifocount_inc = 2;
         else if (free2 & (tm_count >= 2))
               fifocount_inc = 1;
         else if (tm_count >= 1)
               fifocount_inc = 0;
         else
            fifocount_inc = {5{1'b1}};
      end



endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_fifo (
                                                  // inputs:
                                                    atm,
                                                    clk,
                                                    dbrk_traceme,
                                                    dbrk_traceoff,
                                                    dbrk_traceon,
                                                    dct_buffer,
                                                    dct_count,
                                                    dtm,
                                                    itm,
                                                    jrst_n,
                                                    reset_n,
                                                    test_ending,
                                                    test_has_ended,
                                                    trc_on,

                                                  // outputs:
```

```
                                        tw
                                    )
;

   output   [ 35: 0] tw;
   input    [ 35: 0] atm;
   input             clk;
   input             dbrk_traceme;
   input             dbrk_traceoff;
   input             dbrk_traceon;
   input    [ 29: 0] dct_buffer;
   input    [    3: 0] dct_count;
   input    [ 35: 0] dtm;
   input    [ 35: 0] itm;
   input             jrst_n;
   input             reset_n;
   input             test_ending;
   input             test_has_ended;
   input             trc_on;

   wire              atm_valid;
   wire     [    1: 0] compute_tm_count_tm_count;
   wire              dtm_valid;
   wire              empty;
   reg      [ 35: 0] fifo_0;
   wire              fifo_0_enable;
   wire     [ 35: 0] fifo_0_mux;
   reg      [ 35: 0] fifo_1;
   reg      [ 35: 0] fifo_10;
   wire              fifo_10_enable;
   wire     [ 35: 0] fifo_10_mux;
   reg      [ 35: 0] fifo_11;
   wire              fifo_11_enable;
   wire     [ 35: 0] fifo_11_mux;
   reg      [ 35: 0] fifo_12;
   wire              fifo_12_enable;
   wire     [ 35: 0] fifo_12_mux;
   reg      [ 35: 0] fifo_13;
   wire              fifo_13_enable;
   wire     [ 35: 0] fifo_13_mux;
   reg      [ 35: 0] fifo_14;
   wire              fifo_14_enable;
   wire     [ 35: 0] fifo_14_mux;
   reg      [ 35: 0] fifo_15;
   wire              fifo_15_enable;
   wire     [ 35: 0] fifo_15_mux;
   wire              fifo_1_enable;
   wire     [ 35: 0] fifo_1_mux;
   reg      [ 35: 0] fifo_2;
   wire              fifo_2_enable;
   wire     [ 35: 0] fifo_2_mux;
   reg      [ 35: 0] fifo_3;
   wire              fifo_3_enable;
   wire     [ 35: 0] fifo_3_mux;
   reg      [ 35: 0] fifo_4;
   wire              fifo_4_enable;
   wire     [ 35: 0] fifo_4_mux;
   reg      [ 35: 0] fifo_5;
```

```
  wire              fifo_5_enable;
  wire    [ 35: 0] fifo_5_mux;
  reg     [ 35: 0] fifo_6;
  wire              fifo_6_enable;
  wire    [ 35: 0] fifo_6_mux;
  reg     [ 35: 0] fifo_7;
  wire              fifo_7_enable;
  wire    [ 35: 0] fifo_7_mux;
  reg     [ 35: 0] fifo_8;
  wire              fifo_8_enable;
  wire    [ 35: 0] fifo_8_mux;
  reg     [ 35: 0] fifo_9;
  wire              fifo_9_enable;
  wire    [ 35: 0] fifo_9_mux;
  wire    [ 35: 0] fifo_read_mux;
  reg               [    4:    0]    fifocount    /*    synthesis    ALTERA_ATTRIBUTE    =
"SUPPRESS_DA_RULE_INTERNAL=R101"  */;
  wire    [  4: 0] fifocount_inc_fifocount;
  wire    [ 35: 0] fifohead;
  reg               [    3:    0]    fiforp    /*    synthesis    ALTERA_ATTRIBUTE    =
"SUPPRESS_DA_RULE_INTERNAL=R101"  */;
  reg               [    3:    0]    fifowp    /*    synthesis    ALTERA_ATTRIBUTE    =
"SUPPRESS_DA_RULE_INTERNAL=R101"  */;
  wire    [  3: 0] fifowp1;
  wire    [  3: 0] fifowp2;
  wire    [  3: 0] fifowp_inc_fifowp;
  wire              free2;
  wire              free3;
  wire              itm_valid;
  reg                                 ovf_pending    /*    synthesis    ALTERA_ATTRIBUTE    =
"SUPPRESS_DA_RULE_INTERNAL=R101"  */;
  wire    [ 35: 0] ovr_pending_atm;
  wire    [ 35: 0] ovr_pending_dtm;
  wire    [  1: 0] tm_count;
  wire              tm_count_ge1;
  wire              tm_count_ge2;
  wire              tm_count_ge3;
  wire              trc_this;
  wire    [ 35: 0] tw;
  assign trc_this = trc_on | (dbrk_traceon & ~dbrk_traceoff) | dbrk_traceme;
  assign itm_valid = |itm[35 : 32];
  assign atm_valid = |atm[35 : 32] & trc_this;
  assign dtm_valid = |dtm[35 : 32] & trc_this;
  assign free2 = ~fifocount[4];
  assign free3 = ~fifocount[4] & ~&fifocount[3 : 0];
  assign empty = ~|fifocount;
  assign fifowp1 = fifowp + 1;
  assign fifowp2 = fifowp + 2;
  //cpu_0_nios2_oci_compute_tm_count_tm_count, which is an e_instance
  cpu_0_nios2_oci_compute_tm_count cpu_0_nios2_oci_compute_tm_count_tm_count
    (
      .atm_valid              (atm_valid),
      .compute_tm_count (compute_tm_count_tm_count),
      .dtm_valid              (dtm_valid),
      .itm_valid             (itm_valid)
    );

  assign tm_count = compute_tm_count_tm_count;
```

```
  //cpu_0_nios2_oci_fifowp_inc_fifowp, which is an e_instance
  cpu_0_nios2_oci_fifowp_inc cpu_0_nios2_oci_fifowp_inc_fifowp
    (
      .fifowp_inc (fifowp_inc_fifowp),
      .free2      (free2),
      .free3      (free3),
      .tm_count    (tm_count)
    );

  //cpu_0_nios2_oci_fifocount_inc_fifocount, which is an e_instance
  cpu_0_nios2_oci_fifocount_inc cpu_0_nios2_oci_fifocount_inc_fifocount
    (
      .empty              (empty),
      .fifocount_inc (fifocount_inc_fifocount),
      .free2              (free2),
      .free3              (free3),
      .tm_count          (tm_count)
    );

  //the_cpu_0_oci_test_bench, which is an e_instance
  cpu_0_oci_test_bench the_cpu_0_oci_test_bench
    (
      .dct_buffer         (dct_buffer),
      .dct_count          (dct_count),
      .test_ending        (test_ending),
      .test_has_ended (test_has_ended)
    );

  always @(posedge clk or negedge jrst_n)
    begin
      if (jrst_n == 0)
        begin
          fiforp <= 0;
          fifowp <= 0;
          fifocount <= 0;
          ovf_pending <= 1;
        end
      else
        begin
          fifowp <= fifowp + fifowp_inc_fifowp;
          fifocount <= fifocount + fifocount_inc_fifocount;
          if (~empty)
              fiforp <= fiforp + 1;
          if (~trc_this || (~free2 & tm_count[1])      || (~free3 & (&tm_count)))
              ovf_pending <= 1;
          else if (atm_valid | dtm_valid)
              ovf_pending <= 0;
        end
    end


  assign fifohead = fifo_read_mux;
  assign tw = 0 ?   { (empty ?          4'h0          : fifohead[35 : 32]),     fifohead[31 : 0]}   :
itm;
  assign fifo_0_enable = ((fifowp == 4'd0) && tm_count_ge1)   || (free2 && (fifowp1== 4'd0)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd0) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
    begin
```

```
      if (reset_n == 0)
            fifo_0 <= 0;
      else if (fifo_0_enable)
            fifo_0 <= fifo_0_mux;
   end


assign fifo_0_mux = (((fifowp == 4'd0) && itm_valid))? itm :
   (((fifowp == 4'd0) && atm_valid))? ovr_pending_atm :
   (((fifowp == 4'd0) && dtm_valid))? ovr_pending_dtm :
   (((fifowp1 == 4'd0) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
   (((fifowp1 == 4'd0) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
   (((fifowp1 == 4'd0) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
   ovr_pending_dtm;

assign fifo_1_enable = ((fifowp == 4'd1) && tm_count_ge1)   || (free2 && (fifowp1== 4'd1)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd1) && tm_count_ge3);
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            fifo_1 <= 0;
      else if (fifo_1_enable)
            fifo_1 <= fifo_1_mux;
   end


assign fifo_1_mux = (((fifowp == 4'd1) && itm_valid))? itm :
   (((fifowp == 4'd1) && atm_valid))? ovr_pending_atm :
   (((fifowp == 4'd1) && dtm_valid))? ovr_pending_dtm :
   (((fifowp1 == 4'd1) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
   (((fifowp1 == 4'd1) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
   (((fifowp1 == 4'd1) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
   ovr_pending_dtm;

assign fifo_2_enable = ((fifowp == 4'd2) && tm_count_ge1)   || (free2 && (fifowp1== 4'd2)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd2) && tm_count_ge3);
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            fifo_2 <= 0;
      else if (fifo_2_enable)
            fifo_2 <= fifo_2_mux;
   end


assign fifo_2_mux = (((fifowp == 4'd2) && itm_valid))? itm :
   (((fifowp == 4'd2) && atm_valid))? ovr_pending_atm :
   (((fifowp == 4'd2) && dtm_valid))? ovr_pending_dtm :
   (((fifowp1 == 4'd2) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
   (((fifowp1 == 4'd2) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
   (((fifowp1 == 4'd2) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
   ovr_pending_dtm;

assign fifo_3_enable = ((fifowp == 4'd3) && tm_count_ge1)   || (free2 && (fifowp1== 4'd3)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd3) && tm_count_ge3);
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
```

```
            fifo_3 <= 0;
        else if (fifo_3_enable)
            fifo_3 <= fifo_3_mux;
    end


  assign fifo_3_mux = (((fifowp == 4'd3) && itm_valid))? itm :
    (((fifowp == 4'd3) && atm_valid))? ovr_pending_atm :
    (((fifowp == 4'd3) && dtm_valid))? ovr_pending_dtm :
    (((fifowp1 == 4'd3) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
    (((fifowp1 == 4'd3) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
    (((fifowp1 == 4'd3) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
    ovr_pending_dtm;

  assign fifo_4_enable = ((fifowp == 4'd4) && tm_count_ge1)   || (free2 && (fifowp1== 4'd4)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd4) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            fifo_4 <= 0;
      else if (fifo_4_enable)
            fifo_4 <= fifo_4_mux;
    end


  assign fifo_4_mux = (((fifowp == 4'd4) && itm_valid))? itm :
    (((fifowp == 4'd4) && atm_valid))? ovr_pending_atm :
    (((fifowp == 4'd4) && dtm_valid))? ovr_pending_dtm :
    (((fifowp1 == 4'd4) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
    (((fifowp1 == 4'd4) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
    (((fifowp1 == 4'd4) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
    ovr_pending_dtm;

  assign fifo_5_enable = ((fifowp == 4'd5) && tm_count_ge1)   || (free2 && (fifowp1== 4'd5)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd5) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            fifo_5 <= 0;
      else if (fifo_5_enable)
            fifo_5 <= fifo_5_mux;
    end


  assign fifo_5_mux = (((fifowp == 4'd5) && itm_valid))? itm :
    (((fifowp == 4'd5) && atm_valid))? ovr_pending_atm :
    (((fifowp == 4'd5) && dtm_valid))? ovr_pending_dtm :
    (((fifowp1 == 4'd5) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
    (((fifowp1 == 4'd5) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
    (((fifowp1 == 4'd5) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
    ovr_pending_dtm;

  assign fifo_6_enable = ((fifowp == 4'd6) && tm_count_ge1)   || (free2 && (fifowp1== 4'd6)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd6) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            fifo_6 <= 0;
```

```
        else if (fifo_6_enable)
              fifo_6 <= fifo_6_mux;
     end


  assign fifo_6_mux = (((fifowp == 4'd6) && itm_valid))? itm :
     (((fifowp == 4'd6) && atm_valid))? ovr_pending_atm :
     (((fifowp == 4'd6) && dtm_valid))? ovr_pending_dtm :
     (((fifowp1 == 4'd6) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
     (((fifowp1 == 4'd6) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
     (((fifowp1 == 4'd6) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
     ovr_pending_dtm;

  assign fifo_7_enable = ((fifowp == 4'd7) && tm_count_ge1)   || (free2 && (fifowp1== 4'd7)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd7) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
     begin
        if (reset_n == 0)
              fifo_7 <= 0;
        else if (fifo_7_enable)
              fifo_7 <= fifo_7_mux;
     end


  assign fifo_7_mux = (((fifowp == 4'd7) && itm_valid))? itm :
     (((fifowp == 4'd7) && atm_valid))? ovr_pending_atm :
     (((fifowp == 4'd7) && dtm_valid))? ovr_pending_dtm :
     (((fifowp1 == 4'd7) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
     (((fifowp1 == 4'd7) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
     (((fifowp1 == 4'd7) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
     ovr_pending_dtm;

  assign fifo_8_enable = ((fifowp == 4'd8) && tm_count_ge1)   || (free2 && (fifowp1== 4'd8)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd8) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
     begin
        if (reset_n == 0)
              fifo_8 <= 0;
        else if (fifo_8_enable)
              fifo_8 <= fifo_8_mux;
     end


  assign fifo_8_mux = (((fifowp == 4'd8) && itm_valid))? itm :
     (((fifowp == 4'd8) && atm_valid))? ovr_pending_atm :
     (((fifowp == 4'd8) && dtm_valid))? ovr_pending_dtm :
     (((fifowp1 == 4'd8) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
     (((fifowp1 == 4'd8) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
     (((fifowp1 == 4'd8) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
     ovr_pending_dtm;

  assign fifo_9_enable = ((fifowp == 4'd9) && tm_count_ge1)   || (free2 && (fifowp1== 4'd9)
&& tm_count_ge2)   ||(free3 && (fifowp2== 4'd9) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
     begin
        if (reset_n == 0)
              fifo_9 <= 0;
        else if (fifo_9_enable)
```

```
            fifo_9 <= fifo_9_mux;
      end


   assign fifo_9_mux = (((fifowp == 4'd9) && itm_valid))? itm :
      (((fifowp == 4'd9) && atm_valid))? ovr_pending_atm :
      (((fifowp == 4'd9) && dtm_valid))? ovr_pending_dtm :
      (((fifowp1 == 4'd9) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
      (((fifowp1 == 4'd9) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
      (((fifowp1 == 4'd9) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
      ovr_pending_dtm;

   assign fifo_10_enable = ((fifowp == 4'd10) && tm_count_ge1)    || (free2 && (fifowp1== 4'd10)
&& tm_count_ge2)    ||(free3 && (fifowp2== 4'd10) && tm_count_ge3);
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
            fifo_10 <= 0;
         else if (fifo_10_enable)
            fifo_10 <= fifo_10_mux;
      end


   assign fifo_10_mux = (((fifowp == 4'd10) && itm_valid))? itm :
      (((fifowp == 4'd10) && atm_valid))? ovr_pending_atm :
      (((fifowp == 4'd10) && dtm_valid))? ovr_pending_dtm :
      (((fifowp1 == 4'd10) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
      (((fifowp1 == 4'd10) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
      (((fifowp1 == 4'd10) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
      ovr_pending_dtm;

   assign fifo_11_enable = ((fifowp == 4'd11) && tm_count_ge1)    || (free2 && (fifowp1== 4'd11)
&& tm_count_ge2)    ||(free3 && (fifowp2== 4'd11) && tm_count_ge3);
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
            fifo_11 <= 0;
         else if (fifo_11_enable)
            fifo_11 <= fifo_11_mux;
      end


   assign fifo_11_mux = (((fifowp == 4'd11) && itm_valid))? itm :
      (((fifowp == 4'd11) && atm_valid))? ovr_pending_atm :
      (((fifowp == 4'd11) && dtm_valid))? ovr_pending_dtm :
      (((fifowp1 == 4'd11) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
      (((fifowp1 == 4'd11) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
      (((fifowp1 == 4'd11) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
      ovr_pending_dtm;

   assign fifo_12_enable = ((fifowp == 4'd12) && tm_count_ge1)    || (free2 && (fifowp1== 4'd12)
&& tm_count_ge2)    ||(free3 && (fifowp2== 4'd12) && tm_count_ge3);
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
            fifo_12 <= 0;
         else if (fifo_12_enable)
            fifo_12 <= fifo_12_mux;
```

```
    end


  assign fifo_12_mux = (((fifowp == 4'd12) && itm_valid))? itm :
    (((fifowp == 4'd12) && atm_valid))? ovr_pending_atm :
    (((fifowp == 4'd12) && dtm_valid))? ovr_pending_dtm :
    (((fifowp1 == 4'd12) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
    (((fifowp1 == 4'd12) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
    (((fifowp1 == 4'd12) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
    ovr_pending_dtm;

  assign fifo_13_enable = ((fifowp == 4'd13) && tm_count_ge1)    || (free2 && (fifowp1== 4'd13)
&& tm_count_ge2)    ||(free3 && (fifowp2== 4'd13) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          fifo_13 <= 0;
      else if (fifo_13_enable)
          fifo_13 <= fifo_13_mux;
    end


  assign fifo_13_mux = (((fifowp == 4'd13) && itm_valid))? itm :
    (((fifowp == 4'd13) && atm_valid))? ovr_pending_atm :
    (((fifowp == 4'd13) && dtm_valid))? ovr_pending_dtm :
    (((fifowp1 == 4'd13) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
    (((fifowp1 == 4'd13) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
    (((fifowp1 == 4'd13) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
    ovr_pending_dtm;

  assign fifo_14_enable = ((fifowp == 4'd14) && tm_count_ge1)    || (free2 && (fifowp1== 4'd14)
&& tm_count_ge2)    ||(free3 && (fifowp2== 4'd14) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          fifo_14 <= 0;
      else if (fifo_14_enable)
          fifo_14 <= fifo_14_mux;
    end


  assign fifo_14_mux = (((fifowp == 4'd14) && itm_valid))? itm :
    (((fifowp == 4'd14) && atm_valid))? ovr_pending_atm :
    (((fifowp == 4'd14) && dtm_valid))? ovr_pending_dtm :
    (((fifowp1 == 4'd14) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
    (((fifowp1 == 4'd14) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
    (((fifowp1 == 4'd14) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
    ovr_pending_dtm;

  assign fifo_15_enable = ((fifowp == 4'd15) && tm_count_ge1)    || (free2 && (fifowp1== 4'd15)
&& tm_count_ge2)    ||(free3 && (fifowp2== 4'd15) && tm_count_ge3);
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          fifo_15 <= 0;
      else if (fifo_15_enable)
          fifo_15 <= fifo_15_mux;
    end
```

```
    assign fifo_15_mux = (((fifowp == 4'd15) && itm_valid))? itm :
      (((fifowp == 4'd15) && atm_valid))? ovr_pending_atm :
      (((fifowp == 4'd15) && dtm_valid))? ovr_pending_dtm :
      (((fifowp1 == 4'd15) && (free2 & itm_valid & atm_valid)))? ovr_pending_atm :
      (((fifowp1 == 4'd15) && (free2 & itm_valid & dtm_valid)))? ovr_pending_dtm :
      (((fifowp1 == 4'd15) && (free2 & atm_valid & dtm_valid)))? ovr_pending_dtm :
      ovr_pending_dtm;

    assign tm_count_ge1 = |tm_count;
    assign tm_count_ge2 = tm_count[1];
    assign tm_count_ge3 = &tm_count;
    assign ovr_pending_atm = {ovf_pending, atm[34 : 0]};
    assign ovr_pending_dtm = {ovf_pending, dtm[34 : 0]};
    assign fifo_read_mux = (fiforp == 4'd0)? fifo_0 :
      (fiforp == 4'd1)? fifo_1 :
      (fiforp == 4'd2)? fifo_2 :
      (fiforp == 4'd3)? fifo_3 :
      (fiforp == 4'd4)? fifo_4 :
      (fiforp == 4'd5)? fifo_5 :
      (fiforp == 4'd6)? fifo_6 :
      (fiforp == 4'd7)? fifo_7 :
      (fiforp == 4'd8)? fifo_8 :
      (fiforp == 4'd9)? fifo_9 :
      (fiforp == 4'd10)? fifo_10 :
      (fiforp == 4'd11)? fifo_11 :
      (fiforp == 4'd12)? fifo_12 :
      (fiforp == 4'd13)? fifo_13 :
      (fiforp == 4'd14)? fifo_14 :
      fifo_15;


endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_pib (
                             // inputs:
                               clk,
                               clkx2,
                               jrst_n,
                               tw,

                             // outputs:
                               tr_clk,
                               tr_data
                           )
;

    output              tr_clk;
```

```
   output    [ 17: 0] tr_data;
   input               clk;
   input               clkx2;
   input               jrst_n;
   input     [ 35: 0] tw;

   wire                phase;
   wire                tr_clk;
   reg                 tr_clk_reg /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   wire      [ 17: 0] tr_data;
   reg       [ 17: 0] tr_data_reg /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   reg                 x1 /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   reg                 x2 /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=R101" */;
   assign phase = x1^x2;
   always @(posedge clk or negedge jrst_n)
      begin
         if (jrst_n == 0)
               x1 <= 0;
         else
            x1 <= ~x1;
      end


   always @(posedge clkx2 or negedge jrst_n)
      begin
         if (jrst_n == 0)
            begin
               x2 <= 0;
               tr_clk_reg <= 0;
               tr_data_reg <= 0;
            end
         else
            begin
               x2 <= x1;
               tr_clk_reg <= ~phase;
               tr_data_reg <= phase ?     tw[17 : 0] :     tw[35 : 18];
            end
      end


   assign tr_clk = 0 ? tr_clk_reg : 0;
   assign tr_data = 0 ? tr_data_reg : 0;

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030
```

```
module cpu_0_traceram_lpm_dram_bdp_component_module (
                                                      // inputs:
                                                        address_a,
                                                        address_b,
                                                        clock0,
                                                        clock1,
                                                        clocken0,
                                                        clocken1,
                                                        data_a,
                                                        data_b,
                                                        wren_a,
                                                        wren_b,

                                                      // outputs:
                                                        q_a,
                                                        q_b
                                                     )
;

  parameter lpm_file = "UNUSED";


  output    [ 35: 0] q_a;
  output    [ 35: 0] q_b;
  input     [    6: 0] address_a;
  input     [    6: 0] address_b;
  input               clock0;
  input               clock1;
  input               clocken0;
  input               clocken1;
  input     [ 35: 0] data_a;
  input     [ 35: 0] data_b;
  input               wren_a;
  input               wren_b;

  wire      [ 35: 0] q_a;
  wire      [ 35: 0] q_b;
  altsyncram the_altsyncram
    (
      .address_a (address_a),
      .address_b (address_b),
      .clock0 (clock0),
      .clock1 (clock1),
      .clocken0 (clocken0),
      .clocken1 (clocken1),
      .data_a (data_a),
      .data_b (data_b),
      .q_a (q_a),
      .q_b (q_b),
      .wren_a (wren_a),
      .wren_b (wren_b)
    );

  defparam the_altsyncram.address_aclr_a = "NONE",
           the_altsyncram.address_aclr_b = "NONE",
           the_altsyncram.address_reg_b = "CLOCK1",
           the_altsyncram.indata_aclr_a = "NONE",
           the_altsyncram.indata_aclr_b = "NONE",
```

```
                the_altsyncram.init_file = lpm_file,
                the_altsyncram.intended_device_family = "CYCLONEII",
                the_altsyncram.lpm_type = "altsyncram",
                the_altsyncram.numwords_a = 128,
                the_altsyncram.numwords_b = 128,
                the_altsyncram.operation_mode = "BIDIR_DUAL_PORT",
                the_altsyncram.outdata_aclr_a = "NONE",
                the_altsyncram.outdata_aclr_b = "NONE",
                the_altsyncram.outdata_reg_a = "UNREGISTERED",
                the_altsyncram.outdata_reg_b = "UNREGISTERED",
                the_altsyncram.ram_block_type = "AUTO",
                the_altsyncram.read_during_write_mode_mixed_ports = "OLD_DATA",
                the_altsyncram.width_a = 36,
                the_altsyncram.width_b = 36,
                the_altsyncram.widthad_a = 7,
                the_altsyncram.widthad_b = 7,
                the_altsyncram.wrcontrol_aclr_a = "NONE",
                the_altsyncram.wrcontrol_aclr_b = "NONE";


endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci_im (
                                        // inputs:
                                          clk,
                                          jdo,
                                          jrst_n,
                                          reset_n,
                                          take_action_tracectrl,
                                          take_action_tracemem_a,
                                          take_action_tracemem_b,
                                          take_no_action_tracemem_a,
                                          trc_ctrl,
                                          tw,

                                        // outputs:
                                          tracemem_on,
                                          tracemem_trcdata,
                                          tracemem_tw,
                                          trc_enb,
                                          trc_im_addr,
                                          trc_wrap,
                                          xbrk_wrap_traceoff
                                       )
;

  output           tracemem_on;
  output    [ 35: 0] tracemem_trcdata;
  output           tracemem_tw;
```

```
  output              trc_enb;
  output   [  6: 0] trc_im_addr;
  output              trc_wrap;
  output              xbrk_wrap_traceoff;
  input               clk;
  input    [ 37: 0] jdo;
  input               jrst_n;
  input               reset_n;
  input               take_action_tracectrl;
  input               take_action_tracemem_a;
  input               take_action_tracemem_b;
  input               take_no_action_tracemem_a;
  input    [ 15: 0] trc_ctrl;
  input    [ 35: 0] tw;

  wire                tracemem_on;
  wire     [ 35: 0] tracemem_trcdata;
  wire                tracemem_tw;
  wire                trc_enb;
  reg                 [  6:  0] trc_im_addr /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,D103,R101\"" */;
  wire     [ 35: 0] trc_im_data;
  reg                 [ 16:  0] trc_jtag_addr /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=D101" */;
  wire     [ 35: 0] trc_jtag_data;
  wire                trc_on_chip;
  reg                                   trc_wrap /* synthesis ALTERA_ATTRIBUTE =
"SUPPRESS_DA_RULE_INTERNAL=\"D101,D103,R101\"" */;
  wire                tw_valid;
  wire     [ 35: 0] unused_bdpram_port_q_a;
  wire                xbrk_wrap_traceoff;
  assign trc_im_data = tw;
  always @(posedge clk or negedge jrst_n)
    begin
      if (jrst_n == 0)
        begin
          trc_im_addr <= 0;
          trc_wrap <= 0;
        end
      else if (!0)
        begin
          trc_im_addr <= 0;
          trc_wrap <= 0;
        end
      else if (take_action_tracectrl &&
                          (jdo[4] | jdo[3]))
        begin
          if (jdo[4])
              trc_im_addr <= 0;
          if (jdo[3])
              trc_wrap <= 0;
        end
      else if (trc_enb & trc_on_chip & tw_valid)
        begin
          trc_im_addr <= trc_im_addr+1;
          if (&trc_im_addr)
              trc_wrap <= 1;
        end
```

```
      end


   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               trc_jtag_addr <= 0;
         else if (take_action_tracemem_a ||
                        take_no_action_tracemem_a ||
                        take_action_tracemem_b)
               trc_jtag_addr <= take_action_tracemem_a ?
                              jdo[35 : 19] :
                              trc_jtag_addr + 1;

      end


   assign trc_enb = trc_ctrl[0];
   assign trc_on_chip = ~trc_ctrl[8];
   assign tw_valid = |trc_im_data[35 : 32];
   assign xbrk_wrap_traceoff = trc_ctrl[10] & trc_wrap;
   assign tracemem_trcdata = (0) ?
      trc_jtag_data : 0;

   assign tracemem_tw = trc_wrap;
   assign tracemem_on = trc_enb;
//cpu_0_traceram_lpm_dram_bdp_component, which is an nios_tdp_ram
cpu_0_traceram_lpm_dram_bdp_component_module
cpu_0_traceram_lpm_dram_bdp_component
   (
      .address_a (trc_im_addr),
      .address_b (trc_jtag_addr),
      .clock0       (clk),
      .clock1       (clk),
      .clocken0    (1'b1),
      .clocken1    (1'b1),
      .data_a       (trc_im_data),
      .data_b       (jdo[36 : 1]),
      .q_a          (unused_bdpram_port_q_a),
      .q_b          (trc_jtag_data),
      .wren_a       (tw_valid & trc_enb),
      .wren_b       (take_action_tracemem_b)
   );


endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_performance_monitors
;
```

```
endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0_nios2_oci (
                                    // inputs:
                                      D_valid,
                                      E_st_data,
                                      E_valid,
                                      F_pc,
                                      address,
                                      av_ld_data_aligned_filtered,
                                      begintransfer,
                                      byteenable,
                                      chipselect,
                                      clk,
                                      d_address,
                                      d_read,
                                      d_waitrequest,
                                      d_write,
                                      debugaccess,
                                      hbreak_enabled,
                                      reset,
                                      reset_n,
                                      test_ending,
                                      test_has_ended,
                                      write,
                                      writedata,

                                    // outputs:
                                      jtag_debug_module_debugaccess_to_roms,
                                      oci_hbreak_req,
                                      oci_ienable,
                                      oci_single_step_mode,
                                      readdata,
                                      resetrequest
                                   )
;

  output           jtag_debug_module_debugaccess_to_roms;
  output           oci_hbreak_req;
  output   [ 31: 0] oci_ienable;
  output           oci_single_step_mode;
  output   [ 31: 0] readdata;
  output           resetrequest;
  input            D_valid;
  input    [ 31: 0] E_st_data;
  input            E_valid;
```

```
input     [ 22: 0] F_pc;
input     [  8: 0] address;
input     [ 31: 0] av_ld_data_aligned_filtered;
input              begintransfer;
input     [  3: 0] byteenable;
input              chipselect;
input              clk;
input     [ 24: 0] d_address;
input              d_read;
input              d_waitrequest;
input              d_write;
input              debugaccess;
input              hbreak_enabled;
input              reset;
input              reset_n;
input              test_ending;
input              test_has_ended;
input              write;
input     [ 31: 0] writedata;

wire      [ 31: 0] MonDReg;
wire      [ 35: 0] atm;
wire      [ 31: 0] break_readreg;
wire               clkx2;
wire      [ 24: 0] cpu_d_address;
wire               cpu_d_read;
wire      [ 31: 0] cpu_d_readdata;
wire               cpu_d_wait;
wire               cpu_d_write;
wire      [ 31: 0] cpu_d_writedata;
wire               dbrk_break;
wire               dbrk_goto0;
wire               dbrk_goto1;
wire               dbrk_hit0_latch;
wire               dbrk_hit1_latch;
wire               dbrk_hit2_latch;
wire               dbrk_hit3_latch;
wire               dbrk_traceme;
wire               dbrk_traceoff;
wire               dbrk_traceon;
wire               dbrk_trigout;
wire      [ 29: 0] dct_buffer;
wire      [  3: 0] dct_count;
wire               debugack;
wire               debugreq;
wire      [ 35: 0] dtm;
wire               dummy_sink;
wire      [ 35: 0] itm;
wire      [ 37: 0] jdo;
wire               jrst_n;
wire               jtag_debug_module_debugaccess_to_roms;
wire               monitor_error;
wire               monitor_go;
wire               monitor_ready;
wire               oci_hbreak_req;
wire      [ 31: 0] oci_ienable;
wire      [ 31: 0] oci_ram_readdata;
wire      [ 31: 0] oci_reg_readdata;
```

```
wire                    oci_single_step_mode;
wire                    ocireg_ers;
wire                    ocireg_mrs;
wire      [ 31: 0] readdata;
wire                    resetlatch;
wire                    resetrequest;
wire                    st_ready_test_idle;
wire                    take_action_break_a;
wire                    take_action_break_b;
wire                    take_action_break_c;
wire                    take_action_ocimem_a;
wire                    take_action_ocimem_b;
wire                    take_action_ocireg;
wire                    take_action_tracectrl;
wire                    take_action_tracemem_a;
wire                    take_action_tracemem_b;
wire                    take_no_action_break_a;
wire                    take_no_action_break_b;
wire                    take_no_action_break_c;
wire                    take_no_action_ocimem_a;
wire                    take_no_action_tracemem_a;
wire                    tr_clk;
wire      [ 17: 0] tr_data;
wire                    tracemem_on;
wire      [ 35: 0] tracemem_trcdata;
wire                    tracemem_tw;
wire      [ 15: 0] trc_ctrl;
wire                    trc_enb;
wire      [  6: 0] trc_im_addr;
wire                    trc_on;
wire                    trc_wrap;
wire                    trigbrktype;
wire                    trigger_state_0;
wire                    trigger_state_1;
wire                    trigout;
wire      [ 35: 0] tw;
wire                    xbrk_break;
wire      [  7: 0] xbrk_ctrl0;
wire      [  7: 0] xbrk_ctrl1;
wire      [  7: 0] xbrk_ctrl2;
wire      [  7: 0] xbrk_ctrl3;
wire                    xbrk_goto0;
wire                    xbrk_goto1;
wire                    xbrk_traceoff;
wire                    xbrk_traceon;
wire                    xbrk_trigout;
wire                    xbrk_wrap_traceoff;
cpu_0_nios2_oci_debug the_cpu_0_nios2_oci_debug
  (
    .clk                    (clk),
    .dbrk_break             (dbrk_break),
    .debugack               (debugack),
    .debugreq               (debugreq),
    .hbreak_enabled         (hbreak_enabled),
    .jdo                    (jdo),
    .jrst_n                 (jrst_n),
    .monitor_error          (monitor_error),
    .monitor_go             (monitor_go),
```

```
      .monitor_ready              (monitor_ready),
      .oci_hbreak_req             (oci_hbreak_req),
      .ocireg_ers                 (ocireg_ers),
      .ocireg_mrs                 (ocireg_mrs),
      .reset                      (reset),
      .resetlatch                 (resetlatch),
      .resetrequest               (resetrequest),
      .st_ready_test_idle     (st_ready_test_idle),
      .take_action_ocimem_a (take_action_ocimem_a),
      .take_action_ocireg     (take_action_ocireg),
      .xbrk_break                 (xbrk_break)
   );

cpu_0_nios2_ocimem the_cpu_0_nios2_ocimem
   (
      .MonDReg                        (MonDReg),
      .address                        (address),
      .begintransfer                  (begintransfer),
      .byteenable                     (byteenable),
      .chipselect                     (chipselect),
      .clk                            (clk),
      .debugaccess                    (debugaccess),
      .jdo                            (jdo),
      .jrst_n                         (jrst_n),
      .oci_ram_readdata               (oci_ram_readdata),
      .resetrequest                   (resetrequest),
      .take_action_ocimem_a       (take_action_ocimem_a),
      .take_action_ocimem_b       (take_action_ocimem_b),
      .take_no_action_ocimem_a (take_no_action_ocimem_a),
      .write                          (write),
      .writedata                      (writedata)
   );

cpu_0_nios2_avalon_reg the_cpu_0_nios2_avalon_reg
   (
      .address                    (address),
      .chipselect                 (chipselect),
      .clk                        (clk),
      .debugaccess                (debugaccess),
      .monitor_error              (monitor_error),
      .monitor_go                 (monitor_go),
      .monitor_ready              (monitor_ready),
      .oci_ienable                (oci_ienable),
      .oci_reg_readdata           (oci_reg_readdata),
      .oci_single_step_mode (oci_single_step_mode),
      .ocireg_ers                 (ocireg_ers),
      .ocireg_mrs                 (ocireg_mrs),
      .reset_n                    (reset_n),
      .take_action_ocireg     (take_action_ocireg),
      .write                      (write),
      .writedata                  (writedata)
   );

cpu_0_nios2_oci_break the_cpu_0_nios2_oci_break
   (
      .break_readreg              (break_readreg),
      .clk                        (clk),
      .dbrk_break                 (dbrk_break),
```

```
      .dbrk_goto0                    (dbrk_goto0),
      .dbrk_goto1                    (dbrk_goto1),
      .dbrk_hit0_latch               (dbrk_hit0_latch),
      .dbrk_hit1_latch               (dbrk_hit1_latch),
      .dbrk_hit2_latch               (dbrk_hit2_latch),
      .dbrk_hit3_latch               (dbrk_hit3_latch),
      .jdo                           (jdo),
      .jrst_n                        (jrst_n),
      .reset_n                       (reset_n),
      .take_action_break_a           (take_action_break_a),
      .take_action_break_b           (take_action_break_b),
      .take_action_break_c           (take_action_break_c),
      .take_no_action_break_a (take_no_action_break_a),
      .take_no_action_break_b (take_no_action_break_b),
      .take_no_action_break_c (take_no_action_break_c),
      .trigbrktype                   (trigbrktype),
      .trigger_state_0               (trigger_state_0),
      .trigger_state_1               (trigger_state_1),
      .xbrk_ctrl0                    (xbrk_ctrl0),
      .xbrk_ctrl1                    (xbrk_ctrl1),
      .xbrk_ctrl2                    (xbrk_ctrl2),
      .xbrk_ctrl3                    (xbrk_ctrl3),
      .xbrk_goto0                    (xbrk_goto0),
      .xbrk_goto1                    (xbrk_goto1)
    );

cpu_0_nios2_oci_xbrk the_cpu_0_nios2_oci_xbrk
    (
      .D_valid               (D_valid),
      .E_valid               (E_valid),
      .F_pc                  (F_pc),
      .clk                   (clk),
      .reset_n               (reset_n),
      .trigger_state_0 (trigger_state_0),
      .trigger_state_1 (trigger_state_1),
      .xbrk_break            (xbrk_break),
      .xbrk_ctrl0            (xbrk_ctrl0),
      .xbrk_ctrl1            (xbrk_ctrl1),
      .xbrk_ctrl2            (xbrk_ctrl2),
      .xbrk_ctrl3            (xbrk_ctrl3),
      .xbrk_goto0            (xbrk_goto0),
      .xbrk_goto1            (xbrk_goto1),
      .xbrk_traceoff         (xbrk_traceoff),
      .xbrk_traceon          (xbrk_traceon),
      .xbrk_trigout          (xbrk_trigout)
    );

cpu_0_nios2_oci_dbrk the_cpu_0_nios2_oci_dbrk
    (
      .E_st_data                          (E_st_data),
      .av_ld_data_aligned_filtered (av_ld_data_aligned_filtered),
      .clk                                (clk),
      .cpu_d_address                      (cpu_d_address),
      .cpu_d_read                         (cpu_d_read),
      .cpu_d_readdata                     (cpu_d_readdata),
      .cpu_d_wait                         (cpu_d_wait),
      .cpu_d_write                        (cpu_d_write),
      .cpu_d_writedata                    (cpu_d_writedata),
```

```
      .d_address                    (d_address),
      .d_read                       (d_read),
      .d_waitrequest                (d_waitrequest),
      .d_write                      (d_write),
      .dbrk_break                   (dbrk_break),
      .dbrk_goto0                   (dbrk_goto0),
      .dbrk_goto1                   (dbrk_goto1),
      .dbrk_traceme                 (dbrk_traceme),
      .dbrk_traceoff                (dbrk_traceoff),
      .dbrk_traceon                 (dbrk_traceon),
      .dbrk_trigout                 (dbrk_trigout),
      .debugack                     (debugack),
      .reset_n                      (reset_n)
   );

cpu_0_nios2_oci_itrace the_cpu_0_nios2_oci_itrace
   (
      .clk                          (clk),
      .dbrk_traceoff                (dbrk_traceoff),
      .dbrk_traceon                 (dbrk_traceon),
      .dct_buffer                   (dct_buffer),
      .dct_count                    (dct_count),
      .itm                          (itm),
      .jdo                          (jdo),
      .jrst_n                       (jrst_n),
      .take_action_tracectrl (take_action_tracectrl),
      .trc_ctrl                     (trc_ctrl),
      .trc_enb                      (trc_enb),
      .trc_on                       (trc_on),
      .xbrk_traceoff                (xbrk_traceoff),
      .xbrk_traceon                 (xbrk_traceon),
      .xbrk_wrap_traceoff       (xbrk_wrap_traceoff)
   );

cpu_0_nios2_oci_dtrace the_cpu_0_nios2_oci_dtrace
   (
      .atm                      (atm),
      .clk                      (clk),
      .cpu_d_address      (cpu_d_address),
      .cpu_d_read          (cpu_d_read),
      .cpu_d_readdata    (cpu_d_readdata),
      .cpu_d_wait          (cpu_d_wait),
      .cpu_d_write         (cpu_d_write),
      .cpu_d_writedata (cpu_d_writedata),
      .dtm                      (dtm),
      .jrst_n                (jrst_n),
      .trc_ctrl             (trc_ctrl)
   );

cpu_0_nios2_oci_fifo the_cpu_0_nios2_oci_fifo
   (
      .atm                    (atm),
      .clk                    (clk),
      .dbrk_traceme      (dbrk_traceme),
      .dbrk_traceoff    (dbrk_traceoff),
      .dbrk_traceon      (dbrk_traceon),
      .dct_buffer         (dct_buffer),
      .dct_count          (dct_count),
```

```
      .dtm                  (dtm),
      .itm                  (itm),
      .jrst_n               (jrst_n),
      .reset_n              (reset_n),
      .test_ending          (test_ending),
      .test_has_ended (test_has_ended),
      .trc_on               (trc_on),
      .tw                   (tw)
   );

cpu_0_nios2_oci_pib the_cpu_0_nios2_oci_pib
   (
      .clk         (clk),
      .clkx2       (clkx2),
      .jrst_n      (jrst_n),
      .tr_clk      (tr_clk),
      .tr_data (tr_data),
      .tw          (tw)
   );

cpu_0_nios2_oci_im the_cpu_0_nios2_oci_im
   (
      .clk                                (clk),
      .jdo                                (jdo),
      .jrst_n                             (jrst_n),
      .reset_n                            (reset_n),
      .take_action_tracectrl         (take_action_tracectrl),
      .take_action_tracemem_a        (take_action_tracemem_a),
      .take_action_tracemem_b        (take_action_tracemem_b),
      .take_no_action_tracemem_a (take_no_action_tracemem_a),
      .tracemem_on                      (tracemem_on),
      .tracemem_trcdata                 (tracemem_trcdata),
      .tracemem_tw                      (tracemem_tw),
      .trc_ctrl                        (trc_ctrl),
      .trc_enb                          (trc_enb),
      .trc_im_addr                      (trc_im_addr),
      .trc_wrap                         (trc_wrap),
      .tw                               (tw),
      .xbrk_wrap_traceoff              (xbrk_wrap_traceoff)
   );

assign trigout = dbrk_trigout | xbrk_trigout;
assign readdata = address[8] ? oci_reg_readdata : oci_ram_readdata;
assign jtag_debug_module_debugaccess_to_roms = debugack;
cpu_0_jtag_debug_module_wrapper the_cpu_0_jtag_debug_module_wrapper
   (
      .MonDReg                          (MonDReg),
      .break_readreg                   (break_readreg),
      .clk                              (clk),
      .dbrk_hit0_latch                 (dbrk_hit0_latch),
      .dbrk_hit1_latch                 (dbrk_hit1_latch),
      .dbrk_hit2_latch                 (dbrk_hit2_latch),
      .dbrk_hit3_latch                 (dbrk_hit3_latch),
      .debugack                         (debugack),
      .jdo                              (jdo),
      .jrst_n                          (jrst_n),
      .monitor_error                    (monitor_error),
      .monitor_ready                    (monitor_ready),
```

```
        .reset_n                        (reset_n),
        .resetlatch                     (resetlatch),
        .st_ready_test_idle             (st_ready_test_idle),
        .take_action_break_a            (take_action_break_a),
        .take_action_break_b            (take_action_break_b),
        .take_action_break_c            (take_action_break_c),
        .take_action_ocimem_a           (take_action_ocimem_a),
        .take_action_ocimem_b           (take_action_ocimem_b),
        .take_action_tracectrl          (take_action_tracectrl),
        .take_action_tracemem_a         (take_action_tracemem_a),
        .take_action_tracemem_b         (take_action_tracemem_b),
        .take_no_action_break_a         (take_no_action_break_a),
        .take_no_action_break_b         (take_no_action_break_b),
        .take_no_action_break_c         (take_no_action_break_c),
        .take_no_action_ocimem_a        (take_no_action_ocimem_a),
        .take_no_action_tracemem_a      (take_no_action_tracemem_a),
        .tracemem_on                    (tracemem_on),
        .tracemem_trcdata               (tracemem_trcdata),
        .tracemem_tw                    (tracemem_tw),
        .trc_im_addr                    (trc_im_addr),
        .trc_on                         (trc_on),
        .trc_wrap                       (trc_wrap),
        .trigbrktype                    (trigbrktype),
        .trigger_state_1                (trigger_state_1)
      );

    //dummy sink, which is an e_mux
    assign dummy_sink = tr_clk |
      tr_data |
      trigout |
      debugack;

    assign debugreq = 0;
    assign clkx2 = 0;

endmodule


// synthesis translate_off
`timescale 1ns / 1ps
// synthesis translate_on

// turn off superfluous verilog processor warnings
// altera message_level Level1
// altera message_off 10034 10035 10036 10037 10230 10240 10030

module cpu_0 (
                      // inputs:
                        clk,
                        d_irq,
                        d_readdata,
                        d_waitrequest,
                        i_readdata,
                        i_waitrequest,
                        jtag_debug_module_address,
                        jtag_debug_module_begintransfer,
                        jtag_debug_module_byteenable,
                        jtag_debug_module_debugaccess,
```

```
                    jtag_debug_module_select,
                    jtag_debug_module_write,
                    jtag_debug_module_writedata,
                    reset_n,

                  // outputs:
                    d_address,
                    d_byteenable,
                    d_read,
                    d_write,
                    d_writedata,
                    i_address,
                    i_read,
                    jtag_debug_module_debugaccess_to_roms,
                    jtag_debug_module_readdata,
                    jtag_debug_module_resetrequest
                )
;

  output  [ 24: 0] d_address;
  output  [  3: 0] d_byteenable;
  output           d_read;
  output           d_write;
  output  [ 31: 0] d_writedata;
  output  [ 24: 0] i_address;
  output           i_read;
  output           jtag_debug_module_debugaccess_to_roms;
  output  [ 31: 0] jtag_debug_module_readdata;
  output           jtag_debug_module_resetrequest;
  input            clk;
  input   [ 31: 0] d_irq;
  input   [ 31: 0] d_readdata;
  input            d_waitrequest;
  input   [ 31: 0] i_readdata;
  input            i_waitrequest;
  input   [  8: 0] jtag_debug_module_address;
  input            jtag_debug_module_begintransfer;
  input   [  3: 0] jtag_debug_module_byteenable;
  input            jtag_debug_module_debugaccess;
  input            jtag_debug_module_select;
  input            jtag_debug_module_write;
  input   [ 31: 0] jtag_debug_module_writedata;
  input            reset_n;

  wire    [  1: 0] D_compare_op;
  wire             D_ctrl_alu_force_xor;
  wire             D_ctrl_alu_signed_comparison;
  wire             D_ctrl_alu_subtract;
  wire             D_ctrl_b_is_dst;
  wire             D_ctrl_br;
  wire             D_ctrl_br_cmp;
  wire             D_ctrl_br_uncond;
  wire             D_ctrl_break;
  wire             D_ctrl_crst;
  wire             D_ctrl_custom;
  wire             D_ctrl_custom_multi;
  wire             D_ctrl_exception;
  wire             D_ctrl_force_src2_zero;
```

```
wire            D_ctrl_hi_imm16;
wire            D_ctrl_ignore_dst;
wire            D_ctrl_implicit_dst_eretaddr;
wire            D_ctrl_implicit_dst_retaddr;
wire            D_ctrl_jmp_direct;
wire            D_ctrl_jmp_indirect;
wire            D_ctrl_ld;
wire            D_ctrl_ld_io;
wire            D_ctrl_ld_non_io;
wire            D_ctrl_ld_signed;
wire            D_ctrl_logic;
wire            D_ctrl_rdctl_inst;
wire            D_ctrl_retaddr;
wire            D_ctrl_rot_right;
wire            D_ctrl_shift_logical;
wire            D_ctrl_shift_right_arith;
wire            D_ctrl_shift_rot;
wire            D_ctrl_shift_rot_right;
wire            D_ctrl_src2_choose_imm;
wire            D_ctrl_st;
wire            D_ctrl_uncond_cti_non_br;
wire            D_ctrl_unsigned_lo_imm16;
wire            D_ctrl_wrctl_inst;
wire    [  4: 0] D_dst_regnum;
wire    [ 55: 0] D_inst;
reg     [ 31: 0] D_iw /* synthesis ALTERA_IP_DEBUG_VISIBLE = 1 */;
wire    [  4: 0] D_iw_a;
wire    [  4: 0] D_iw_b;
wire    [  4: 0] D_iw_c;
wire    [  2: 0] D_iw_control_regnum;
wire    [  7: 0] D_iw_custom_n;
wire            D_iw_custom_readra;
wire            D_iw_custom_readrb;
wire            D_iw_custom_writerc;
wire    [ 15: 0] D_iw_imm16;
wire    [ 25: 0] D_iw_imm26;
wire    [  4: 0] D_iw_imm5;
wire    [  1: 0] D_iw_memsz;
wire    [  5: 0] D_iw_op;
wire    [  5: 0] D_iw_opx;
wire    [  4: 0] D_iw_shift_imm5;
wire    [  4: 0] D_iw_trap_break_imm5;
wire    [ 22: 0] D_jmp_direct_target_waddr;
wire    [  1: 0] D_logic_op;
wire    [  1: 0] D_logic_op_raw;
wire            D_mem16;
wire            D_mem32;
wire            D_mem8;
wire            D_op_add;
wire            D_op_addi;
wire            D_op_and;
wire            D_op_andhi;
wire            D_op_andi;
wire            D_op_beq;
wire            D_op_bge;
wire            D_op_bgeu;
wire            D_op_blt;
wire            D_op_bltu;
```

```
wire        D_op_bne;
wire        D_op_br;
wire        D_op_break;
wire        D_op_bret;
wire        D_op_call;
wire        D_op_callr;
wire        D_op_cmpeq;
wire        D_op_cmpeqi;
wire        D_op_cmpge;
wire        D_op_cmpgei;
wire        D_op_cmpgeu;
wire        D_op_cmpgeui;
wire        D_op_cmplt;
wire        D_op_cmplti;
wire        D_op_cmpltu;
wire        D_op_cmpltui;
wire        D_op_cmpne;
wire        D_op_cmpnei;
wire        D_op_crst;
wire        D_op_custom;
wire        D_op_div;
wire        D_op_divu;
wire        D_op_eret;
wire        D_op_flushd;
wire        D_op_flushda;
wire        D_op_flushi;
wire        D_op_flushp;
wire        D_op_hbreak;
wire        D_op_initd;
wire        D_op_initda;
wire        D_op_initi;
wire        D_op_intr;
wire        D_op_jmp;
wire        D_op_jmpi;
wire        D_op_ldb;
wire        D_op_ldbio;
wire        D_op_ldbu;
wire        D_op_ldbuio;
wire        D_op_ldh;
wire        D_op_ldhio;
wire        D_op_ldhu;
wire        D_op_ldhuio;
wire        D_op_ldl;
wire        D_op_ldw;
wire        D_op_ldwio;
wire        D_op_mul;
wire        D_op_muli;
wire        D_op_mulxss;
wire        D_op_mulxsu;
wire        D_op_mulxuu;
wire        D_op_nextpc;
wire        D_op_nor;
wire        D_op_opx;
wire        D_op_or;
wire        D_op_orhi;
wire        D_op_ori;
wire        D_op_rdctl;
wire        D_op_rdprs;
```

```
wire            D_op_ret;
wire            D_op_rol;
wire            D_op_roli;
wire            D_op_ror;
wire            D_op_rsv02;
wire            D_op_rsv09;
wire            D_op_rsv10;
wire            D_op_rsv17;
wire            D_op_rsv18;
wire            D_op_rsv25;
wire            D_op_rsv26;
wire            D_op_rsv33;
wire            D_op_rsv34;
wire            D_op_rsv41;
wire            D_op_rsv42;
wire            D_op_rsv49;
wire            D_op_rsv57;
wire            D_op_rsv61;
wire            D_op_rsv62;
wire            D_op_rsv63;
wire            D_op_rsvx00;
wire            D_op_rsvx10;
wire            D_op_rsvx15;
wire            D_op_rsvx17;
wire            D_op_rsvx21;
wire            D_op_rsvx25;
wire            D_op_rsvx33;
wire            D_op_rsvx34;
wire            D_op_rsvx35;
wire            D_op_rsvx42;
wire            D_op_rsvx43;
wire            D_op_rsvx44;
wire            D_op_rsvx47;
wire            D_op_rsvx50;
wire            D_op_rsvx51;
wire            D_op_rsvx55;
wire            D_op_rsvx56;
wire            D_op_rsvx60;
wire            D_op_rsvx63;
wire            D_op_sll;
wire            D_op_slli;
wire            D_op_sra;
wire            D_op_srai;
wire            D_op_srl;
wire            D_op_srli;
wire            D_op_stb;
wire            D_op_stbio;
wire            D_op_stc;
wire            D_op_sth;
wire            D_op_sthio;
wire            D_op_stw;
wire            D_op_stwio;
wire            D_op_sub;
wire            D_op_sync;
wire            D_op_trap;
wire            D_op_wrctl;
wire            D_op_wrprs;
wire            D_op_xor;
```

```
wire            D_op_xorhi;
wire            D_op_xori;
reg             D_valid;
wire    [ 55: 0] D_vinst;
wire            D_wr_dst_reg;
wire    [ 31: 0] E_alu_result;
reg             E_alu_sub;
wire    [ 32: 0] E_arith_result;
wire    [ 31: 0] E_arith_src1;
wire    [ 31: 0] E_arith_src2;
wire            E_ci_multi_stall;
wire    [ 31: 0] E_ci_result;
wire            E_cmp_result;
wire    [ 31: 0] E_control_rd_data;
wire            E_eq;
reg             E_invert_arith_src_msb;
wire            E_ld_stall;
wire    [ 31: 0] E_logic_result;
wire            E_logic_result_is_0;
wire            E_lt;
wire    [ 24: 0] E_mem_baddr;
wire    [  3: 0] E_mem_byte_en;
reg             E_new_inst;
reg     [  4: 0] E_shift_rot_cnt;
wire    [  4: 0] E_shift_rot_cnt_nxt;
wire            E_shift_rot_done;
wire            E_shift_rot_fill_bit;
reg     [ 31: 0] E_shift_rot_result;
wire    [ 31: 0] E_shift_rot_result_nxt;
wire            E_shift_rot_stall;
reg     [ 31: 0] E_src1;
reg     [ 31: 0] E_src2;
wire    [ 31: 0] E_st_data;
wire            E_st_stall;
wire            E_stall;
reg             E_valid;
wire    [ 55: 0] E_vinst;
wire            E_wrctl_bstatus;
wire            E_wrctl_estatus;
wire            E_wrctl_ienable;
wire            E_wrctl_status;
wire    [ 31: 0] F_av_iw;
wire    [  4: 0] F_av_iw_a;
wire    [  4: 0] F_av_iw_b;
wire    [  4: 0] F_av_iw_c;
wire    [  2: 0] F_av_iw_control_regnum;
wire    [  7: 0] F_av_iw_custom_n;
wire            F_av_iw_custom_readra;
wire            F_av_iw_custom_readrb;
wire            F_av_iw_custom_writerc;
wire    [ 15: 0] F_av_iw_imm16;
wire    [ 25: 0] F_av_iw_imm26;
wire    [  4: 0] F_av_iw_imm5;
wire    [  1: 0] F_av_iw_memsz;
wire    [  5: 0] F_av_iw_op;
wire    [  5: 0] F_av_iw_opx;
wire    [  4: 0] F_av_iw_shift_imm5;
wire    [  4: 0] F_av_iw_trap_break_imm5;
```

```
wire            F_av_mem16;
wire            F_av_mem32;
wire            F_av_mem8;
wire    [ 55: 0] F_inst;
wire    [ 31: 0] F_iw;
wire    [  4: 0] F_iw_a;
wire    [  4: 0] F_iw_b;
wire    [  4: 0] F_iw_c;
wire    [  2: 0] F_iw_control_regnum;
wire    [  7: 0] F_iw_custom_n;
wire            F_iw_custom_readra;
wire            F_iw_custom_readrb;
wire            F_iw_custom_writerc;
wire    [ 15: 0] F_iw_imm16;
wire    [ 25: 0] F_iw_imm26;
wire    [  4: 0] F_iw_imm5;
wire    [  1: 0] F_iw_memsz;
wire    [  5: 0] F_iw_op;
wire    [  5: 0] F_iw_opx;
wire    [  4: 0] F_iw_shift_imm5;
wire    [  4: 0] F_iw_trap_break_imm5;
wire            F_mem16;
wire            F_mem32;
wire            F_mem8;
wire            F_op_add;
wire            F_op_addi;
wire            F_op_and;
wire            F_op_andhi;
wire            F_op_andi;
wire            F_op_beq;
wire            F_op_bge;
wire            F_op_bgeu;
wire            F_op_blt;
wire            F_op_bltu;
wire            F_op_bne;
wire            F_op_br;
wire            F_op_break;
wire            F_op_bret;
wire            F_op_call;
wire            F_op_callr;
wire            F_op_cmpeq;
wire            F_op_cmpeqi;
wire            F_op_cmpge;
wire            F_op_cmpgei;
wire            F_op_cmpgeu;
wire            F_op_cmpgeui;
wire            F_op_cmplt;
wire            F_op_cmplti;
wire            F_op_cmpltu;
wire            F_op_cmpltui;
wire            F_op_cmpne;
wire            F_op_cmpnei;
wire            F_op_crst;
wire            F_op_custom;
wire            F_op_div;
wire            F_op_divu;
wire            F_op_eret;
wire            F_op_flushd;
```

```
wire        F_op_flushda;
wire        F_op_flushi;
wire        F_op_flushp;
wire        F_op_hbreak;
wire        F_op_initd;
wire        F_op_initda;
wire        F_op_initi;
wire        F_op_intr;
wire        F_op_jmp;
wire        F_op_jmpi;
wire        F_op_ldb;
wire        F_op_ldbio;
wire        F_op_ldbu;
wire        F_op_ldbuio;
wire        F_op_ldh;
wire        F_op_ldhio;
wire        F_op_ldhu;
wire        F_op_ldhuio;
wire        F_op_ldl;
wire        F_op_ldw;
wire        F_op_ldwio;
wire        F_op_mul;
wire        F_op_muli;
wire        F_op_mulxss;
wire        F_op_mulxsu;
wire        F_op_mulxuu;
wire        F_op_nextpc;
wire        F_op_nor;
wire        F_op_opx;
wire        F_op_or;
wire        F_op_orhi;
wire        F_op_ori;
wire        F_op_rdctl;
wire        F_op_rdprs;
wire        F_op_ret;
wire        F_op_rol;
wire        F_op_roli;
wire        F_op_ror;
wire        F_op_rsv02;
wire        F_op_rsv09;
wire        F_op_rsv10;
wire        F_op_rsv17;
wire        F_op_rsv18;
wire        F_op_rsv25;
wire        F_op_rsv26;
wire        F_op_rsv33;
wire        F_op_rsv34;
wire        F_op_rsv41;
wire        F_op_rsv42;
wire        F_op_rsv49;
wire        F_op_rsv57;
wire        F_op_rsv61;
wire        F_op_rsv62;
wire        F_op_rsv63;
wire        F_op_rsvx00;
wire        F_op_rsvx10;
wire        F_op_rsvx15;
wire        F_op_rsvx17;
```

```
wire            F_op_rsvx21;
wire            F_op_rsvx25;
wire            F_op_rsvx33;
wire            F_op_rsvx34;
wire            F_op_rsvx35;
wire            F_op_rsvx42;
wire            F_op_rsvx43;
wire            F_op_rsvx44;
wire            F_op_rsvx47;
wire            F_op_rsvx50;
wire            F_op_rsvx51;
wire            F_op_rsvx55;
wire            F_op_rsvx56;
wire            F_op_rsvx60;
wire            F_op_rsvx63;
wire            F_op_sll;
wire            F_op_slli;
wire            F_op_sra;
wire            F_op_srai;
wire            F_op_srl;
wire            F_op_srli;
wire            F_op_stb;
wire            F_op_stbio;
wire            F_op_stc;
wire            F_op_sth;
wire            F_op_sthio;
wire            F_op_stw;
wire            F_op_stwio;
wire            F_op_sub;
wire            F_op_sync;
wire            F_op_trap;
wire            F_op_wrctl;
wire            F_op_wrprs;
wire            F_op_xor;
wire            F_op_xorhi;
wire            F_op_xori;
reg     [ 22: 0] F_pc /* synthesis ALTERA_IP_DEBUG_VISIBLE = 1 */;
wire            F_pc_en;
wire    [ 22: 0] F_pc_no_crst_nxt;
wire    [ 22: 0] F_pc_nxt;
wire    [ 22: 0] F_pc_plus_one;
wire    [  1: 0] F_pc_sel_nxt;
wire    [ 24: 0] F_pcb;
wire    [ 24: 0] F_pcb_nxt;
wire    [ 24: 0] F_pcb_plus_four;
wire            F_valid;
wire    [ 55: 0] F_vinst;
reg     [  1: 0] R_compare_op;
reg             R_ctrl_alu_force_xor;
wire            R_ctrl_alu_force_xor_nxt;
reg             R_ctrl_alu_signed_comparison;
wire            R_ctrl_alu_signed_comparison_nxt;
reg             R_ctrl_alu_subtract;
wire            R_ctrl_alu_subtract_nxt;
reg             R_ctrl_b_is_dst;
wire            R_ctrl_b_is_dst_nxt;
reg             R_ctrl_br;
reg             R_ctrl_br_cmp;
```

```
wire        R_ctrl_br_cmp_nxt;
wire        R_ctrl_br_nxt;
reg         R_ctrl_br_uncond;
wire        R_ctrl_br_uncond_nxt;
reg         R_ctrl_break;
wire        R_ctrl_break_nxt;
reg         R_ctrl_crst;
wire        R_ctrl_crst_nxt;
reg         R_ctrl_custom;
reg         R_ctrl_custom_multi;
wire        R_ctrl_custom_multi_nxt;
wire        R_ctrl_custom_nxt;
reg         R_ctrl_exception;
wire        R_ctrl_exception_nxt;
reg         R_ctrl_force_src2_zero;
wire        R_ctrl_force_src2_zero_nxt;
reg         R_ctrl_hi_imm16;
wire        R_ctrl_hi_imm16_nxt;
reg         R_ctrl_ignore_dst;
wire        R_ctrl_ignore_dst_nxt;
reg         R_ctrl_implicit_dst_eretaddr;
wire        R_ctrl_implicit_dst_eretaddr_nxt;
reg         R_ctrl_implicit_dst_retaddr;
wire        R_ctrl_implicit_dst_retaddr_nxt;
reg         R_ctrl_jmp_direct;
wire        R_ctrl_jmp_direct_nxt;
reg         R_ctrl_jmp_indirect;
wire        R_ctrl_jmp_indirect_nxt;
reg         R_ctrl_ld;
reg         R_ctrl_ld_io;
wire        R_ctrl_ld_io_nxt;
reg         R_ctrl_ld_non_io;
wire        R_ctrl_ld_non_io_nxt;
wire        R_ctrl_ld_nxt;
reg         R_ctrl_ld_signed;
wire        R_ctrl_ld_signed_nxt;
reg         R_ctrl_logic;
wire        R_ctrl_logic_nxt;
reg         R_ctrl_rdctl_inst;
wire        R_ctrl_rdctl_inst_nxt;
reg         R_ctrl_retaddr;
wire        R_ctrl_retaddr_nxt;
reg         R_ctrl_rot_right;
wire        R_ctrl_rot_right_nxt;
reg         R_ctrl_shift_logical;
wire        R_ctrl_shift_logical_nxt;
reg         R_ctrl_shift_right_arith;
wire        R_ctrl_shift_right_arith_nxt;
reg         R_ctrl_shift_rot;
wire        R_ctrl_shift_rot_nxt;
reg         R_ctrl_shift_rot_right;
wire        R_ctrl_shift_rot_right_nxt;
reg         R_ctrl_src2_choose_imm;
wire        R_ctrl_src2_choose_imm_nxt;
reg         R_ctrl_st;
wire        R_ctrl_st_nxt;
reg         R_ctrl_uncond_cti_non_br;
wire        R_ctrl_uncond_cti_non_br_nxt;
```

```
reg             R_ctrl_unsigned_lo_imm16;
wire            R_ctrl_unsigned_lo_imm16_nxt;
reg             R_ctrl_wrctl_inst;
wire            R_ctrl_wrctl_inst_nxt;
reg     [  4: 0] R_dst_regnum /* synthesis ALTERA_IP_DEBUG_VISIBLE = 1 */;
wire            R_en;
reg     [  1: 0] R_logic_op;
wire    [ 31: 0] R_rf_a;
wire    [ 31: 0] R_rf_b;
wire    [ 31: 0] R_src1;
wire    [ 31: 0] R_src2;
wire    [ 15: 0] R_src2_hi;
wire    [ 15: 0] R_src2_lo;
reg             R_src2_use_imm;
wire    [  7: 0] R_stb_data;
wire    [ 15: 0] R_sth_data;
reg             R_valid;
wire    [ 55: 0] R_vinst;
reg             R_wr_dst_reg;
reg     [ 31: 0] W_alu_result;
wire            W_br_taken;
reg             W_bstatus_reg;
wire            W_bstatus_reg_inst_nxt;
wire            W_bstatus_reg_nxt;
reg             W_cmp_result;
reg     [ 31: 0] W_control_rd_data;
reg             W_estatus_reg;
wire            W_estatus_reg_inst_nxt;
wire            W_estatus_reg_nxt;
reg     [ 31: 0] W_ienable_reg;
wire    [ 31: 0] W_ienable_reg_nxt;
reg     [ 31: 0] W_ipending_reg;
wire    [ 31: 0] W_ipending_reg_nxt;
wire    [ 24: 0] W_mem_baddr;
wire    [ 31: 0] W_rf_wr_data;
wire            W_rf_wren;
wire            W_status_reg;
reg             W_status_reg_pie;
wire            W_status_reg_pie_inst_nxt;
wire            W_status_reg_pie_nxt;
reg             W_valid /* synthesis ALTERA_IP_DEBUG_VISIBLE = 1 */;
wire    [ 55: 0] W_vinst;
wire    [ 31: 0] W_wr_data;
wire    [ 31: 0] W_wr_data_non_zero;
wire            av_fill_bit;
reg     [  1: 0] av_ld_align_cycle;
wire    [  1: 0] av_ld_align_cycle_nxt;
wire            av_ld_align_one_more_cycle;
reg             av_ld_aligning_data;
wire            av_ld_aligning_data_nxt;
reg     [  7: 0] av_ld_byte0_data;
wire    [  7: 0] av_ld_byte0_data_nxt;
reg     [  7: 0] av_ld_byte1_data;
wire            av_ld_byte1_data_en;
wire    [  7: 0] av_ld_byte1_data_nxt;
reg     [  7: 0] av_ld_byte2_data;
wire    [  7: 0] av_ld_byte2_data_nxt;
reg     [  7: 0] av_ld_byte3_data;
```

```
wire       [    7: 0] av_ld_byte3_data_nxt;
wire       [ 31: 0] av_ld_data_aligned_filtered;
wire       [ 31: 0] av_ld_data_aligned_unfiltered;
wire                av_ld_done;
wire                av_ld_extend;
wire                av_ld_getting_data;
wire                av_ld_rshift8;
reg                 av_ld_waiting_for_data;
wire                av_ld_waiting_for_data_nxt;
wire                av_sign_bit;
wire       [ 24: 0] d_address;
reg        [    3: 0] d_byteenable;
reg                 d_read;
wire                d_read_nxt;
wire                d_write;
wire                d_write_nxt;
reg        [ 31: 0] d_writedata;
reg                 hbreak_enabled;
reg                 hbreak_pending;
wire                hbreak_pending_nxt;
wire                hbreak_req;
wire       [ 24: 0] i_address;
reg                 i_read;
wire                i_read_nxt;
wire       [ 31: 0] iactive;
wire                intr_req;
wire                jtag_debug_module_clk;
wire                jtag_debug_module_debugaccess_to_roms;
wire       [ 31: 0] jtag_debug_module_readdata;
wire                jtag_debug_module_reset;
wire                jtag_debug_module_resetrequest;
wire                oci_hbreak_req;
wire       [ 31: 0] oci_ienable;
wire                oci_single_step_mode;
wire                oci_tb_hbreak_req;
wire                test_ending;
wire                test_has_ended;
reg                 wait_for_one_post_bret_inst;
//the_cpu_0_test_bench, which is an e_instance
cpu_0_test_bench the_cpu_0_test_bench
  (
    .D_iw                               (D_iw),
    .D_iw_op                            (D_iw_op),
    .D_iw_opx                           (D_iw_opx),
    .D_valid                           (D_valid),
    .E_alu_result                      (E_alu_result),
    .E_mem_byte_en                        (E_mem_byte_en),
    .E_st_data                         (E_st_data),
    .E_valid                           (E_valid),
    .F_pcb                              (F_pcb),
    .F_valid                           (F_valid),
    .R_ctrl_exception                  (R_ctrl_exception),
    .R_ctrl_ld                         (R_ctrl_ld),
    .R_ctrl_ld_non_io                  (R_ctrl_ld_non_io),
    .R_dst_regnum                         (R_dst_regnum),
    .R_wr_dst_reg                      (R_wr_dst_reg),
    .W_bstatus_reg                     (W_bstatus_reg),
    .W_cmp_result                       (W_cmp_result),
```

```
        .W_estatus_reg                      (W_estatus_reg),
        .W_ienable_reg                      (W_ienable_reg),
        .W_ipending_reg                     (W_ipending_reg),
        .W_mem_baddr                        (W_mem_baddr),
        .W_rf_wr_data                       (W_rf_wr_data),
        .W_status_reg                       (W_status_reg),
        .W_valid                            (W_valid),
        .W_vinst                            (W_vinst),
        .W_wr_data                          (W_wr_data),
        .av_ld_data_aligned_filtered    (av_ld_data_aligned_filtered),
        .av_ld_data_aligned_unfiltered (av_ld_data_aligned_unfiltered),
        .clk                                (clk),
        .d_address                          (d_address),
        .d_byteenable                       (d_byteenable),
        .d_read                             (d_read),
        .d_write                            (d_write),
        .d_write_nxt                        (d_write_nxt),
        .i_address                          (i_address),
        .i_read                             (i_read),
        .i_readdata                         (i_readdata),
        .i_waitrequest                      (i_waitrequest),
        .reset_n                            (reset_n),
        .test_has_ended                     (test_has_ended)
    );

assign F_av_iw_a = F_av_iw[31 : 27];
assign F_av_iw_b = F_av_iw[26 : 22];
assign F_av_iw_c = F_av_iw[21 : 17];
assign F_av_iw_custom_n = F_av_iw[13 : 6];
assign F_av_iw_custom_readra = F_av_iw[16];
assign F_av_iw_custom_readrb = F_av_iw[15];
assign F_av_iw_custom_writerc = F_av_iw[14];
assign F_av_iw_opx = F_av_iw[16 : 11];
assign F_av_iw_op = F_av_iw[5 : 0];
assign F_av_iw_shift_imm5 = F_av_iw[10 : 6];
assign F_av_iw_trap_break_imm5 = F_av_iw[10 : 6];
assign F_av_iw_imm5 = F_av_iw[10 : 6];
assign F_av_iw_imm16 = F_av_iw[21 : 6];
assign F_av_iw_imm26 = F_av_iw[31 : 6];
assign F_av_iw_memsz = F_av_iw[4 : 3];
assign F_av_iw_control_regnum = F_av_iw[8 : 6];
assign F_av_mem8 = F_av_iw_memsz == 2'b00;
assign F_av_mem16 = F_av_iw_memsz == 2'b01;
assign F_av_mem32 = F_av_iw_memsz[1] == 1'b1;
assign F_iw_a = F_iw[31 : 27];
assign F_iw_b = F_iw[26 : 22];
assign F_iw_c = F_iw[21 : 17];
assign F_iw_custom_n = F_iw[13 : 6];
assign F_iw_custom_readra = F_iw[16];
assign F_iw_custom_readrb = F_iw[15];
assign F_iw_custom_writerc = F_iw[14];
assign F_iw_opx = F_iw[16 : 11];
assign F_iw_op = F_iw[5 : 0];
assign F_iw_shift_imm5 = F_iw[10 : 6];
assign F_iw_trap_break_imm5 = F_iw[10 : 6];
assign F_iw_imm5 = F_iw[10 : 6];
assign F_iw_imm16 = F_iw[21 : 6];
assign F_iw_imm26 = F_iw[31 : 6];
```

```
assign F_iw_memsz = F_iw[4 : 3];
assign F_iw_control_regnum = F_iw[8 : 6];
assign F_mem8 = F_iw_memsz == 2'b00;
assign F_mem16 = F_iw_memsz == 2'b01;
assign F_mem32 = F_iw_memsz[1] == 1'b1;
assign D_iw_a = D_iw[31 : 27];
assign D_iw_b = D_iw[26 : 22];
assign D_iw_c = D_iw[21 : 17];
assign D_iw_custom_n = D_iw[13 : 6];
assign D_iw_custom_readra = D_iw[16];
assign D_iw_custom_readrb = D_iw[15];
assign D_iw_custom_writerc = D_iw[14];
assign D_iw_opx = D_iw[16 : 11];
assign D_iw_op = D_iw[5 : 0];
assign D_iw_shift_imm5 = D_iw[10 : 6];
assign D_iw_trap_break_imm5 = D_iw[10 : 6];
assign D_iw_imm5 = D_iw[10 : 6];
assign D_iw_imm16 = D_iw[21 : 6];
assign D_iw_imm26 = D_iw[31 : 6];
assign D_iw_memsz = D_iw[4 : 3];
assign D_iw_control_regnum = D_iw[8 : 6];
assign D_mem8 = D_iw_memsz == 2'b00;
assign D_mem16 = D_iw_memsz == 2'b01;
assign D_mem32 = D_iw_memsz[1] == 1'b1;
assign F_op_call = F_iw_op == 0;
assign F_op_jmpi = F_iw_op == 1;
assign F_op_ldbu = F_iw_op == 3;
assign F_op_addi = F_iw_op == 4;
assign F_op_stb = F_iw_op == 5;
assign F_op_br = F_iw_op == 6;
assign F_op_ldb = F_iw_op == 7;
assign F_op_cmpgei = F_iw_op == 8;
assign F_op_ldhu = F_iw_op == 11;
assign F_op_andi = F_iw_op == 12;
assign F_op_sth = F_iw_op == 13;
assign F_op_bge = F_iw_op == 14;
assign F_op_ldh = F_iw_op == 15;
assign F_op_cmplti = F_iw_op == 16;
assign F_op_initda = F_iw_op == 19;
assign F_op_ori = F_iw_op == 20;
assign F_op_stw = F_iw_op == 21;
assign F_op_blt = F_iw_op == 22;
assign F_op_ldw = F_iw_op == 23;
assign F_op_cmpnei = F_iw_op == 24;
assign F_op_flushda = F_iw_op == 27;
assign F_op_xori = F_iw_op == 28;
assign F_op_stc = F_iw_op == 29;
assign F_op_bne = F_iw_op == 30;
assign F_op_ldl = F_iw_op == 31;
assign F_op_cmpeqi = F_iw_op == 32;
assign F_op_ldbuio = F_iw_op == 35;
assign F_op_muli = F_iw_op == 36;
assign F_op_stbio = F_iw_op == 37;
assign F_op_beq = F_iw_op == 38;
assign F_op_ldbio = F_iw_op == 39;
assign F_op_cmpgeui = F_iw_op == 40;
assign F_op_ldhuio = F_iw_op == 43;
assign F_op_andhi = F_iw_op == 44;
```

```
assign F_op_sthio = F_iw_op == 45;
assign F_op_bgeu = F_iw_op == 46;
assign F_op_ldhio = F_iw_op == 47;
assign F_op_cmpltui = F_iw_op == 48;
assign F_op_initd = F_iw_op == 51;
assign F_op_orhi = F_iw_op == 52;
assign F_op_stwio = F_iw_op == 53;
assign F_op_bltu = F_iw_op == 54;
assign F_op_ldwio = F_iw_op == 55;
assign F_op_rdprs = F_iw_op == 56;
assign F_op_flushd = F_iw_op == 59;
assign F_op_xorhi = F_iw_op == 60;
assign F_op_rsv02 = F_iw_op == 2;
assign F_op_rsv09 = F_iw_op == 9;
assign F_op_rsv10 = F_iw_op == 10;
assign F_op_rsv17 = F_iw_op == 17;
assign F_op_rsv18 = F_iw_op == 18;
assign F_op_rsv25 = F_iw_op == 25;
assign F_op_rsv26 = F_iw_op == 26;
assign F_op_rsv33 = F_iw_op == 33;
assign F_op_rsv34 = F_iw_op == 34;
assign F_op_rsv41 = F_iw_op == 41;
assign F_op_rsv42 = F_iw_op == 42;
assign F_op_rsv49 = F_iw_op == 49;
assign F_op_rsv57 = F_iw_op == 57;
assign F_op_rsv61 = F_iw_op == 61;
assign F_op_rsv62 = F_iw_op == 62;
assign F_op_rsv63 = F_iw_op == 63;
assign F_op_eret = F_op_opx & (F_iw_opx == 1);
assign F_op_roli = F_op_opx & (F_iw_opx == 2);
assign F_op_rol = F_op_opx & (F_iw_opx == 3);
assign F_op_flushp = F_op_opx & (F_iw_opx == 4);
assign F_op_ret = F_op_opx & (F_iw_opx == 5);
assign F_op_nor = F_op_opx & (F_iw_opx == 6);
assign F_op_mulxuu = F_op_opx & (F_iw_opx == 7);
assign F_op_cmpge = F_op_opx & (F_iw_opx == 8);
assign F_op_bret = F_op_opx & (F_iw_opx == 9);
assign F_op_ror = F_op_opx & (F_iw_opx == 11);
assign F_op_flushi = F_op_opx & (F_iw_opx == 12);
assign F_op_jmp = F_op_opx & (F_iw_opx == 13);
assign F_op_and = F_op_opx & (F_iw_opx == 14);
assign F_op_cmplt = F_op_opx & (F_iw_opx == 16);
assign F_op_slli = F_op_opx & (F_iw_opx == 18);
assign F_op_sll = F_op_opx & (F_iw_opx == 19);
assign F_op_wrprs = F_op_opx & (F_iw_opx == 20);
assign F_op_or = F_op_opx & (F_iw_opx == 22);
assign F_op_mulxsu = F_op_opx & (F_iw_opx == 23);
assign F_op_cmpne = F_op_opx & (F_iw_opx == 24);
assign F_op_srli = F_op_opx & (F_iw_opx == 26);
assign F_op_srl = F_op_opx & (F_iw_opx == 27);
assign F_op_nextpc = F_op_opx & (F_iw_opx == 28);
assign F_op_callr = F_op_opx & (F_iw_opx == 29);
assign F_op_xor = F_op_opx & (F_iw_opx == 30);
assign F_op_mulxss = F_op_opx & (F_iw_opx == 31);
assign F_op_cmpeq = F_op_opx & (F_iw_opx == 32);
assign F_op_divu = F_op_opx & (F_iw_opx == 36);
assign F_op_div = F_op_opx & (F_iw_opx == 37);
assign F_op_rdctl = F_op_opx & (F_iw_opx == 38);
```

```
assign F_op_mul = F_op_opx & (F_iw_opx == 39);
assign F_op_cmpgeu = F_op_opx & (F_iw_opx == 40);
assign F_op_initi = F_op_opx & (F_iw_opx == 41);
assign F_op_trap = F_op_opx & (F_iw_opx == 45);
assign F_op_wrctl = F_op_opx & (F_iw_opx == 46);
assign F_op_cmpltu = F_op_opx & (F_iw_opx == 48);
assign F_op_add = F_op_opx & (F_iw_opx == 49);
assign F_op_break = F_op_opx & (F_iw_opx == 52);
assign F_op_hbreak = F_op_opx & (F_iw_opx == 53);
assign F_op_sync = F_op_opx & (F_iw_opx == 54);
assign F_op_sub = F_op_opx & (F_iw_opx == 57);
assign F_op_srai = F_op_opx & (F_iw_opx == 58);
assign F_op_sra = F_op_opx & (F_iw_opx == 59);
assign F_op_intr = F_op_opx & (F_iw_opx == 61);
assign F_op_crst = F_op_opx & (F_iw_opx == 62);
assign F_op_rsvx00 = F_op_opx & (F_iw_opx == 0);
assign F_op_rsvx10 = F_op_opx & (F_iw_opx == 10);
assign F_op_rsvx15 = F_op_opx & (F_iw_opx == 15);
assign F_op_rsvx17 = F_op_opx & (F_iw_opx == 17);
assign F_op_rsvx21 = F_op_opx & (F_iw_opx == 21);
assign F_op_rsvx25 = F_op_opx & (F_iw_opx == 25);
assign F_op_rsvx33 = F_op_opx & (F_iw_opx == 33);
assign F_op_rsvx34 = F_op_opx & (F_iw_opx == 34);
assign F_op_rsvx35 = F_op_opx & (F_iw_opx == 35);
assign F_op_rsvx42 = F_op_opx & (F_iw_opx == 42);
assign F_op_rsvx43 = F_op_opx & (F_iw_opx == 43);
assign F_op_rsvx44 = F_op_opx & (F_iw_opx == 44);
assign F_op_rsvx47 = F_op_opx & (F_iw_opx == 47);
assign F_op_rsvx50 = F_op_opx & (F_iw_opx == 50);
assign F_op_rsvx51 = F_op_opx & (F_iw_opx == 51);
assign F_op_rsvx55 = F_op_opx & (F_iw_opx == 55);
assign F_op_rsvx56 = F_op_opx & (F_iw_opx == 56);
assign F_op_rsvx60 = F_op_opx & (F_iw_opx == 60);
assign F_op_rsvx63 = F_op_opx & (F_iw_opx == 63);
assign F_op_opx = F_iw_op == 58;
assign F_op_custom = F_iw_op == 50;
assign D_op_call = D_iw_op == 0;
assign D_op_jmpi = D_iw_op == 1;
assign D_op_ldbu = D_iw_op == 3;
assign D_op_addi = D_iw_op == 4;
assign D_op_stb = D_iw_op == 5;
assign D_op_br = D_iw_op == 6;
assign D_op_ldb = D_iw_op == 7;
assign D_op_cmpgei = D_iw_op == 8;
assign D_op_ldhu = D_iw_op == 11;
assign D_op_andi = D_iw_op == 12;
assign D_op_sth = D_iw_op == 13;
assign D_op_bge = D_iw_op == 14;
assign D_op_ldh = D_iw_op == 15;
assign D_op_cmplti = D_iw_op == 16;
assign D_op_initda = D_iw_op == 19;
assign D_op_ori = D_iw_op == 20;
assign D_op_stw = D_iw_op == 21;
assign D_op_blt = D_iw_op == 22;
assign D_op_ldw = D_iw_op == 23;
assign D_op_cmpnei = D_iw_op == 24;
assign D_op_flushda = D_iw_op == 27;
assign D_op_xori = D_iw_op == 28;
```

```
assign D_op_stc = D_iw_op == 29;
assign D_op_bne = D_iw_op == 30;
assign D_op_ldl = D_iw_op == 31;
assign D_op_cmpeqi = D_iw_op == 32;
assign D_op_ldbuio = D_iw_op == 35;
assign D_op_muli = D_iw_op == 36;
assign D_op_stbio = D_iw_op == 37;
assign D_op_beq = D_iw_op == 38;
assign D_op_ldbio = D_iw_op == 39;
assign D_op_cmpgeui = D_iw_op == 40;
assign D_op_ldhuio = D_iw_op == 43;
assign D_op_andhi = D_iw_op == 44;
assign D_op_sthio = D_iw_op == 45;
assign D_op_bgeu = D_iw_op == 46;
assign D_op_ldhio = D_iw_op == 47;
assign D_op_cmpltui = D_iw_op == 48;
assign D_op_initd = D_iw_op == 51;
assign D_op_orhi = D_iw_op == 52;
assign D_op_stwio = D_iw_op == 53;
assign D_op_bltu = D_iw_op == 54;
assign D_op_ldwio = D_iw_op == 55;
assign D_op_rdprs = D_iw_op == 56;
assign D_op_flushd = D_iw_op == 59;
assign D_op_xorhi = D_iw_op == 60;
assign D_op_rsv02 = D_iw_op == 2;
assign D_op_rsv09 = D_iw_op == 9;
assign D_op_rsv10 = D_iw_op == 10;
assign D_op_rsv17 = D_iw_op == 17;
assign D_op_rsv18 = D_iw_op == 18;
assign D_op_rsv25 = D_iw_op == 25;
assign D_op_rsv26 = D_iw_op == 26;
assign D_op_rsv33 = D_iw_op == 33;
assign D_op_rsv34 = D_iw_op == 34;
assign D_op_rsv41 = D_iw_op == 41;
assign D_op_rsv42 = D_iw_op == 42;
assign D_op_rsv49 = D_iw_op == 49;
assign D_op_rsv57 = D_iw_op == 57;
assign D_op_rsv61 = D_iw_op == 61;
assign D_op_rsv62 = D_iw_op == 62;
assign D_op_rsv63 = D_iw_op == 63;
assign D_op_eret = D_op_opx & (D_iw_opx == 1);
assign D_op_roli = D_op_opx & (D_iw_opx == 2);
assign D_op_rol = D_op_opx & (D_iw_opx == 3);
assign D_op_flushp = D_op_opx & (D_iw_opx == 4);
assign D_op_ret = D_op_opx & (D_iw_opx == 5);
assign D_op_nor = D_op_opx & (D_iw_opx == 6);
assign D_op_mulxuu = D_op_opx & (D_iw_opx == 7);
assign D_op_cmpge = D_op_opx & (D_iw_opx == 8);
assign D_op_bret = D_op_opx & (D_iw_opx == 9);
assign D_op_ror = D_op_opx & (D_iw_opx == 11);
assign D_op_flushi = D_op_opx & (D_iw_opx == 12);
assign D_op_jmp = D_op_opx & (D_iw_opx == 13);
assign D_op_and = D_op_opx & (D_iw_opx == 14);
assign D_op_cmplt = D_op_opx & (D_iw_opx == 16);
assign D_op_slli = D_op_opx & (D_iw_opx == 18);
assign D_op_sll = D_op_opx & (D_iw_opx == 19);
assign D_op_wrprs = D_op_opx & (D_iw_opx == 20);
assign D_op_or = D_op_opx & (D_iw_opx == 22);
```

```
assign D_op_mulxsu = D_op_opx & (D_iw_opx == 23);
assign D_op_cmpne = D_op_opx & (D_iw_opx == 24);
assign D_op_srli = D_op_opx & (D_iw_opx == 26);
assign D_op_srl = D_op_opx & (D_iw_opx == 27);
assign D_op_nextpc = D_op_opx & (D_iw_opx == 28);
assign D_op_callr = D_op_opx & (D_iw_opx == 29);
assign D_op_xor = D_op_opx & (D_iw_opx == 30);
assign D_op_mulxss = D_op_opx & (D_iw_opx == 31);
assign D_op_cmpeq = D_op_opx & (D_iw_opx == 32);
assign D_op_divu = D_op_opx & (D_iw_opx == 36);
assign D_op_div = D_op_opx & (D_iw_opx == 37);
assign D_op_rdctl = D_op_opx & (D_iw_opx == 38);
assign D_op_mul = D_op_opx & (D_iw_opx == 39);
assign D_op_cmpgeu = D_op_opx & (D_iw_opx == 40);
assign D_op_initi = D_op_opx & (D_iw_opx == 41);
assign D_op_trap = D_op_opx & (D_iw_opx == 45);
assign D_op_wrctl = D_op_opx & (D_iw_opx == 46);
assign D_op_cmpltu = D_op_opx & (D_iw_opx == 48);
assign D_op_add = D_op_opx & (D_iw_opx == 49);
assign D_op_break = D_op_opx & (D_iw_opx == 52);
assign D_op_hbreak = D_op_opx & (D_iw_opx == 53);
assign D_op_sync = D_op_opx & (D_iw_opx == 54);
assign D_op_sub = D_op_opx & (D_iw_opx == 57);
assign D_op_srai = D_op_opx & (D_iw_opx == 58);
assign D_op_sra = D_op_opx & (D_iw_opx == 59);
assign D_op_intr = D_op_opx & (D_iw_opx == 61);
assign D_op_crst = D_op_opx & (D_iw_opx == 62);
assign D_op_rsvx00 = D_op_opx & (D_iw_opx == 0);
assign D_op_rsvx10 = D_op_opx & (D_iw_opx == 10);
assign D_op_rsvx15 = D_op_opx & (D_iw_opx == 15);
assign D_op_rsvx17 = D_op_opx & (D_iw_opx == 17);
assign D_op_rsvx21 = D_op_opx & (D_iw_opx == 21);
assign D_op_rsvx25 = D_op_opx & (D_iw_opx == 25);
assign D_op_rsvx33 = D_op_opx & (D_iw_opx == 33);
assign D_op_rsvx34 = D_op_opx & (D_iw_opx == 34);
assign D_op_rsvx35 = D_op_opx & (D_iw_opx == 35);
assign D_op_rsvx42 = D_op_opx & (D_iw_opx == 42);
assign D_op_rsvx43 = D_op_opx & (D_iw_opx == 43);
assign D_op_rsvx44 = D_op_opx & (D_iw_opx == 44);
assign D_op_rsvx47 = D_op_opx & (D_iw_opx == 47);
assign D_op_rsvx50 = D_op_opx & (D_iw_opx == 50);
assign D_op_rsvx51 = D_op_opx & (D_iw_opx == 51);
assign D_op_rsvx55 = D_op_opx & (D_iw_opx == 55);
assign D_op_rsvx56 = D_op_opx & (D_iw_opx == 56);
assign D_op_rsvx60 = D_op_opx & (D_iw_opx == 60);
assign D_op_rsvx63 = D_op_opx & (D_iw_opx == 63);
assign D_op_opx = D_iw_op == 58;
assign D_op_custom = D_iw_op == 50;
assign R_en = 1'b1;
assign E_ci_result = 0;
assign E_ci_multi_stall = 1'b0;
assign iactive = d_irq[31 : 0] & 32'b00000000000000000000000000000111;
assign F_pc_sel_nxt = R_ctrl_exception                                  ? 2'b00 :
   R_ctrl_break                                 ? 2'b01 :
   (W_br_taken | R_ctrl_uncond_cti_non_br)      ? 2'b10 :
   2'b11;

assign F_pc_no_crst_nxt = (F_pc_sel_nxt == 2'b00)? 2097160 :
```

```
      (F_pc_sel_nxt == 2'b01)? 4194824 :
      (F_pc_sel_nxt == 2'b10)? E_arith_result[24 : 2] :
      F_pc_plus_one;

   assign F_pc_nxt = F_pc_no_crst_nxt;
   assign F_pcb_nxt = {F_pc_nxt, 2'b00};
   assign F_pc_en = W_valid;
   assign F_pc_plus_one = F_pc + 1;
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               F_pc <= 2097152;
         else if (F_pc_en)
               F_pc <= F_pc_nxt;
      end


   assign F_pcb = {F_pc, 2'b00};
   assign F_pcb_plus_four = {F_pc_plus_one, 2'b00};
   assign F_valid = i_read & ~i_waitrequest;
   assign i_read_nxt = W_valid | (i_read & i_waitrequest);
   assign i_address = {F_pc, 2'b00};
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               i_read <= 1'b1;
         else
            i_read <= i_read_nxt;
      end


   assign oci_tb_hbreak_req = oci_hbreak_req;
   assign hbreak_req = (oci_tb_hbreak_req | hbreak_pending) & hbreak_enabled &
~(wait_for_one_post_bret_inst & ~W_valid);
   assign hbreak_pending_nxt = hbreak_pending ? hbreak_enabled
      : hbreak_req;

   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               wait_for_one_post_bret_inst <= 1'b0;
         else
            wait_for_one_post_bret_inst <= (~hbreak_enabled & oci_single_step_mode) ? 1'b1    :
(F_valid | ~oci_single_step_mode) ? 1'b0    : wait_for_one_post_bret_inst;
      end


   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               hbreak_pending <= 1'b0;
         else
            hbreak_pending <= hbreak_pending_nxt;
      end


   assign intr_req = W_status_reg_pie & (W_ipending_reg != 0);
   assign F_av_iw = i_readdata;
```

```
assign F_iw = hbreak_req          ? 4040762 :
   1'b0      ? 127034 :
   intr_req             ? 3926074 :
   F_av_iw;

always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            D_iw <= 0;
      else if (F_valid)
            D_iw <= F_iw;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            D_valid <= 0;
      else
         D_valid <= F_valid;
   end


assign D_dst_regnum = D_ctrl_implicit_dst_retaddr        ? 5'd31 :
   D_ctrl_implicit_dst_eretaddr      ? 5'd29 :
   D_ctrl_b_is_dst                          ? D_iw_b :
   D_iw_c;

assign D_wr_dst_reg = (D_dst_regnum != 0) & ~D_ctrl_ignore_dst;
assign D_logic_op_raw = D_op_opx ? D_iw_opx[4 : 3] :
   D_iw_op[4 : 3];

assign D_logic_op = D_ctrl_alu_force_xor ? 2'b11 : D_logic_op_raw;
assign D_compare_op = D_op_opx ? D_iw_opx[4 : 3] :
   D_iw_op[4 : 3];

assign D_jmp_direct_target_waddr = D_iw[31 : 6];
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_valid <= 0;
      else
         R_valid <= D_valid;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_wr_dst_reg <= 0;
      else
         R_wr_dst_reg <= D_wr_dst_reg;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
```

```
            R_dst_regnum <= 0;
        else
            R_dst_regnum <= D_dst_regnum;
    end


  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            R_logic_op <= 0;
      else
        R_logic_op <= D_logic_op;
    end


  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            R_compare_op <= 0;
      else
        R_compare_op <= D_compare_op;
    end


  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            R_src2_use_imm <= 0;
      else
        R_src2_use_imm <= D_ctrl_src2_choose_imm | (D_ctrl_br & R_valid);
    end


  assign W_rf_wren = (R_wr_dst_reg & W_valid) | ~reset_n;
  assign W_rf_wr_data = R_ctrl_ld ? av_ld_data_aligned_filtered : W_wr_data;
//cpu_0_register_bank_a, which is an nios_sdp_ram
cpu_0_register_bank_a_module cpu_0_register_bank_a
  (
    .clock      (clk),
    .data       (W_rf_wr_data),
    .q          (R_rf_a),
    .rdaddress (D_iw_a),
    .wraddress (R_dst_regnum),
    .wren       (W_rf_wren)
  );

//synthesis translate_off
`ifdef NO_PLI
defparam cpu_0_register_bank_a.lpm_file = "cpu_0_rf_ram_a.dat";
`else
defparam cpu_0_register_bank_a.lpm_file = "cpu_0_rf_ram_a.hex";
`endif
//synthesis translate_on
//synthesis read_comments_as_HDL on
//defparam cpu_0_register_bank_a.lpm_file = "cpu_0_rf_ram_a.mif";
//synthesis read_comments_as_HDL off
//cpu_0_register_bank_b, which is an nios_sdp_ram
cpu_0_register_bank_b_module cpu_0_register_bank_b
```

```
   (
      .clock         (clk),
      .data          (W_rf_wr_data),
      .q             (R_rf_b),
      .rdaddress (D_iw_b),
      .wraddress (R_dst_regnum),
      .wren          (W_rf_wren)
   );

//synthesis translate_off
`ifdef NO_PLI
defparam cpu_0_register_bank_b.lpm_file = "cpu_0_rf_ram_b.dat";
`else
defparam cpu_0_register_bank_b.lpm_file = "cpu_0_rf_ram_b.hex";
`endif
//synthesis translate_on
//synthesis read_comments_as_HDL on
//defparam cpu_0_register_bank_b.lpm_file = "cpu_0_rf_ram_b.mif";
//synthesis read_comments_as_HDL off
   assign R_src1 = (((R_ctrl_br & E_valid) | (R_ctrl_retaddr & R_valid)))? {F_pc_plus_one, 2'b00} :
      ((R_ctrl_jmp_direct & E_valid))? {D_jmp_direct_target_waddr, 2'b00} :
      R_rf_a;

   assign R_src2_lo = ((R_ctrl_force_src2_zero|R_ctrl_hi_imm16))? 16'b0 :
      (R_src2_use_imm)? D_iw_imm16 :
      R_rf_b[15 : 0];

   assign R_src2_hi = ((R_ctrl_force_src2_zero|R_ctrl_unsigned_lo_imm16))? 16'b0 :
      (R_ctrl_hi_imm16)? D_iw_imm16 :
      (R_src2_use_imm)? {16 {D_iw_imm16[15]}} :
      R_rf_b[31 : 16];

   assign R_src2 = {R_src2_hi, R_src2_lo};
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               E_valid <= 0;
         else
            E_valid <= R_valid | E_stall;
      end


   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               E_new_inst <= 0;
         else
            E_new_inst <= R_valid;
      end


   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               E_src1 <= 0;
         else
            E_src1 <= R_src1;
      end
```

```
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            E_src2 <= 0;
      else
         E_src2 <= R_src2;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            E_invert_arith_src_msb <= 0;
      else
         E_invert_arith_src_msb <= D_ctrl_alu_signed_comparison & R_valid;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            E_alu_sub <= 0;
      else
         E_alu_sub <= D_ctrl_alu_subtract & R_valid;
   end


assign E_stall = E_shift_rot_stall | E_ld_stall | E_st_stall | E_ci_multi_stall;
assign E_arith_src1 = { E_src1[31] ^ E_invert_arith_src_msb,
   E_src1[30 : 0]};

assign E_arith_src2 = { E_src2[31] ^ E_invert_arith_src_msb,
   E_src2[30 : 0]};

assign E_arith_result = E_alu_sub ?
   E_arith_src1 - E_arith_src2 :
   E_arith_src1 + E_arith_src2;

assign E_mem_baddr = E_arith_result[24 : 0];
assign E_logic_result = (R_logic_op == 2'b00)? (~(E_src1 | E_src2)) :
   (R_logic_op == 2'b01)? (E_src1 & E_src2) :
   (R_logic_op == 2'b10)? (E_src1 | E_src2) :
   (E_src1 ^ E_src2);

assign E_logic_result_is_0 = E_logic_result == 0;
assign E_eq = E_logic_result_is_0;
assign E_lt = E_arith_result[32];
assign E_cmp_result = (R_compare_op == 2'b00)? E_eq :
   (R_compare_op == 2'b01)? ~E_lt :
   (R_compare_op == 2'b10)? E_lt :
   ~E_eq;

assign E_shift_rot_cnt_nxt = E_new_inst ? E_src2[4 : 0] : E_shift_rot_cnt-1;
assign E_shift_rot_done = (E_shift_rot_cnt == 0) & ~E_new_inst;
assign E_shift_rot_stall = R_ctrl_shift_rot & E_valid & ~E_shift_rot_done;
assign E_shift_rot_fill_bit = R_ctrl_shift_logical ? 1'b0 :
```

```
    (R_ctrl_rot_right ? E_shift_rot_result[0] :
    E_shift_rot_result[31]);

assign E_shift_rot_result_nxt = (E_new_inst)? E_src1 :
    (R_ctrl_shift_rot_right)? {E_shift_rot_fill_bit, E_shift_rot_result[31 : 1]} :
    {E_shift_rot_result[30 : 0], E_shift_rot_fill_bit};

always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            E_shift_rot_result <= 0;
      else
         E_shift_rot_result <= E_shift_rot_result_nxt;
    end


always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            E_shift_rot_cnt <= 0;
      else
         E_shift_rot_cnt <= E_shift_rot_cnt_nxt;
    end


assign E_control_rd_data = (D_iw_control_regnum == 3'd0)? W_status_reg :
    (D_iw_control_regnum == 3'd1)? W_estatus_reg :
    (D_iw_control_regnum == 3'd2)? W_bstatus_reg :
    (D_iw_control_regnum == 3'd3)? W_ienable_reg :
    (D_iw_control_regnum == 3'd4)? W_ipending_reg :
    0;

assign E_alu_result = ((R_ctrl_br_cmp | R_ctrl_rdctl_inst))? 0 :
    (R_ctrl_shift_rot)? E_shift_rot_result :
    (R_ctrl_logic)? E_logic_result :
    (R_ctrl_custom)? E_ci_result :
    E_arith_result;

assign R_stb_data = R_rf_b[7 : 0];
assign R_sth_data = R_rf_b[15 : 0];
assign E_st_data = (D_mem8)? {R_stb_data, R_stb_data, R_stb_data, R_stb_data} :
    (D_mem16)? {R_sth_data, R_sth_data} :
    R_rf_b;

assign E_mem_byte_en = ({D_iw_memsz, E_mem_baddr[1 : 0]} == {2'b00, 2'b00})? 4'b0001 :
    ({D_iw_memsz, E_mem_baddr[1 : 0]} == {2'b00, 2'b01})? 4'b0010 :
    ({D_iw_memsz, E_mem_baddr[1 : 0]} == {2'b00, 2'b10})? 4'b0100 :
    ({D_iw_memsz, E_mem_baddr[1 : 0]} == {2'b00, 2'b11})? 4'b1000 :
    ({D_iw_memsz, E_mem_baddr[1 : 0]} == {2'b01, 2'b00})? 4'b0011 :
    ({D_iw_memsz, E_mem_baddr[1 : 0]} == {2'b01, 2'b01})? 4'b0011 :
    ({D_iw_memsz, E_mem_baddr[1 : 0]} == {2'b01, 2'b10})? 4'b1100 :
    ({D_iw_memsz, E_mem_baddr[1 : 0]} == {2'b01, 2'b11})? 4'b1100 :
    4'b1111;

assign d_read_nxt = (R_ctrl_ld & E_new_inst) | (d_read & d_waitrequest);
assign E_ld_stall = R_ctrl_ld & ((E_valid & ~av_ld_done) | E_new_inst);
assign d_write_nxt = (R_ctrl_st & E_new_inst) | (d_write & d_waitrequest);
assign E_st_stall = d_write_nxt;
```

```
assign d_address = W_mem_baddr;
assign av_ld_getting_data = d_read & ~d_waitrequest;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            d_read <= 0;
      else
         d_read <= d_read_nxt;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            d_writedata <= 0;
      else
         d_writedata <= E_st_data;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            d_byteenable <= 0;
      else
         d_byteenable <= E_mem_byte_en;
   end


assign av_ld_align_cycle_nxt = av_ld_getting_data ? 0 : (av_ld_align_cycle+1);
assign av_ld_align_one_more_cycle = av_ld_align_cycle == (D_mem16 ? 2 : 3);
assign av_ld_aligning_data_nxt = av_ld_aligning_data ?
   ~av_ld_align_one_more_cycle :
   (~D_mem32 & av_ld_getting_data);

assign av_ld_waiting_for_data_nxt = av_ld_waiting_for_data ?
   ~av_ld_getting_data :
   (R_ctrl_ld & E_new_inst);

assign av_ld_done = ~av_ld_waiting_for_data_nxt & (D_mem32 | ~av_ld_aligning_data_nxt);
assign av_ld_rshift8 = av_ld_aligning_data &
   (av_ld_align_cycle < (W_mem_baddr[1 : 0]));

assign av_ld_extend = av_ld_aligning_data;
assign av_ld_byte0_data_nxt = av_ld_rshift8          ? av_ld_byte1_data :
   av_ld_extend          ? av_ld_byte0_data :
   d_readdata[7 : 0];

assign av_ld_byte1_data_nxt = av_ld_rshift8          ? av_ld_byte2_data :
   av_ld_extend          ? {8 {av_fill_bit}} :
   d_readdata[15 : 8];

assign av_ld_byte2_data_nxt = av_ld_rshift8          ? av_ld_byte3_data :
   av_ld_extend          ? {8 {av_fill_bit}} :
   d_readdata[23 : 16];

assign av_ld_byte3_data_nxt = av_ld_rshift8          ? av_ld_byte3_data :
   av_ld_extend          ? {8 {av_fill_bit}} :
```

```
   d_readdata[31 : 24];

assign av_ld_byte1_data_en = ~(av_ld_extend & D_mem16 & ~av_ld_rshift8);
assign av_ld_data_aligned_unfiltered = {av_ld_byte3_data, av_ld_byte2_data,
   av_ld_byte1_data, av_ld_byte0_data};

assign av_sign_bit = D_mem16 ? av_ld_byte1_data[7] : av_ld_byte0_data[7];
assign av_fill_bit = av_sign_bit & R_ctrl_ld_signed;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            av_ld_align_cycle <= 0;
      else
         av_ld_align_cycle <= av_ld_align_cycle_nxt;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            av_ld_waiting_for_data <= 0;
      else
         av_ld_waiting_for_data <= av_ld_waiting_for_data_nxt;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            av_ld_aligning_data <= 0;
      else
         av_ld_aligning_data <= av_ld_aligning_data_nxt;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            av_ld_byte0_data <= 0;
      else
         av_ld_byte0_data <= av_ld_byte0_data_nxt;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            av_ld_byte1_data <= 0;
      else if (av_ld_byte1_data_en)
            av_ld_byte1_data <= av_ld_byte1_data_nxt;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            av_ld_byte2_data <= 0;
      else
```

```
        av_ld_byte2_data <= av_ld_byte2_data_nxt;
    end


always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            av_ld_byte3_data <= 0;
        else
            av_ld_byte3_data <= av_ld_byte3_data_nxt;
    end


always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            W_valid <= 0;
        else
            W_valid <= E_valid & ~E_stall;
    end


always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            W_control_rd_data <= 0;
        else
            W_control_rd_data <= E_control_rd_data;
    end


always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            W_cmp_result <= 0;
        else
            W_cmp_result <= E_cmp_result;
    end


always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            W_alu_result <= 0;
        else
            W_alu_result <= E_alu_result;
    end


always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            W_status_reg_pie <= 0;
        else
            W_status_reg_pie <= W_status_reg_pie_nxt;
    end
```

```
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            W_estatus_reg <= 0;
      else
         W_estatus_reg <= W_estatus_reg_nxt;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            W_bstatus_reg <= 0;
      else
         W_bstatus_reg <= W_bstatus_reg_nxt;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            W_ienable_reg <= 0;
      else
         W_ienable_reg <= W_ienable_reg_nxt;
   end


always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            W_ipending_reg <= 0;
      else
         W_ipending_reg <= W_ipending_reg_nxt;
   end


assign W_wr_data_non_zero = R_ctrl_br_cmp ? W_cmp_result :
   R_ctrl_rdctl_inst          ? W_control_rd_data :
   W_alu_result[31 : 0];

assign W_wr_data = W_wr_data_non_zero;
assign W_br_taken = R_ctrl_br & W_cmp_result;
assign W_mem_baddr = W_alu_result[24 : 0];
assign W_status_reg = W_status_reg_pie;
assign E_wrctl_status = R_ctrl_wrctl_inst &
   (D_iw_control_regnum == 3'd0);

assign E_wrctl_estatus = R_ctrl_wrctl_inst &
   (D_iw_control_regnum == 3'd1);

assign E_wrctl_bstatus = R_ctrl_wrctl_inst &
   (D_iw_control_regnum == 3'd2);

assign E_wrctl_ienable = R_ctrl_wrctl_inst &
   (D_iw_control_regnum == 3'd3);

assign W_status_reg_pie_inst_nxt = (R_ctrl_exception | R_ctrl_break | R_ctrl_crst) ? 1'b0 :
   (D_op_eret)                          ? W_estatus_reg :
```

```
      (D_op_bret)                              ? W_bstatus_reg :
      (E_wrctl_status)                         ? E_src1[0] :
      W_status_reg_pie;

   assign W_status_reg_pie_nxt = E_valid ? W_status_reg_pie_inst_nxt : W_status_reg_pie;
   assign W_estatus_reg_inst_nxt = (R_ctrl_crst)              ? 0 :
      (R_ctrl_exception)      ? W_status_reg :
      (E_wrctl_estatus)       ? E_src1[0] :
      W_estatus_reg;

   assign W_estatus_reg_nxt = E_valid ? W_estatus_reg_inst_nxt : W_estatus_reg;
   assign W_bstatus_reg_inst_nxt = (R_ctrl_break)             ? W_status_reg :
      (E_wrctl_bstatus)       ? E_src1[0] :
      W_bstatus_reg;

   assign W_bstatus_reg_nxt = E_valid ? W_bstatus_reg_inst_nxt : W_bstatus_reg;
   assign W_ienable_reg_nxt = ((E_wrctl_ienable & E_valid) ?
      E_src1[31 : 0] : W_ienable_reg) & 32'b00000000000000000000000000000111;

   assign W_ipending_reg_nxt = iactive & W_ienable_reg & oci_ienable &
32'b00000000000000000000000000000111;
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               hbreak_enabled <= 1'b1;
         else if (E_valid)
               hbreak_enabled <= R_ctrl_break ? 1'b0 : D_op_bret ? 1'b1 : hbreak_enabled;
      end


   cpu_0_nios2_oci the_cpu_0_nios2_oci
      (
         .D_valid                                (D_valid),
         .E_st_data                              (E_st_data),
         .E_valid                                (E_valid),
         .F_pc                                   (F_pc),
         .address                                (jtag_debug_module_address),
         .av_ld_data_aligned_filtered            (av_ld_data_aligned_filtered),
         .begintransfer                          (jtag_debug_module_begintransfer),
         .byteenable                             (jtag_debug_module_byteenable),
         .chipselect                             (jtag_debug_module_select),
         .clk                                    (jtag_debug_module_clk),
         .d_address                              (d_address),
         .d_read                                 (d_read),
         .d_waitrequest                          (d_waitrequest),
         .d_write                                (d_write),
         .debugaccess                            (jtag_debug_module_debugaccess),
         .hbreak_enabled                         (hbreak_enabled),
         .jtag_debug_module_debugaccess_to_roms
(jtag_debug_module_debugaccess_to_roms),
         .oci_hbreak_req                         (oci_hbreak_req),
         .oci_ienable                            (oci_ienable),
         .oci_single_step_mode                   (oci_single_step_mode),
         .readdata                               (jtag_debug_module_readdata),
         .reset                                  (jtag_debug_module_reset),
         .reset_n                                (reset_n),
         .resetrequest                           (jtag_debug_module_resetrequest),
         .test_ending                            (test_ending),
```

```
      .test_has_ended                       (test_has_ended),
      .write                                (jtag_debug_module_write),
      .writedata                            (jtag_debug_module_writedata)
   );

//jtag_debug_module, which is an e_avalon_slave
assign jtag_debug_module_clk = clk;
assign jtag_debug_module_reset = ~reset_n;
assign D_ctrl_custom = 1'b0;
assign R_ctrl_custom_nxt = D_ctrl_custom;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_custom <= 0;
      else if (R_en)
            R_ctrl_custom <= R_ctrl_custom_nxt;
   end


assign D_ctrl_custom_multi = 1'b0;
assign R_ctrl_custom_multi_nxt = D_ctrl_custom_multi;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_custom_multi <= 0;
      else if (R_en)
            R_ctrl_custom_multi <= R_ctrl_custom_multi_nxt;
   end


assign D_ctrl_jmp_indirect = D_op_eret|
   D_op_bret|
   D_op_rsvx17|
   D_op_rsvx25|
   D_op_ret|
   D_op_jmp|
   D_op_rsvx21|
   D_op_callr;

assign R_ctrl_jmp_indirect_nxt = D_ctrl_jmp_indirect;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_jmp_indirect <= 0;
      else if (R_en)
            R_ctrl_jmp_indirect <= R_ctrl_jmp_indirect_nxt;
   end


assign D_ctrl_jmp_direct = D_op_call|D_op_jmpi;
assign R_ctrl_jmp_direct_nxt = D_ctrl_jmp_direct;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_jmp_direct <= 0;
      else if (R_en)
            R_ctrl_jmp_direct <= R_ctrl_jmp_direct_nxt;
   end
```

```
  assign D_ctrl_implicit_dst_retaddr = D_op_call|D_op_rsv02;
  assign R_ctrl_implicit_dst_retaddr_nxt = D_ctrl_implicit_dst_retaddr;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          R_ctrl_implicit_dst_retaddr <= 0;
      else if (R_en)
          R_ctrl_implicit_dst_retaddr <= R_ctrl_implicit_dst_retaddr_nxt;
    end


  assign                                D_ctrl_implicit_dst_eretaddr                                =
D_op_div|D_op_divu|D_op_mul|D_op_muli|D_op_mulxss|D_op_mulxsu|D_op_mulxuu;
  assign R_ctrl_implicit_dst_eretaddr_nxt = D_ctrl_implicit_dst_eretaddr;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          R_ctrl_implicit_dst_eretaddr <= 0;
      else if (R_en)
          R_ctrl_implicit_dst_eretaddr <= R_ctrl_implicit_dst_eretaddr_nxt;
    end


  assign D_ctrl_exception = D_op_trap|
    D_op_rsvx44|
    D_op_div|
    D_op_divu|
    D_op_mul|
    D_op_muli|
    D_op_mulxss|
    D_op_mulxsu|
    D_op_mulxuu|
    D_op_intr|
    D_op_rsvx60;

  assign R_ctrl_exception_nxt = D_ctrl_exception;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          R_ctrl_exception <= 0;
      else if (R_en)
          R_ctrl_exception <= R_ctrl_exception_nxt;
    end


  assign D_ctrl_break = D_op_break|D_op_hbreak;
  assign R_ctrl_break_nxt = D_ctrl_break;
  always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
          R_ctrl_break <= 0;
      else if (R_en)
          R_ctrl_break <= R_ctrl_break_nxt;
    end
```

```
assign D_ctrl_crst = D_op_crst|D_op_rsvx63;
assign R_ctrl_crst_nxt = D_ctrl_crst;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_crst <= 0;
      else if (R_en)
            R_ctrl_crst <= R_ctrl_crst_nxt;
   end


assign D_ctrl_uncond_cti_non_br = D_op_call|
   D_op_jmpi|
   D_op_eret|
   D_op_bret|
   D_op_rsvx17|
   D_op_rsvx25|
   D_op_ret|
   D_op_jmp|
   D_op_rsvx21|
   D_op_callr;

assign R_ctrl_uncond_cti_non_br_nxt = D_ctrl_uncond_cti_non_br;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_uncond_cti_non_br <= 0;
      else if (R_en)
            R_ctrl_uncond_cti_non_br <= R_ctrl_uncond_cti_non_br_nxt;
   end


assign D_ctrl_retaddr = D_op_call|
   D_op_rsv02|
   D_op_nextpc|
   D_op_callr|
   D_op_trap|
   D_op_rsvx44|
   D_op_div|
   D_op_divu|
   D_op_mul|
   D_op_muli|
   D_op_mulxss|
   D_op_mulxsu|
   D_op_mulxuu|
   D_op_intr|
   D_op_rsvx60|
   D_op_break|
   D_op_hbreak;

assign R_ctrl_retaddr_nxt = D_ctrl_retaddr;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_retaddr <= 0;
      else if (R_en)
            R_ctrl_retaddr <= R_ctrl_retaddr_nxt;
   end
```

```
assign D_ctrl_shift_logical = D_op_slli|D_op_sll|D_op_srli|D_op_srl;
assign R_ctrl_shift_logical_nxt = D_ctrl_shift_logical;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_shift_logical <= 0;
      else if (R_en)
            R_ctrl_shift_logical <= R_ctrl_shift_logical_nxt;
   end


assign D_ctrl_shift_right_arith = D_op_srai|D_op_sra;
assign R_ctrl_shift_right_arith_nxt = D_ctrl_shift_right_arith;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_shift_right_arith <= 0;
      else if (R_en)
            R_ctrl_shift_right_arith <= R_ctrl_shift_right_arith_nxt;
   end


assign D_ctrl_rot_right = D_op_rsvx10|D_op_ror|D_op_rsvx42|D_op_rsvx43;
assign R_ctrl_rot_right_nxt = D_ctrl_rot_right;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_rot_right <= 0;
      else if (R_en)
            R_ctrl_rot_right <= R_ctrl_rot_right_nxt;
   end


assign D_ctrl_shift_rot_right = D_op_srli|
   D_op_srl|
   D_op_srai|
   D_op_sra|
   D_op_rsvx10|
   D_op_ror|
   D_op_rsvx42|
   D_op_rsvx43;

assign R_ctrl_shift_rot_right_nxt = D_ctrl_shift_rot_right;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_shift_rot_right <= 0;
      else if (R_en)
            R_ctrl_shift_rot_right <= R_ctrl_shift_rot_right_nxt;
   end


assign D_ctrl_shift_rot = D_op_slli|
   D_op_rsvx50|
   D_op_sll|
   D_op_rsvx51|
```

```
    D_op_roli|
    D_op_rsvx34|
    D_op_rol|
    D_op_rsvx35|
    D_op_srli|
    D_op_srl|
    D_op_srai|
    D_op_sra|
    D_op_rsvx10|
    D_op_ror|
    D_op_rsvx42|
    D_op_rsvx43;

assign R_ctrl_shift_rot_nxt = D_ctrl_shift_rot;
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            R_ctrl_shift_rot <= 0;
        else if (R_en)
            R_ctrl_shift_rot <= R_ctrl_shift_rot_nxt;
    end


assign D_ctrl_logic = D_op_and|
    D_op_or|
    D_op_xor|
    D_op_nor|
    D_op_andhi|
    D_op_orhi|
    D_op_xorhi|
    D_op_andi|
    D_op_ori|
    D_op_xori;

assign R_ctrl_logic_nxt = D_ctrl_logic;
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            R_ctrl_logic <= 0;
        else if (R_en)
            R_ctrl_logic <= R_ctrl_logic_nxt;
    end


assign D_ctrl_hi_imm16 = D_op_andhi|D_op_orhi|D_op_xorhi;
assign R_ctrl_hi_imm16_nxt = D_ctrl_hi_imm16;
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
            R_ctrl_hi_imm16 <= 0;
        else if (R_en)
            R_ctrl_hi_imm16 <= R_ctrl_hi_imm16_nxt;
    end


assign D_ctrl_unsigned_lo_imm16 = D_op_cmpgeui|
    D_op_cmpltui|
    D_op_andi|
```

```
    D_op_ori|
    D_op_xori|
    D_op_roli|
    D_op_rsvx10|
    D_op_slli|
    D_op_srli|
    D_op_rsvx34|
    D_op_rsvx42|
    D_op_rsvx50|
    D_op_srai;

assign R_ctrl_unsigned_lo_imm16_nxt = D_ctrl_unsigned_lo_imm16;
always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            R_ctrl_unsigned_lo_imm16 <= 0;
      else if (R_en)
            R_ctrl_unsigned_lo_imm16 <= R_ctrl_unsigned_lo_imm16_nxt;
    end


assign D_ctrl_br_uncond = D_op_br|D_op_rsv02;
assign R_ctrl_br_uncond_nxt = D_ctrl_br_uncond;
always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            R_ctrl_br_uncond <= 0;
      else if (R_en)
            R_ctrl_br_uncond <= R_ctrl_br_uncond_nxt;
    end


assign D_ctrl_br = D_op_br|
    D_op_bge|
    D_op_blt|
    D_op_bne|
    D_op_beq|
    D_op_bgeu|
    D_op_bltu|
    D_op_rsv62;

assign R_ctrl_br_nxt = D_ctrl_br;
always @(posedge clk or negedge reset_n)
    begin
      if (reset_n == 0)
            R_ctrl_br <= 0;
      else if (R_en)
            R_ctrl_br <= R_ctrl_br_nxt;
    end


assign D_ctrl_alu_subtract = D_op_sub|
    D_op_rsvx25|
    D_op_cmplti|
    D_op_cmpltui|
    D_op_cmplt|
    D_op_cmpltu|
    D_op_blt|
```

```
      D_op_bltu|
      D_op_cmpgei|
      D_op_cmpgeui|
      D_op_cmpge|
      D_op_cmpgeu|
      D_op_bge|
      D_op_rsv10|
      D_op_bgeu|
      D_op_rsv42;

   assign R_ctrl_alu_subtract_nxt = D_ctrl_alu_subtract;
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               R_ctrl_alu_subtract <= 0;
         else if (R_en)
               R_ctrl_alu_subtract <= R_ctrl_alu_subtract_nxt;
      end


   assign                                D_ctrl_alu_signed_comparison                                =
D_op_cmpge|D_op_cmpgei|D_op_cmplt|D_op_cmplti|D_op_bge|D_op_blt;
   assign R_ctrl_alu_signed_comparison_nxt = D_ctrl_alu_signed_comparison;
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               R_ctrl_alu_signed_comparison <= 0;
         else if (R_en)
               R_ctrl_alu_signed_comparison <= R_ctrl_alu_signed_comparison_nxt;
      end


   assign D_ctrl_br_cmp = D_op_br|
      D_op_bge|
      D_op_blt|
      D_op_bne|
      D_op_beq|
      D_op_bgeu|
      D_op_bltu|
      D_op_rsv62|
      D_op_cmpgei|
      D_op_cmplti|
      D_op_cmpnei|
      D_op_cmpgeui|
      D_op_cmpltui|
      D_op_cmpeqi|
      D_op_rsvx00|
      D_op_cmpge|
      D_op_cmplt|
      D_op_cmpne|
      D_op_cmpgeu|
      D_op_cmpltu|
      D_op_cmpeq|
      D_op_rsvx56;

   assign R_ctrl_br_cmp_nxt = D_ctrl_br_cmp;
   always @(posedge clk or negedge reset_n)
      begin
```

```
    if (reset_n == 0)
        R_ctrl_br_cmp <= 0;
    else if (R_en)
        R_ctrl_br_cmp <= R_ctrl_br_cmp_nxt;
  end


assign D_ctrl_ld_signed = D_op_ldb|
  D_op_ldh|
  D_op_ldl|
  D_op_ldw|
  D_op_ldbio|
  D_op_ldhio|
  D_op_ldwio|
  D_op_rsv63;

assign R_ctrl_ld_signed_nxt = D_ctrl_ld_signed;
always @(posedge clk or negedge reset_n)
  begin
    if (reset_n == 0)
        R_ctrl_ld_signed <= 0;
    else if (R_en)
        R_ctrl_ld_signed <= R_ctrl_ld_signed_nxt;
  end


assign D_ctrl_ld = D_op_ldb|
  D_op_ldh|
  D_op_ldl|
  D_op_ldw|
  D_op_ldbio|
  D_op_ldhio|
  D_op_ldwio|
  D_op_rsv63|
  D_op_ldbu|
  D_op_ldhu|
  D_op_ldbuio|
  D_op_ldhuio;

assign R_ctrl_ld_nxt = D_ctrl_ld;
always @(posedge clk or negedge reset_n)
  begin
    if (reset_n == 0)
        R_ctrl_ld <= 0;
    else if (R_en)
        R_ctrl_ld <= R_ctrl_ld_nxt;
  end


assign D_ctrl_ld_non_io = D_op_ldbu|D_op_ldhu|D_op_ldb|D_op_ldh|D_op_ldw|D_op_ldl;
assign R_ctrl_ld_non_io_nxt = D_ctrl_ld_non_io;
always @(posedge clk or negedge reset_n)
  begin
    if (reset_n == 0)
        R_ctrl_ld_non_io <= 0;
    else if (R_en)
        R_ctrl_ld_non_io <= R_ctrl_ld_non_io_nxt;
  end
```

```
  assign D_ctrl_st = D_op_stb|
     D_op_sth|
     D_op_stw|
     D_op_stc|
     D_op_stbio|
     D_op_sthio|
     D_op_stwio|
     D_op_rsv61;

  assign R_ctrl_st_nxt = D_ctrl_st;
  always @(posedge clk or negedge reset_n)
     begin
        if (reset_n == 0)
              R_ctrl_st <= 0;
        else if (R_en)
              R_ctrl_st <= R_ctrl_st_nxt;
     end


  assign                                    D_ctrl_ld_io                                    =
D_op_ldbuio|D_op_ldhuio|D_op_ldbio|D_op_ldhio|D_op_ldwio|D_op_rsv63;
  assign R_ctrl_ld_io_nxt = D_ctrl_ld_io;
  always @(posedge clk or negedge reset_n)
     begin
        if (reset_n == 0)
              R_ctrl_ld_io <= 0;
        else if (R_en)
              R_ctrl_ld_io <= R_ctrl_ld_io_nxt;
     end


  assign D_ctrl_b_is_dst = D_op_addi|
     D_op_andhi|
     D_op_orhi|
     D_op_xorhi|
     D_op_andi|
     D_op_ori|
     D_op_xori|
     D_op_call|
     D_op_rdprs|
     D_op_cmpgei|
     D_op_cmplti|
     D_op_cmpnei|
     D_op_cmpgeui|
     D_op_cmpltui|
     D_op_cmpeqi|
     D_op_jmpi|
     D_op_rsv09|
     D_op_rsv17|
     D_op_rsv25|
     D_op_rsv33|
     D_op_rsv41|
     D_op_rsv49|
     D_op_rsv57|
     D_op_ldb|
     D_op_ldh|
```

```
    D_op_ldl|
    D_op_ldw|
    D_op_ldbio|
    D_op_ldhio|
    D_op_ldwio|
    D_op_rsv63|
    D_op_ldbu|
    D_op_ldhu|
    D_op_ldbuio|
    D_op_ldhuio|
    D_op_initd|
    D_op_initda|
    D_op_flushd|
    D_op_flushda;

assign R_ctrl_b_is_dst_nxt = D_ctrl_b_is_dst;
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
                R_ctrl_b_is_dst <= 0;
        else if (R_en)
                R_ctrl_b_is_dst <= R_ctrl_b_is_dst_nxt;
    end


assign D_ctrl_ignore_dst = D_op_br|
    D_op_bge|
    D_op_blt|
    D_op_bne|
    D_op_beq|
    D_op_bgeu|
    D_op_bltu|
    D_op_rsv62|
    D_op_stb|
    D_op_sth|
    D_op_stw|
    D_op_stc|
    D_op_stbio|
    D_op_sthio|
    D_op_stwio|
    D_op_rsv61|
    D_op_jmpi|
    D_op_rsv09|
    D_op_rsv17|
    D_op_rsv25|
    D_op_rsv33|
    D_op_rsv41|
    D_op_rsv49|
    D_op_rsv57;

assign R_ctrl_ignore_dst_nxt = D_ctrl_ignore_dst;
always @(posedge clk or negedge reset_n)
    begin
        if (reset_n == 0)
                R_ctrl_ignore_dst <= 0;
        else if (R_en)
                R_ctrl_ignore_dst <= R_ctrl_ignore_dst_nxt;
    end
```

```
assign D_ctrl_src2_choose_imm = D_op_addi|
    D_op_andhi|
    D_op_orhi|
    D_op_xorhi|
    D_op_andi|
    D_op_ori|
    D_op_xori|
    D_op_call|
    D_op_rdprs|
    D_op_cmpgei|
    D_op_cmplti|
    D_op_cmpnei|
    D_op_cmpgeui|
    D_op_cmpltui|
    D_op_cmpeqi|
    D_op_jmpi|
    D_op_rsv09|
    D_op_rsv17|
    D_op_rsv25|
    D_op_rsv33|
    D_op_rsv41|
    D_op_rsv49|
    D_op_rsv57|
    D_op_ldb|
    D_op_ldh|
    D_op_ldl|
    D_op_ldw|
    D_op_ldbio|
    D_op_ldhio|
    D_op_ldwio|
    D_op_rsv63|
    D_op_ldbu|
    D_op_ldhu|
    D_op_ldbuio|
    D_op_ldhuio|
    D_op_initd|
    D_op_initda|
    D_op_flushd|
    D_op_flushda|
    D_op_stb|
    D_op_sth|
    D_op_stw|
    D_op_stc|
    D_op_stbio|
    D_op_sthio|
    D_op_stwio|
    D_op_rsv61|
    D_op_roli|
    D_op_rsvx10|
    D_op_slli|
    D_op_srli|
    D_op_rsvx34|
    D_op_rsvx42|
    D_op_rsvx50|
    D_op_srai;
```

```
assign R_ctrl_src2_choose_imm_nxt = D_ctrl_src2_choose_imm;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_src2_choose_imm <= 0;
      else if (R_en)
            R_ctrl_src2_choose_imm <= R_ctrl_src2_choose_imm_nxt;
   end


assign D_ctrl_wrctl_inst = D_op_wrctl;
assign R_ctrl_wrctl_inst_nxt = D_ctrl_wrctl_inst;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_wrctl_inst <= 0;
      else if (R_en)
            R_ctrl_wrctl_inst <= R_ctrl_wrctl_inst_nxt;
   end


assign D_ctrl_rdctl_inst = D_op_rdctl;
assign R_ctrl_rdctl_inst_nxt = D_ctrl_rdctl_inst;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_rdctl_inst <= 0;
      else if (R_en)
            R_ctrl_rdctl_inst <= R_ctrl_rdctl_inst_nxt;
   end


assign D_ctrl_force_src2_zero = D_op_call|
   D_op_rsv02|
   D_op_nextpc|
   D_op_callr|
   D_op_trap|
   D_op_rsvx44|
   D_op_intr|
   D_op_rsvx60|
   D_op_break|
   D_op_hbreak|
   D_op_eret|
   D_op_bret|
   D_op_rsvx17|
   D_op_rsvx25|
   D_op_ret|
   D_op_jmp|
   D_op_rsvx21|
   D_op_jmpi;

assign R_ctrl_force_src2_zero_nxt = D_ctrl_force_src2_zero;
always @(posedge clk or negedge reset_n)
   begin
      if (reset_n == 0)
            R_ctrl_force_src2_zero <= 0;
      else if (R_en)
            R_ctrl_force_src2_zero <= R_ctrl_force_src2_zero_nxt;
```

```
      end


   assign D_ctrl_alu_force_xor = D_op_cmpgei|
      D_op_cmpgeui|
      D_op_cmpeqi|
      D_op_cmpge|
      D_op_cmpgeu|
      D_op_cmpeq|
      D_op_cmpnei|
      D_op_cmpne|
      D_op_bge|
      D_op_rsv10|
      D_op_bgeu|
      D_op_rsv42|
      D_op_beq|
      D_op_rsv34|
      D_op_bne|
      D_op_rsv62|
      D_op_br|
      D_op_rsv02;

   assign R_ctrl_alu_force_xor_nxt = D_ctrl_alu_force_xor;
   always @(posedge clk or negedge reset_n)
      begin
         if (reset_n == 0)
               R_ctrl_alu_force_xor <= 0;
         else if (R_en)
               R_ctrl_alu_force_xor <= R_ctrl_alu_force_xor_nxt;
      end


   //data_master, which is an e_avalon_master
   //instruction_master, which is an e_avalon_master

//synthesis translate_off
//////////////// SIMULATION-ONLY CONTENTS
   assign F_inst = (F_op_call)? 56'h20202063616c6c :
      (F_op_jmpi)? 56'h2020206a6d7069 :
      (F_op_ldbu)? 56'h2020206c646275 :
      (F_op_addi)? 56'h20202061646469 :
      (F_op_stb)? 56'h20202020737462 :
      (F_op_br)? 56'h20202020206272 :
      (F_op_ldb)? 56'h202020206c6462 :
      (F_op_cmpgei)? 56'h20636d70676569 :
      (F_op_ldhu)? 56'h2020206c646875 :
      (F_op_andi)? 56'h202020616e6469 :
      (F_op_sth)? 56'h20202020737468 :
      (F_op_bge)? 56'h20202020626765 :
      (F_op_ldh)? 56'h202020206c6468 :
      (F_op_cmplti)? 56'h20636d706c7469 :
      (F_op_initda)? 56'h20696e69746461 :
      (F_op_ori)? 56'h202020206f7269 :
      (F_op_stw)? 56'h20202020737477 :
      (F_op_blt)? 56'h20202020626c74 :
      (F_op_ldw)? 56'h202020206c6477 :
      (F_op_cmpnei)? 56'h20636d706e6569 :
      (F_op_flushda)? 56'h666c7573686461 :
```

(F_op_xori)? 56'h202020786f7269 :
(F_op_bne)? 56'h20202020626e65 :
(F_op_cmpeqi)? 56'h20636d70657169 :
(F_op_ldbuio)? 56'h206c646275696f :
(F_op_muli)? 56'h2020206d756c69 :
(F_op_stbio)? 56'h2020737462696f :
(F_op_beq)? 56'h20202020626571 :
(F_op_ldbio)? 56'h20206c6462696f :
(F_op_cmpgeui)? 56'h636d7067657569 :
(F_op_ldhuio)? 56'h206c646875696f :
(F_op_andhi)? 56'h2020616e646869 :
(F_op_sthio)? 56'h2020737468696f :
(F_op_bgeu)? 56'h20202062676575 :
(F_op_ldhio)? 56'h20206c6468696f :
(F_op_cmpltui)? 56'h636d706c747569 :
(F_op_initd)? 56'h2020696e697464 :
(F_op_orhi)? 56'h2020206f726869 :
(F_op_stwio)? 56'h2020737477696f :
(F_op_bltu)? 56'h202020626c7475 :
(F_op_ldwio)? 56'h20206c6477696f :
(F_op_flushd)? 56'h20666c75736864 :
(F_op_xorhi)? 56'h2020786f726869 :
(F_op_eret)? 56'h20202065726574 :
(F_op_roli)? 56'h202020726f6c69 :
(F_op_rol)? 56'h20202020726f6c :
(F_op_flushp)? 56'h20666c75736870 :
(F_op_ret)? 56'h20202020726574 :
(F_op_nor)? 56'h202020206e6f72 :
(F_op_mulxuu)? 56'h206d756c787575 :
(F_op_cmpge)? 56'h2020636d706765 :
(F_op_bret)? 56'h20202062726574 :
(F_op_ror)? 56'h20202020726f72 :
(F_op_flushi)? 56'h20666c75736869 :
(F_op_jmp)? 56'h202020206a6d70 :
(F_op_and)? 56'h20202020616e64 :
(F_op_cmplt)? 56'h2020636d706c74 :
(F_op_slli)? 56'h202020736c6c69 :
(F_op_sll)? 56'h20202020736c6c :
(F_op_or)? 56'h20202020206f72 :
(F_op_mulxsu)? 56'h206d756c787375 :
(F_op_cmpne)? 56'h2020636d706e65 :
(F_op_srli)? 56'h20202073726c69 :
(F_op_srl)? 56'h2020202073726c :
(F_op_nextpc)? 56'h206e6578747063 :
(F_op_callr)? 56'h202063616c6c72 :
(F_op_xor)? 56'h20202020786f72 :
(F_op_mulxss)? 56'h206d756c787373 :
(F_op_cmpeq)? 56'h2020636d706571 :
(F_op_divu)? 56'h20202064697675 :
(F_op_div)? 56'h20202020646976 :
(F_op_rdctl)? 56'h2020726463746c :
(F_op_mul)? 56'h202020206d756c :
(F_op_cmpgeu)? 56'h20636d70676575 :
(F_op_initi)? 56'h2020696e697469 :
(F_op_trap)? 56'h20202074726170 :
(F_op_wrctl)? 56'h2020777263746c :
(F_op_cmpltu)? 56'h20636d706c7475 :
(F_op_add)? 56'h20202020616464 :

```
    (F_op_break)? 56'h2020627265616b :
    (F_op_hbreak)? 56'h2068627265616b :
    (F_op_sync)? 56'h20202073796e63 :
    (F_op_sub)? 56'h20202020737562 :
    (F_op_srai)? 56'h20202073726169 :
    (F_op_sra)? 56'h20202020737261 :
    (F_op_intr)? 56'h202020696e7472 :
    56'h20202020424144;

assign D_inst = (D_op_call)? 56'h20202063616c6c :
    (D_op_jmpi)? 56'h2020206a6d7069 :
    (D_op_ldbu)? 56'h2020206c646275 :
    (D_op_addi)? 56'h20202061646469 :
    (D_op_stb)? 56'h20202020737462 :
    (D_op_br)? 56'h20202020206272 :
    (D_op_ldb)? 56'h202020206c6462 :
    (D_op_cmpgei)? 56'h20636d70676569 :
    (D_op_ldhu)? 56'h2020206c646875 :
    (D_op_andi)? 56'h202020616e6469 :
    (D_op_sth)? 56'h20202020737468 :
    (D_op_bge)? 56'h20202020626765 :
    (D_op_ldh)? 56'h202020206c6468 :
    (D_op_cmplti)? 56'h20636d706c7469 :
    (D_op_initda)? 56'h20696e69746461 :
    (D_op_ori)? 56'h202020206f7269 :
    (D_op_stw)? 56'h20202020737477 :
    (D_op_blt)? 56'h20202020626c74 :
    (D_op_ldw)? 56'h202020206c6477 :
    (D_op_cmpnei)? 56'h20636d706e6569 :
    (D_op_flushda)? 56'h666c7573686461 :
    (D_op_xori)? 56'h202020786f7269 :
    (D_op_bne)? 56'h20202020626e65 :
    (D_op_cmpeqi)? 56'h20636d70657169 :
    (D_op_ldbuio)? 56'h206c646275696f :
    (D_op_muli)? 56'h2020206d756c69 :
    (D_op_stbio)? 56'h20207374626f :
    (D_op_beq)? 56'h20202020626571 :
    (D_op_ldbio)? 56'h20206c6462696f :
    (D_op_cmpgeui)? 56'h636d7067657569 :
    (D_op_ldhuio)? 56'h206c646875696f :
    (D_op_andhi)? 56'h2020616e646869 :
    (D_op_sthio)? 56'h20207374686f :
    (D_op_bgeu)? 56'h20202062676575 :
    (D_op_ldhio)? 56'h20206c6468696f :
    (D_op_cmpltui)? 56'h636d706c747569 :
    (D_op_initd)? 56'h2020696e697464 :
    (D_op_orhi)? 56'h2020206f726869 :
    (D_op_stwio)? 56'h20207374776f :
    (D_op_bltu)? 56'h202020626c7475 :
    (D_op_ldwio)? 56'h20206c6477696f :
    (D_op_flushd)? 56'h20666c75736864 :
    (D_op_xorhi)? 56'h2020786f726869 :
    (D_op_eret)? 56'h20202065726574 :
    (D_op_roli)? 56'h202020726f6c69 :
    (D_op_rol)? 56'h20202020726f6c :
    (D_op_flushp)? 56'h20666c75736870 :
    (D_op_ret)? 56'h20202020726574 :
    (D_op_nor)? 56'h202020206e6f72 :
```

```
      (D_op_mulxuu)? 56'h206d756c787575 :
      (D_op_cmpge)? 56'h2020636d706765 :
      (D_op_bret)? 56'h20202062726574 :
      (D_op_ror)? 56'h20202020726f72 :
      (D_op_flushi)? 56'h20666c75736869 :
      (D_op_jmp)? 56'h202020206a6d70 :
      (D_op_and)? 56'h20202020616e64 :
      (D_op_cmplt)? 56'h2020636d706c74 :
      (D_op_slli)? 56'h202020736c6c69 :
      (D_op_sll)? 56'h20202020736c6c :
      (D_op_or)? 56'h20202020206f72 :
      (D_op_mulxsu)? 56'h206d756c787375 :
      (D_op_cmpne)? 56'h2020636d706e65 :
      (D_op_srli)? 56'h20202073726c69 :
      (D_op_srl)? 56'h2020202073726c :
      (D_op_nextpc)? 56'h206e6578747063 :
      (D_op_callr)? 56'h202063616c6c72 :
      (D_op_xor)? 56'h20202020786f72 :
      (D_op_mulxss)? 56'h206d756c787373 :
      (D_op_cmpeq)? 56'h2020636d706571 :
      (D_op_divu)? 56'h20202064697675 :
      (D_op_div)? 56'h20202020646976 :
      (D_op_rdctl)? 56'h2020726463746c :
      (D_op_mul)? 56'h202020206d756c :
      (D_op_cmpgeu)? 56'h20636d70676575 :
      (D_op_initi)? 56'h2020696e697469 :
      (D_op_trap)? 56'h20202074726170 :
      (D_op_wrctl)? 56'h2020777263746c :
      (D_op_cmpltu)? 56'h20636d706c7475 :
      (D_op_add)? 56'h20202020616464 :
      (D_op_break)? 56'h2020627265616b :
      (D_op_hbreak)? 56'h2068627265616b :
      (D_op_sync)? 56'h20202073796e63 :
      (D_op_sub)? 56'h20202020737562 :
      (D_op_srai)? 56'h20202073726169 :
      (D_op_sra)? 56'h20202020737261 :
      (D_op_intr)? 56'h202020696e7472 :
      56'h20202020424144;

   assign F_vinst = F_valid ? F_inst : {7{8'h2d}};
   assign D_vinst = D_valid ? D_inst : {7{8'h2d}};
   assign R_vinst = R_valid ? D_inst : {7{8'h2d}};
   assign E_vinst = E_valid ? D_inst : {7{8'h2d}};
   assign W_vinst = W_valid ? D_inst : {7{8'h2d}};

//////////////// END SIMULATION-ONLY CONTENTS

//synthesis translate_on

endmodule
```

De2_i2c_av_config.v

```
/*
  * I2C bus control for initializing the audio and video chips on the DE2 board
```

```
 *
 * Adapted by Stephen A. Edwards, Columbia University, sedwards@cs.columbia.edu
 */

module de2_i2c_av_config( iCLK, iRST_N, I2C_SCLK, I2C_SDAT );

// From host

input     iCLK;
input     iRST_N;

// I2C bus

output    I2C_SCLK;
inout     I2C_SDAT;

// Internal Registers/Wires
reg       [15:0]   mI2C_CLK_DIV;
reg       [23:0]   mI2C_DATA;
reg                mI2C_CTRL_CLK;
reg                mI2C_GO;
wire               mI2C_END;
wire               mI2C_ACK;
reg       [15:0]   LUT_DATA;
reg       [5:0]    LUT_INDEX;
reg       [3:0]    mSetup_ST;

// Clock frequencies
parameter CLK_Freq          = 50000000; // 50 MHz
parameter I2C_Freq          = 20000;       // 20 kHz

parameter LUT_SIZE          = 50;

// Audio Data Index
parameter SET_LIN_L         = 0;
parameter SET_LIN_R         = 1;
parameter SET_HEAD_L        = 2;
parameter SET_HEAD_R        = 3;
parameter A_PATH_CTRL       = 4;
parameter D_PATH_CTRL       = 5;
parameter POWER_ON          = 6;
parameter SET_FORMAT        = 7;
parameter SAMPLE_CTRL       = 8;
parameter SET_ACTIVE        = 9;
// Video Data Index
parameter SET_VIDEO         = 10;

// I2C Control Clock
always @(posedge iCLK or negedge iRST_N)
begin
   if (!iRST_N)
      begin
         mI2C_CTRL_CLK <= 0;
         mI2C_CLK_DIV    <= 0;
      end
   else
      begin
        if ( mI2C_CLK_DIV < (CLK_Freq/I2C_Freq) )
```

```
            mI2C_CLK_DIV <= mI2C_CLK_DIV+1;
        else
           begin
              mI2C_CLK_DIV    <= 0;
              mI2C_CTRL_CLK <= ~mI2C_CTRL_CLK;
           end
      end
end


de2_i2c_controller u0 (
   .CLOCK(mI2C_CTRL_CLK), //    Controller Work Clock
   .I2C_SCLK(I2C_SCLK),      //    I2C CLOCK
   .I2C_SDAT(I2C_SDAT),      //    I2C DATA
   .I2C_DATA(mI2C_DATA),    //    DATA:[SLAVE_ADDR,SUB_ADDR,DATA]
   .GO(mI2C_GO),                 //    GO transfor
   .END(mI2C_END),              //    END transfor
   .ACK(mI2C_ACK),              //    ACK
   .RESET(iRST_N)
);

// Configuration control
always @(posedge mI2C_CTRL_CLK or negedge iRST_N)
begin
   if(!iRST_N)
   begin
      LUT_INDEX <= 0;
      mSetup_ST <= 0;
      mI2C_GO      <= 0;
   end
   else
   begin
      if (LUT_INDEX<LUT_SIZE)
      begin
         case(mSetup_ST)
         0:    begin
                  if(LUT_INDEX<SET_VIDEO)
                     mI2C_DATA <= {8'h34,LUT_DATA};
                  else
                     mI2C_DATA <= {8'h40,LUT_DATA};
                  mI2C_GO      <= 1;
                  mSetup_ST <= 1;
               end
         1:    begin
                  if (mI2C_END)
                     begin
                        if (!mI2C_ACK)
                           mSetup_ST <= 2;
                        else
                           mSetup_ST <= 0;
                     mI2C_GO <= 0;
                  end
               end
         2:    begin
                  LUT_INDEX <= LUT_INDEX+1;
                  mSetup_ST <= 0;
               end
         endcase
```

```
      end
   end
end

// Configuration data LUT
always
begin
   case (LUT_INDEX)
   // Audio Config Data
   SET_LIN_L       : LUT_DATA <= 16'h001A;
   SET_LIN_R       : LUT_DATA <= 16'h021A;
   SET_HEAD_L      : LUT_DATA <= 16'h047B;
   SET_HEAD_R      : LUT_DATA <= 16'h067B;
   A_PATH_CTRL     : LUT_DATA <= 16'h08F8;
   D_PATH_CTRL     : LUT_DATA <= 16'h0A06;
   POWER_ON        : LUT_DATA <= 16'h0C00;
   SET_FORMAT      : LUT_DATA <= 16'h0E01;
   SAMPLE_CTRL     : LUT_DATA <= 16'h1002;
   SET_ACTIVE      : LUT_DATA <= 16'h1201;
   // Video Config Data
   SET_VIDEO+0     : LUT_DATA <= 16'h1500;
   SET_VIDEO+1     : LUT_DATA <= 16'h1741;
   SET_VIDEO+2     : LUT_DATA <= 16'h3a16;
   SET_VIDEO+3     : LUT_DATA <= 16'h5004;
   SET_VIDEO+4     : LUT_DATA <= 16'hc305;
   SET_VIDEO+5     : LUT_DATA <= 16'hc480;
   SET_VIDEO+6     : LUT_DATA <= 16'h0e80;
   SET_VIDEO+7     : LUT_DATA <= 16'h5020;
   SET_VIDEO+8     : LUT_DATA <= 16'h5218;
   SET_VIDEO+9     : LUT_DATA <= 16'h58ed;
   SET_VIDEO+10 : LUT_DATA <= 16'h77c5;
   SET_VIDEO+11 : LUT_DATA <= 16'h7c93;
   SET_VIDEO+12 : LUT_DATA <= 16'h7d00;
   SET_VIDEO+13 : LUT_DATA <= 16'hd048;
   SET_VIDEO+14 : LUT_DATA <= 16'hd5a0;
   SET_VIDEO+15 : LUT_DATA <= 16'hd7ea;
   SET_VIDEO+16 : LUT_DATA <= 16'he43e;
   SET_VIDEO+17 : LUT_DATA <= 16'hea0f;
   SET_VIDEO+18 : LUT_DATA <= 16'h3112;
   SET_VIDEO+19 : LUT_DATA <= 16'h3281;
   SET_VIDEO+20 : LUT_DATA <= 16'h3384;
   SET_VIDEO+21 : LUT_DATA <= 16'h37A0;
   SET_VIDEO+22 : LUT_DATA <= 16'he580;
   SET_VIDEO+23 : LUT_DATA <= 16'he603;
   SET_VIDEO+24 : LUT_DATA <= 16'he785;
   SET_VIDEO+25 : LUT_DATA <= 16'h5000;
   SET_VIDEO+26 : LUT_DATA <= 16'h5100;
   SET_VIDEO+27 : LUT_DATA <= 16'h0050;
   SET_VIDEO+28 : LUT_DATA <= 16'h1000;
   SET_VIDEO+29 : LUT_DATA <= 16'h0402;
   SET_VIDEO+30 : LUT_DATA <= 16'h0b00;
   SET_VIDEO+31 : LUT_DATA <= 16'h0a20;
   SET_VIDEO+32 : LUT_DATA <= 16'h1100;
   SET_VIDEO+33 : LUT_DATA <= 16'h2b00;
   SET_VIDEO+34 : LUT_DATA <= 16'h2c8c;
   SET_VIDEO+35 : LUT_DATA <= 16'h2df2;
   SET_VIDEO+36 : LUT_DATA <= 16'h2eee;
   SET_VIDEO+37 : LUT_DATA <= 16'h2ff4;
```

```
   SET_VIDEO+38 : LUT_DATA <= 16'h30d2;
   SET_VIDEO+39 : LUT_DATA <= 16'h0e05;
   default      : LUT_DATA <= 16'hxxxx;
   endcase
end

endmodule
```

De2_i2c_controller.v

```
// --------------------------------------------------------------------
// Copyright (c) 2005 by Terasic Technologies Inc.
// --------------------------------------------------------------------
//
// Permission:
//
//     Terasic grants permission to use and modify this code for use
//     in synthesis for all Terasic Development Boards and Altrea Development
//     Kits made by Terasic.  Other use of this code, including the selling,
//     duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
//     This VHDL or Verilog source code is intended as a design reference
//     which illustrates how these types of functions can be implemented.
//     It is the user's responsibility to verify their design for
//     consistency and functionality through the use of formal
//     verification methods.  Terasic provides no warranty regarding the use
//     or functionality of this code.
//
// --------------------------------------------------------------------
//
//                     Terasic Technologies Inc
//                     356 Fu-Shin E. Rd Sec. 1. JhuBei City,
//                     HsinChu County, Taiwan
//                     302
//
//                     web: http://www.terasic.com/
//                     email: support@terasic.com
//
// --------------------------------------------------------------------
//
// Major Functions:i2c controller
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
//     Ver  :| Author            :| Mod. Date :| Changes Made:
//     V1.0 :| Joe Yang          :| 05/07/10  :|      Initial Revision
// --------------------------------------------------------------------
module de2_i2c_controller (
   CLOCK,
   I2C_SCLK, // I2C CLOCK
   I2C_SDAT, // I2C DATA
   I2C_DATA, // DATA:[SLAVE_ADDR,SUB_ADDR,DATA]
```

```
   GO,            // GO transfor
   END,           // END transfor
   W_R,           // W_R
   ACK,           // ACK
   RESET,
   // TEST
   SD_COUNTER,
   SDO
);

input    CLOCK;
input    [23:0] I2C_DATA;
input    GO;
input    RESET;
input    W_R;
inout    I2C_SDAT;
output I2C_SCLK;
output END;
output ACK;

// TEST
output [5:0] SD_COUNTER;
output SDO;

reg SDO;
reg SCLK;
reg END;
reg [23:0] SD;
reg [5:0] SD_COUNTER;

wire I2C_SCLK = SCLK | (((SD_COUNTER >= 4) & (SD_COUNTER <= 30))? ~CLOCK : 0);
wire I2C_SDAT = SDO ? 1'bz : 0;

reg ACK1, ACK2, ACK3;
wire ACK = ACK1 | ACK2 | ACK3;

//--I2C COUNTER

always @(negedge RESET or posedge CLOCK)
begin
   if (!RESET)
      SD_COUNTER = 6'b111111;
   else
      begin
         if (GO == 0)
            SD_COUNTER = 0;
         else
            if (SD_COUNTER < 6'b111111)
               SD_COUNTER = SD_COUNTER + 1;
      end
end

always @(negedge RESET or posedge CLOCK )
begin
   if (!RESET)
      begin
         SCLK = 1;
         SDO = 1;
```

```
        ACK1 = 0;
        ACK2 = 0;
        ACK3 = 0;
        END = 1;
      end
else
  case (SD_COUNTER)
  6'd0  : begin ACK1 = 0; ACK2 = 0; ACK3 = 0; END = 0; SDO = 1; SCLK = 1; end

  // Start
  6'd1  : begin SD = I2C_DATA; SDO = 0; end
  6'd2  : SCLK = 0;

  // Slave Address
  6'd3  : SDO = SD[23];
  6'd4  : SDO = SD[22];
  6'd5  : SDO = SD[21];
  6'd6  : SDO = SD[20];
  6'd7  : SDO = SD[19];
  6'd8  : SDO = SD[18];
  6'd9  : SDO = SD[17];
  6'd10 : SDO = SD[16];
  6'd11 : SDO = 1'b1;     //ACK

  // Sub-address
  6'd12  : begin SDO = SD[15]; ACK1 = I2C_SDAT; end
  6'd13  : SDO = SD[14];
  6'd14  : SDO = SD[13];
  6'd15  : SDO = SD[12];
  6'd16  : SDO = SD[11];
  6'd17  : SDO = SD[10];
  6'd18  : SDO = SD[9];
  6'd19  : SDO = SD[8];
  6'd20  : SDO = 1'b1; // ACK

  // Data
  6'd21  : begin SDO = SD[7]; ACK2 = I2C_SDAT; end
  6'd22  : SDO = SD[6];
  6'd23  : SDO = SD[5];
  6'd24  : SDO = SD[4];
  6'd25  : SDO = SD[3];
  6'd26  : SDO = SD[2];
  6'd27  : SDO = SD[1];
  6'd28  : SDO = SD[0];
  6'd29  : SDO = 1'b1;   // ACK

  // Stop
  6'd30 : begin SDO = 1'b0; SCLK = 1'b0; ACK3 = I2C_SDAT; end
  6'd31 : SCLK = 1'b1;
  6'd32 : begin SDO = 1'b1; END = 1; end
  endcase
end

endmodule
```

De2_LTM_Test.v

```
// --------------------------------------------------------------------
// Copyright (c) 2005 by Terasic Technologies Inc.
// --------------------------------------------------------------------
//
// Permission:
//
//     Terasic grants permission to use and modify this code for use
//     in synthesis for all Terasic Development Boards and Altera Development
//     Kits made by Terasic.    Other use of this code, including the selling
//     ,duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
//     This VHDL/Verilog or C/C++ source code is intended as a design reference
//     which illustrates how these types of functions can be implemented.
//     It is the user's responsibility to verify their design for
//     consistency and functionality through the use of formal
//     verification methods.    Terasic provides no warranty regarding the use
//     or functionality of this code.
//
// --------------------------------------------------------------------
//
//                              Terasic Technologies Inc
//                              356 Fu-Shin E. Rd Sec. 1. JhuBei City,
//                              HsinChu County, Taiwan
//                              302
//
//                              web: http://www.terasic.com/
//                              email: support@terasic.com
//
// --------------------------------------------------------------------
//
// Major Functions: DE2 LTM    Test
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
//     Ver    :| Author                        :| Mod. Date :| Changes Made:
//     V1.0 :| Joe Yang ,Johnny Fan      :| 07/04/12    :| Initial Revision
//     V1.1 :| ben                             :| 11/03/14    :| Quartus 10.1 sp1
// --------------------------------------------------------------------

module DE2_LTM_Test
    (
        ////////////////////// Clock Input          //////////////////////
        CLOCK_27,                                  //    27 MHz
        CLOCK_50,                                  //    50 MHz
        EXT_CLOCK,                                 //    External Clock
        ////////////////////// Push Button          //////////////////////
        KEY,                                       //    Pushbutton[3:0]
        ////////////////////// DPDT Switch          //////////////////////
        SW,                                        //    Toggle Switch[17:0]
        ////////////////////// 7-SEG Dispaly  //////////////////////
        HEX0,                                            //    Seven Segment Digit 0
        HEX1,                                            //    Seven Segment Digit 1
```

```
HEX2,                                   //  Seven Segment Digit 2
HEX3,                                   //  Seven Segment Digit 3
HEX4,                                   //  Seven Segment Digit 4
HEX5,                                   //  Seven Segment Digit 5
HEX6,                                   //  Seven Segment Digit 6
HEX7,                                   //  Seven Segment Digit 7
//////////////////////  LED     //////////////////////
LEDG,                                   //  LED Green[8:0]
LEDR,                                   //  LED Red[17:0]
//////////////////////  UART    //////////////////////
UART_TXD,                       //  UART Transmitter
UART_RXD,                       //  UART Receiver
//////////////////////  IRDA//////////////////////
IRDA_TXD,                       //  IRDA Transmitter
IRDA_RXD,                       //  IRDA Receiver
////////////////////    SDRAM Interface     ////////////////
DRAM_DQ,                                //  SDRAM Data bus 16 Bits
DRAM_ADDR,                      //  SDRAM Address bus 12 Bits
DRAM_LDQM,                      //  SDRAM Low-byte Data Mask
DRAM_UDQM,                              //  SDRAM High-byte Data Mask
DRAM_WE_N,                      //  SDRAM Write Enable
DRAM_CAS_N,                             //  SDRAM Column Address Strobe
DRAM_RAS_N,                             //  SDRAM Row Address Strobe
DRAM_CS_N,                      //  SDRAM Chip Select
DRAM_BA_0,                      //  SDRAM Bank Address 0
DRAM_BA_1,                      //  SDRAM Bank Address 0
DRAM_CLK,                       //  SDRAM Clock
DRAM_CKE,                       //  SDRAM Clock Enable
////////////////////// Flash Interface  ////////////////
FL_DQ,                          //  FLASH Data bus 8 Bits
FL_ADDR,                        //  FLASH Address bus 22 Bits
FL_WE_N,                                //  FLASH Write Enable
FL_RST_N,                       //  FLASH Reset
FL_OE_N,                        //  FLASH Output Enable
FL_CE_N,                        //  FLASH Chip Enable
////////////////////// SRAM Interface       ////////////////
SRAM_DQ,                                //  SRAM Data bus 16 Bits
SRAM_ADDR,                      //  SRAM Address bus 18 Bits
SRAM_UB_N,                      //  SRAM High-byte Data Mask
SRAM_LB_N,                      //  SRAM Low-byte Data Mask
SRAM_WE_N,                      //  SRAM Write Enable
SRAM_CE_N,                      //  SRAM Chip Enable
SRAM_OE_N,                      //  SRAM Output Enable
////////////////////// ISP1362 Interface    ////////////////
OTG_DATA,                       //  ISP1362 Data bus 16 Bits
OTG_ADDR,                       //  ISP1362 Address 2 Bits
OTG_CS_N,                       //  ISP1362 Chip Select
OTG_RD_N,                       //  ISP1362 Write
OTG_WR_N,                       //  ISP1362 Read
OTG_RST_N,                      //  ISP1362 Reset
OTG_FSPEED,                     //  USB Full Speed,     0 = Enable, Z = Disable
OTG_LSPEED,                     //  USB Low Speed,      0 = Enable, Z = Disable
OTG_INT0,                       //  ISP1362 Interrupt 0
OTG_INT1,                       //  ISP1362 Interrupt 1
OTG_DREQ0,                      //  ISP1362 DMA Request 0
OTG_DREQ1,                      //  ISP1362 DMA Request 1
OTG_DACK0_N,                    //  ISP1362 DMA Acknowledge 0
OTG_DACK1_N,                    //  ISP1362 DMA Acknowledge 1
```

```
        ////////////////////  LCD Module 16X2          ////////////////
        LCD_ON,                                 //    LCD Power ON/OFF
        LCD_BLON,                               //    LCD Back Light ON/OFF
        LCD_RW,                                 //    LCD Read/Write Select, 0 = Write, 1 = Read
        LCD_EN,                                 //    LCD Enable
        LCD_RS,                                 //    LCD Command/Data Select, 0 = Command,
1 = Data
        LCD_DATA,                               //    LCD Data bus 8 bits
        ////////////////////  SD_Card Interface  ////////////////
        SD_DAT,                                 //    SD Card Data
        SD_DAT3,                                      //    SD Card Data 3
        SD_CMD,                                 //    SD Card Command Signal
        SD_CLK,                                 //    SD Card Clock
        ////////////////////  USB JTAG link  ////////////////////
        TDI,                                    // CPLD -> FPGA (data in)
        TCK,                                    // CPLD -> FPGA (clk)
        TCS,                                    // CPLD -> FPGA (CS)
        TDO,                                    // FPGA -> CPLD (data out)
        ////////////////////  I2C         ////////////////////////////
        I2C_SDAT,                               //    I2C Data
        I2C_SCLK,                         //    I2C Clock
        ////////////////////  PS2         ////////////////////////////
        PS2_DAT,                                //    PS2 Data
        PS2_CLK,                                //    PS2 Clock
        ////////////////////  VGA         ////////////////////////////
        VGA_CLK,                                //    VGA Clock
        VGA_HS,                                 //    VGA H_SYNC
        VGA_VS,                                 //    VGA V_SYNC
        VGA_BLANK,                              //    VGA BLANK
        VGA_SYNC,                               //    VGA SYNC
        VGA_R,                                  //    VGA Red[9:0]
        VGA_G,                                  //    VGA Green[9:0]
        VGA_B,                                        //    VGA Blue[9:0]
        ////////////     Ethernet Interface  ////////////////////////
        ENET_DATA,                              //    DM9000A DATA bus 16Bits
        ENET_CMD,                               //    DM9000A    Command/Data    Select,    0    =
Command, 1 = Data
        ENET_CS_N,                              //    DM9000A Chip Select
        ENET_WR_N,                              //    DM9000A Write
        ENET_RD_N,                              //    DM9000A Read
        ENET_RST_N,                             //    DM9000A Reset
        ENET_INT,                               //    DM9000A Interrupt
        ENET_CLK,                               //    DM9000A Clock 25 MHz
        ////////////////      Audio CODEC       ////////////////////////
        AUD_ADCLRCK,                            //    Audio CODEC ADC LR Clock
        AUD_ADCDAT,                             //    Audio CODEC ADC Data
        AUD_DACLRCK,                            //    Audio CODEC DAC LR Clock
        AUD_DACDAT,                             //    Audio CODEC DAC Data
        AUD_BCLK,                               //    Audio CODEC Bit-Stream Clock
        AUD_XCK,                                      //    Audio CODEC Chip Clock
        ////////////////      TV Decoder        ////////////////////////
        TD_DATA,                                //    TV Decoder Data bus 8 bits
        TD_HS,                                  //    TV Decoder H_SYNC
        TD_VS,                                  //    TV Decoder V_SYNC
        TD_RESET,                               //    TV Decoder Reset
        ////////////////////  GPIO        ////////////////////////////
        GPIO_0,                                 //    GPIO Connection 0
        GPIO_1,                                 //    GPIO Connection 1
```

```
    );
//=============================================================================
// PORT declarations
//=============================================================================

///////////////////////  Clock Input          ///////////////////////
input               CLOCK_27;                   //    27 MHz
input               CLOCK_50;                   //    50 MHz
input               EXT_CLOCK;                  //    External Clock
///////////////////////  Push Button          ///////////////////////
input     [3:0]  KEY;                           //    Pushbutton[3:0]
///////////////////////  DPDT Switch          ///////////////////////
input     [17:0]     SW;                        //    Toggle Switch[17:0]
///////////////////////  7-SEG Dispaly  ///////////////////////
output    [6:0]  HEX0;                          //    Seven Segment Digit 0
output    [6:0]  HEX1;                          //    Seven Segment Digit 1
output    [6:0]  HEX2;                          //    Seven Segment Digit 2
output    [6:0]  HEX3;                          //    Seven Segment Digit 3
output    [6:0]  HEX4;                          //    Seven Segment Digit 4
output    [6:0]  HEX5;                          //    Seven Segment Digit 5
output    [6:0]  HEX6;                          //    Seven Segment Digit 6
output    [6:0]  HEX7;                          //    Seven Segment Digit 7
////////////////////////////    LED        ////////////////////////////
output    [8:0]  LEDG;                          //    LED Green[8:0]
output    [17:0]LEDR;                           //    LED Red[17:0]
////////////////////////////    UART       ////////////////////////////
output          UART_TXD;                       //    UART Transmitter
input               UART_RXD;                   //    UART Receiver
////////////////////////////    IRDA////////////////////////////
output          IRDA_TXD;                       //    IRDA Transmitter
input               IRDA_RXD;                   //    IRDA Receiver
///////////////////////         SDRAM Interface    ///////////////////////
inout     [15:0]     DRAM_DQ;                        //    SDRAM Data bus 16 Bits
output[11:0]   DRAM_ADDR;                  //    SDRAM Address bus 12 Bits
output              DRAM_LDQM;                  //    SDRAM Low-byte Data Mask
output              DRAM_UDQM;                      //    SDRAM High-byte Data Mask
output              DRAM_WE_N;                  //    SDRAM Write Enable
output              DRAM_CAS_N;                     //    SDRAM Column Address Strobe
output              DRAM_RAS_N;                     //    SDRAM Row Address Strobe
output              DRAM_CS_N;                  //    SDRAM Chip Select
output              DRAM_BA_0;                  //    SDRAM Bank Address 0
output              DRAM_BA_1;                  //    SDRAM Bank Address 0
output              DRAM_CLK;                   //    SDRAM Clock
output              DRAM_CKE;                   //    SDRAM Clock Enable
///////////////////////  Flash Interface////////////////////////
inout           [7:0]  FL_DQ;                  //    FLASH Data bus 8 Bits
output    [21:0]FL_ADDR;                       //    FLASH Address bus 22 Bits
output              FL_WE_N;                        //    FLASH Write Enable
output              FL_RST_N;                   //    FLASH Reset
output              FL_OE_N;                    //    FLASH Output Enable
output              FL_CE_N;                    //    FLASH Chip Enable
///////////////////////  SRAM Interface     ///////////////////////
inout     [15:0]     SRAM_DQ;                        //    SRAM Data bus 16 Bits
output[17:0]   SRAM_ADDR;                  //    SRAM Address bus 18 Bits
output              SRAM_UB_N;                  //    SRAM High-byte Data Mask
output              SRAM_LB_N;                  //    SRAM Low-byte Data Mask
output              SRAM_WE_N;                  //    SRAM Write Enable
output              SRAM_CE_N;                  //    SRAM Chip Enable
```

```
output              SRAM_OE_N;              //    SRAM Output Enable
//////////////////// ISP1362 Interface   ////////////////////////
inout     [15:0]    OTG_DATA;               //    ISP1362 Data bus 16 Bits
output[1:0]         OTG_ADDR;               //    ISP1362 Address 2 Bits
output              OTG_CS_N;               //    ISP1362 Chip Select
output              OTG_RD_N;               //    ISP1362 Write
output              OTG_WR_N;               //    ISP1362 Read
output              OTG_RST_N;              //    ISP1362 Reset
output              OTG_FSPEED;             //    USB Full Speed,     0 = Enable, Z = Disable
output              OTG_LSPEED;             //    USB Low Speed,      0 = Enable, Z = Disable
input                   OTG_INT0;               //    ISP1362 Interrupt 0
input                   OTG_INT1;               //    ISP1362 Interrupt 1
input                   OTG_DREQ0;              //    ISP1362 DMA Request 0
input                   OTG_DREQ1;              //    ISP1362 DMA Request 1
output              OTG_DACK0_N;            //    ISP1362 DMA Acknowledge 0
output              OTG_DACK1_N;            //    ISP1362 DMA Acknowledge 1
//////////////////// LCD Module 16X2   ////////////////////////////
inout           [7:0]LCD_DATA;              //    LCD Data bus 8 bits
output              LCD_ON;                 //    LCD Power ON/OFF
output              LCD_BLON;               //    LCD Back Light ON/OFF
output              LCD_RW;                 //    LCD Read/Write Select, 0 = Write, 1 = Read
output              LCD_EN;                 //    LCD Enable
output              LCD_RS;                 //    LCD Command/Data Select, 0 = Command,
1 = Data
//////////////////// SD Card Interface   ////////////////////////
inout                   SD_DAT;                 //    SD Card Data
inout                   SD_DAT3;                    //    SD Card Data 3
inout                   SD_CMD;                 //    SD Card Command Signal
output              SD_CLK;                 //    SD Card Clock
////////////////////////  I2C       ////////////////////////////////
inout                   I2C_SDAT;                   //    I2C Data
output              I2C_SCLK;               //    I2C Clock
////////////////////////  PS2       ////////////////////////////////
input                   PS2_DAT;                //    PS2 Data
input                   PS2_CLK;                //    PS2 Clock
//////////////////// USB JTAG link  ////////////////////////////
input               TDI;                    // CPLD -> FPGA (data in)
input               TCK;                    // CPLD -> FPGA (clk)
input               TCS;                    // CPLD -> FPGA (CS)
output              TDO;                        // FPGA -> CPLD (data out)
////////////////////////  VGA           ////////////////////////////
output              VGA_CLK;                //    VGA Clock
output              VGA_HS;                 //    VGA H_SYNC
output              VGA_VS;                 //    VGA V_SYNC
output              VGA_BLANK;              //    VGA BLANK
output              VGA_SYNC;               //    VGA SYNC
output    [9:0]VGA_R;                   //    VGA Red[9:0]
output    [9:0]VGA_G;                   //    VGA Green[9:0]
output    [9:0]VGA_B;                   //    VGA Blue[9:0]
////////////////      Ethernet Interface  ////////////////////////////
inout     [15:0]    ENET_DATA;              //    DM9000A DATA bus 16Bits
output              ENET_CMD;               //    DM9000A    Command/Data    Select,    0    =
Command, 1 = Data
output              ENET_CS_N;              //    DM9000A Chip Select
output              ENET_WR_N;              //    DM9000A Write
output              ENET_RD_N;              //    DM9000A Read
output              ENET_RST_N;             //    DM9000A Reset
input                   ENET_INT;               //    DM9000A Interrupt
```

```
output              ENET_CLK;                   //    DM9000A Clock 25 MHz
////////////////////// Audio CODEC        ////////////////////////////
inout                    AUD_ADCLRCK;           //    Audio CODEC ADC LR Clock
input                    AUD_ADCDAT;                 //    Audio CODEC ADC Data
inout                    AUD_DACLRCK;           //    Audio CODEC DAC LR Clock
output              AUD_DACDAT;                 //    Audio CODEC DAC Data
inout                    AUD_BCLK;              //    Audio CODEC Bit-Stream Clock
output              AUD_XCK;                    //    Audio CODEC Chip Clock
////////////////////// TV Devoder         ////////////////////////////
input          [7:0] TD_DATA;                   //    TV Decoder Data bus 8 bits
input                    TD_HS;                 //    TV Decoder H_SYNC
input                    TD_VS;                 //    TV Decoder V_SYNC
output              TD_RESET;                   //    TV Decoder Reset
////////////////////////   GPIO      ////////////////////////////////
inout     [35:0]    GPIO_0;                     //    GPIO Connection 0
inout     [35:0]    GPIO_1;                     //    GPIO Connection 1

assign    LCD_ON        =    1'b1;
assign    LCD_BLON      =    1'b1;

//    All inout port turn to tri-state
assign    DRAM_DQ           =    16'hzzzz;
assign    FL_DQ             =    8'hzz;
assign    SRAM_DQ           =    16'hzzzz;
assign    OTG_DATA          =    16'hzzzz;
assign    LCD_DATA          =    8'hzz;
assign    SD_DAT        =    1'bz;
assign    ENET_DATA     =    16'hzzzz;
assign    AUD_ADCLRCK       =    1'bz;
assign    AUD_DACLRCK       =    1'bz;
assign    AUD_BCLK          =    1'bz;
assign    GPIO_1        =    36'hzzzzzzzzz;
//=============================================================================
// REG/WIRE declarations
//=============================================================================
// Touch panel signal //
wire [7:0] ltm_r;          //    LTM Red Data 8 Bits
wire [7:0] ltm_g;          //    LTM Green Data 8 Bits
wire [7:0] ltm_b;          //    LTM Blue Data 8 Bits
wire       ltm_nclk; //    LTM Clcok
wire       ltm_hd;
wire       ltm_vd;
wire       ltm_den;
wire       ltm_grst;
// lcd 3wire interface//
wire       ltm_sclk;
wire       ltm_sda;
wire       ltm_scen;
wire            ltm_3wirebusy_n;
// Touch Screen Digitizer ADC
wire            adc_dclk;
wire            adc_cs;
wire            adc_penirq_n;
wire            adc_busy;
wire            adc_din;
wire            adc_dout;
wire            adc_ltm_sclk;
```

```
wire [11:0]      x_coord;
wire [11:0]      y_coord;
wire             new_coord;
wire [1:0] display_mode;
reg  [31:0]      div;

////////////// GPIO_O /////////////////
assign      adc_penirq_n    =GPIO_0[0];
assign      adc_dout        =GPIO_0[1];
assign      adc_busy        =GPIO_0[2];
assign      GPIO_0[3]       =adc_din;
assign      GPIO_0[4]       =adc_ltm_sclk;
assign      GPIO_0[5]       =ltm_b[3];
assign      GPIO_0[6]       =ltm_b[2];
assign      GPIO_0[7]       =ltm_b[1];
assign      GPIO_0[8]       =ltm_b[0];
assign      GPIO_0[9]       =ltm_nclk;
assign      GPIO_0[10]      =ltm_den;
assign      GPIO_0[11]      =ltm_hd;
assign      GPIO_0[12]      =ltm_vd;
assign      GPIO_0[13]      =ltm_b[4];
assign      GPIO_0[14]      =ltm_b[5];
assign      GPIO_0[15]      =ltm_b[6];
assign      GPIO_0[16]      =ltm_b[7];
assign      GPIO_0[17]      =ltm_g[0];
assign      GPIO_0[18]      =ltm_g[1];
assign      GPIO_0[19]      =ltm_g[2];
assign      GPIO_0[20]      =ltm_g[3];
assign      GPIO_0[21]      =ltm_g[4];
assign      GPIO_0[22]      =ltm_g[5];
assign      GPIO_0[23]      =ltm_g[6];
assign      GPIO_0[24]      =ltm_g[7];
assign      GPIO_0[25]      =ltm_r[0];
assign      GPIO_0[26]      =ltm_r[1];
assign      GPIO_0[27]      =ltm_r[2];
assign      GPIO_0[28]      =ltm_r[3];
assign      GPIO_0[29]      =ltm_r[4];
assign      GPIO_0[30]      =ltm_r[5];
assign      GPIO_0[31]      =ltm_r[6];
assign      GPIO_0[32]      =ltm_r[7];
assign      GPIO_0[33]      =ltm_grst;
assign      GPIO_0[34]      =ltm_scen;
assign      GPIO_0[35]      =ltm_sda;

assign adc_ltm_sclk= ( adc_dclk & ltm_3wirebusy_n )    |    ( ~ltm_3wirebusy_n & ltm_sclk );
assign ltm_nclk = div[0]; // 25 Mhz
assign ltm_grst = KEY[0];
always @(posedge CLOCK_50)
      begin
            div <= div+1;
      end

SEG7_LUT_8          u1    (
                          .oSEG0(HEX0),
                          .oSEG1(HEX1),
                          .oSEG2(HEX2),
                          .oSEG3(HEX3),
                          .oSEG4(HEX4),
```

```
                        .oSEG5(HEX5),
                        .oSEG6(HEX6),
                        .oSEG7(HEX7),
                        .iDIG({ 4'h0 , x_coord , 4'h0 , y_coord }),
                        .ON_OFF(8'b01110111)
                        );

// lcd 3 wire interface configuration    //
lcd_spi_cotroller    u2    (
                        // Host Side
                        .iCLK(CLOCK_50),
                        .iRST_n(DLY0),
                        // 3wire Side
                        .o3WIRE_SCLK(ltm_sclk),
                        .io3WIRE_SDAT(ltm_sda),
                        .o3WIRE_SCEN(ltm_scen),
                        .o3WIRE_BUSY_n(ltm_3wirebusy_n)
                        );

// system reset    //
Reset_Delay          u3    (.iCLK(CLOCK_50),
                        .iRST(KEY[0]),
                        .oRST_0(DLY0),
                        .oRST_1(DLY1),
                        .oRST_2(DLY2)
                        );
// Touch Screen Digitizer ADC configuration //
//adc_spi_controller            u4    (
//                        .iCLK(CLOCK_50),
//                        .iRST_n(DLY0),
//                        .oADC_DIN(adc_din),
//                        .oADC_DCLK(adc_dclk),
//                        .oADC_CS(adc_cs),
//                        .iADC_DOUT(adc_dout),
//                        .iADC_BUSY(adc_busy),
//                        .iADC_PENIRQ_n(adc_penirq_n),
//                        .oTOUCH_IRQ(touch_irq),
//                        .oX_COORD(x_coord),
//                        .oY_COORD(y_coord),
//                        .oNEW_COORD(new_coord),
//                        );
//
//touch_irq_detector    u5    (
//                        .clk(CLOCK_50),
//                        .reset_n(DLY0),
//                        .iTOUCH_IRQ(touch_irq),
//                        .iX_COORD(x_coord),
//                        .iY_COORD(y_coord),
//                        .iNEW_COORD(new_coord),
//                        .oDISPLAY_MODE(display_mode),
//                        );
//
//lcd_timing_controller        u6    (
//                        .clk(ltm_nclk),
//                        .reset_n(DLY2),
//                        // lcd side
//                        .oLCD_R(ltm_r),
//                        .oLCD_G(ltm_g),
```

```
//                          .oLCD_B(ltm_b),
//                          .oHD(ltm_hd),
//                          .oVD(ltm_vd),
//                          .oDEN(ltm_den),
//                          .iDISPLAY_MODE(display_mode),
//                          );

nios_system DUT
      (
         .SRAM_ADDR_from_the_sram                                  (SRAM_ADDR),
         .SRAM_CE_N_from_the_sram                                  (SRAM_CE_N),
         .SRAM_DQ_to_and_from_the_sram                             (SRAM_DQ),
         .SRAM_LB_N_from_the_sram                                  (SRAM_LB_N),
         .SRAM_OE_N_from_the_sram                                  (SRAM_OE_N),
         .SRAM_UB_N_from_the_sram                                  (SRAM_UB_N),
         .SRAM_WE_N_from_the_sram                                  (SRAM_WE_N),
         .clk_0                                                    (CLOCK_50),
         .iADC_BUSY_to_the_adc_spi_controller_0                    (GPIO_0[2]),
         .iADC_DOUT_to_the_adc_spi_controller_0                    (GPIO_0[1]),
         .iADC_PENIRQ_n_to_the_adc_spi_controller_0                (GPIO_0[0]),
         .iDISPLAY_MODE_to_the_lcd_timing_controller_0             (display_mode),
         .oADC_CS_from_the_adc_spi_controller_0                    (adc_cs),
         .oADC_DCLK_from_the_adc_spi_controller_0                  (adc_dclk),
         .oADC_DIN_from_the_adc_spi_controller_0                   (adc_din),
         .oDEN_from_the_lcd_timing_controller_0                    (ltm_den),
         .oHD_from_the_lcd_timing_controller_0                     (ltm_hd),
         .oLCD_B_from_the_lcd_timing_controller_0                  (ltm_b),
         .oLCD_G_from_the_lcd_timing_controller_0                  (ltm_g),
         .oLCD_R_from_the_lcd_timing_controller_0                  (ltm_r),
         .oNEW_COORD_from_the_adc_spi_controller_0                 (new_coord),
         .oTOUCH_IRQ_from_the_adc_spi_controller_0                 (touch_irq),
         .oVD_from_the_lcd_timing_controller_0                     (ltm_vd),
         .oX_COORD_from_the_adc_spi_controller_0                   (x_coord),
         .oY_COORD_from_the_adc_spi_controller_0                   (y_coord),
         .reset_n                                                  (1)
      );

endmodule
```

DE2_TOP_PROJECT.v

```
// --------------------------------------------------------------------
// Copyright (c) 2005 by Terasic Technologies Inc.
// --------------------------------------------------------------------
//
// Permission:
//
//     Terasic grants permission to use and modify this code for use
//     in synthesis for all Terasic Development Boards and Altera Development
//     Kits made by Terasic.     Other use of this code, including the selling
//     ,duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
```

```
//      This VHDL/Verilog or C/C++ source code is intended as a design reference
//      which illustrates how these types of functions can be implemented.
//      It is the user's responsibility to verify their design for
//      consistency and functionality through the use of formal
//      verification methods.    Terasic provides no warranty regarding the use
//      or functionality of this code.
//
// --------------------------------------------------------------------
//
//                              Terasic Technologies Inc
//                              356 Fu-Shin E. Rd Sec. 1. JhuBei City,
//                              HsinChu County, Taiwan
//                              302
//
//                              web: http://www.terasic.com/
//                              email: support@terasic.com
//
// --------------------------------------------------------------------
//
// Major Functions: DE2 LTM    Test
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
//      Ver    :| Author                    :| Mod. Date :| Changes Made:
//      V1.0 :| Joe Yang ,Johnny Fan      :| 07/04/12    :| Initial Revision
//      V1.1 :| ben                         :| 11/03/14    :| Quartus 10.1 sp1
// --------------------------------------------------------------------

module DE2_TOP_PROJECT
      (
            ////////////////////// Clock Input           //////////////////////
            CLOCK_27,                                   //    27 MHz
            CLOCK_50,                                   //    50 MHz
            EXT_CLOCK,                                  //    External Clock
            ////////////////////// Push Button           //////////////////////
            KEY,                                        //    Pushbutton[3:0]
            ////////////////////// DPDT Switch           //////////////////////
            SW,                                         //    Toggle Switch[17:0]
            ////////////////////// 7-SEG Dispaly  //////////////////////
            HEX0,                                             //    Seven Segment Digit 0
            HEX1,                                             //    Seven Segment Digit 1
            HEX2,                                             //    Seven Segment Digit 2
            HEX3,                                             //    Seven Segment Digit 3
            HEX4,                                             //    Seven Segment Digit 4
            HEX5,                                             //    Seven Segment Digit 5
            HEX6,                                             //    Seven Segment Digit 6
            HEX7,                                             //    Seven Segment Digit 7
            ////////////////////////   LED        ////////////////////////
            LEDG,                                             //    LED Green[8:0]
            LEDR,                                             //    LED Red[17:0]
            ////////////////////////   UART       ////////////////////////
            UART_TXD,                                   //    UART Transmitter
            UART_RXD,                                   //    UART Receiver
            ////////////////////////   IRDA////////////////////////
            IRDA_TXD,                                   //    IRDA Transmitter
            IRDA_RXD,                                   //    IRDA Receiver
```

```
        ////////////////////     SDRAM Interface        ////////////////
        DRAM_DQ,                                    //    SDRAM Data bus 16 Bits
        DRAM_ADDR,                          //    SDRAM Address bus 12 Bits
        DRAM_LDQM,                          //    SDRAM Low-byte Data Mask
        DRAM_UDQM,                                  //    SDRAM High-byte Data Mask
        DRAM_WE_N,                          //    SDRAM Write Enable
        DRAM_CAS_N,                                 //    SDRAM Column Address Strobe
        DRAM_RAS_N,                                 //    SDRAM Row Address Strobe
        DRAM_CS_N,                          //    SDRAM Chip Select
        DRAM_BA_0,                          //    SDRAM Bank Address 0
        DRAM_BA_1,                          //    SDRAM Bank Address 0
        DRAM_CLK,                           //    SDRAM Clock
        DRAM_CKE,                           //    SDRAM Clock Enable
        ////////////////////// Flash Interface     ////////////////
        FL_DQ,                              //    FLASH Data bus 8 Bits
        FL_ADDR,                            //    FLASH Address bus 22 Bits
        FL_WE_N,                                    //    FLASH Write Enable
        FL_RST_N,                           //    FLASH Reset
        FL_OE_N,                            //    FLASH Output Enable
        FL_CE_N,                            //    FLASH Chip Enable
        ////////////////////// SRAM Interface          ////////////////
        SRAM_DQ,                                    //    SRAM Data bus 16 Bits
        SRAM_ADDR,                          //    SRAM Address bus 18 Bits
        SRAM_UB_N,                          //    SRAM High-byte Data Mask
        SRAM_LB_N,                          //    SRAM Low-byte Data Mask
        SRAM_WE_N,                          //    SRAM Write Enable
        SRAM_CE_N,                          //    SRAM Chip Enable
        SRAM_OE_N,                          //    SRAM Output Enable
        ////////////////////// ISP1362 Interface   ////////////////
        OTG_DATA,                           //    ISP1362 Data bus 16 Bits
        OTG_ADDR,                           //    ISP1362 Address 2 Bits
        OTG_CS_N,                           //    ISP1362 Chip Select
        OTG_RD_N,                           //    ISP1362 Write
        OTG_WR_N,                           //    ISP1362 Read
        OTG_RST_N,                          //    ISP1362 Reset
        OTG_FSPEED,                         //    USB Full Speed,     0 = Enable, Z = Disable
        OTG_LSPEED,                         //    USB Low Speed,      0 = Enable, Z = Disable
        OTG_INT0,                           //    ISP1362 Interrupt 0
        OTG_INT1,                           //    ISP1362 Interrupt 1
        OTG_DREQ0,                          //    ISP1362 DMA Request 0
        OTG_DREQ1,                          //    ISP1362 DMA Request 1
        OTG_DACK0_N,                        //    ISP1362 DMA Acknowledge 0
        OTG_DACK1_N,                        //    ISP1362 DMA Acknowledge 1
        ////////////////////// LCD Module 16X2     ////////////////
        LCD_ON,                             //    LCD Power ON/OFF
        LCD_BLON,                           //    LCD Back Light ON/OFF
        LCD_RW,                             //    LCD Read/Write Select, 0 = Write, 1 = Read
        LCD_EN,                             //    LCD Enable
        LCD_RS,                             //    LCD Command/Data Select, 0 = Command,
1 = Data
        LCD_DATA,                           //    LCD Data bus 8 bits
        ////////////////////// SD_Card Interface   ////////////////
        SD_DAT,                             //    SD Card Data
        SD_DAT3,                                    //    SD Card Data 3
        SD_CMD,                             //    SD Card Command Signal
        SD_CLK,                             //    SD Card Clock
        ////////////////////// USB JTAG link  ////////////////////
        TDI,                                // CPLD -> FPGA (data in)
```

```
        TCK,                                // CPLD -> FPGA (clk)
        TCS,                                // CPLD -> FPGA (CS)
        TDO,                                // FPGA -> CPLD (data out)
        ////////////////// I2C         ////////////////////////////
        I2C_SDAT,                           //    I2C Data
        I2C_SCLK,                       //    I2C Clock
        ////////////////// PS2         ////////////////////////////
        PS2_DAT,                            //    PS2 Data
        PS2_CLK,                            //    PS2 Clock
        ////////////////// VGA         ////////////////////////////
        VGA_CLK,                            //    VGA Clock
        VGA_HS,                             //    VGA H_SYNC
        VGA_VS,                             //    VGA V_SYNC
        VGA_BLANK,                          //    VGA BLANK
        VGA_SYNC,                           //    VGA SYNC
        VGA_R,                              //    VGA Red[9:0]
        VGA_G,                              //    VGA Green[9:0]
        VGA_B,                                  //    VGA Blue[9:0]
        ////////////    Ethernet Interface  ////////////////////////
        ENET_DATA,                          //    DM9000A DATA bus 16Bits
        ENET_CMD,                           //    DM9000A    Command/Data    Select,    0    =
Command, 1 = Data
        ENET_CS_N,                          //    DM9000A Chip Select
        ENET_WR_N,                          //    DM9000A Write
        ENET_RD_N,                          //    DM9000A Read
        ENET_RST_N,                         //    DM9000A Reset
        ENET_INT,                           //    DM9000A Interrupt
        ENET_CLK,                           //    DM9000A Clock 25 MHz
        ////////////////    Audio CODEC     ////////////////////////
        AUD_ADCLRCK,                        //    Audio CODEC ADC LR Clock
        AUD_ADCDAT,                         //    Audio CODEC ADC Data
        AUD_DACLRCK,                        //    Audio CODEC DAC LR Clock
        AUD_DACDAT,                         //    Audio CODEC DAC Data
        AUD_BCLK,                           //    Audio CODEC Bit-Stream Clock
        AUD_XCK,                                //    Audio CODEC Chip Clock
        ////////////////    TV Decoder      ////////////////////////
        TD_DATA,                            //    TV Decoder Data bus 8 bits
        TD_HS,                              //    TV Decoder H_SYNC
        TD_VS,                              //    TV Decoder V_SYNC
        TD_RESET,                           //    TV Decoder Reset
        ////////////////// GPIO        ////////////////////////////
        GPIO_0,                             //    GPIO Connection 0
        GPIO_1,                             //    GPIO Connection 1
    );
//=============================================================================
// PORT declarations
//=============================================================================

////////////////////// Clock Input          ////////////////////////
input               CLOCK_27;               //    27 MHz
input               CLOCK_50;               //    50 MHz
input               EXT_CLOCK;              //    External Clock
////////////////////// Push Button          ////////////////////////
input   [3:0] KEY;                          //    Pushbutton[3:0]
////////////////////// DPDT Switch          ////////////////////////
input   [17:0]  SW;                         //    Toggle Switch[17:0]
////////////////////// 7-SEG Dispaly        ////////////////////////
output  [6:0] HEX0;                         //    Seven Segment Digit 0
```

```
output    [6:0] HEX1;                    //    Seven Segment Digit 1
output    [6:0] HEX2;                    //    Seven Segment Digit 2
output    [6:0] HEX3;                    //    Seven Segment Digit 3
output    [6:0] HEX4;                    //    Seven Segment Digit 4
output    [6:0] HEX5;                    //    Seven Segment Digit 5
output    [6:0] HEX6;                    //    Seven Segment Digit 6
output    [6:0] HEX7;                    //    Seven Segment Digit 7
//////////////////////////    LED    //////////////////////////
output    [8:0] LEDG;                    //    LED Green[8:0]
output    [17:0]LEDR;                    //    LED Red[17:0]
//////////////////////////    UART    //////////////////////////
output          UART_TXD;                //    UART Transmitter
input           UART_RXD;                //    UART Receiver
//////////////////////////    IRDA//////////////////////////
output          IRDA_TXD;                //    IRDA Transmitter
input           IRDA_RXD;                //    IRDA Receiver
//////////////////////        SDRAM Interface    //////////////////////
inout     [15:0]    DRAM_DQ;                    //    SDRAM Data bus 16 Bits
output[11:0]  DRAM_ADDR;                //    SDRAM Address bus 12 Bits
output          DRAM_LDQM;               //    SDRAM Low-byte Data Mask
output          DRAM_UDQM;                    //    SDRAM High-byte Data Mask
output          DRAM_WE_N;               //    SDRAM Write Enable
output          DRAM_CAS_N;                   //    SDRAM Column Address Strobe
output          DRAM_RAS_N;                   //    SDRAM Row Address Strobe
output          DRAM_CS_N;               //    SDRAM Chip Select
output          DRAM_BA_0;               //    SDRAM Bank Address 0
output          DRAM_BA_1;               //    SDRAM Bank Address 0
output          DRAM_CLK;                //    SDRAM Clock
output          DRAM_CKE;                //    SDRAM Clock Enable
//////////////////////  Flash Interface//////////////////////
inout           [7:0] FL_DQ;             //    FLASH Data bus 8 Bits
output    [21:0]FL_ADDR;                 //    FLASH Address bus 22 Bits
output          FL_WE_N;                      //    FLASH Write Enable
output          FL_RST_N;                //    FLASH Reset
output          FL_OE_N;                 //    FLASH Output Enable
output          FL_CE_N;                 //    FLASH Chip Enable
//////////////////////  SRAM Interface    //////////////////////
inout     [15:0]    SRAM_DQ;                    //    SRAM Data bus 16 Bits
output[17:0]  SRAM_ADDR;                //    SRAM Address bus 18 Bits
output          SRAM_UB_N;               //    SRAM High-byte Data Mask
output          SRAM_LB_N;               //    SRAM Low-byte Data Mask
output          SRAM_WE_N;               //    SRAM Write Enable
output          SRAM_CE_N;               //    SRAM Chip Enable
output          SRAM_OE_N;               //    SRAM Output Enable
////////////////////  ISP1362 Interface    //////////////////////
inout     [15:0]    OTG_DATA;                //    ISP1362 Data bus 16 Bits
output[1:0]       OTG_ADDR;                //    ISP1362 Address 2 Bits
output          OTG_CS_N;                //    ISP1362 Chip Select
output          OTG_RD_N;                //    ISP1362 Write
output          OTG_WR_N;                //    ISP1362 Read
output          OTG_RST_N;               //    ISP1362 Reset
output          OTG_FSPEED;              //    USB Full Speed,     0 = Enable, Z = Disable
output          OTG_LSPEED;              //    USB Low Speed,     0 = Enable, Z = Disable
input                OTG_INT0;                   //    ISP1362 Interrupt 0
input                OTG_INT1;                   //    ISP1362 Interrupt 1
input                OTG_DREQ0;                  //    ISP1362 DMA Request 0
input                OTG_DREQ1;                  //    ISP1362 DMA Request 1
output          OTG_DACK0_N;             //    ISP1362 DMA Acknowledge 0
```

```
output              OTG_DACK1_N;                //    ISP1362 DMA Acknowledge 1
////////////////////  LCD Module 16X2  ////////////////////////////
inout         [7:0] LCD_DATA;                   //    LCD Data bus 8 bits
output              LCD_ON;                     //    LCD Power ON/OFF
output              LCD_BLON;                   //    LCD Back Light ON/OFF
output              LCD_RW;                     //    LCD Read/Write Select, 0 = Write, 1 = Read
output              LCD_EN;                     //    LCD Enable
output              LCD_RS;                     //    LCD Command/Data Select, 0 = Command,
1 = Data
////////////////////  SD Card Interface   ////////////////////////
inout                   SD_DAT;                 //    SD Card Data
inout                   SD_DAT3;                //    SD Card Data 3
inout                   SD_CMD;                 //    SD Card Command Signal
output              SD_CLK;                     //    SD Card Clock
////////////////////////  I2C       ////////////////////////////////
inout                   I2C_SDAT;               //    I2C Data
output              I2C_SCLK;               //    I2C Clock
////////////////////////  PS2       ////////////////////////////////
input                   PS2_DAT;                //    PS2 Data
input                   PS2_CLK;                //    PS2 Clock
////////////////////  USB JTAG link  //////////////////////////////
input               TDI;                        // CPLD -> FPGA (data in)
input               TCK;                        // CPLD -> FPGA (clk)
input               TCS;                        // CPLD -> FPGA (CS)
output              TDO;                        // FPGA -> CPLD (data out)
////////////////////////  VGA           ////////////////////////////
output              VGA_CLK;                    //    VGA Clock
output              VGA_HS;                     //    VGA H_SYNC
output              VGA_VS;                     //    VGA V_SYNC
output              VGA_BLANK;                  //    VGA BLANK
output              VGA_SYNC;                   //    VGA SYNC
output    [9:0]     VGA_R;                  //    VGA Red[9:0]
output    [9:0]     VGA_G;                  //    VGA Green[9:0]
output    [9:0]     VGA_B;                  //    VGA Blue[9:0]
////////////////     Ethernet Interface  ////////////////////////////
inout     [15:0]    ENET_DATA;                  //    DM9000A DATA bus 16Bits
output              ENET_CMD;                   //    DM9000A   Command/Data   Select,   0   =
Command, 1 = Data
output              ENET_CS_N;                  //    DM9000A Chip Select
output              ENET_WR_N;                  //    DM9000A Write
output              ENET_RD_N;                  //    DM9000A Read
output              ENET_RST_N;                 //    DM9000A Reset
input                   ENET_INT;               //    DM9000A Interrupt
output              ENET_CLK;                   //    DM9000A Clock 25 MHz
////////////////////  Audio CODEC       ////////////////////////////
inout                   AUD_ADCLRCK;            //    Audio CODEC ADC LR Clock
input                   AUD_ADCDAT;             //    Audio CODEC ADC Data
inout                   AUD_DACLRCK;            //    Audio CODEC DAC LR Clock
output              AUD_DACDAT;                 //    Audio CODEC DAC Data
inout                   AUD_BCLK;               //    Audio CODEC Bit-Stream Clock
output              AUD_XCK;                    //    Audio CODEC Chip Clock
////////////////////  TV Devoder        ////////////////////////////
input         [7:0] TD_DATA;                    //    TV Decoder Data bus 8 bits
input                   TD_HS;                  //    TV Decoder H_SYNC
input                   TD_VS;                  //    TV Decoder V_SYNC
output              TD_RESET;                   //    TV Decoder Reset
////////////////////////  GPIO      ////////////////////////////////
inout     [35:0]    GPIO_0;                     //    GPIO Connection 0
```

```
inout     [35:0]    GPIO_1;                    //    GPIO Connection 1

assign    LCD_ON        =    1'b1;
assign    LCD_BLON      =    1'b1;

//    All inout port turn to tri-state
assign    DRAM_DQ           =    16'hzzzz;
assign    FL_DQ             =    8'hzz;
assign    SRAM_DQ           =    16'hzzzz;
assign    OTG_DATA          =    16'hzzzz;
assign    LCD_DATA          =    8'hzz;
assign    SD_DAT        =    1'bz;
assign    ENET_DATA     =    16'hzzzz;
assign    AUD_ADCLRCK       =    1'bz;
assign    AUD_DACLRCK       =    1'bz;
assign    AUD_BCLK          =    1'bz;
assign    GPIO_1        =    36'hzzzzzzzzz;
//=============================================================================
// REG/WIRE declarations
//=============================================================================
// Touch panel signal //
wire [7:0] ltm_r;          //    LTM Red Data 8 Bits
wire [7:0] ltm_g;          //    LTM Green Data 8 Bits
wire [7:0] ltm_b;          //    LTM Blue Data 8 Bits
wire       ltm_nclk; //    LTM Clcok
wire       ltm_hd;
wire       ltm_vd;
wire       ltm_den;
wire       ltm_grst;
// lcd 3wire interface//
wire       ltm_sclk;
wire       ltm_sda;
wire       ltm_scen;
wire            ltm_3wirebusy_n;
// Touch Screen Digitizer ADC
wire            adc_dclk;
wire            adc_cs;
wire            adc_penirq_n;
wire            adc_busy;
wire            adc_din;
wire            adc_dout;
wire            adc_ltm_sclk;
wire [11:0]     x_coord;
wire [11:0]     y_coord;
wire       new_coord;
wire touch_irq;
wire display_mode;
reg  [31:0]     div;
reg     [2:0] irq_num;
wire pll_c1;
wire [17:0] VGA_ADDR;
wire [15:0] VGA_DQ;
wire VGA_WE_N;
wire VGA_OE_N;
wire [10:0] x_cnt;
wire [9:0] y_cnt;
wire [31:0]Panel_command;
wire OverridePANEL;
```

```
wire [17:0]PANEL_ADDR;
wire [31:0]Mole_command;
wire OverrideMOLE;
wire [17:0]MOLE_ADDR;
wire [11:0] x_coord1;
wire [11:0] y_coord1;
wire [3:0] state;
wire clk_18;
wire irq;
reg [1:0] interrupt;

////////////// GPIO_O /////////////////
assign      adc_penirq_n    =GPIO_0[0];
assign      adc_dout        =GPIO_0[1];
assign      adc_busy        =GPIO_0[2];
assign      GPIO_0[3]       =adc_din;
assign      GPIO_0[4]       =adc_ltm_sclk;
assign      GPIO_0[5]       =ltm_b[3];
assign      GPIO_0[6]       =ltm_b[2];
assign      GPIO_0[7]       =ltm_b[1];
assign      GPIO_0[8]       =ltm_b[0];
assign      GPIO_0[9]       =ltm_nclk;
assign      GPIO_0[10]      =ltm_den;
assign      GPIO_0[11]      =ltm_hd;
assign      GPIO_0[12]      =ltm_vd;
assign      GPIO_0[13]      =ltm_b[4];
assign      GPIO_0[14]      =ltm_b[5];
assign      GPIO_0[15]      =ltm_b[6];
assign      GPIO_0[16]      =ltm_b[7];
assign      GPIO_0[17]      =ltm_g[0];
assign      GPIO_0[18]      =ltm_g[1];
assign      GPIO_0[19]      =ltm_g[2];
assign      GPIO_0[20]      =ltm_g[3];
assign      GPIO_0[21]      =ltm_g[4];
assign      GPIO_0[22]      =ltm_g[5];
assign      GPIO_0[23]      =ltm_g[6];
assign      GPIO_0[24]      =ltm_g[7];
assign      GPIO_0[25]      =ltm_r[0];
assign      GPIO_0[26]      =ltm_r[1];
assign      GPIO_0[27]      =ltm_r[2];
assign      GPIO_0[28]      =ltm_r[3];
assign      GPIO_0[29]      =ltm_r[4];
assign      GPIO_0[30]      =ltm_r[5];
assign      GPIO_0[31]      =ltm_r[6];
assign      GPIO_0[32]      =ltm_r[7];
assign      GPIO_0[33]      =ltm_grst;
assign      GPIO_0[34]      =ltm_scen;
assign      GPIO_0[35]      =ltm_sda;

assign adc_ltm_sclk= ( adc_dclk & ltm_3wirebusy_n )    |    ( ~ltm_3wirebusy_n & ltm_sclk );
assign ltm_nclk = div[0]; // 25 Mhz
assign ltm_grst = KEY[0];

assign LEDR[17] = display_mode;
assign LEDR[16] = irq;
assign LEDR[15] = irq_num[2];
assign LEDR[14] = irq_num[1];
assign LEDR[13] = irq_num[0];
```

```
assign VGA_R[9] = ltm_r[7];
assign VGA_R[8] = ltm_r[6];
assign VGA_R[7] = ltm_r[5];
assign VGA_R[6] = ltm_r[4];
assign VGA_R[5] = ltm_r[3];
assign VGA_R[4] = ltm_r[2];
assign VGA_R[3] = ltm_r[1];
assign VGA_R[2] = ltm_r[0];
assign VGA_R[1] = 0;
assign VGA_R[0] = 0;

assign VGA_B[9] = ltm_b[7];
assign VGA_B[8] = ltm_b[6];
assign VGA_B[7] = ltm_b[5];
assign VGA_B[6] = ltm_b[4];
assign VGA_B[5] = ltm_b[3];
assign VGA_B[4] = ltm_b[2];
assign VGA_B[3] = ltm_b[1];
assign VGA_B[2] = ltm_b[0];
assign VGA_B[1] = 0;
assign VGA_B[0] = 0;

assign VGA_G[9] = ltm_g[7];
assign VGA_G[8] = ltm_g[6];
assign VGA_G[7] = ltm_g[5];
assign VGA_G[6] = ltm_g[4];
assign VGA_G[5] = ltm_g[3];
assign VGA_G[4] = ltm_g[2];
assign VGA_G[3] = ltm_g[1];
assign VGA_G[2] = ltm_g[0];
assign VGA_G[1] = 0;
assign VGA_G[0] = 0;

assign      VGA_HS   =ltm_hd;
assign      VGA_VS   =ltm_vd;
assign      VGA_CLK    = ltm_nclk;
assign      VGA_SYNC = 0;

always @(posedge pll_c1)
      begin
            div <= div+1;
      end

always@(posedge pll_c1)
begin
if(touch_irq)
begin
irq_num <= irq_num +1;
interrupt <= interrupt +1;
end
else
begin
irq_num <= irq_num;
interrupt <= irq_num;
end
end
```

```
de2_i2c_av_config de2_i2c_av_config1 (
      .iCLK(pll_c1),
      .iRST_N(1'b1),
      .I2C_SCLK(I2C_SCLK),
      .I2C_SDAT(I2C_SDAT)
   );

double_pll pll1(
      .inclk0(CLOCK_50),
      .c0(DRAM_CLK),
      .c1(pll_c1),
      .c2(clk_18));

SEG7_LUT_8        u1   (
                       .oSEG0(HEX0),
                       .oSEG1(HEX1),
                       .oSEG2(HEX2),
                       .oSEG3(HEX3),
                       .oSEG4(HEX4),
                       .oSEG5(HEX5),
                       .oSEG6(HEX6),
                       .oSEG7(HEX7),
                       .iDIG({ 4'h0 , x_coord , 4'h0 , y_coord }),
                       .ON_OFF(8'b01110111)
                       );

// lcd 3 wire interface configuration    //
lcd_spi_cotroller    u2   (
                       // Host Side
                       .iCLK(pll_c1),
                       .iRST_n(DLY0),
                       // 3wire Side
                       .o3WIRE_SCLK(ltm_sclk),
                       .io3WIRE_SDAT(ltm_sda),
                       .o3WIRE_SCEN(ltm_scen),
                       .o3WIRE_BUSY_n(ltm_3wirebusy_n)
                       );

// system reset    //
Reset_Delay          u3   (.iCLK(pll_c1),
                       .iRST(KEY[0]),
                       .oRST_0(DLY0),
                       .oRST_1(DLY1),
                       .oRST_2(DLY2)
                       );

// Touch Screen Digitizer ADC configuration //
adc_spi_controller  u4   (
                       .iCLK(pll_c1),
                       .iRST_n(DLY0),
                       .oADC_DIN(adc_din),
                       .oADC_DCLK(adc_dclk),
                       .oADC_CS(adc_cs),
                       .iADC_DOUT(adc_dout),
                       .iADC_BUSY(adc_busy),
                       .iADC_PENIRQ_n(adc_penirq_n),
```

```
                        .oTOUCH_IRQ(touch_irq),
                        .oX_COORD(x_coord),
                        .oY_COORD(y_coord),
                        .oNEW_COORD(new_coord),
                        );

touch_irq_detector detector1(
                        .clk(pll_c1),
                        .reset_n(DLY0),
                        .iTOUCH_IRQ(touch_irq),
                        .iX_COORD(x_coord),
                        .iY_COORD(y_coord),
                        .iNEW_COORD(new_coord),
                        .oDISPLAY_MODE(display_mode)
                        );

single_clk_irq single_clk_irq1(
                        .clk(pll_c1),
                        .reset_n(DLY0),
                        .idisplay_mode(display_mode),
                        .irq(irq));

lcd_controller u5(
        .clk(pll_c1),
        .clk_ltm(ltm_nclk),
        .reset_n(DLY2),
        .iMOLECTRL(Mole_command),
        .iPANELCTRL(Panel_command),
        .oHD(ltm_hd),                   // LCD Horizontal sync
        .oVD(ltm_vd),                   // LCD Vertical sync
        .oDEN(ltm_den),                     // LCD Data Enable
        .oLCD_R(ltm_r),                     // LCD Red color data
        .oLCD_G(ltm_g),                   // LCD Green color data
        .oLCD_B(ltm_b),                   // LCD Blue color data
        .oVGA_ADDR(VGA_ADDR),
        .oVGA_WE_N(VGA_WE_N),
        .oVGA_OE_N(VGA_OE_N),
        .iVGA_DQ(VGA_DQ)
        );

assign LEDG[0] = Mole_command[0];
assign LEDG[1] = Mole_command[1];
assign LEDG[2] = Mole_command[2];
assign LEDG[3] = Mole_command[3];
assign LEDG[4] = Mole_command[8];
assign LEDG[5] = Mole_command[9];
assign LEDG[6] = Mole_command[10];
assign LEDG[7] = Mole_command[11];


nios_system DUT
    (
        .AUD_ADCDAT_to_the_musiccontroller_0          (AUD_ADCDAT),
      .AUD_ADCLRCK_from_the_musiccontroller_0      (AUD_ADCLRCK),
      .AUD_BCLK_to_and_from_the_musiccontroller_0 (AUD_BCLK),
      .AUD_DACDAT_from_the_musiccontroller_0       (AUD_DACDAT),
      .AUD_DACLRCK_from_the_musiccontroller_0      (AUD_DACLRCK),
        .AUD_XCK_from_the_musiccontroller_0          (AUD_XCK),
```

```
        .SRAM_ADDR_from_the_sram_mux_0                  (SRAM_ADDR),
        .SRAM_CE_N_from_the_sram_mux_0                  (SRAM_CE_N),
        .SRAM_DQ_to_and_from_the_sram_mux_0             (SRAM_DQ),
        .SRAM_LB_N_from_the_sram_mux_0                  (SRAM_LB),
        .SRAM_OE_N_from_the_sram_mux_0                  (SRAM_OE_N),
        .SRAM_UB_N_from_the_sram_mux_0                  (SRAM_UB_N),
        .SRAM_WE_N_from_the_sram_mux_0                  (SRAM_WE_N),
        .VGA_ADDR_to_the_sram_mux_0                     (VGA_ADDR),
        .VGA_DQ_from_the_sram_mux_0                     (VGA_DQ),
        .VGA_OE_N_to_the_sram_mux_0                     (VGA_OE_N),
        .VGA_WE_N_to_the_sram_mux_0                     (VGA_WE_N),
        .clk_0                                          (pll_c1),
            .clk_18_to_the_musiccontroller_0          (clk_18),
        .iX_COORD_to_the_xycoord_interface_0          (x_coord),
        .iY_COORD_to_the_xycoord_interface_0          (y_coord),
           .irq_adc_to_the_interrupt_0                (display_mode),
        .oMOLECTRL_from_the_command_0                       (Mole_command),
        .oPANELCTRL_from_the_command_0                  (Panel_command),
        .reset_n                                        (DLY0),
           .zs_addr_from_the_sdram_0                    (DRAM_ADDR),
        .zs_ba_from_the_sdram_0                         ({DRAM_BA_1,DRAM_BA_0}),
        .zs_cas_n_from_the_sdram_0                      (DRAM_CAS_N),
        .zs_cke_from_the_sdram_0                        (DRAM_CKE),
        .zs_cs_n_from_the_sdram_0                       (DRAM_CS_N),
        .zs_dq_to_and_from_the_sdram_0                  (DRAM_DQ),
        .zs_dqm_from_the_sdram_0                        ({DRAM_UDQM,DRAM_LDQM}),
        .zs_ras_n_from_the_sdram_0                      (DRAM_RAS_N),
        .zs_we_n_from_the_sdram_0                       (DRAM_WE_N)
    );
endmodule
```

interrupt.v

```
module interrupt(
                clk,
                reset_n,
                chipselect,
                read,
                write,
                address,
                readdata,
                writedata,
                irq,
                irq_adc
                );

//================================
//IO PORT
//================================
input clk;
input reset_n;
input chipselect;
input read;
input write;
input [4:0] address;
```

```
output [31:0] readdata;
input [31:0] writedata;
output irq;
input irq_adc;
//==================================
//REG and WIRE
//==================================
reg irq;


//==================================
//STRUCTURE LOGIC
//==================================
always@(posedge clk or negedge reset_n)
begin
	if(!reset_n)
		begin
		irq <= 0;
		end
	else if(irq_adc == 1)
		irq <= 1;
	else if(write && chipselect)
		irq <= 0;
end

endmodule
```

irq_gen.v

```
module irq_gen(
			iclk,
			ireset,
			xcoord,
			ycoord,
			irq);

input iclk;
input ireset;
input [11:0] xcoord;
input [11:0] ycoord;
output irq;

reg [11:0] prex;
reg [11:0] prey;
reg irq;

always@(posedge iclk or negedge ireset)
begin
	if(!ireset)
	begin
		irq <= 0;
		prex<= 0;
		prey <= 0;
	end
	else
	begin
```

```
            if((prex != xcoord)|| (prey != ycoord))
            begin
                  irq <= 1;
                  prex<= xcoord;
                  prey <= ycoord;
            end
            else
                  irq <=0;
      end
end

endmodule
```

lcd_controller.v

```
module lcd_controller(
            clk, //50M
            clk_ltm, //25M
            reset_n,
            iMOLECTRL,
            iPANELCTRL,
            oHD,                        // LCD Horizontal sync
            oVD,                  // LCD Vertical sync
            oDEN,                       // LCD Data Enable
            oLCD_R,                     // LCD Red color data
            oLCD_G,                     // LCD Green color data
            oLCD_B,                     // LCD Blue color data
            oVGA_ADDR,
            oVGA_WE_N,
            oVGA_OE_N,
            iVGA_DQ,
            );

//=============================================================================
// PARAMETER declarations
//=============================================================================
parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
//=============================================================================

////IO PORT DECLARATION
//=============================================================================
input clk;
input clk_ltm;
input reset_n;
input [31:0] iMOLECTRL;
input [31:0] iPANELCTRL;
output     [7:0] oLCD_R;
output    [7:0]  oLCD_G;
output    [7:0]  oLCD_B;
output     oHD;
output     oVD;
```

```
output     oDEN;
output [17:0] oVGA_ADDR;
output oVGA_WE_N;
output oVGA_OE_N;
input [15:0]iVGA_DQ;


///reg and wire
reg         [10:0]    x_cnt;
reg         [9:0] y_cnt;
wire [7:0] mred;
wire [7:0] mgreen;
wire [7:0] mblue;
wire                display_area;
reg                       mhd;
reg                       mvd;
reg                       mden;
reg                       oHD;
reg                       oVD;
reg                       oDEN;
reg         [7:0] oLCD_R;
reg         [7:0] oLCD_G;
reg         [7:0] oLCD_B;
reg         [7:0] red_1;
reg   [7:0] green_1;
reg   [7:0] blue_1;

reg [17:0] BACK_ADDR;
wire [17:0] WELCOME_ADDR;
wire [17:0] PLAY_ADDR;
wire [17:0] WIN_ADDR;
wire [17:0] LOSE_ADDR;
reg [17:0] address;
reg [3:0] state;
reg [31:0] mole_ctrl;
reg [31:0] panel_ctrl;

assign oVGA_ADDR = BACK_ADDR;
assign oVGA_WE_N = 1'b0;
assign oVGA_OE_N = display_area ? 1'b1 : 1'b0;
//assign state = iPANELCTRL[3:0];

always@(posedge clk_ltm or negedge reset_n)
      begin
            if (!reset_n)
            begin
            state <= 4'b0;
            mole_ctrl <= 32'b0;
            panel_ctrl <= 32'b0;
            end
            else if(mvd == 1)
            begin
            state <= iPANELCTRL[3:0];
            mole_ctrl <= iMOLECTRL;
            panel_ctrl <= iPANELCTRL;
            end
            else
            begin
```

```
            state <= state;
            mole_ctrl <= mole_ctrl;
            panel_ctrl <= panel_ctrl;
            end
      end

// This signal indicate the lcd display area .
assign      display_area = ((x_cnt>(Hsync_Blank-1)&& //>215
                              (x_cnt<(H_LINE-Hsync_Front_Porch))&& //< 1016
                              (y_cnt>(Vertical_Back_Porch-1))&&
                              (y_cnt<(V_LINE - Vertical_Front_Porch))
                              ))    ? 1'b1 : 1'b0;


////////////////////////// x    y counter    and lcd hd generator ///////////////////
always@(posedge clk_ltm or negedge reset_n)
      begin
            if (!reset_n)
            begin
                  x_cnt <= 11'd0;
                  mhd    <= 1'd0;
            end
            else if (x_cnt == (H_LINE-1))
            begin
                  x_cnt <= 11'd0;
                  mhd    <= 1'd0;
            end
            else
            begin
                  x_cnt <= x_cnt + 11'd1;
                  mhd    <= 1'd1;
            end
      end

always@(posedge clk_ltm or negedge reset_n)
      begin
            if (!reset_n)
                  y_cnt <= 10'd0;
            else if (x_cnt == (H_LINE-1))
            begin
                  if (y_cnt == (V_LINE-1))
                        y_cnt <= 10'd0;
                  else
                        y_cnt <= y_cnt + 10'd1;
            end
      end
////////////////////////////// touch panel timing ///////////////////

always@(posedge clk_ltm    or negedge reset_n)
      begin
            if (!reset_n)
                  mvd    <= 1'b1;
            else if (y_cnt == 10'd0)
                  mvd    <= 1'b0;
            else
                  mvd    <= 1'b1;
      end
```

```
always@(posedge clk_ltm    or negedge reset_n)
      begin
            if (!reset_n)
                  mden    <= 1'b0;
            else if (display_area)
                  mden    <= 1'b1;
            else
                  mden    <= 1'b0;
      end

//=============================================================
//Assigning Address=============================================
//=============================================================
always@(posedge clk_ltm or negedge reset_n)
begin
      if(!reset_n)
            BACK_ADDR <= 18'bZ;
      else if (state == 4'b0000 || state==4'b0001 || state ==4'b0010 || state == 4'b0011)
            BACK_ADDR <= WELCOME_ADDR;
      else if (state == 4'b0100 || state==4'b0101 || state ==4'b0110 || state == 4'b0111)
            BACK_ADDR <= PLAY_ADDR;
      else if (state == 4'b1000 || state==4'b1001 || state ==4'b1010 || state == 4'b1011)
            BACK_ADDR <= WIN_ADDR;
      else if (state == 4'b1100 || state==4'b1101 || state ==4'b1110 || state == 4'b1111)
            BACK_ADDR <= LOSE_ADDR;
end




//=============================================================
//Welcome Mode==================================================
//=============================================================
welcome welcome1(
                        .iclk(clk_ltm),
                        .ireset_n(reset_n),
                        .ix_cnt(x_cnt),
                        .iy_cnt(y_cnt),
                        .istate(state),
                        .oADDR(WELCOME_ADDR));

//=============================================================
//Running Mode==================================================
//=============================================================
play play1(
                        .iclk(clk_ltm),
                        .ireset_n(reset_n),
                        .ix_cnt(x_cnt), // input from lcd controlelr
                        .iy_cnt(y_cnt), // input from lcd controller
                        .iPANELCTRL(panel_ctrl), // command from nios containing 32 bits
                        .iMOLECTRL(mole_ctrl),
                        .oADDR(PLAY_ADDR), //the output address to the lcd controller
                              );
//=============================================================
//Win Mode======================================================
//=============================================================
win win1(
                        .iclk(clk_ltm),
```

```
                        .ireset_n(reset_n),
                        .ix_cnt(x_cnt),
                        .iy_cnt(y_cnt),
                        .oADDR(WIN_ADDR));
//=============================================================
//Lose Mode=================================================
//=============================================================
lose lose1(
                        .iclk(clk_ltm),
                        .ireset_n(reset_n),
                        .ix_cnt(x_cnt),
                        .iy_cnt(y_cnt),
                        .oADDR(LOSE_ADDR));

//=============================================================
//OUTPUT LOGIC==============================================
//=============================================================
always@(posedge clk_ltm or negedge reset_n)
      begin
            if (!reset_n)
                  begin
                  red_1 <= 8'd255;
                  green_1 <= 8'd255;
                  blue_1 <= 8'd255;
                  end
            else
                  begin
                  red_1 <= (iVGA_DQ[15:11]<<3);
                  green_1 <= (iVGA_DQ[10:5]<<2);
                  blue_1 <= (iVGA_DQ[4:0]<<3);
                  end
      end

///Assign RGB to OUPUT BLOCK
assign mred = red_1;
assign mgreen = green_1;
assign mblue    = blue_1;

////////////////////////////////////OUTPUT BLOCK///    ///////////////
always@(posedge clk_ltm or negedge reset_n)
      begin
            if (!reset_n)
                  begin
                        oHD <= 1'd0;
                        oVD <= 1'd0;
                        oDEN <= 1'd0;
                        oLCD_R <= 8'd0;
                        oLCD_G <= 8'd0;
                        oLCD_B <= 8'd0;
                  end
            else
                  begin
                        oHD <= mhd;
                        oVD <= mvd;
                        oDEN <= display_area;
                        oLCD_R <= mred;
                        oLCD_G <= mgreen;
                        oLCD_B <= mblue;
```

```
                end
        end

endmodule
```

lcd_spi_controller.v

```
// --------------------------------------------------------------------

// Copyright (c) 2005 by Terasic Technologies Inc.

// --------------------------------------------------------------------

//

// Permission:

//

//     Terasic grants permission to use and modify this code for use

//     in synthesis for all Terasic Development Boards and Altera Development

//     Kits made by Terasic.    Other use of this code, including the selling

//     ,duplication, or modification of any portion is strictly prohibited.

//

// Disclaimer:

//

//     This VHDL/Verilog or C/C++ source code is intended as a design reference

//     which illustrates how these types of functions can be implemented.

//     It is the user's responsibility to verify their design for

//     consistency and functionality through the use of formal

//     verification methods.    Terasic provides no warranty regarding the use

//     or functionality of this code.

//

// --------------------------------------------------------------------

//

//                                 Terasic Technologies Inc

//                                 356 Fu-Shin E. Rd Sec. 1. JhuBei City,

//                                 HsinChu County, Taiwan
```

```
//                              302
//
//                              web: http://www.terasic.com/
//                              email: support@terasic.com
//
// ---------------------------------------------------------------------
//
// Major Functions: This function will transmit the lcd register setting
//
// ---------------------------------------------------------------------
//
// Revision History :
// ---------------------------------------------------------------------
//     Ver   :| Author                    :| Mod. Date :| Changes Made:
//     V1.0 :| Johnny Fan                 :| 07/06/30   :| Initial Revision
// ---------------------------------------------------------------------
module lcd_spi_cotroller (//  Host Side
                          iCLK,
                          iRST_n,
                          //    3wire interface side
                          o3WIRE_SCLK,
                          io3WIRE_SDAT,
                          o3WIRE_SCEN,
                          o3WIRE_BUSY_n
                          );
//=============================================================================
// PARAMETER declarations
//=============================================================================
```

```
parameter      LUT_SIZE =      20; // Total setting register numbers
//=============================================================================
// PORT declarations
//=============================================================================
//     Host Side
output          o3WIRE_BUSY_n;
input           iCLK;
input           iRST_n;
//     3wire interface side
output          o3WIRE_SCLK;
inout           io3WIRE_SDAT;
output          o3WIRE_SCEN;
//     Internal Registers/Wires
//=============================================================================
// REG/WIRE declarations
//=============================================================================
reg             m3wire_str;
wire        m3wire_rdy;
wire        m3wire_ack;
wire        m3wire_clk;
reg   [15:0]    m3wire_data;
reg   [15:0]    lut_data;
reg   [5:0] lut_index;
reg   [3:0] msetup_st;
reg             o3WIRE_BUSY_n;
wire        v_reverse; // display Vertical reverse function
wire        h_reverse; // display Horizontal reverse function
wire [9:0] g0;
wire [9:0] g1;
```

```
wire [9:0] g2;
wire [9:0] g3;
wire [9:0] g4;
wire [9:0] g5;
wire [9:0] g6;
wire [9:0] g7;
wire [9:0] g8;
wire [9:0] g9;
wire [9:0] g10;
wire [9:0] g11;

//=============================================================================
// Structural coding
//=============================================================================

assign      h_reverse = 1'b1;
assign      v_reverse = 1'b0;

three_wire_controller    u0   (      //     Host Side
                                 .iCLK(iCLK),
                                 .iRST(iRST_n),
                                 .iDATA(m3wire_data),
                                 .iSTR(m3wire_str),
                                 .oACK(m3wire_ack),
                                 .oRDY(m3wire_rdy),
                                 .oCLK(m3wire_clk),
                                 //     Serial Side
                                 .oSCEN(o3WIRE_SCEN),
                                 .SDA(io3WIRE_SDAT),
```

```
                .oSCLK(o3WIRE_SCLK)
            );
//////////////////////  Config Control //////////////////////////
always@(posedge m3wire_clk or negedge iRST_n)
begin
    if(!iRST_n)
    begin
        lut_index <=  0;
        msetup_st   <=  0;
        m3wire_str  <=  0;
        o3WIRE_BUSY_n  <=  0;
    end
    else
    begin
        if(lut_index<LUT_SIZE)
        begin
        o3WIRE_BUSY_n  <=  0;
            case(msetup_st)
            0:  begin
                    msetup_st   <=  1;
                end
            1:  begin
                    msetup_st   <=  2;
                end

            2:  begin
                    m3wire_data <=  lut_data;
                    m3wire_str  <=  1;
                    msetup_st   <=  3;
```

```
                    end
                3:  begin
                        if(m3wire_rdy)
                        begin
                            if(m3wire_ack)
                            msetup_st     <=  4;
                            else
                            msetup_st     <=  0;
                            m3wire_str    <=  0;
                        end
                    end
                4:  begin
                        lut_index <=  lut_index+1;
                        msetup_st     <=  0;
                    end
                endcase
            end
            else            o3WIRE_BUSY_n   <=  1;
        end
end

assign    g0 =106;
assign    g1 =200;
assign    g2 =289;
assign    g3 =375;
assign    g4 =460;
assign    g5 =543;
assign    g6 =625;
assign    g7 =705;
```

```
assign     g8 =785;

assign     g9 =864;

assign     g10 = 942;

assign     g11 = 1020;


////////////////////     Config Data LUT        ////////////////////////

always

begin

      case(lut_index)


      0          :     lut_data  <=   {6'h11,2'b01,g0[9:8],g1[9:8],g2[9:8],g3[9:8]};

      1          :     lut_data  <=   {6'h12,2'b01,g4[9:8],g5[9:8],g6[9:8],g7[9:8]};

      2          :     lut_data  <=   {6'h13,2'b01,g8[9:8],g9[9:8],g10[9:8],g11[9:8]};

      3          :     lut_data  <=   {6'h14,2'b01,g0[7:0]};

      4          :     lut_data  <=   {6'h15,2'b01,g1[7:0]};

      5          :     lut_data  <=   {6'h16,2'b01,g2[7:0]};

      6          :     lut_data  <=   {6'h17,2'b01,g3[7:0]};

      7          :     lut_data  <=   {6'h18,2'b01,g4[7:0]};

      8          :     lut_data  <=   {6'h19,2'b01,g5[7:0]};

      9          :     lut_data  <=   {6'h1a,2'b01,g6[7:0]};

      10         :     lut_data  <=   {6'h1b,2'b01,g7[7:0]};

      11         :     lut_data  <=   {6'h1c,2'b01,g8[7:0]};

      12         :     lut_data  <=   {6'h1d,2'b01,g9[7:0]};

      13         :     lut_data  <=   {6'h1e,2'b01,g10[7:0]};

      14         :     lut_data  <=   {6'h1f,2'b01,g11[7:0]};

      15         :     lut_data  <=   {6'h20,2'b01,4'hf,4'h0};

      16         :     lut_data  <=   {6'h21,2'b01,4'hf,4'h0};

      17         :     lut_data  <=   {6'h03, 2'b01, 8'hdf};

      18         :     lut_data  <=   {6'h02, 2'b01, 8'h07};
```

```
    19      :   lut_data  <=  {6'h04, 2'b01, 6'b000101,!v_reverse,!h_reverse};
    default :   lut_data  <=  16'h0000;
    endcase
end
////////////////////////////////////////////////////////////////////

endmodule
```

lcd_timing_controller.v

```
// --------------------------------------------------------------------
// Copyright (c) 2005 by Terasic Technologies Inc.
// --------------------------------------------------------------------
//
// Permission:
//
//   Terasic grants permission to use and modify this code for use
//   in synthesis for all Terasic Development Boards and Altera Development
//   Kits made by Terasic.  Other use of this code, including the selling
//   ,duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
//   This VHDL/Verilog or C/C++ source code is intended as a design reference
//   which illustrates how these types of functions can be implemented.
//   It is the user's responsibility to verify their design for
//   consistency and functionality through the use of formal
//   verification methods.  Terasic provides no warranty regarding the use
//   or functionality of this code.
//
```

```
// --------------------------------------------------------------------
//
//                          Terasic Technologies Inc
//                          356 Fu-Shin E. Rd Sec. 1. JhuBei City,
//                          HsinChu County, Taiwan
//                          302
//
//                          web: http://www.terasic.com/
//                          email: support@terasic.com
//
// --------------------------------------------------------------------
//
// Major Functions: DE2 LTM module Timing control and color pattern
//                          generator
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
//     Ver  :| Author                :| Mod. Date :| Changes Made:
//     V1.0 :| Johnny Fan            :| 07/06/30  :| Initial Revision
// --------------------------------------------------------------------
module lcd_timing_controller        (
                                    clk,            // LCD display clock
                                    reset_n,        // systen reset
                                    read,
                                    write,
                                    chipselect,
                                    address,
```

```
                        readdata,
                        writedata,
                        //LCD SIDE
                        oHD,                    // LCD Horizontal sync
                        oVD,                  // LCD Vertical sync
                        oDEN,                   // LCD Data Enable
                        oLCD_R,                 // LCD Red color data
                        oLCD_G,                 // LCD Green color data
                        oLCD_B,                 // LCD Blue color data
                        oVGA_ADDR,
                        oVGA_WE_N,
                        oVGA_OE_N,
                        iVGA_DQ,
                        iDISPLAY_MODE
                        );
//=============================================================================
// PARAMETER declarations
//=============================================================================
parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
//=============================================================================
// PORT declarations
//=============================================================================
input           clk;
input           reset_n;
```

```
input              read;
input              write;
input              chipselect;
input [4:0]address;
output [15:0]readdata;
input [15:0]writedata;
output     [7:0] oLCD_R;
output    [7:0]  oLCD_G;
output    [7:0]  oLCD_B;
output                   oHD;
output                   oVD;
output                   oDEN;
input       [1:0] iDISPLAY_MODE;
output [17:0] oVGA_ADDR;
output oVGA_WE_N;
output oVGA_OE_N;
input [15:0] iVGA_DQ;
//=============================================================================
// REG/WIRE declarations
//=============================================================================
reg          [10:0]    x_cnt;
reg          [9:0] y_cnt;
wire [7:0] mred;
wire [7:0] mgreen;
wire [7:0] mblue;
wire               display_area;
reg                      mhd;
reg                      mvd;
reg                      mden;
```

```
reg                 oHD;
reg                 oVD;
reg                 oDEN;
reg       [7:0] oLCD_R;
reg       [7:0] oLCD_G;
reg       [7:0] oLCD_B;
wire [1:0] msel;
reg       [7:0] red_1;
reg   [7:0] green_1;
reg   [7:0] blue_1;
reg   [7:0] graycnt;
reg       [7:0] pattern_data;
reg [31:0] div;
wire [15:0] area;
reg [23:0]oRGB_DATA; //information containing mole state
reg [2:0]mole0;
reg [2:0]mole1;
reg [2:0]mole2;
reg [2:0]mole3;
reg [2:0]mole4;
reg [2:0]mole5;
reg [2:0]mole6;
reg [2:0]mole7;
reg [2:0]mole8;
reg [2:0]mole9;
wire [7:0]LED1_R;
wire [7:0]LED1_G;
wire [7:0]LED1_B;
wire [7:0]LED2_R;
```

```
wire [7:0]LED2_G;

wire [7:0]LED2_B;

wire [7:0]LED0_R;

wire [7:0]LED0_G;

wire [7:0]LED0_B;

wire [2:0]oOVERRIDE;

reg [3:0] inumber_s;

reg [3:0] inumber_ts;

reg [3:0] inumber_min;

reg [26:0] counter;


wire [3:0] num_input_s;

wire [3:0] num_input_ts;

wire [3:0] num_input_min;

wire [26:0] oCNT;

wire flag1;

wire flag2;

wire clk_ltm;

assign clk_ltm = div[0]; // 25 Mhz

reg [14:0] area_counter[0:14];

//=============================================================================

// Structural coding

//=============================================================================

always @(posedge clk)

	begin

		div <= div+1;

	end


// This signal indicate the lcd display area .
```

```
assign      display_area = ((x_cnt>(Hsync_Blank-1)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0;



////////////////////////// x    y counter    and lcd hd generator //////////////////

always@(posedge clk_ltm or negedge reset_n)
      begin
            if (!reset_n)
            begin
                  x_cnt <= 11'd0;
                  mhd    <= 1'd0;
            end
            else if (x_cnt == (H_LINE-1))
            begin
                  x_cnt <= 11'd0;
                  mhd    <= 1'd0;
            end
            else
            begin
                  x_cnt <= x_cnt + 11'd1;
                  mhd    <= 1'd1;
            end
      end


always@(posedge clk_ltm or negedge reset_n)
      begin
```

```
        if (!reset_n)
            y_cnt <= 10'd0;
        else if (x_cnt == (H_LINE-1))
        begin
            if (y_cnt == (V_LINE-1))
                y_cnt <= 10'd0;
            else
                y_cnt <= y_cnt + 10'd1;
        end
    end
////////////////////////////// touch panel timing ///////////////////

always@(posedge clk_ltm   or negedge reset_n)
    begin
        if (!reset_n)
            mvd   <= 1'b1;
        else if (y_cnt == 10'd0)
            mvd   <= 1'b0;
        else
            mvd   <= 1'b1;
    end

always@(posedge clk_ltm   or negedge reset_n)
    begin
        if (!reset_n)
            mden   <= 1'b0;
        else if (display_area)
            mden   <= 1'b1;
        else
```

```
            mden    <= 1'b0;
    end



//Detect which area you are in// 0~14 different areas//offset corrected
assign area[0] = ((x_cnt>(Hsync_Blank-2+400)&& //>214
                        (x_cnt<(H_LINE-Hsync_Front_Porch-1-200))&& //< 1015
                        (y_cnt>(Vertical_Back_Porch-1))&&
                        (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                        ))    ? 1'b1 : 1'b0 ; //mole0
assign area[1] = ((x_cnt>(Hsync_Blank-2+200)&& //>215
                        (x_cnt<(H_LINE-Hsync_Front_Porch-1-400))&& //< 1016
                        (y_cnt>(Vertical_Back_Porch-1))&&
                        (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                        ))    ? 1'b1 : 1'b0;//mole1
assign area[2] = ((x_cnt>(Hsync_Blank-2)&& //>215
                        (x_cnt<(H_LINE-Hsync_Front_Porch-1-600))&& //< 1016
                        (y_cnt>(Vertical_Back_Porch-1))&&
                        (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                        ))    ? 1'b1 : 1'b0 ;//mole2
assign area[3] = ((x_cnt>(Hsync_Blank-2+400)&& //>215
                        (x_cnt<(H_LINE-Hsync_Front_Porch-1-200))&& //< 1016
                        (y_cnt>(Vertical_Back_Porch-1+160))&&
                        (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                        ))    ? 1'b1 : 1'b0 ;    //mole3
assign area[4] = ((x_cnt>(Hsync_Blank-2+200)&& //>215
                        (x_cnt<(H_LINE-Hsync_Front_Porch-1-400))&& //< 1016
                        (y_cnt>(Vertical_Back_Porch-1+160))&&
```

```
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ; //mole4
assign area[5] = ((x_cnt>(Hsync_Blank-2)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ;//mole5
assign area[6] = ((x_cnt>(Hsync_Blank-2+400)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-200))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;    //mole6
assign area[7] = ((x_cnt>(Hsync_Blank-2+200)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0; //mole7
assign area[8] = ((x_cnt>(Hsync_Blank-2)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;//mole8
assign area[9] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-400))
                                ))    ? 1'b1 : 1'b0 ;//area 9 Time
assign area[10] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
```

```
                                (y_cnt>(Vertical_Back_Porch-1+80))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0 ;//area 10 life
assign area[11] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-240))
                                ))    ? 1'b1 : 1'b0 ;//area 11 score
assign area[12] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+240))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ;//area 12 start/pause
assign area[13] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-80))
                                ))    ? 1'b1 : 1'b0 ;//area 13 exit
assign area[14] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+400))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;//area 14 label
assign area[15] = 1'b0;
//area 15 is the logo of the team defalt black

//////////////////////Timmer//////////////////////////
No_GEN LED0(
                        .iclk(clk_ltm),
```

```
                .iReset_n(reset_n),
                .ixcount(x_cnt),
                .iycount(y_cnt),
                .iled_pos_x(Hsync_Blank-1+600),
                .iled_pos_y(Vertical_Back_Porch),
                .inumber(num_input_s),
                .oOVERRIDE(oOVERRIDE[0]),
                .oR(LED0_R),
                .oG(LED0_G),
                .oB(LED0_B)
);

No_GEN LED1(
                .iclk(clk_ltm),
                .iReset_n(reset_n),
                .ixcount(x_cnt),
                .iycount(y_cnt),
                .iled_pos_x(Hsync_Blank-1+640),
                .iled_pos_y(Vertical_Back_Porch),
                .inumber(num_input_ts),
                .oOVERRIDE(oOVERRIDE[1]),
                .oR(LED1_R),
                .oG(LED1_G),
                .oB(LED1_B)
);

No_GEN LED2(
                .iclk(clk_ltm),
                .iReset_n(reset_n),
```

```
                        .ixcount(x_cnt),
                        .iycount(y_cnt),
                        .iled_pos_x(Hsync_Blank-1+680),
                        .iled_pos_y(Vertical_Back_Porch),
                        .inumber(num_input_min),
                        .oOVERRIDE(oOVERRIDE[2]),
                        .oR(LED2_R),
                        .oG(LED2_G),
                        .oB(LED2_B)
);

assign flag1 = (oCNT == 26'd24999999) ? 1'b1 : 1'b0;
assign flag2 = (inumber_s == 0) ? 1'b1 : 1'b0;
assign num_input_s = inumber_s;
assign num_input_ts = inumber_ts;
assign num_input_min = inumber_min;
assign oCNT = counter;

always@(posedge clk_ltm or negedge reset_n)
begin
    if(!reset_n)
    begin
    inumber_min <= 4'd2;
    inumber_ts <= 4'd0;
    inumber_s <= 4'd0;
    counter <= 26'b0;
    end
    else if(flag1)
        begin
```

```
if(flag2)
    begin
        if(inumber_ts != 0)
            begin
                inumber_ts <= inumber_ts - 1;
                inumber_s <= 4'd9;
                counter <= 26'b0;
            end
        else
            begin
                if(inumber_min != 0)
                    begin
                        inumber_min <= inumber_min -1;
                        inumber_ts    <= 4'd5;
                        inumber_s     <= 4'd9;
                        counter <= 26'b0;
                    end
                else
                    begin
                        inumber_min <= 4'd2;
                        inumber_ts <= 4'd0;
                        inumber_s <= 4'd0;
                        counter <= 26'b0;
                    end
            end
    end
else
begin
inumber_s <= inumber_s-1;
```

```
                counter <= 26'b0;
                end
            end
        else
            counter <= counter +1;
end

assign oVGA_ADDR = area[0] ? area_counter[0] : 18'bZ;
assign oVGA_OE_N = area[0] ? 1'b1 : 1'b0;
assign oVGA_WE_N = 0;

///////////////Test --Allocate color to each block//////////////////////
always@(posedge clk_ltm or negedge reset_n)
    begin
    if (!reset_n)
        begin
        red_1 <= 8'd255;
        green_1 <= 8'd255;
        blue_1 <= 8'd255;
        end
    else if (!(oOVERRIDE[2] || oOVERRIDE[1] || oOVERRIDE[0]))
        case(area)
            16'b0000000000000001:
            begin
            if(area_counter[0] == 15'd31999)
                area_counter[0] <= 0;
            else
                area_counter[0] <= area_counter[0] +1;
            red_1 <= (iVGA_DQ[15:11]<<3);
```

```
                green_1 <= (iVGA_DQ[10:5]<<2);
                blue_1 <= (iVGA_DQ[4:0]<<3);
                end//area 1
                16'b0000000000000010: begin red_1 <= 8'd255;green_1 <= 8'd128;blue_1 <=
8'd0;end
                16'b0000000000000100: begin red_1 <= 8'd255;green_1 <= 8'd255;blue_1 <=
8'd0;end
               16'b0000000000001000: begin red_1 <= 8'd128;green_1 <= 8'd255;blue_1 <= 8'd0;
end
                16'b0000000000010000: begin red_1 <= 8'd0;green_1 <= 8'd255;blue_1 <=
8'd0;end
                16'b0000000000100000: begin red_1 <= 8'd0;green_1 <= 8'd255;blue_1 <=
8'd128;end
                16'b0000000001000000: begin red_1 <= 8'd0;green_1 <= 8'd255;blue_1 <=
8'd255;end
                16'b0000000010000000: begin red_1 <= 8'd0;green_1 <= 8'd128;blue_1 <=
8'd255;end
                16'b0000000100000000: begin red_1 <= 8'd0;green_1 <= 8'd0;blue_1 <=
8'd255;end
                16'b0000001000000000: begin red_1 <= 8'd127;green_1 <= 8'd0;blue_1 <=
8'd255;end
                16'b0000010000000000: begin red_1 <= 8'd255;green_1 <= 8'd0;blue_1 <=
8'd255;end
                16'b0000100000000000: begin red_1 <= 8'd255;green_1 <= 8'd0;blue_1 <=
8'd127;end
                16'b0001000000000000: begin red_1 <= 8'd128;green_1 <= 8'd128;blue_1 <=
8'd128;end
                16'b0010000000000000: begin red_1 <= 8'd0;green_1 <= 8'd0;blue_1 <= 8'd0;end
                16'b0100000000000000: begin red_1 <= 8'd255;green_1 <= 8'd255;blue_1 <=
8'd255;end
                16'b1000000000000000: begin red_1 <= 8'd255;green_1 <= 8'd204;blue_1 <=
8'd204;end
                default: begin red_1 <= 8'd255;green_1 <= 8'd255;blue_1 <= 8'd255; end
            endcase
        else
            begin
```

```
                case(oOVERRIDE)
                    3'b001:
                        begin
                            red_1 <= LED0_R;
                            green_1 <= LED0_G;
                            blue_1 <= LED0_B;
                        end
                    3'b010:
                        begin
                            red_1 <= LED1_R;
                            green_1 <= LED1_G;
                            blue_1 <= LED1_B;
                        end
                    3'b100:
                        begin
                            red_1 <= LED2_R;
                            green_1 <= LED2_G;
                            blue_1 <= LED2_B;
                        end
                endcase
            end
        end

assign mred = red_1;
assign mgreen = green_1;
assign mblue    = blue_1;


always@(posedge clk_ltm or negedge reset_n)
```

```
    begin
        if (!reset_n)
            begin
                oHD <= 1'd0;
                oVD <= 1'd0;
                oDEN <= 1'd0;
                oLCD_R <= 8'd0;
                oLCD_G <= 8'd0;
                oLCD_B <= 8'd0;
            end
        else
            begin
                oHD <= mhd;
                oVD <= mvd;
                oDEN <= display_area;
                oLCD_R <= mred;
                oLCD_G <= mgreen;
                oLCD_B <= mblue;
            end
    end

endmodule




//////////////////////Number Display Module//////////////////////////
module No_GEN(
                iclk,
                iReset_n,
```

```
                ixcount,
                iycount,
                iled_pos_x,
                iled_pos_y,
                inumber,
                oOVERRIDE,
                oR,
                oG,
                oB
                );

//Port IO
input iclk;
input iReset_n;
input [3:0]inumber;
input [10:0] ixcount;
input [9:0]    iycount;
input [10:0] iled_pos_x;
input [10:0] iled_pos_y;
output oOVERRIDE;
output [7:0] oR;
output [7:0] oG;
output [7:0] oB;

//Wire & Reg
reg [7:0]    oR;
reg [7:0]    oG;
reg [7:0]    oB;
reg oOVERRIDE;
```

```
wire display_area;

wire number_area;

wire a;

wire b;

wire c;

wire d;

wire e;

wire f;

wire g;

wire a_en;

wire b_en;

wire c_en;

wire d_en;

wire e_en;

wire f_en;

wire g_en;


//Structure

assign display_area = ((ixcount > (iled_pos_x -1))
                                        && (ixcount < (iled_pos_x+ 60) )
                                        && (iycount > (iled_pos_y -1))
                                        && (iycount < (iled_pos_y +80))) ? 1'b1:1'b0;


assign number_area = ((ixcount > (iled_pos_x +4))
                                        && (ixcount < (iled_pos_x+ 35) )
                                        && (iycount > (iled_pos_y + 9))
                                        && (iycount < (iled_pos_y +70))) ? 1'b1:1'b0;


assign a = ((ixcount > (iled_pos_x + 4))
```

```
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 9))
                    &&(iycount < (iled_pos_y + 18))) ? 1'b1 : 1'b0;
assign b =      ((ixcount > (iled_pos_x + 4))
                    && (ixcount < (iled_pos_x + 13))
                    &&(iycount > (iled_pos_y + 9))
                    &&(iycount < (iled_pos_y + 40))) ? 1'b1 : 1'b0;
assign c =   ((ixcount > (iled_pos_x + 4))
                    && (ixcount < (iled_pos_x + 13))
                    &&(iycount > (iled_pos_y + 39))
                    &&(iycount < (iled_pos_y + 70))) ? 1'b1 : 1'b0;
assign d =      ((ixcount > (iled_pos_x + 4))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 61))
                    &&(iycount < (iled_pos_y + 70))) ? 1'b1 : 1'b0;
assign e =      ((ixcount > (iled_pos_x + 26))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 39))
                    &&(iycount < (iled_pos_y + 70))) ? 1'b1 : 1'b0;
assign f =      ((ixcount > (iled_pos_x + 26))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 9))
                    &&(iycount < (iled_pos_y + 40))) ? 1'b1 : 1'b0;
assign g =      ((ixcount > (iled_pos_x + 4))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 35))
                    &&(iycount < (iled_pos_y + 44))) ? 1'b1 : 1'b0;

//assign a_en = ((inumber != 1) && (inumber != 4)) ? 1'b1 : 1'b0;
```

```
//assign b_en = ((inumber != 5) && (inumber != 6)) ? 1'b1 : 1'b0;

//assign c_en = ((inumber != 2)) ? 1'b1 : 1'b0;

//assign d_en = ((inumber != 1) && (inumber != 4) && (inumber != 7)) ? 1'b1 : 1'b0;

//assign e_en = ((inumber == 2) || (inumber == 6) || (inumber == 8) || (inumber == 0)) ? 1'b1:
1'b0;

//assign f_en = ((inumber == 4) || (inumber == 5) || (inumber == 6) || (inumber == 8)

//                              || (inumber == 9) || (inumber == 0)) ? 1'b1: 1'b0;

//assign g_en = ((inumber != 1) && (inumber != 7) && (inumber != 0)) ? 1'b1 : 1'b0;


always @(posedge iclk or negedge iReset_n)

begin

    if(!iReset_n)

            oOVERRIDE <= 1'b0;

    else

    case(inumber)

        0: oOVERRIDE <= (a||b||c||d||e||f);

        1: oOVERRIDE <= (b||c);

        2: oOVERRIDE <= (a||b||g||e||d);

        3: oOVERRIDE <= (a||b||c||d||g);

        4: oOVERRIDE <= (b||c||f||g);

        5: oOVERRIDE <= (a||c||d||f||g);

        6: oOVERRIDE <= (a||c||d||e||f||g);

        7: oOVERRIDE <= (a||b||c);

        8: oOVERRIDE <= (a||b||c||d||e||f||g);

        9: oOVERRIDE <= (a||b||c||d||f||g);

        default : oOVERRIDE <= 1'b0;

    endcase

    oR <= 8'd255;

    oG <= 8'd255;

    oB <= 8'd255;
```

```
end

endmodule
```

letter.v

```
module letter(
                        iclk,
                        ireset_n,
                        ix_cnt,
                        iy_cnt,
                        iletter_pos_x,
                        iletter_pos_y,
                        iletter,
                        oOVERRIDE,
                        oADDR
                        );

//=============================================
//Parameter================================
//=============================================
parameter A_base = 62400;
parameter B_base = 63424;
parameter C_base = 64448;
parameter D_base = 65472;
parameter E_base = 66496;
parameter F_base = 67520;
parameter G_base = 68544;
parameter H_base = 69568;
parameter I_base = 70592;
parameter J_base = 71616;
parameter K_base = 72640;
parameter L_base = 73664;
parameter M_base = 74688;
parameter N_base = 75712;
parameter O_base = 76736;
parameter P_base = 77760;
parameter Q_base = 78784;
parameter R_base = 79808;
parameter S_base = 80832;
parameter T_base = 81856;
parameter U_base = 82880;
parameter V_base = 83904;
parameter W_base = 84928;
parameter X_base = 85952;
parameter Y_base = 86976;
parameter Z_base = 88000;

parameter A = 5'd1;
parameter B = 5'd2;
parameter C = 5'd3;
parameter D = 5'd4;
parameter E = 5'd5;
parameter F = 5'd6;
```

```
parameter G = 5'd7;
parameter H = 5'd8;
parameter I = 5'd9;
parameter J = 5'd10;
parameter K = 5'd11;
parameter L = 5'd12;
parameter M = 5'd13;
parameter N = 5'd14;
parameter O = 5'd15;
parameter P = 5'd16;
parameter Q = 5'd17;
parameter R = 5'd18;
parameter S = 5'd19;
parameter T = 5'd20;
parameter U = 5'd21;
parameter V = 5'd22;
parameter W = 5'd23;
parameter X = 5'd24;
parameter Y = 5'd25;
parameter Z = 5'd26;

//=============================================
//I/O DECALARATION=============================
//=============================================
input iclk;
input ireset_n;
input [10:0] ix_cnt;
input [9:0] iy_cnt;
input [10:0]iletter_pos_x;
input [9:0]iletter_pos_y;
input [4:0] iletter;
output oOVERRIDE;
output [17:0] oADDR;


//=============================================
//WIRE & REG==================================
//=============================================
reg    oOVERRIDE;
reg [17:0] oADDR;
reg [4:0] counter;
reg [17:0] address;
wire display_area;
reg white[1024];


//=============================================
//Structured logic============================
//=============================================
assign display_area = ((ix_cnt > iletter_pos_x-1) && (ix_cnt < iletter_pos_x + 32)
                                        && (iy_cnt > iletter_pos_y -1) && (iy_cnt < iletter_pos_y
+ 32)) ? 1'b1:1'b0;


always@(negedge ireset_n)
begin
      if(!ireset_n)
            begin
```

```
case(iletter)
A: begin
        white[ 0 ]<=0;
        white[ 1 ]<=0;
        white[ 2 ]<=0;
        white[ 3 ]<=0;
        white[ 4 ]<=0;
        white[ 5 ]<=0;
        white[ 6 ]<=0;
        white[ 7 ]<=0;
        white[ 8 ]<=0;
        white[ 9 ]<=0;
        white[ 10 ]<=0;
        white[ 11 ]<=0;
        white[ 12 ]<=0;
        white[ 13 ]<=0;
        white[ 14 ]<=0;
        white[ 15 ]<=0;
        white[ 16 ]<=0;
        white[ 17 ]<=0;
        white[ 18 ]<=0;
        white[ 19 ]<=0;
        white[ 20 ]<=0;
        white[ 21 ]<=0;
        white[ 22 ]<=0;
        white[ 23 ]<=0;
        white[ 24 ]<=0;
        white[ 25 ]<=0;
        white[ 26 ]<=0;
        white[ 27 ]<=0;
        white[ 28 ]<=0;
        white[ 29 ]<=0;
        white[ 30 ]<=0;
        white[ 31 ]<=0;
        white[ 32 ]<=0;
        white[ 33 ]<=0;
        white[ 34 ]<=0;
        white[ 35 ]<=0;
        white[ 36 ]<=0;
        white[ 37 ]<=0;
        white[ 38 ]<=0;
        white[ 39 ]<=0;
        white[ 40 ]<=0;
        white[ 41 ]<=0;
        white[ 42 ]<=0;
        white[ 43 ]<=0;
        white[ 44 ]<=0;
        white[ 45 ]<=0;
        white[ 46 ]<=0;
        white[ 47 ]<=0;
        white[ 48 ]<=0;
        white[ 49 ]<=0;
        white[ 50 ]<=0;
        white[ 51 ]<=0;
        white[ 52 ]<=0;
        white[ 53 ]<=0;
        white[ 54 ]<=0;
        white[ 55 ]<=0;
```

```
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
```

```
white[ 114 ]<=1;
white[ 115 ]<=1;
white[ 116 ]<=1;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=0;
white[ 168 ]<=0;
white[ 169 ]<=0;
white[ 170 ]<=0;
white[ 171 ]<=1;
```

```
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=0;
white[ 198 ]<=0;
white[ 199 ]<=0;
white[ 200 ]<=0;
white[ 201 ]<=0;
white[ 202 ]<=0;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=0;
white[ 216 ]<=0;
white[ 217 ]<=0;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=0;
```

```
white[ 230 ]<=0;
white[ 231 ]<=0;
white[ 232 ]<=0;
white[ 233 ]<=0;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=0;
white[ 249 ]<=0;
white[ 250 ]<=0;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=0;
white[ 261 ]<=0;
white[ 262 ]<=0;
white[ 263 ]<=0;
white[ 264 ]<=0;
white[ 265 ]<=0;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=0;
white[ 281 ]<=0;
white[ 282 ]<=0;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
```

```
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=0;
white[ 294 ]<=0;
white[ 295 ]<=0;
white[ 296 ]<=0;
white[ 297 ]<=0;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=0;
white[ 313 ]<=0;
white[ 314 ]<=0;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=0;
white[ 325 ]<=0;
white[ 326 ]<=0;
white[ 327 ]<=0;
white[ 328 ]<=0;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=0;
```

```
white[ 346 ]<=0;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=0;
white[ 358 ]<=0;
white[ 359 ]<=0;
white[ 360 ]<=0;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=0;
white[ 378 ]<=0;
white[ 379 ]<=0;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
white[ 389 ]<=0;
white[ 390 ]<=0;
white[ 391 ]<=0;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
```

```
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=0;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=0;
white[ 423 ]<=0;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=0;
white[ 454 ]<=0;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
```

```
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=0;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=1;
white[ 519 ]<=1;
```

```
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
```

```
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=0;
white[ 581 ]<=0;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
```

```
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
```

```
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
```

```
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=1;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=0;
white[ 800 ]<=1;
white[ 801 ]<=1;
white[ 802 ]<=1;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
```

```
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=1;
white[ 832 ]<=1;
white[ 833 ]<=1;
white[ 834 ]<=1;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=0;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=1;
white[ 863 ]<=1;
white[ 864 ]<=1;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
```

```
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=0;
white[ 881 ]<=0;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=1;
white[ 895 ]<=1;
white[ 896 ]<=1;
white[ 897 ]<=1;
white[ 898 ]<=1;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=0;
white[ 913 ]<=0;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
```

```
white[ 926 ]<=1;
white[ 927 ]<=1;
white[ 928 ]<=1;
white[ 929 ]<=1;
white[ 930 ]<=1;
white[ 931 ]<=1;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=0;
white[ 943 ]<=0;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=1;
white[ 955 ]<=1;
white[ 956 ]<=1;
white[ 957 ]<=1;
white[ 958 ]<=1;
white[ 959 ]<=1;
white[ 960 ]<=0;
white[ 961 ]<=1;
white[ 962 ]<=1;
white[ 963 ]<=1;
white[ 964 ]<=1;
white[ 965 ]<=1;
white[ 966 ]<=1;
white[ 967 ]<=1;
white[ 968 ]<=1;
white[ 969 ]<=1;
white[ 970 ]<=1;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=1;
white[ 981 ]<=1;
white[ 982 ]<=1;
white[ 983 ]<=1;
```

```
        white[ 984 ]<=1;
        white[ 985 ]<=1;
        white[ 986 ]<=1;
        white[ 987 ]<=1;
        white[ 988 ]<=1;
        white[ 989 ]<=1;
        white[ 990 ]<=1;
        white[ 991 ]<=0;
        white[ 992 ]<=0;
        white[ 993 ]<=0;
        white[ 994 ]<=0;
        white[ 995 ]<=0;
        white[ 996 ]<=0;
        white[ 997 ]<=0;
        white[ 998 ]<=0;
        white[ 999 ]<=0;
        white[ 1000 ]<=0;
        white[ 1001 ]<=0;
        white[ 1002 ]<=0;
        white[ 1003 ]<=0;
        white[ 1004 ]<=0;
        white[ 1005 ]<=0;
        white[ 1006 ]<=0;
        white[ 1007 ]<=0;
        white[ 1008 ]<=0;
        white[ 1009 ]<=0;
        white[ 1010 ]<=0;
        white[ 1011 ]<=0;
        white[ 1012 ]<=0;
        white[ 1013 ]<=0;
        white[ 1014 ]<=0;
        white[ 1015 ]<=0;
        white[ 1016 ]<=0;
        white[ 1017 ]<=0;
        white[ 1018 ]<=0;
        white[ 1019 ]<=0;
        white[ 1020 ]<=0;
        white[ 1021 ]<=0;
        white[ 1022 ]<=0;
        white[ 1023 ]<=0;
        end

    B: begin
        white[ 0 ]<=0;
        white[ 1 ]<=0;
        white[ 2 ]<=0;
        white[ 3 ]<=0;
        white[ 4 ]<=0;
        white[ 5 ]<=0;
        white[ 6 ]<=0;
        white[ 7 ]<=0;
        white[ 8 ]<=0;
        white[ 9 ]<=0;
        white[ 10 ]<=0;
        white[ 11 ]<=0;
        white[ 12 ]<=0;
        white[ 13 ]<=0;
        white[ 14 ]<=0;
```

```
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
```

```
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=1;
white[ 101 ]<=1;
white[ 102 ]<=1;
white[ 103 ]<=1;
white[ 104 ]<=1;
white[ 105 ]<=1;
white[ 106 ]<=1;
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=1;
```

```
white[ 131 ]<=1;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=1;
white[ 162 ]<=1;
white[ 163 ]<=1;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=0;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
```

```
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=1;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=0;
white[ 216 ]<=0;
white[ 217 ]<=0;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=1;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
```

```
white[ 247 ]<=1;
white[ 248 ]<=0;
white[ 249 ]<=0;
white[ 250 ]<=0;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=1;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=0;
white[ 282 ]<=0;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
```

```
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=0;
white[ 314 ]<=0;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=0;
white[ 346 ]<=0;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
```

```
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=0;
white[ 378 ]<=0;
white[ 379 ]<=0;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=0;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=1;
white[ 420 ]<=1;
```

```
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=0;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
```

```
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
```

```
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
```

```
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
```

```
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
```

```
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
```

```
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=1;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
```

```
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=1;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
```

```
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=0;
white[ 893 ]<=1;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=1;
white[ 898 ]<=1;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=1;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
```

```
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=1;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=1;
white[ 965 ]<=1;
white[ 966 ]<=1;
white[ 967 ]<=1;
white[ 968 ]<=1;
white[ 969 ]<=1;
white[ 970 ]<=1;
white[ 971 ]<=1;
white[ 972 ]<=1;
white[ 973 ]<=1;
white[ 974 ]<=1;
white[ 975 ]<=1;
white[ 976 ]<=1;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=1;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
```

```
        white[ 1001 ]<=0;
        white[ 1002 ]<=0;
        white[ 1003 ]<=0;
        white[ 1004 ]<=0;
        white[ 1005 ]<=0;
        white[ 1006 ]<=0;
        white[ 1007 ]<=0;
        white[ 1008 ]<=0;
        white[ 1009 ]<=0;
        white[ 1010 ]<=0;
        white[ 1011 ]<=0;
        white[ 1012 ]<=0;
        white[ 1013 ]<=0;
        white[ 1014 ]<=0;
        white[ 1015 ]<=0;
        white[ 1016 ]<=0;
        white[ 1017 ]<=0;
        white[ 1018 ]<=0;
        white[ 1019 ]<=0;
        white[ 1020 ]<=0;
        white[ 1021 ]<=0;
        white[ 1022 ]<=0;
        white[ 1023 ]<=0;
        end

    C: begin
        white[ 0 ]<=0;
        white[ 1 ]<=0;
        white[ 2 ]<=0;
        white[ 3 ]<=0;
        white[ 4 ]<=0;
        white[ 5 ]<=0;
        white[ 6 ]<=0;
        white[ 7 ]<=0;
        white[ 8 ]<=0;
        white[ 9 ]<=0;
        white[ 10 ]<=0;
        white[ 11 ]<=0;
        white[ 12 ]<=0;
        white[ 13 ]<=0;
        white[ 14 ]<=0;
        white[ 15 ]<=0;
        white[ 16 ]<=0;
        white[ 17 ]<=0;
        white[ 18 ]<=0;
        white[ 19 ]<=0;
        white[ 20 ]<=0;
        white[ 21 ]<=0;
        white[ 22 ]<=0;
        white[ 23 ]<=0;
        white[ 24 ]<=0;
        white[ 25 ]<=0;
        white[ 26 ]<=0;
        white[ 27 ]<=0;
        white[ 28 ]<=0;
        white[ 29 ]<=0;
        white[ 30 ]<=0;
        white[ 31 ]<=0;
```

```
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=1;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
```

```
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=1;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
```

```
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=0;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=0;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
```

```
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=0;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=0;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=0;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
```

```
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
```

```
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=0;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
```

```
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=0;
white[ 400 ]<=0;
white[ 401 ]<=0;
white[ 402 ]<=0;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=0;
white[ 431 ]<=0;
white[ 432 ]<=0;
white[ 433 ]<=0;
white[ 434 ]<=0;
white[ 435 ]<=0;
white[ 436 ]<=1;
white[ 437 ]<=1;
```

```
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=0;
white[ 462 ]<=0;
white[ 463 ]<=0;
white[ 464 ]<=0;
white[ 465 ]<=0;
white[ 466 ]<=0;
white[ 467 ]<=0;
white[ 468 ]<=0;
white[ 469 ]<=0;
white[ 470 ]<=0;
white[ 471 ]<=0;
white[ 472 ]<=0;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=0;
white[ 494 ]<=0;
white[ 495 ]<=0;
```

```
white[ 496 ]<=0;
white[ 497 ]<=0;
white[ 498 ]<=0;
white[ 499 ]<=0;
white[ 500 ]<=0;
white[ 501 ]<=0;
white[ 502 ]<=0;
white[ 503 ]<=0;
white[ 504 ]<=0;
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=0;
white[ 526 ]<=0;
white[ 527 ]<=0;
white[ 528 ]<=0;
white[ 529 ]<=0;
white[ 530 ]<=0;
white[ 531 ]<=0;
white[ 532 ]<=0;
white[ 533 ]<=0;
white[ 534 ]<=0;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
```

```
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=0;
white[ 558 ]<=0;
white[ 559 ]<=0;
white[ 560 ]<=0;
white[ 561 ]<=0;
white[ 562 ]<=0;
white[ 563 ]<=0;
white[ 564 ]<=0;
white[ 565 ]<=0;
white[ 566 ]<=0;
white[ 567 ]<=0;
white[ 568 ]<=0;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=0;
white[ 590 ]<=0;
white[ 591 ]<=0;
white[ 592 ]<=0;
white[ 593 ]<=0;
white[ 594 ]<=0;
white[ 595 ]<=0;
white[ 596 ]<=0;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=1;
```

```
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=0;
white[ 622 ]<=0;
white[ 623 ]<=0;
white[ 624 ]<=0;
white[ 625 ]<=0;
white[ 626 ]<=0;
white[ 627 ]<=0;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=0;
white[ 655 ]<=0;
white[ 656 ]<=0;
white[ 657 ]<=0;
white[ 658 ]<=0;
white[ 659 ]<=0;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
```

```
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=0;
white[ 688 ]<=0;
white[ 689 ]<=0;
white[ 690 ]<=0;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=0;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
```

```
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
```

```
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=0;
white[ 805 ]<=0;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=0;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=0;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
```

```
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=0;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=0;
white[ 870 ]<=0;
white[ 871 ]<=0;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
```

```
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=0;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
```

```
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=1;
white[ 976 ]<=1;
white[ 977 ]<=1;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
```

```
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

D:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
```

```
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=1;
white[ 67 ]<=1;
white[ 68 ]<=1;
white[ 69 ]<=1;
white[ 70 ]<=1;
white[ 71 ]<=1;
white[ 72 ]<=1;
white[ 73 ]<=1;
white[ 74 ]<=1;
white[ 75 ]<=1;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=1;
white[ 99 ]<=1;
white[ 100 ]<=1;
white[ 101 ]<=1;
white[ 102 ]<=1;
white[ 103 ]<=1;
white[ 104 ]<=1;
white[ 105 ]<=1;
white[ 106 ]<=1;
```

```
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=1;
white[ 130 ]<=1;
white[ 131 ]<=1;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=1;
white[ 163 ]<=1;
white[ 164 ]<=1;
```

```
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=0;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=0;
white[ 217 ]<=0;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
```

```
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=0;
white[ 250 ]<=0;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
```

```
white[ 281 ]<=1;
white[ 282 ]<=0;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=0;
white[ 335 ]<=0;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
```

```
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=0;
white[ 367 ]<=0;
white[ 368 ]<=0;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
```

```
white[ 397 ]<=1;
white[ 398 ]<=0;
white[ 399 ]<=0;
white[ 400 ]<=0;
white[ 401 ]<=0;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=0;
white[ 431 ]<=0;
white[ 432 ]<=0;
white[ 433 ]<=0;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
```

```
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=0;
white[ 463 ]<=0;
white[ 464 ]<=0;
white[ 465 ]<=0;
white[ 466 ]<=0;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=1;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=0;
white[ 494 ]<=0;
white[ 495 ]<=0;
white[ 496 ]<=0;
white[ 497 ]<=0;
white[ 498 ]<=0;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
```

```
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=0;
white[ 526 ]<=0;
white[ 527 ]<=0;
white[ 528 ]<=0;
white[ 529 ]<=0;
white[ 530 ]<=0;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=1;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=0;
white[ 558 ]<=0;
white[ 559 ]<=0;
white[ 560 ]<=0;
white[ 561 ]<=0;
white[ 562 ]<=0;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
```

```
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=0;
white[ 590 ]<=0;
white[ 591 ]<=0;
white[ 592 ]<=0;
white[ 593 ]<=0;
white[ 594 ]<=0;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=1;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=0;
white[ 622 ]<=0;
white[ 623 ]<=0;
white[ 624 ]<=0;
white[ 625 ]<=0;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
```

```
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=0;
white[ 654 ]<=0;
white[ 655 ]<=0;
white[ 656 ]<=0;
white[ 657 ]<=0;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=1;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=0;
white[ 686 ]<=0;
```

```
white[ 687 ]<=0;
white[ 688 ]<=0;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=1;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=0;
white[ 718 ]<=0;
white[ 719 ]<=0;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=1;
white[ 738 ]<=1;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
```

```
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=1;
white[ 770 ]<=1;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=1;
white[ 801 ]<=1;
white[ 802 ]<=1;
```

```
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=1;
white[ 833 ]<=1;
white[ 834 ]<=1;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=0;
white[ 860 ]<=0;
```

```
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=1;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
```

```
white[ 919 ]<=0;
white[ 920 ]<=0;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=1;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
```

```
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

E:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
```

```
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
```

```
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=1;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=1;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=1;
white[ 99 ]<=1;
white[ 100 ]<=1;
white[ 101 ]<=1;
white[ 102 ]<=1;
white[ 103 ]<=1;
white[ 104 ]<=1;
white[ 105 ]<=1;
white[ 106 ]<=1;
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=1;
white[ 116 ]<=1;
white[ 117 ]<=1;
white[ 118 ]<=1;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
```

```
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=1;
white[ 131 ]<=1;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=1;
white[ 162 ]<=1;
white[ 163 ]<=1;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
```

```
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=1;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=1;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
```

```
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=1;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
```

```
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=1;
```

```
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=0;
white[ 404 ]<=0;
white[ 405 ]<=0;
white[ 406 ]<=0;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=0;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
```

```
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=0;
white[ 437 ]<=0;
white[ 438 ]<=0;
white[ 439 ]<=0;
white[ 440 ]<=0;
white[ 441 ]<=0;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=0;
white[ 470 ]<=0;
white[ 471 ]<=0;
```

```
white[ 472 ]<=0;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=0;
white[ 503 ]<=0;
white[ 504 ]<=0;
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
```

```
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=0;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
```

```
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=0;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=0;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=0;
white[ 638 ]<=1;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
```

```
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=0;
white[ 663 ]<=0;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=1;
white[ 703 ]<=0;
```

```
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=1;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
```

```
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=1;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=1;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=1;
white[ 802 ]<=1;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
```

```
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=1;
white[ 834 ]<=1;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
```

```
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=1;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
```

```
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=0;
white[ 940 ]<=0;
white[ 941 ]<=0;
white[ 942 ]<=0;
white[ 943 ]<=0;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=0;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
```

```
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=1;
white[ 1018 ]<=0;
white[ 1019 ]<=1;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

F:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
```

```
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
```

```
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=1;
white[ 87 ]<=1;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=1;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=1;
white[ 101 ]<=1;
white[ 102 ]<=1;
white[ 103 ]<=1;
white[ 104 ]<=1;
white[ 105 ]<=1;
white[ 106 ]<=1;
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=1;
white[ 116 ]<=1;
white[ 117 ]<=1;
white[ 118 ]<=1;
white[ 119 ]<=1;
white[ 120 ]<=1;
white[ 121 ]<=1;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=1;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
```

```
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=1;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=1;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=1;
white[ 188 ]<=1;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
```

```
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=1;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=1;
white[ 255 ]<=0;
white[ 256 ]<=0;
```

```
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=1;
white[ 287 ]<=1;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
```

```
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=1;
white[ 318 ]<=1;
white[ 319 ]<=1;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=1;
white[ 351 ]<=1;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
```

```
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=1;
white[ 383 ]<=1;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=0;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=1;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
```

```
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=0;
white[ 441 ]<=0;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=0;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=1;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
```

```
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
```

```
white[ 547 ]<=0;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=0;
white[ 604 ]<=0;
```

```
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=0;
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
```

```
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=0;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=0;
white[ 697 ]<=0;
white[ 698 ]<=0;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
```

```
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=0;
white[ 724 ]<=0;
white[ 725 ]<=0;
white[ 726 ]<=0;
white[ 727 ]<=0;
white[ 728 ]<=0;
white[ 729 ]<=0;
white[ 730 ]<=0;
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=0;
white[ 755 ]<=0;
white[ 756 ]<=0;
white[ 757 ]<=0;
white[ 758 ]<=0;
white[ 759 ]<=0;
white[ 760 ]<=0;
white[ 761 ]<=0;
white[ 762 ]<=0;
white[ 763 ]<=0;
white[ 764 ]<=0;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
```

```
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=0;
white[ 787 ]<=0;
white[ 788 ]<=0;
white[ 789 ]<=0;
white[ 790 ]<=0;
white[ 791 ]<=0;
white[ 792 ]<=0;
white[ 793 ]<=0;
white[ 794 ]<=0;
white[ 795 ]<=0;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=0;
white[ 820 ]<=0;
white[ 821 ]<=0;
white[ 822 ]<=0;
white[ 823 ]<=0;
white[ 824 ]<=0;
white[ 825 ]<=0;
white[ 826 ]<=0;
white[ 827 ]<=0;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=1;
white[ 836 ]<=1;
```

```
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=0;
white[ 853 ]<=0;
white[ 854 ]<=0;
white[ 855 ]<=0;
white[ 856 ]<=0;
white[ 857 ]<=0;
white[ 858 ]<=0;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=0;
white[ 885 ]<=0;
white[ 886 ]<=0;
white[ 887 ]<=0;
white[ 888 ]<=0;
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
```

```
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=0;
white[ 917 ]<=0;
white[ 918 ]<=0;
white[ 919 ]<=0;
white[ 920 ]<=0;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=0;
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
```

```
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=1;
white[ 966 ]<=1;
white[ 967 ]<=1;
white[ 968 ]<=1;
white[ 969 ]<=1;
white[ 970 ]<=1;
white[ 971 ]<=1;
white[ 972 ]<=1;
white[ 973 ]<=1;
white[ 974 ]<=1;
white[ 975 ]<=1;
white[ 976 ]<=1;
white[ 977 ]<=1;
white[ 978 ]<=1;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=1;
white[ 1001 ]<=1;
white[ 1002 ]<=1;
white[ 1003 ]<=1;
white[ 1004 ]<=1;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
```

```
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

G:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
```

```
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
```

```
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
```

```
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=0;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
```

```
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
```

```
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=1;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
```

```
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=1;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=1;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
```

```
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=0;
white[ 404 ]<=1;
white[ 405 ]<=0;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=1;
white[ 418 ]<=1;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=0;
white[ 434 ]<=0;
white[ 435 ]<=0;
white[ 436 ]<=0;
white[ 437 ]<=0;
white[ 438 ]<=0;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
```

```
white[ 448 ]<=0;
white[ 449 ]<=1;
white[ 450 ]<=1;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=0;
white[ 465 ]<=0;
white[ 466 ]<=0;
white[ 467 ]<=0;
white[ 468 ]<=0;
white[ 469 ]<=0;
white[ 470 ]<=0;
white[ 471 ]<=0;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=1;
white[ 482 ]<=1;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=0;
white[ 496 ]<=0;
white[ 497 ]<=0;
white[ 498 ]<=0;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=0;
```

```
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=0;
white[ 510 ]<=1;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=1;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=0;
white[ 528 ]<=0;
white[ 529 ]<=0;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=1;
white[ 546 ]<=1;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=0;
white[ 560 ]<=0;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
```

```
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=1;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=0;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=1;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=1;
white[ 610 ]<=1;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
```

```
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=1;
white[ 640 ]<=0;
white[ 641 ]<=1;
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=0;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=1;
white[ 671 ]<=1;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
```

```
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=0;
white[ 691 ]<=0;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=1;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=1;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=1;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
```

```
white[ 738 ]<=0;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
```

```
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
```

```
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=1;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=0;
white[ 870 ]<=0;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=1;
white[ 895 ]<=1;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
```

```
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=1;
white[ 927 ]<=1;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=1;
white[ 955 ]<=1;
white[ 956 ]<=1;
white[ 957 ]<=1;
white[ 958 ]<=1;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
```

```
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=1;
white[ 975 ]<=1;
white[ 976 ]<=1;
white[ 977 ]<=1;
white[ 978 ]<=1;
white[ 979 ]<=1;
white[ 980 ]<=1;
white[ 981 ]<=1;
white[ 982 ]<=1;
white[ 983 ]<=1;
white[ 984 ]<=1;
white[ 985 ]<=1;
white[ 986 ]<=1;
white[ 987 ]<=1;
white[ 988 ]<=1;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

H:begin
white[ 0 ]<=0;
```

```
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
```

```
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=1;
white[ 89 ]<=1;
white[ 90 ]<=1;
white[ 91 ]<=1;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=1;
white[ 100 ]<=1;
white[ 101 ]<=1;
white[ 102 ]<=1;
white[ 103 ]<=1;
white[ 104 ]<=1;
white[ 105 ]<=1;
white[ 106 ]<=1;
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=1;
white[ 116 ]<=1;
```

```
white[ 117 ]<=1;
white[ 118 ]<=1;
white[ 119 ]<=1;
white[ 120 ]<=1;
white[ 121 ]<=1;
white[ 122 ]<=1;
white[ 123 ]<=1;
white[ 124 ]<=1;
white[ 125 ]<=1;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=1;
white[ 129 ]<=1;
white[ 130 ]<=1;
white[ 131 ]<=1;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=1;
white[ 155 ]<=1;
white[ 156 ]<=1;
white[ 157 ]<=1;
white[ 158 ]<=1;
white[ 159 ]<=0;
white[ 160 ]<=1;
white[ 161 ]<=1;
white[ 162 ]<=1;
white[ 163 ]<=1;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
```

```
white[ 175 ]<=0;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=1;
white[ 188 ]<=1;
white[ 189 ]<=1;
white[ 190 ]<=1;
white[ 191 ]<=0;
white[ 192 ]<=1;
white[ 193 ]<=1;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=1;
white[ 222 ]<=1;
white[ 223 ]<=1;
white[ 224 ]<=1;
white[ 225 ]<=1;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
```

```
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=1;
white[ 255 ]<=1;
white[ 256 ]<=1;
white[ 257 ]<=1;
white[ 258 ]<=1;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=1;
white[ 287 ]<=1;
white[ 288 ]<=0;
white[ 289 ]<=1;
white[ 290 ]<=1;
```

```
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=0;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=1;
white[ 318 ]<=1;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=1;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=0;
white[ 336 ]<=0;
white[ 337 ]<=0;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
```

```
white[ 349 ]<=1;
white[ 350 ]<=1;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=1;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=1;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=1;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
```

```
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=1;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=1;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=1;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=1;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
```

```
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=1;
white[ 478 ]<=1;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=1;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=1;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
```

```
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=1;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=1;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=1;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
```

```
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=1;
white[ 607 ]<=1;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=1;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
```

```
white[ 639 ]<=1;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=1;
white[ 671 ]<=1;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
```

```
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=1;
white[ 703 ]<=1;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=1;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=0;
white[ 722 ]<=0;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=1;
white[ 735 ]<=1;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=1;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=0;
white[ 753 ]<=0;
white[ 754 ]<=0;
```

```
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=1;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=1;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=0;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=1;
white[ 800 ]<=0;
white[ 801 ]<=1;
white[ 802 ]<=1;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
```

```
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=1;
white[ 832 ]<=0;
white[ 833 ]<=1;
white[ 834 ]<=1;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=1;
white[ 863 ]<=1;
white[ 864 ]<=1;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
```

```
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=1;
white[ 895 ]<=1;
white[ 896 ]<=0;
white[ 897 ]<=1;
white[ 898 ]<=1;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=1;
white[ 927 ]<=1;
white[ 928 ]<=0;
```

```
white[ 929 ]<=0;
white[ 930 ]<=1;
white[ 931 ]<=1;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=0;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=1;
white[ 955 ]<=1;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=1;
white[ 964 ]<=1;
white[ 965 ]<=1;
white[ 966 ]<=1;
white[ 967 ]<=1;
white[ 968 ]<=1;
white[ 969 ]<=1;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
```

```
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

I:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
```

```
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
```

```
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=1;
white[ 104 ]<=1;
white[ 105 ]<=1;
white[ 106 ]<=1;
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=1;
white[ 116 ]<=1;
white[ 117 ]<=1;
white[ 118 ]<=1;
white[ 119 ]<=1;
white[ 120 ]<=1;
white[ 121 ]<=1;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
```

```
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=1;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
```

```
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
```

```
white[ 250 ]<=1;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=0;
white[ 261 ]<=0;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=0;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=0;
white[ 294 ]<=0;
white[ 295 ]<=0;
white[ 296 ]<=0;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
```

```
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=0;
white[ 312 ]<=0;
white[ 313 ]<=0;
white[ 314 ]<=0;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=0;
white[ 325 ]<=0;
white[ 326 ]<=1;
white[ 327 ]<=0;
white[ 328 ]<=0;
white[ 329 ]<=0;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=0;
white[ 343 ]<=0;
white[ 344 ]<=0;
white[ 345 ]<=0;
white[ 346 ]<=0;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=1;
white[ 358 ]<=0;
white[ 359 ]<=0;
white[ 360 ]<=0;
white[ 361 ]<=0;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
```

```
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=0;
white[ 375 ]<=1;
white[ 376 ]<=0;
white[ 377 ]<=0;
white[ 378 ]<=0;
white[ 379 ]<=0;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
white[ 389 ]<=0;
white[ 390 ]<=0;
white[ 391 ]<=0;
white[ 392 ]<=0;
white[ 393 ]<=0;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=0;
white[ 407 ]<=0;
white[ 408 ]<=0;
white[ 409 ]<=0;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=0;
white[ 423 ]<=0;
```

```
white[ 424 ]<=0;
white[ 425 ]<=0;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=0;
white[ 439 ]<=0;
white[ 440 ]<=0;
white[ 441 ]<=0;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=0;
white[ 454 ]<=0;
white[ 455 ]<=0;
white[ 456 ]<=0;
white[ 457 ]<=0;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=0;
white[ 471 ]<=0;
white[ 472 ]<=0;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
```

```
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=0;
white[ 487 ]<=0;
white[ 488 ]<=0;
white[ 489 ]<=0;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=0;
white[ 503 ]<=0;
white[ 504 ]<=0;
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=0;
white[ 519 ]<=0;
white[ 520 ]<=0;
white[ 521 ]<=0;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=0;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
```

```
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=0;
white[ 551 ]<=0;
white[ 552 ]<=0;
white[ 553 ]<=0;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=0;
white[ 567 ]<=0;
white[ 568 ]<=0;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=0;
white[ 581 ]<=0;
white[ 582 ]<=0;
white[ 583 ]<=0;
white[ 584 ]<=0;
white[ 585 ]<=0;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
```

```
white[ 598 ]<=0;
white[ 599 ]<=0;
white[ 600 ]<=0;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=0;
white[ 614 ]<=0;
white[ 615 ]<=0;
white[ 616 ]<=0;
white[ 617 ]<=0;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=0;
white[ 631 ]<=0;
white[ 632 ]<=0;
white[ 633 ]<=0;
white[ 634 ]<=0;
white[ 635 ]<=0;
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=0;
white[ 645 ]<=0;
white[ 646 ]<=0;
white[ 647 ]<=0;
white[ 648 ]<=0;
white[ 649 ]<=0;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
```

```
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=0;
white[ 663 ]<=0;
white[ 664 ]<=0;
white[ 665 ]<=0;
white[ 666 ]<=0;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=0;
white[ 677 ]<=0;
white[ 678 ]<=0;
white[ 679 ]<=0;
white[ 680 ]<=0;
white[ 681 ]<=0;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=0;
white[ 695 ]<=0;
white[ 696 ]<=0;
white[ 697 ]<=0;
white[ 698 ]<=0;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=0;
white[ 710 ]<=0;
white[ 711 ]<=0;
white[ 712 ]<=0;
white[ 713 ]<=0;
```

```
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=0;
white[ 727 ]<=0;
white[ 728 ]<=0;
white[ 729 ]<=0;
white[ 730 ]<=0;
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=0;
white[ 742 ]<=0;
white[ 743 ]<=0;
white[ 744 ]<=0;
white[ 745 ]<=0;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=0;
white[ 760 ]<=0;
white[ 761 ]<=0;
white[ 762 ]<=0;
white[ 763 ]<=0;
white[ 764 ]<=0;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
```

```
white[ 772 ]<=0;
white[ 773 ]<=0;
white[ 774 ]<=0;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=0;
white[ 795 ]<=0;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=1;
white[ 804 ]<=0;
white[ 805 ]<=0;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=0;
white[ 828 ]<=0;
white[ 829 ]<=0;
```

```
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
```

```
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
```

```
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=1;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
```

```
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

J:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=1;
white[ 22 ]<=0;
white[ 23 ]<=1;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
```

```
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=1;
white[ 81 ]<=1;
white[ 82 ]<=1;
white[ 83 ]<=1;
white[ 84 ]<=1;
white[ 85 ]<=1;
white[ 86 ]<=1;
white[ 87 ]<=1;
white[ 88 ]<=1;
white[ 89 ]<=1;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
```

```
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=1;
white[ 116 ]<=1;
white[ 117 ]<=1;
white[ 118 ]<=1;
white[ 119 ]<=1;
white[ 120 ]<=1;
white[ 121 ]<=1;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
```

```
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=0;
white[ 168 ]<=0;
white[ 169 ]<=0;
white[ 170 ]<=0;
white[ 171 ]<=0;
white[ 172 ]<=0;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=0;
white[ 198 ]<=0;
white[ 199 ]<=0;
white[ 200 ]<=0;
white[ 201 ]<=0;
white[ 202 ]<=0;
white[ 203 ]<=0;
white[ 204 ]<=0;
white[ 205 ]<=0;
white[ 206 ]<=0;
white[ 207 ]<=0;
white[ 208 ]<=1;
```

```
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=0;
white[ 230 ]<=0;
white[ 231 ]<=0;
white[ 232 ]<=0;
white[ 233 ]<=0;
white[ 234 ]<=0;
white[ 235 ]<=0;
white[ 236 ]<=0;
white[ 237 ]<=0;
white[ 238 ]<=0;
white[ 239 ]<=0;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=0;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=0;
white[ 261 ]<=0;
white[ 262 ]<=0;
white[ 263 ]<=0;
white[ 264 ]<=0;
white[ 265 ]<=0;
white[ 266 ]<=0;
```

```
white[ 267 ]<=0;
white[ 268 ]<=0;
white[ 269 ]<=0;
white[ 270 ]<=0;
white[ 271 ]<=0;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=0;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=0;
white[ 294 ]<=0;
white[ 295 ]<=0;
white[ 296 ]<=0;
white[ 297 ]<=0;
white[ 298 ]<=0;
white[ 299 ]<=0;
white[ 300 ]<=0;
white[ 301 ]<=0;
white[ 302 ]<=0;
white[ 303 ]<=0;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=0;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=0;
```

```
white[ 325 ]<=0;
white[ 326 ]<=0;
white[ 327 ]<=0;
white[ 328 ]<=0;
white[ 329 ]<=0;
white[ 330 ]<=0;
white[ 331 ]<=0;
white[ 332 ]<=0;
white[ 333 ]<=0;
white[ 334 ]<=0;
white[ 335 ]<=0;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=0;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=0;
white[ 358 ]<=0;
white[ 359 ]<=0;
white[ 360 ]<=0;
white[ 361 ]<=0;
white[ 362 ]<=0;
white[ 363 ]<=0;
white[ 364 ]<=0;
white[ 365 ]<=0;
white[ 366 ]<=0;
white[ 367 ]<=0;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=0;
white[ 379 ]<=0;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
```

```
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
white[ 389 ]<=0;
white[ 390 ]<=0;
white[ 391 ]<=0;
white[ 392 ]<=0;
white[ 393 ]<=0;
white[ 394 ]<=0;
white[ 395 ]<=0;
white[ 396 ]<=0;
white[ 397 ]<=0;
white[ 398 ]<=0;
white[ 399 ]<=0;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=0;
white[ 423 ]<=0;
white[ 424 ]<=0;
white[ 425 ]<=0;
white[ 426 ]<=0;
white[ 427 ]<=0;
white[ 428 ]<=0;
white[ 429 ]<=0;
white[ 430 ]<=0;
white[ 431 ]<=0;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
```

```
white[ 441 ]<=1;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=0;
white[ 454 ]<=0;
white[ 455 ]<=0;
white[ 456 ]<=0;
white[ 457 ]<=0;
white[ 458 ]<=0;
white[ 459 ]<=0;
white[ 460 ]<=0;
white[ 461 ]<=0;
white[ 462 ]<=0;
white[ 463 ]<=0;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=0;
white[ 487 ]<=0;
white[ 488 ]<=0;
white[ 489 ]<=0;
white[ 490 ]<=0;
white[ 491 ]<=0;
white[ 492 ]<=0;
white[ 493 ]<=0;
white[ 494 ]<=0;
white[ 495 ]<=0;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
```

```
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=0;
white[ 519 ]<=0;
white[ 520 ]<=0;
white[ 521 ]<=0;
white[ 522 ]<=0;
white[ 523 ]<=0;
white[ 524 ]<=0;
white[ 525 ]<=0;
white[ 526 ]<=0;
white[ 527 ]<=0;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=1;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=0;
white[ 551 ]<=0;
white[ 552 ]<=0;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=0;
white[ 556 ]<=0;
```

```
white[ 557 ]<=0;
white[ 558 ]<=0;
white[ 559 ]<=0;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=0;
white[ 590 ]<=0;
white[ 591 ]<=0;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
```

```
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=0;
white[ 622 ]<=0;
white[ 623 ]<=0;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=0;
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=0;
white[ 654 ]<=0;
white[ 655 ]<=0;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
```

```
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=0;
white[ 686 ]<=0;
white[ 687 ]<=0;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=0;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=0;
```

```
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=0;
white[ 763 ]<=0;
white[ 764 ]<=0;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
```

```
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=0;
white[ 795 ]<=0;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=0;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=0;
white[ 826 ]<=0;
white[ 827 ]<=0;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=1;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
```

```
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=0;
white[ 857 ]<=0;
white[ 858 ]<=0;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=0;
white[ 870 ]<=0;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=0;
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=1;
white[ 904 ]<=1;
```

```
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=0;
white[ 920 ]<=0;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=1;
white[ 935 ]<=0;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=1;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
```

```
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=1;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=1;
white[ 973 ]<=1;
white[ 974 ]<=1;
white[ 975 ]<=1;
white[ 976 ]<=1;
white[ 977 ]<=1;
white[ 978 ]<=1;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
```

```
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

K:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
```

```
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
```

```
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=1;
white[ 155 ]<=1;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=1;
white[ 163 ]<=0;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
```

```
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=1;
white[ 188 ]<=1;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=0;
white[ 207 ]<=0;
white[ 208 ]<=0;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
```

```
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=0;
white[ 239 ]<=0;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=0;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=0;
```

```
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=0;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
```

```
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=0;
white[ 346 ]<=0;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=0;
white[ 377 ]<=0;
white[ 378 ]<=0;
white[ 379 ]<=0;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
```

```
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=0;
white[ 409 ]<=0;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=0;
white[ 440 ]<=0;
white[ 441 ]<=0;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
```

```
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=0;
white[ 471 ]<=0;
white[ 472 ]<=0;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=0;
white[ 504 ]<=0;
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
```

```
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=0;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
```

```
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
```

```
white[ 632 ]<=1;
white[ 633 ]<=0;
white[ 634 ]<=0;
white[ 635 ]<=0;
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=0;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
```

```
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=0;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=0;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
```

```
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=0;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=0;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=0;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
```

```
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=0;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=0;
white[ 848 ]<=0;
white[ 849 ]<=0;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
```

```
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=0;
white[ 879 ]<=0;
white[ 880 ]<=0;
white[ 881 ]<=0;
white[ 882 ]<=0;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=0;
white[ 911 ]<=0;
white[ 912 ]<=0;
white[ 913 ]<=0;
white[ 914 ]<=0;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
```

```
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=1;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=0;
white[ 941 ]<=0;
white[ 942 ]<=0;
white[ 943 ]<=0;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=1;
white[ 955 ]<=1;
white[ 956 ]<=1;
white[ 957 ]<=1;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=1;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
```

```
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

L:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
```

```
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
```

```
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
```

```
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=0;
white[ 146 ]<=0;
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=0;
white[ 179 ]<=0;
white[ 180 ]<=0;
white[ 181 ]<=0;
white[ 182 ]<=0;
white[ 183 ]<=0;
white[ 184 ]<=0;
```

```
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=0;
white[ 212 ]<=0;
white[ 213 ]<=0;
white[ 214 ]<=0;
white[ 215 ]<=0;
white[ 216 ]<=0;
white[ 217 ]<=0;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
```

```
white[ 243 ]<=1;
white[ 244 ]<=0;
white[ 245 ]<=0;
white[ 246 ]<=0;
white[ 247 ]<=0;
white[ 248 ]<=0;
white[ 249 ]<=0;
white[ 250 ]<=0;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=0;
white[ 277 ]<=0;
white[ 278 ]<=0;
white[ 279 ]<=0;
white[ 280 ]<=0;
white[ 281 ]<=0;
white[ 282 ]<=0;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
```

```
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=0;
white[ 309 ]<=0;
white[ 310 ]<=0;
white[ 311 ]<=0;
white[ 312 ]<=0;
white[ 313 ]<=0;
white[ 314 ]<=0;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=0;
white[ 325 ]<=0;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=0;
white[ 340 ]<=0;
white[ 341 ]<=0;
white[ 342 ]<=0;
white[ 343 ]<=0;
white[ 344 ]<=0;
white[ 345 ]<=0;
white[ 346 ]<=0;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=1;
white[ 358 ]<=1;
```

```
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=0;
white[ 372 ]<=0;
white[ 373 ]<=0;
white[ 374 ]<=0;
white[ 375 ]<=0;
white[ 376 ]<=0;
white[ 377 ]<=0;
white[ 378 ]<=0;
white[ 379 ]<=0;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
white[ 389 ]<=0;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=0;
white[ 404 ]<=0;
white[ 405 ]<=0;
white[ 406 ]<=0;
white[ 407 ]<=0;
white[ 408 ]<=0;
white[ 409 ]<=0;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
```

```
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=0;
white[ 435 ]<=0;
white[ 436 ]<=0;
white[ 437 ]<=0;
white[ 438 ]<=0;
white[ 439 ]<=0;
white[ 440 ]<=0;
white[ 441 ]<=0;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=0;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=0;
white[ 467 ]<=0;
white[ 468 ]<=0;
white[ 469 ]<=0;
white[ 470 ]<=0;
white[ 471 ]<=0;
white[ 472 ]<=0;
white[ 473 ]<=0;
white[ 474 ]<=0;
```

```
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=0;
white[ 499 ]<=0;
white[ 500 ]<=0;
white[ 501 ]<=0;
white[ 502 ]<=0;
white[ 503 ]<=0;
white[ 504 ]<=0;
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=0;
white[ 531 ]<=0;
white[ 532 ]<=0;
```

```
white[ 533 ]<=0;
white[ 534 ]<=0;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=0;
white[ 563 ]<=0;
white[ 564 ]<=0;
white[ 565 ]<=0;
white[ 566 ]<=0;
white[ 567 ]<=0;
white[ 568 ]<=0;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=0;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
```

```
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=0;
white[ 595 ]<=0;
white[ 596 ]<=0;
white[ 597 ]<=0;
white[ 598 ]<=0;
white[ 599 ]<=0;
white[ 600 ]<=0;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=0;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=0;
white[ 627 ]<=0;
white[ 628 ]<=0;
white[ 629 ]<=0;
white[ 630 ]<=0;
white[ 631 ]<=0;
white[ 632 ]<=0;
white[ 633 ]<=0;
white[ 634 ]<=0;
white[ 635 ]<=0;
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=0;
white[ 645 ]<=0;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
```

```
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=0;
white[ 659 ]<=0;
white[ 660 ]<=0;
white[ 661 ]<=0;
white[ 662 ]<=0;
white[ 663 ]<=0;
white[ 664 ]<=0;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=0;
white[ 677 ]<=0;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=0;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
```

```
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=0;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
```

```
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
```

```
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
```

```
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
```

```
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=1;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
```

```
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

M:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
```

```
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
```

```
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
```

```
white[ 144 ]<=1;
white[ 145 ]<=0;
white[ 146 ]<=0;
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=1;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=1;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=1;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=1;
white[ 188 ]<=1;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
```

```
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=0;
white[ 207 ]<=0;
white[ 208 ]<=0;
white[ 209 ]<=0;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=1;
white[ 222 ]<=1;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=1;
white[ 255 ]<=1;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=1;
white[ 259 ]<=1;
```

```
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=1;
white[ 287 ]<=1;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=1;
```

```
white[ 318 ]<=1;
white[ 319 ]<=1;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=1;
white[ 351 ]<=1;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
```

```
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=1;
white[ 383 ]<=1;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=1;
white[ 415 ]<=1;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=1;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
```

```
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=1;
white[ 447 ]<=1;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=1;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=1;
white[ 478 ]<=1;
white[ 479 ]<=1;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=1;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
```

```
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=1;
white[ 511 ]<=1;
white[ 512 ]<=0;
white[ 513 ]<=1;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=1;
white[ 543 ]<=1;
white[ 544 ]<=0;
white[ 545 ]<=1;
white[ 546 ]<=1;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
```

```
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=1;
white[ 576 ]<=0;
white[ 577 ]<=1;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=1;
white[ 607 ]<=1;
```

```
white[ 608 ]<=0;
white[ 609 ]<=1;
white[ 610 ]<=1;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=1;
white[ 640 ]<=1;
white[ 641 ]<=1;
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
```

```
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=1;
white[ 671 ]<=1;
white[ 672 ]<=1;
white[ 673 ]<=1;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=1;
white[ 703 ]<=1;
white[ 704 ]<=1;
white[ 705 ]<=1;
white[ 706 ]<=1;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
```

```
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=1;
white[ 735 ]<=1;
white[ 736 ]<=1;
white[ 737 ]<=1;
white[ 738 ]<=1;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=1;
white[ 768 ]<=1;
white[ 769 ]<=1;
white[ 770 ]<=1;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
```

```
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=1;
white[ 800 ]<=1;
white[ 801 ]<=1;
white[ 802 ]<=1;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=1;
white[ 832 ]<=1;
white[ 833 ]<=1;
white[ 834 ]<=1;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
```

```
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=1;
white[ 863 ]<=1;
white[ 864 ]<=1;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=1;
white[ 895 ]<=1;
white[ 896 ]<=1;
white[ 897 ]<=1;
```

```
white[ 898 ]<=1;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=0;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=1;
white[ 927 ]<=1;
white[ 928 ]<=1;
white[ 929 ]<=1;
white[ 930 ]<=1;
white[ 931 ]<=1;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=0;
white[ 940 ]<=0;
white[ 941 ]<=0;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=0;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
```

```
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
```

```
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

N:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
```

```
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
```

```
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=0;
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
```

```
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=0;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=0;
white[ 180 ]<=0;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=1;
white[ 188 ]<=1;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=0;
white[ 206 ]<=0;
white[ 207 ]<=0;
white[ 208 ]<=0;
white[ 209 ]<=0;
white[ 210 ]<=0;
white[ 211 ]<=0;
white[ 212 ]<=0;
white[ 213 ]<=0;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
```

```
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=1;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=0;
white[ 239 ]<=0;
white[ 240 ]<=0;
white[ 241 ]<=0;
white[ 242 ]<=0;
white[ 243 ]<=0;
white[ 244 ]<=0;
white[ 245 ]<=0;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=0;
white[ 272 ]<=0;
white[ 273 ]<=0;
white[ 274 ]<=0;
white[ 275 ]<=0;
white[ 276 ]<=0;
```

```
white[ 277 ]<=0;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=0;
white[ 305 ]<=0;
white[ 306 ]<=0;
white[ 307 ]<=0;
white[ 308 ]<=0;
white[ 309 ]<=0;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=1;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
```

```
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=0;
white[ 338 ]<=0;
white[ 339 ]<=0;
white[ 340 ]<=0;
white[ 341 ]<=0;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=0;
white[ 370 ]<=0;
white[ 371 ]<=0;
white[ 372 ]<=0;
white[ 373 ]<=0;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
```

```
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=0;
white[ 403 ]<=0;
white[ 404 ]<=0;
white[ 405 ]<=0;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=0;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
```

```
white[ 451 ]<=0;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=1;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
```

```
white[ 509 ]<=1;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
```

```
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
```

```
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=0;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
```

```
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=0;
white[ 686 ]<=0;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=0;
white[ 717 ]<=0;
white[ 718 ]<=0;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=1;
```

```
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=0;
white[ 749 ]<=0;
white[ 750 ]<=0;
white[ 751 ]<=0;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=0;
white[ 781 ]<=0;
white[ 782 ]<=0;
white[ 783 ]<=0;
white[ 784 ]<=0;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=0;
```

```
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=0;
white[ 812 ]<=0;
white[ 813 ]<=0;
white[ 814 ]<=0;
white[ 815 ]<=0;
white[ 816 ]<=0;
white[ 817 ]<=0;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=0;
white[ 844 ]<=0;
white[ 845 ]<=0;
white[ 846 ]<=0;
white[ 847 ]<=0;
white[ 848 ]<=1;
white[ 849 ]<=0;
white[ 850 ]<=0;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
```

```
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=0;
white[ 876 ]<=0;
white[ 877 ]<=0;
white[ 878 ]<=0;
white[ 879 ]<=0;
white[ 880 ]<=0;
white[ 881 ]<=0;
white[ 882 ]<=0;
white[ 883 ]<=0;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=0;
white[ 908 ]<=0;
white[ 909 ]<=0;
white[ 910 ]<=0;
white[ 911 ]<=0;
white[ 912 ]<=0;
white[ 913 ]<=0;
white[ 914 ]<=0;
```

```
white[ 915 ]<=0;
white[ 916 ]<=0;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=0;
white[ 940 ]<=0;
white[ 941 ]<=0;
white[ 942 ]<=0;
white[ 943 ]<=0;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=0;
white[ 949 ]<=0;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=1;
white[ 955 ]<=1;
white[ 956 ]<=1;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=1;
white[ 966 ]<=1;
white[ 967 ]<=1;
white[ 968 ]<=1;
white[ 969 ]<=1;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
```

```
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=1;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=1;
white[ 985 ]<=1;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

O:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
```

```
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
```

```
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
```

```
white[ 120 ]<=1;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
```

```
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=0;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=0;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
```

```
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=1;
white[ 293 ]<=1;
```

```
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=0;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
```

```
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=1;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=0;
white[ 370 ]<=0;
white[ 371 ]<=0;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=1;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=0;
white[ 401 ]<=0;
white[ 402 ]<=0;
white[ 403 ]<=0;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
```

```
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=1;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=1;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=0;
white[ 432 ]<=0;
white[ 433 ]<=0;
white[ 434 ]<=0;
white[ 435 ]<=0;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=1;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=1;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=0;
white[ 463 ]<=0;
white[ 464 ]<=0;
white[ 465 ]<=0;
white[ 466 ]<=0;
white[ 467 ]<=0;
```

```
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=1;
white[ 478 ]<=1;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=1;
white[ 482 ]<=1;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=0;
white[ 495 ]<=0;
white[ 496 ]<=0;
white[ 497 ]<=0;
white[ 498 ]<=0;
white[ 499 ]<=0;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=1;
white[ 511 ]<=1;
white[ 512 ]<=0;
white[ 513 ]<=1;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
```

```
white[ 526 ]<=0;
white[ 527 ]<=0;
white[ 528 ]<=0;
white[ 529 ]<=0;
white[ 530 ]<=0;
white[ 531 ]<=0;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=1;
white[ 543 ]<=1;
white[ 544 ]<=0;
white[ 545 ]<=1;
white[ 546 ]<=1;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=0;
white[ 558 ]<=0;
white[ 559 ]<=0;
white[ 560 ]<=0;
white[ 561 ]<=0;
white[ 562 ]<=0;
white[ 563 ]<=0;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=1;
white[ 576 ]<=0;
white[ 577 ]<=1;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
```

```
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=0;
white[ 591 ]<=0;
white[ 592 ]<=0;
white[ 593 ]<=0;
white[ 594 ]<=0;
white[ 595 ]<=0;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=1;
white[ 607 ]<=1;
white[ 608 ]<=0;
white[ 609 ]<=1;
white[ 610 ]<=1;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=0;
white[ 622 ]<=0;
white[ 623 ]<=0;
white[ 624 ]<=0;
white[ 625 ]<=0;
white[ 626 ]<=0;
white[ 627 ]<=0;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=1;
white[ 640 ]<=0;
white[ 641 ]<=0;
```

```
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=0;
white[ 654 ]<=0;
white[ 655 ]<=0;
white[ 656 ]<=0;
white[ 657 ]<=0;
white[ 658 ]<=0;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=1;
white[ 671 ]<=1;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=0;
white[ 687 ]<=0;
white[ 688 ]<=0;
white[ 689 ]<=0;
white[ 690 ]<=0;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
```

```
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=1;
white[ 703 ]<=1;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=1;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=0;
white[ 719 ]<=0;
white[ 720 ]<=0;
white[ 721 ]<=0;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=1;
white[ 735 ]<=1;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=1;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=0;
white[ 751 ]<=0;
white[ 752 ]<=0;
white[ 753 ]<=0;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
```

```
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
```

```
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=0;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
```

```
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
```

```
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=1;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=1;
white[ 977 ]<=1;
white[ 978 ]<=1;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
```

```
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

P:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
```

```
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
```

```
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=1;
white[ 102 ]<=1;
white[ 103 ]<=1;
white[ 104 ]<=1;
white[ 105 ]<=1;
white[ 106 ]<=1;
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=1;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
```

```
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=1;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=1;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
```

```
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=0;
```

```
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
```

```
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=0;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
```

```
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=1;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
```

```
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=1;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=1;
white[ 478 ]<=1;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
```

```
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=1;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=1;
```

```
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=0;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
```

```
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=0;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
```

```
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=0;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=0;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
```

```
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=0;
white[ 729 ]<=0;
white[ 730 ]<=0;
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=0;
white[ 757 ]<=0;
white[ 758 ]<=0;
white[ 759 ]<=0;
white[ 760 ]<=0;
white[ 761 ]<=0;
white[ 762 ]<=0;
white[ 763 ]<=0;
white[ 764 ]<=0;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=1;
white[ 774 ]<=1;
```

```
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=0;
white[ 786 ]<=0;
white[ 787 ]<=0;
white[ 788 ]<=0;
white[ 789 ]<=0;
white[ 790 ]<=0;
white[ 791 ]<=0;
white[ 792 ]<=0;
white[ 793 ]<=0;
white[ 794 ]<=0;
white[ 795 ]<=0;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=0;
white[ 818 ]<=0;
white[ 819 ]<=0;
white[ 820 ]<=0;
white[ 821 ]<=0;
white[ 822 ]<=0;
white[ 823 ]<=0;
white[ 824 ]<=0;
white[ 825 ]<=0;
white[ 826 ]<=0;
white[ 827 ]<=0;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
```

```
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=0;
white[ 851 ]<=0;
white[ 852 ]<=0;
white[ 853 ]<=0;
white[ 854 ]<=0;
white[ 855 ]<=0;
white[ 856 ]<=0;
white[ 857 ]<=0;
white[ 858 ]<=0;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=0;
white[ 883 ]<=0;
white[ 884 ]<=0;
white[ 885 ]<=0;
white[ 886 ]<=0;
white[ 887 ]<=0;
white[ 888 ]<=0;
white[ 889 ]<=0;
white[ 890 ]<=0;
```

```
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=0;
white[ 916 ]<=0;
white[ 917 ]<=0;
white[ 918 ]<=0;
white[ 919 ]<=0;
white[ 920 ]<=0;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=0;
```

```
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=1;
white[ 967 ]<=1;
white[ 968 ]<=1;
white[ 969 ]<=1;
white[ 970 ]<=1;
white[ 971 ]<=1;
white[ 972 ]<=1;
white[ 973 ]<=0;
white[ 974 ]<=1;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
```

```
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

Q:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
```

```
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=1;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=1;
white[ 77 ]<=1;
white[ 78 ]<=1;
white[ 79 ]<=1;
white[ 80 ]<=1;
white[ 81 ]<=1;
white[ 82 ]<=1;
white[ 83 ]<=1;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
```

```
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=1;
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=1;
white[ 116 ]<=1;
white[ 117 ]<=1;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
```

```
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=0;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
```

```
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
```

```
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
```

```
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=0;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=0;
white[ 368 ]<=0;
white[ 369 ]<=0;
white[ 370 ]<=0;
white[ 371 ]<=0;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
```

```
white[ 386 ]<=0;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=0;
white[ 399 ]<=0;
white[ 400 ]<=0;
white[ 401 ]<=0;
white[ 402 ]<=0;
white[ 403 ]<=0;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=1;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=0;
white[ 430 ]<=0;
white[ 431 ]<=0;
white[ 432 ]<=0;
white[ 433 ]<=0;
white[ 434 ]<=0;
white[ 435 ]<=0;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
```

```
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=1;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=0;
white[ 462 ]<=0;
white[ 463 ]<=0;
white[ 464 ]<=0;
white[ 465 ]<=0;
white[ 466 ]<=0;
white[ 467 ]<=0;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=1;
white[ 478 ]<=1;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=1;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=0;
white[ 493 ]<=0;
white[ 494 ]<=0;
white[ 495 ]<=0;
white[ 496 ]<=0;
white[ 497 ]<=0;
white[ 498 ]<=0;
white[ 499 ]<=0;
white[ 500 ]<=1;
white[ 501 ]<=1;
```

```
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=1;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=0;
white[ 525 ]<=0;
white[ 526 ]<=0;
white[ 527 ]<=0;
white[ 528 ]<=0;
white[ 529 ]<=1;
white[ 530 ]<=0;
white[ 531 ]<=0;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=1;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=1;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=0;
white[ 557 ]<=0;
white[ 558 ]<=0;
white[ 559 ]<=0;
```

```
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=0;
white[ 590 ]<=0;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=1;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=1;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
```

```
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=0;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=1;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=1;
```

```
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=0;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=0;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
```

```
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
```

```
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=0;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=0;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
```

```
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=0;
white[ 870 ]<=0;
white[ 871 ]<=0;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=0;
white[ 905 ]<=0;
white[ 906 ]<=1;
white[ 907 ]<=1;
```

```
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=0;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=1;
white[ 955 ]<=1;
white[ 956 ]<=1;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
```

```
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=1;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=1;
white[ 979 ]<=0;
white[ 980 ]<=1;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=1;
white[ 984 ]<=1;
white[ 985 ]<=1;
white[ 986 ]<=1;
white[ 987 ]<=1;
white[ 988 ]<=1;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=1;
white[ 1016 ]<=1;
white[ 1017 ]<=1;
white[ 1018 ]<=1;
white[ 1019 ]<=1;
white[ 1020 ]<=1;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
```

```
end

R:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
```

```
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
```

```
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=1;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=1;
white[ 162 ]<=1;
white[ 163 ]<=1;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
```

```
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=1;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=0;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=1;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
```

```
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=0;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=1;
white[ 258 ]<=1;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=0;
white[ 284 ]<=1;
white[ 285 ]<=0;
white[ 286 ]<=0;
```

```
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=1;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
```

```
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=0;
white[ 399 ]<=0;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
```

```
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=0;
white[ 431 ]<=0;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
```

```
white[ 461 ]<=1;
white[ 462 ]<=0;
white[ 463 ]<=0;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=0;
white[ 495 ]<=0;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
```

```
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
```

```
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
```

```
white[ 635 ]<=0;
white[ 636 ]<=1;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=0;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
```

```
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
```

```
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=1;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=1;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
```

```
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=1;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=1;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=1;
```

```
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=1;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=0;
white[ 913 ]<=0;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
```

```
white[ 925 ]<=1;
white[ 926 ]<=0;
white[ 927 ]<=1;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=1;
white[ 931 ]<=1;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=0;
white[ 941 ]<=0;
white[ 942 ]<=0;
white[ 943 ]<=0;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=0;
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=1;
white[ 959 ]<=1;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
```

```
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

S:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
```

```
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
```

```
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=1;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=1;
white[ 107 ]<=1;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=1;
white[ 113 ]<=1;
white[ 114 ]<=1;
white[ 115 ]<=1;
white[ 116 ]<=1;
white[ 117 ]<=1;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
```

```
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
```

```
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=0;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
```

```
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=0;
white[ 250 ]<=0;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=0;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
```

```
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
```

```
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=0;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=0;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
```

```
white[ 420 ]<=0;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=0;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
```

```
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=0;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
```

```
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=0;
white[ 551 ]<=0;
white[ 552 ]<=0;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=0;
white[ 584 ]<=0;
white[ 585 ]<=0;
white[ 586 ]<=0;
white[ 587 ]<=0;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
```

```
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=0;
white[ 617 ]<=0;
white[ 618 ]<=0;
white[ 619 ]<=0;
white[ 620 ]<=0;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=0;
white[ 651 ]<=0;
```

```
white[ 652 ]<=0;
white[ 653 ]<=0;
white[ 654 ]<=0;
white[ 655 ]<=0;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=0;
white[ 685 ]<=0;
white[ 686 ]<=0;
white[ 687 ]<=0;
white[ 688 ]<=0;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
```

```
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
```

```
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
```

```
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=1;
white[ 836 ]<=0;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
```

```
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=0;
white[ 939 ]<=0;
white[ 940 ]<=0;
white[ 941 ]<=0;
```

```
white[ 942 ]<=0;
white[ 943 ]<=0;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
```

```
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

T:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
```

```
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
```

```
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=0;
```

```
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=0;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=0;
white[ 171 ]<=0;
white[ 172 ]<=0;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=0;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
```

```
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
```

```
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
```

```
white[ 321 ]<=0;
white[ 322 ]<=1;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=1;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
```

```
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=1;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=1;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
```

```
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=0;
white[ 456 ]<=0;
white[ 457 ]<=0;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=0;
white[ 472 ]<=0;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=0;
white[ 487 ]<=0;
white[ 488 ]<=0;
white[ 489 ]<=0;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
```

```
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=0;
white[ 504 ]<=0;
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=1;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=0;
white[ 519 ]<=0;
white[ 520 ]<=0;
white[ 521 ]<=0;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=0;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=0;
white[ 551 ]<=0;
white[ 552 ]<=0;
```

```
white[ 553 ]<=0;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=0;
white[ 567 ]<=0;
white[ 568 ]<=0;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=0;
white[ 581 ]<=0;
white[ 582 ]<=0;
white[ 583 ]<=0;
white[ 584 ]<=0;
white[ 585 ]<=0;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=0;
white[ 599 ]<=0;
white[ 600 ]<=0;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
```

```
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=0;
white[ 614 ]<=0;
white[ 615 ]<=0;
white[ 616 ]<=0;
white[ 617 ]<=0;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=0;
white[ 631 ]<=0;
white[ 632 ]<=0;
white[ 633 ]<=0;
white[ 634 ]<=0;
white[ 635 ]<=0;
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=0;
white[ 645 ]<=0;
white[ 646 ]<=0;
white[ 647 ]<=0;
white[ 648 ]<=0;
white[ 649 ]<=0;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=0;
white[ 663 ]<=0;
white[ 664 ]<=0;
white[ 665 ]<=0;
white[ 666 ]<=0;
white[ 667 ]<=0;
white[ 668 ]<=0;
```

```
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=0;
white[ 677 ]<=0;
white[ 678 ]<=0;
white[ 679 ]<=0;
white[ 680 ]<=0;
white[ 681 ]<=0;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=0;
white[ 695 ]<=0;
white[ 696 ]<=0;
white[ 697 ]<=0;
white[ 698 ]<=0;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=0;
white[ 710 ]<=0;
white[ 711 ]<=0;
white[ 712 ]<=0;
white[ 713 ]<=0;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=0;
```

```
white[ 727 ]<=0;
white[ 728 ]<=0;
white[ 729 ]<=0;
white[ 730 ]<=0;
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=0;
white[ 742 ]<=0;
white[ 743 ]<=0;
white[ 744 ]<=0;
white[ 745 ]<=0;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=0;
white[ 759 ]<=0;
white[ 760 ]<=0;
white[ 761 ]<=0;
white[ 762 ]<=0;
white[ 763 ]<=0;
white[ 764 ]<=0;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=0;
white[ 774 ]<=0;
white[ 775 ]<=0;
white[ 776 ]<=0;
white[ 777 ]<=0;
white[ 778 ]<=0;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
```

```
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=0;
white[ 791 ]<=1;
white[ 792 ]<=0;
white[ 793 ]<=0;
white[ 794 ]<=0;
white[ 795 ]<=0;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=0;
white[ 805 ]<=0;
white[ 806 ]<=0;
white[ 807 ]<=0;
white[ 808 ]<=0;
white[ 809 ]<=0;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=0;
white[ 823 ]<=0;
white[ 824 ]<=0;
white[ 825 ]<=0;
white[ 826 ]<=0;
white[ 827 ]<=0;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=0;
white[ 839 ]<=0;
white[ 840 ]<=0;
white[ 841 ]<=1;
white[ 842 ]<=1;
```

```
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=0;
white[ 856 ]<=0;
white[ 857 ]<=0;
white[ 858 ]<=0;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=0;
white[ 870 ]<=0;
white[ 871 ]<=0;
white[ 872 ]<=0;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=0;
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
```

```
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=0;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=0;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
```

```
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
```

```
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

U:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
```

```
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
```

```
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=1;
white[ 109 ]<=1;
white[ 110 ]<=1;
white[ 111 ]<=1;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=1;
white[ 117 ]<=1;
white[ 118 ]<=1;
white[ 119 ]<=1;
white[ 120 ]<=1;
white[ 121 ]<=1;
white[ 122 ]<=1;
white[ 123 ]<=1;
white[ 124 ]<=1;
white[ 125 ]<=1;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=0;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=1;
white[ 155 ]<=1;
white[ 156 ]<=1;
white[ 157 ]<=1;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
```

```
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=1;
white[ 188 ]<=1;
white[ 189 ]<=1;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=1;
```

```
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=0;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=0;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
```

```
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=0;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=0;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
```

```
white[ 338 ]<=1;
white[ 339 ]<=0;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=0;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
```

```
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=0;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=0;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=1;
```

```
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=0;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=1;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=0;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=0;
white[ 511 ]<=0;
```

```
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=0;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=0;
white[ 564 ]<=0;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
```

```
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=0;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=0;
white[ 596 ]<=0;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=0;
```

```
white[ 628 ]<=0;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=0;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=0;
white[ 660 ]<=0;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=0;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
```

```
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=0;
white[ 692 ]<=0;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=0;
white[ 724 ]<=0;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=0;
white[ 742 ]<=1;
white[ 743 ]<=1;
```

```
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=0;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
```

```
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=0;
white[ 805 ]<=0;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=0;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
```

```
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=1;
white[ 868 ]<=0;
white[ 869 ]<=0;
white[ 870 ]<=0;
white[ 871 ]<=0;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=0;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
```

```
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
```

```
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

V:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
```

```
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
```

```
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
```

```
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=1;
white[ 131 ]<=1;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=0;
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=1;
white[ 155 ]<=1;
white[ 156 ]<=1;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=1;
white[ 162 ]<=1;
white[ 163 ]<=1;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=0;
white[ 180 ]<=1;
```

```
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=1;
white[ 188 ]<=1;
white[ 189 ]<=1;
white[ 190 ]<=1;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=1;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=0;
white[ 209 ]<=0;
white[ 210 ]<=0;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=1;
white[ 222 ]<=1;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=1;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
```

```
white[ 239 ]<=1;
white[ 240 ]<=0;
white[ 241 ]<=0;
white[ 242 ]<=0;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=1;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=1;
white[ 258 ]<=1;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=0;
white[ 274 ]<=0;
white[ 275 ]<=0;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=1;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=1;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
```

```
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=0;
white[ 306 ]<=0;
white[ 307 ]<=0;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=1;
white[ 318 ]<=1;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=0;
white[ 337 ]<=0;
white[ 338 ]<=0;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
```

```
white[ 355 ]<=0;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=0;
white[ 370 ]<=0;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=0;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
```

```
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=0;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
```

```
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
```

```
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=0;
white[ 581 ]<=0;
white[ 582 ]<=0;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
```

```
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=0;
white[ 614 ]<=0;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=0;
white[ 634 ]<=0;
white[ 635 ]<=0;
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=0;
```

```
white[ 645 ]<=0;
white[ 646 ]<=0;
white[ 647 ]<=0;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=0;
white[ 666 ]<=0;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=0;
white[ 677 ]<=0;
white[ 678 ]<=0;
white[ 679 ]<=0;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=0;
white[ 698 ]<=0;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
```

```
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=0;
white[ 710 ]<=0;
white[ 711 ]<=0;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=0;
white[ 729 ]<=0;
white[ 730 ]<=0;
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=0;
white[ 742 ]<=0;
white[ 743 ]<=0;
white[ 744 ]<=0;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=0;
```

```
white[ 761 ]<=0;
white[ 762 ]<=0;
white[ 763 ]<=0;
white[ 764 ]<=0;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=0;
white[ 774 ]<=0;
white[ 775 ]<=0;
white[ 776 ]<=0;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=0;
white[ 792 ]<=0;
white[ 793 ]<=0;
white[ 794 ]<=0;
white[ 795 ]<=0;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=0;
white[ 805 ]<=0;
white[ 806 ]<=0;
white[ 807 ]<=0;
white[ 808 ]<=0;
white[ 809 ]<=0;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
```

```
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=0;
white[ 823 ]<=0;
white[ 824 ]<=0;
white[ 825 ]<=0;
white[ 826 ]<=0;
white[ 827 ]<=0;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=0;
white[ 839 ]<=0;
white[ 840 ]<=0;
white[ 841 ]<=0;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=0;
white[ 855 ]<=0;
white[ 856 ]<=0;
white[ 857 ]<=0;
white[ 858 ]<=0;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=0;
white[ 870 ]<=0;
white[ 871 ]<=0;
white[ 872 ]<=0;
white[ 873 ]<=0;
white[ 874 ]<=0;
white[ 875 ]<=1;
white[ 876 ]<=1;
```

```
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=0;
white[ 886 ]<=0;
white[ 887 ]<=0;
white[ 888 ]<=0;
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=0;
white[ 905 ]<=0;
white[ 906 ]<=0;
white[ 907 ]<=0;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=0;
white[ 917 ]<=0;
white[ 918 ]<=0;
white[ 919 ]<=0;
white[ 920 ]<=0;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
```

```
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=0;
white[ 940 ]<=0;
white[ 941 ]<=0;
white[ 942 ]<=0;
white[ 943 ]<=0;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=0;
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
```

```
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

W:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
```

```
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
```

```
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
```

```
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=0;
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=0;
white[ 168 ]<=0;
white[ 169 ]<=0;
white[ 170 ]<=1;
white[ 171 ]<=0;
white[ 172 ]<=0;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=1;
white[ 180 ]<=0;
white[ 181 ]<=0;
white[ 182 ]<=0;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=1;
white[ 188 ]<=1;
white[ 189 ]<=1;
white[ 190 ]<=1;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=0;
```

```
white[ 198 ]<=0;
white[ 199 ]<=0;
white[ 200 ]<=0;
white[ 201 ]<=0;
white[ 202 ]<=0;
white[ 203 ]<=1;
white[ 204 ]<=0;
white[ 205 ]<=0;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=0;
white[ 211 ]<=0;
white[ 212 ]<=0;
white[ 213 ]<=0;
white[ 214 ]<=0;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=1;
white[ 222 ]<=1;
white[ 223 ]<=1;
white[ 224 ]<=1;
white[ 225 ]<=1;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=0;
white[ 234 ]<=0;
white[ 235 ]<=0;
white[ 236 ]<=0;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=0;
white[ 245 ]<=0;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=1;
white[ 255 ]<=1;
```

```
white[ 256 ]<=1;
white[ 257 ]<=1;
white[ 258 ]<=1;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=0;
white[ 267 ]<=0;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=1;
white[ 287 ]<=1;
white[ 288 ]<=1;
white[ 289 ]<=1;
white[ 290 ]<=1;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=0;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
```

```
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=1;
white[ 318 ]<=1;
white[ 319 ]<=1;
white[ 320 ]<=1;
white[ 321 ]<=1;
white[ 322 ]<=1;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=1;
white[ 351 ]<=1;
white[ 352 ]<=1;
white[ 353 ]<=1;
white[ 354 ]<=1;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
```

```
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=1;
white[ 383 ]<=1;
white[ 384 ]<=1;
white[ 385 ]<=1;
white[ 386 ]<=1;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=1;
white[ 415 ]<=1;
white[ 416 ]<=1;
white[ 417 ]<=1;
white[ 418 ]<=1;
white[ 419 ]<=1;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
```

```
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=1;
white[ 447 ]<=1;
white[ 448 ]<=1;
white[ 449 ]<=1;
white[ 450 ]<=1;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=1;
white[ 477 ]<=1;
white[ 478 ]<=1;
white[ 479 ]<=1;
white[ 480 ]<=1;
white[ 481 ]<=1;
white[ 482 ]<=1;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
```

```
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=1;
white[ 511 ]<=1;
white[ 512 ]<=0;
white[ 513 ]<=1;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=1;
white[ 538 ]<=1;
white[ 539 ]<=1;
white[ 540 ]<=1;
white[ 541 ]<=1;
white[ 542 ]<=1;
white[ 543 ]<=1;
white[ 544 ]<=0;
white[ 545 ]<=1;
```

```
white[ 546 ]<=1;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=1;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=1;
```

```
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=1;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=1;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
```

```
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=1;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
```

```
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
```

```
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=0;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=0;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=1;
```

```
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=0;
white[ 881 ]<=0;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=0;
```

```
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=0;
white[ 912 ]<=0;
white[ 913 ]<=0;
white[ 914 ]<=0;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=0;
white[ 940 ]<=0;
white[ 941 ]<=0;
white[ 942 ]<=0;
white[ 943 ]<=0;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=0;
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=0;
```

```
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
```

```
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

X:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
```

```
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
```

```
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=0;
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
```

```
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=0;
white[ 168 ]<=0;
white[ 169 ]<=0;
white[ 170 ]<=0;
white[ 171 ]<=0;
white[ 172 ]<=0;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=1;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=1;
white[ 194 ]<=0;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=0;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
```

```
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=1;
white[ 258 ]<=1;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
```

```
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=1;
white[ 290 ]<=1;
white[ 291 ]<=1;
white[ 292 ]<=1;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=1;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=1;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
```

```
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=1;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
```

```
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=1;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
```

```
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=0;
white[ 454 ]<=0;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=0;
white[ 487 ]<=0;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
```

```
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=0;
white[ 519 ]<=0;
white[ 520 ]<=0;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=0;
white[ 551 ]<=0;
white[ 552 ]<=0;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
```

```
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=0;
white[ 581 ]<=0;
white[ 582 ]<=0;
white[ 583 ]<=0;
white[ 584 ]<=0;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=1;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=0;
white[ 614 ]<=0;
white[ 615 ]<=0;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
```

```
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=0;
white[ 635 ]<=0;
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=0;
white[ 645 ]<=0;
white[ 646 ]<=0;
white[ 647 ]<=0;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=0;
white[ 677 ]<=0;
white[ 678 ]<=0;
```

```
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=0;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
```

```
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
```

```
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=1;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
```

```
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=1;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=1;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=1;
white[ 899 ]<=1;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
```

```
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=1;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=1;
white[ 931 ]<=1;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=0;
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
```

```
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=1;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

Y:begin
```

```
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
```

```
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
```

```
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=0;
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=1;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=0;
white[ 168 ]<=0;
white[ 169 ]<=0;
white[ 170 ]<=0;
white[ 171 ]<=0;
white[ 172 ]<=0;
white[ 173 ]<=0;
```

```
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=0;
white[ 180 ]<=0;
white[ 181 ]<=0;
white[ 182 ]<=0;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=0;
white[ 206 ]<=0;
white[ 207 ]<=0;
white[ 208 ]<=0;
white[ 209 ]<=0;
white[ 210 ]<=0;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=1;
white[ 230 ]<=1;
white[ 231 ]<=1;
```

```
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=0;
white[ 240 ]<=0;
white[ 241 ]<=0;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
```

```
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=1;
white[ 324 ]<=0;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
```

```
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=0;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
white[ 389 ]<=0;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
```

```
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=0;
white[ 423 ]<=1;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=0;
white[ 454 ]<=0;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
```

```
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=0;
white[ 487 ]<=0;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=0;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
```

```
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=0;
white[ 551 ]<=0;
white[ 552 ]<=0;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=0;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
```

```
white[ 580 ]<=0;
white[ 581 ]<=0;
white[ 582 ]<=0;
white[ 583 ]<=0;
white[ 584 ]<=0;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=1;
white[ 598 ]<=1;
white[ 599 ]<=1;
white[ 600 ]<=0;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=0;
white[ 614 ]<=0;
white[ 615 ]<=0;
white[ 616 ]<=0;
white[ 617 ]<=0;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=0;
white[ 632 ]<=0;
white[ 633 ]<=0;
white[ 634 ]<=0;
white[ 635 ]<=0;
white[ 636 ]<=0;
white[ 637 ]<=0;
```

```
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=0;
white[ 645 ]<=0;
white[ 646 ]<=0;
white[ 647 ]<=0;
white[ 648 ]<=0;
white[ 649 ]<=1;
white[ 650 ]<=0;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=1;
white[ 659 ]<=1;
white[ 660 ]<=1;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=0;
white[ 664 ]<=0;
white[ 665 ]<=0;
white[ 666 ]<=0;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=0;
white[ 677 ]<=0;
white[ 678 ]<=0;
white[ 679 ]<=0;
white[ 680 ]<=0;
white[ 681 ]<=1;
white[ 682 ]<=0;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=0;
```

```
white[ 696 ]<=0;
white[ 697 ]<=0;
white[ 698 ]<=0;
white[ 699 ]<=0;
white[ 700 ]<=0;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=0;
white[ 710 ]<=0;
white[ 711 ]<=0;
white[ 712 ]<=0;
white[ 713 ]<=1;
white[ 714 ]<=0;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=0;
white[ 727 ]<=0;
white[ 728 ]<=0;
white[ 729 ]<=0;
white[ 730 ]<=0;
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=0;
white[ 742 ]<=0;
white[ 743 ]<=0;
white[ 744 ]<=0;
white[ 745 ]<=0;
white[ 746 ]<=0;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
```

```
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=0;
white[ 759 ]<=0;
white[ 760 ]<=0;
white[ 761 ]<=0;
white[ 762 ]<=0;
white[ 763 ]<=0;
white[ 764 ]<=0;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=0;
white[ 774 ]<=0;
white[ 775 ]<=0;
white[ 776 ]<=0;
white[ 777 ]<=0;
white[ 778 ]<=0;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=0;
white[ 791 ]<=0;
white[ 792 ]<=0;
white[ 793 ]<=0;
white[ 794 ]<=0;
white[ 795 ]<=0;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=0;
white[ 805 ]<=0;
white[ 806 ]<=0;
white[ 807 ]<=0;
white[ 808 ]<=0;
white[ 809 ]<=0;
white[ 810 ]<=0;
white[ 811 ]<=1;
```

```
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=0;
white[ 823 ]<=0;
white[ 824 ]<=0;
white[ 825 ]<=0;
white[ 826 ]<=0;
white[ 827 ]<=0;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=0;
white[ 839 ]<=0;
white[ 840 ]<=0;
white[ 841 ]<=0;
white[ 842 ]<=0;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=0;
white[ 855 ]<=0;
white[ 856 ]<=0;
white[ 857 ]<=0;
white[ 858 ]<=0;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=0;
```

```
white[ 870 ]<=0;
white[ 871 ]<=0;
white[ 872 ]<=0;
white[ 873 ]<=0;
white[ 874 ]<=0;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=0;
white[ 887 ]<=0;
white[ 888 ]<=0;
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=0;
white[ 905 ]<=0;
white[ 906 ]<=0;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=0;
white[ 918 ]<=0;
white[ 919 ]<=0;
white[ 920 ]<=0;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
```

```
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
```

```
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=0;
white[ 1005 ]<=0;
white[ 1006 ]<=0;
white[ 1007 ]<=0;
white[ 1008 ]<=0;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
end

Z: begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
```

```
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
```

```
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=1;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
```

```
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=0;
white[ 155 ]<=1;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=1;
white[ 168 ]<=1;
white[ 169 ]<=1;
white[ 170 ]<=1;
white[ 171 ]<=1;
white[ 172 ]<=1;
white[ 173 ]<=1;
white[ 174 ]<=1;
white[ 175 ]<=1;
white[ 176 ]<=1;
white[ 177 ]<=1;
white[ 178 ]<=1;
white[ 179 ]<=1;
white[ 180 ]<=1;
white[ 181 ]<=1;
white[ 182 ]<=1;
white[ 183 ]<=1;
white[ 184 ]<=1;
white[ 185 ]<=1;
white[ 186 ]<=1;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
```

```
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=1;
white[ 197 ]<=0;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=0;
white[ 230 ]<=1;
white[ 231 ]<=1;
white[ 232 ]<=1;
white[ 233 ]<=1;
white[ 234 ]<=1;
white[ 235 ]<=1;
white[ 236 ]<=1;
white[ 237 ]<=1;
white[ 238 ]<=1;
white[ 239 ]<=1;
white[ 240 ]<=1;
white[ 241 ]<=1;
white[ 242 ]<=1;
white[ 243 ]<=1;
white[ 244 ]<=1;
white[ 245 ]<=1;
white[ 246 ]<=1;
white[ 247 ]<=1;
white[ 248 ]<=1;
```

```
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=0;
white[ 260 ]<=0;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=1;
white[ 294 ]<=1;
white[ 295 ]<=1;
white[ 296 ]<=1;
white[ 297 ]<=1;
white[ 298 ]<=1;
white[ 299 ]<=1;
white[ 300 ]<=1;
white[ 301 ]<=1;
white[ 302 ]<=1;
white[ 303 ]<=1;
white[ 304 ]<=1;
white[ 305 ]<=1;
white[ 306 ]<=1;
```

```
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=0;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=0;
white[ 319 ]<=0;
white[ 320 ]<=0;
white[ 321 ]<=0;
white[ 322 ]<=0;
white[ 323 ]<=0;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=0;
white[ 346 ]<=0;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=1;
white[ 357 ]<=1;
white[ 358 ]<=1;
white[ 359 ]<=1;
white[ 360 ]<=1;
white[ 361 ]<=1;
white[ 362 ]<=1;
white[ 363 ]<=1;
white[ 364 ]<=1;
```

```
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=0;
white[ 378 ]<=0;
white[ 379 ]<=0;
white[ 380 ]<=0;
white[ 381 ]<=0;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=0;
white[ 395 ]<=0;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=0;
white[ 409 ]<=0;
white[ 410 ]<=1;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=1;
white[ 421 ]<=1;
white[ 422 ]<=1;
```

```
white[ 423 ]<=1;
white[ 424 ]<=0;
white[ 425 ]<=0;
white[ 426 ]<=0;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=0;
white[ 441 ]<=0;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=0;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=0;
white[ 454 ]<=0;
white[ 455 ]<=0;
white[ 456 ]<=0;
white[ 457 ]<=0;
white[ 458 ]<=0;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=0;
white[ 472 ]<=0;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
```

```
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=0;
white[ 487 ]<=1;
white[ 488 ]<=0;
white[ 489 ]<=0;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=0;
white[ 503 ]<=0;
white[ 504 ]<=0;
white[ 505 ]<=0;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=0;
white[ 515 ]<=0;
white[ 516 ]<=0;
white[ 517 ]<=0;
white[ 518 ]<=0;
white[ 519 ]<=0;
white[ 520 ]<=0;
white[ 521 ]<=0;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=0;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
```

```
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=0;
white[ 551 ]<=0;
white[ 552 ]<=0;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=0;
white[ 566 ]<=0;
white[ 567 ]<=0;
white[ 568 ]<=0;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=0;
white[ 574 ]<=0;
white[ 575 ]<=0;
white[ 576 ]<=0;
white[ 577 ]<=0;
white[ 578 ]<=0;
white[ 579 ]<=0;
white[ 580 ]<=0;
white[ 581 ]<=0;
white[ 582 ]<=0;
white[ 583 ]<=0;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=0;
```

```
white[ 597 ]<=1;
white[ 598 ]<=0;
white[ 599 ]<=0;
white[ 600 ]<=0;
white[ 601 ]<=1;
white[ 602 ]<=1;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=0;
white[ 614 ]<=0;
white[ 615 ]<=0;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=0;
white[ 628 ]<=0;
white[ 629 ]<=0;
white[ 630 ]<=0;
white[ 631 ]<=0;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=0;
white[ 637 ]<=0;
white[ 638 ]<=0;
white[ 639 ]<=0;
white[ 640 ]<=0;
white[ 641 ]<=0;
white[ 642 ]<=0;
white[ 643 ]<=0;
white[ 644 ]<=0;
white[ 645 ]<=0;
white[ 646 ]<=0;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
```

```
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=0;
white[ 659 ]<=0;
white[ 660 ]<=0;
white[ 661 ]<=1;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=0;
white[ 676 ]<=0;
white[ 677 ]<=0;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=0;
white[ 702 ]<=0;
white[ 703 ]<=0;
white[ 704 ]<=0;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=0;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
```

```
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=1;
white[ 719 ]<=1;
white[ 720 ]<=1;
white[ 721 ]<=1;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=0;
white[ 734 ]<=0;
white[ 735 ]<=0;
white[ 736 ]<=0;
white[ 737 ]<=0;
white[ 738 ]<=0;
white[ 739 ]<=0;
white[ 740 ]<=0;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=0;
white[ 766 ]<=0;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
```

```
white[ 771 ]<=0;
white[ 772 ]<=0;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=1;
white[ 779 ]<=1;
white[ 780 ]<=1;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=0;
white[ 797 ]<=0;
white[ 798 ]<=0;
white[ 799 ]<=0;
white[ 800 ]<=0;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=0;
```

```
white[ 829 ]<=0;
white[ 830 ]<=0;
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=1;
white[ 839 ]<=1;
white[ 840 ]<=1;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=0;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
```

```
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=1;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=0;
white[ 911 ]<=0;
white[ 912 ]<=0;
white[ 913 ]<=0;
white[ 914 ]<=0;
white[ 915 ]<=0;
white[ 916 ]<=0;
white[ 917 ]<=0;
white[ 918 ]<=0;
white[ 919 ]<=0;
white[ 920 ]<=0;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=0;
white[ 929 ]<=0;
white[ 930 ]<=0;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=0;
white[ 940 ]<=0;
white[ 941 ]<=0;
white[ 942 ]<=0;
white[ 943 ]<=0;
white[ 944 ]<=0;
```

```
white[ 945 ]<=0;
white[ 946 ]<=0;
white[ 947 ]<=0;
white[ 948 ]<=0;
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=0;
white[ 973 ]<=0;
white[ 974 ]<=0;
white[ 975 ]<=0;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=0;
white[ 989 ]<=0;
white[ 990 ]<=0;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
```

```
                white[ 1003 ]<=0;
                white[ 1004 ]<=0;
                white[ 1005 ]<=0;
                white[ 1006 ]<=0;
                white[ 1007 ]<=0;
                white[ 1008 ]<=0;
                white[ 1009 ]<=0;
                white[ 1010 ]<=0;
                white[ 1011 ]<=0;
                white[ 1012 ]<=0;
                white[ 1013 ]<=0;
                white[ 1014 ]<=0;
                white[ 1015 ]<=0;
                white[ 1016 ]<=0;
                white[ 1017 ]<=0;
                white[ 1018 ]<=0;
                white[ 1019 ]<=0;
                white[ 1020 ]<=0;
                white[ 1021 ]<=0;
                white[ 1022 ]<=0;
                white[ 1023 ]<=0;
                end
                endcase
        end
end

always@(posedge iclk or negedge ireset_n)
begin
    if(!ireset_n)
        begin
            address <= 18'bZ;
        end
    else
        case(iletter)
        A: address <= A_base;
        B: address <= B_base;
        C: address <= C_base;
        D: address <= D_base;
        E: address <= E_base;
        F: address <= F_base;
        G: address <= G_base;
        H: address <= H_base;
        I: address <= I_base;
        J: address <= J_base;
        K: address <= K_base;
        L: address <= L_base;
        M: address <= M_base;
        N: address <= N_base;
        O: address <= O_base;
        P: address <= P_base;
        Q: address <= Q_base;
        R: address <= R_base;
        S: address <= S_base;
        T: address <= T_base;
        U: address <= U_base;
        V: address <= V_base;
        W: address <= W_base;
        X: address <= X_base;
```

```
            Y: address <= Y_base;
            Z: address <= Z_base;
        endcase
end

always@(posedge iclk or negedge ireset_n)
begin
        if(!ireset_n)
        begin
            oOVERRIDE<= 0;
            oADDR <= 18'bZ;
            counter <= 5'd31;
        end
        else if(display_area)
        begin
            if(counter == 0)
                counter <= 5'd31;
            else
                counter <= counter-1;
            oOVERRIDE <= white[counter + 32*(iy_cnt-iletter_pos_y)];
            oADDR <= address + counter + 32*(iy_cnt-iletter_pos_y);
        end
        else
        begin
            oOVERRIDE <= 0;
            oADDR <= 18'bZ;
        end
end

endmodule
```

life.v

```
module Life(
                clk_ltm,
                reset_n,
                x_cnt,
                y_cnt,
                life,
                oOVERRIDE,
                oADDR
                );

input clk_ltm;
input reset_n;
input [10:0] x_cnt;
input [9:0] y_cnt;
input [3:0]life;
output oOVERRIDE;
output [17:0] oADDR;
//PARAMETER
parameter BASE_ADDR = 89025;
parameter INIT = 2'b00;
parameter START = 2'b01;
parameter PAUSE = 2'b10;
parameter H_LINE = 1056;
```

```
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
parameter heart_base = 18'd92639;
//WIRE & REG
wire [17:0] address;
wire [1:0] OVERRIDE;
symbol heart(
                        .iclk(clk_ltm),
                        .ireset_n(reset_n),
                        .ix_cnt(x_cnt),
                        .iy_cnt(y_cnt),
                        .isymbol_pos_x(Hsync_Blank-1+730),
                        .isymbol_pos_y(Vertical_Back_Porch+90),
                        .iBASE_ADDR(heart_base),
                        .oOVERRIDE(OVERRIDE[1]),
                        .oADDR(address)
);

number life1(
                        .iclk(clk_ltm),
                        .iReset_n(reset_n),
                        .ixcount(x_cnt),
                        .iycount(y_cnt),
                        .iled_pos_x(Hsync_Blank-1+600),
                        .iled_pos_y(Vertical_Back_Porch+80),
                        .inumber(life),
                        .oOVERRIDE(OVERRIDE[0])
);

////OUTPUT
assign oOVERRIDE = OVERRIDE[0] || OVERRIDE[1];
assign oADDR = OVERRIDE[0] ? BASE_ADDR :
                            OVERRIDE[1] ? address : 18'bZ;

endmodule
```

lose.v

```
module lose(
                        iclk,
                        ireset_n,
                        ix_cnt,
                        iy_cnt,
                        oADDR);

//=============================================
//Parameter=============================
//=============================================
parameter A = 5'd1;
parameter B = 5'd2;
parameter C = 5'd3;
parameter D = 5'd4;
```

```
parameter E = 5'd5;
parameter F = 5'd6;
parameter G = 5'd7;
parameter H = 5'd8;
parameter I = 5'd9;
parameter J = 5'd10;
parameter K = 5'd11;
parameter L = 5'd12;
parameter M = 5'd13;
parameter N = 5'd14;
parameter O = 5'd15;
parameter P = 5'd16;
parameter Q = 5'd17;
parameter R = 5'd18;
parameter S = 5'd19;
parameter T = 5'd20;
parameter U = 5'd21;
parameter V = 5'd22;
parameter W = 5'd23;
parameter X = 5'd24;
parameter Y = 5'd25;
parameter Z = 5'd26;

parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
//=============================================
//I/O DECALARATION=============================
//=============================================
input iclk;
input ireset_n;
input [10:0] ix_cnt;
input [9:0] iy_cnt;
output [17:0] oADDR;

//=============================================
//WIRE & REG===================================
//=============================================
wire [4:0] letter[0:20];
wire [10:0] letter_pos_x[0:20];
wire [9:0] letter_pos_y[0:20];
wire [20:0] override;
wire [17:0] address[0:20];
reg [17:0] oADDR;
//=============================================
//Structured logic=============================
//=============================================
//Assign Letters
assign letter[0] = Y;
assign letter[1] = O;
assign letter[2] = U;
assign letter[3] = L;
assign letter[4] = O;
assign letter[5] = S;
assign letter[6] = E;
```

```
assign letter[7] = P;
assign letter[8] = L;
assign letter[9] = E;
assign letter[10] = A;
assign letter[11] = S;
assign letter[12] = E;
assign letter[13] = T;
assign letter[14] = R;
assign letter[15] = Y;
assign letter[16] = A;
assign letter[17] = G;
assign letter[18] = A;
assign letter[19] = I;
assign letter[20] = N;


//assign letter position
assign letter_pos_x[0] = Hsync_Blank-2+496;
assign letter_pos_x[1] = Hsync_Blank-2+464;
assign letter_pos_x[2] = Hsync_Blank-2+432;
assign letter_pos_x[3] = Hsync_Blank-2+368;
assign letter_pos_x[4] = Hsync_Blank-2+336;
assign letter_pos_x[5] = Hsync_Blank-2+304;
assign letter_pos_x[6] = Hsync_Blank-2+272;

assign letter_pos_x[7] = Hsync_Blank-2+624;
assign letter_pos_x[8] = Hsync_Blank-2+592;
assign letter_pos_x[9] = Hsync_Blank-2+560;
assign letter_pos_x[10] = Hsync_Blank-2+528;
assign letter_pos_x[11] = Hsync_Blank-2+496;
assign letter_pos_x[12] = Hsync_Blank-2+464;

assign letter_pos_x[13] = Hsync_Blank-2+400;
assign letter_pos_x[14] = Hsync_Blank-2+368;
assign letter_pos_x[15] = Hsync_Blank-2+336;

assign letter_pos_x[16] = Hsync_Blank-2+272;
assign letter_pos_x[17] = Hsync_Blank-2+240;
assign letter_pos_x[18] = Hsync_Blank-2+208;
assign letter_pos_x[19] = Hsync_Blank-2+176;
assign letter_pos_x[20] = Hsync_Blank-2+144;


assign letter_pos_y[0] = Vertical_Back_Porch-1+160;
assign letter_pos_y[1] = Vertical_Back_Porch-1+160;
assign letter_pos_y[2] = Vertical_Back_Porch-1+160;
assign letter_pos_y[3] = Vertical_Back_Porch-1+160;
assign letter_pos_y[4] = Vertical_Back_Porch-1+160;
assign letter_pos_y[5] = Vertical_Back_Porch-1+160;
assign letter_pos_y[6] = Vertical_Back_Porch-1+160;

assign letter_pos_y[7] = Vertical_Back_Porch-1+240;
assign letter_pos_y[8] = Vertical_Back_Porch-1+240;
assign letter_pos_y[9] = Vertical_Back_Porch-1+240;
assign letter_pos_y[10] = Vertical_Back_Porch-1+240;
assign letter_pos_y[11] = Vertical_Back_Porch-1+240;
assign letter_pos_y[12] = Vertical_Back_Porch-1+240;
assign letter_pos_y[13] = Vertical_Back_Porch-1+240;
```

```
assign letter_pos_y[14] = Vertical_Back_Porch-1+240;
assign letter_pos_y[15] = Vertical_Back_Porch-1+240;
assign letter_pos_y[16] = Vertical_Back_Porch-1+240;
assign letter_pos_y[17] = Vertical_Back_Porch-1+240;
assign letter_pos_y[18] = Vertical_Back_Porch-1+240;
assign letter_pos_y[19] = Vertical_Back_Porch-1+240;
assign letter_pos_y[20] = Vertical_Back_Porch-1+240;


letter letter0(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[0]),
                        .iletter_pos_y(letter_pos_y[0]),
                        .iletter(letter[0]),
                        .oOVERRIDE(override[0]),
                        .oADDR(address[0])
                        );

letter letter1(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[1]),
                        .iletter_pos_y(letter_pos_y[1]),
                        .iletter(letter[1]),
                        .oOVERRIDE(override[1]),
                        .oADDR(address[1])
                        );

letter letter2(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[2]),
                        .iletter_pos_y(letter_pos_y[2]),
                        .iletter(letter[2]),
                        .oOVERRIDE(override[2]),
                        .oADDR(address[2])
                        );

letter letter3(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[3]),
                        .iletter_pos_y(letter_pos_y[3]),
                        .iletter(letter[3]),
                        .oOVERRIDE(override[3]),
                        .oADDR(address[3])
                        );

letter letter4(
```

```
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[4]),
                .iletter_pos_y(letter_pos_y[4]),
                .iletter(letter[4]),
                .oOVERRIDE(override[4]),
                .oADDR(address[4])
                );

letter letter5(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[5]),
                .iletter_pos_y(letter_pos_y[5]),
                .iletter(letter[5]),
                .oOVERRIDE(override[5]),
                .oADDR(address[5])
                );

letter letter6(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[6]),
                .iletter_pos_y(letter_pos_y[6]),
                .iletter(letter[6]),
                .oOVERRIDE(override[6]),
                .oADDR(address[6])
                );

letter letter7(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[7]),
                .iletter_pos_y(letter_pos_y[7]),
                .iletter(letter[7]),
                .oOVERRIDE(override[7]),
                .oADDR(address[7])
                );

letter letter8(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[8]),
                .iletter_pos_y(letter_pos_y[8]),
                .iletter(letter[8]),
                .oOVERRIDE(override[8]),
                .oADDR(address[8])
                );
```

```
letter letter9(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[9]),
                        .iletter_pos_y(letter_pos_y[9]),
                        .iletter(letter[9]),
                        .oOVERRIDE(override[9]),
                        .oADDR(address[9])
                        );

letter letter10(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[10]),
                        .iletter_pos_y(letter_pos_y[10]),
                        .iletter(letter[10]),
                        .oOVERRIDE(override[10]),
                        .oADDR(address[10])
                        );

letter letter11(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[11]),
                        .iletter_pos_y(letter_pos_y[11]),
                        .iletter(letter[11]),
                        .oOVERRIDE(override[11]),
                        .oADDR(address[11])
                        );

letter letter12(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[12]),
                        .iletter_pos_y(letter_pos_y[12]),
                        .iletter(letter[12]),
                        .oOVERRIDE(override[12]),
                        .oADDR(address[12])
                        );

letter letter13(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[13]),
                        .iletter_pos_y(letter_pos_y[13]),
                        .iletter(letter[13]),
                        .oOVERRIDE(override[13]),
```

```
                .oADDR(address[13])
                );

letter letter14(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[14]),
                .iletter_pos_y(letter_pos_y[14]),
                .iletter(letter[14]),
                .oOVERRIDE(override[14]),
                .oADDR(address[14])
                );

letter letter15(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[15]),
                .iletter_pos_y(letter_pos_y[15]),
                .iletter(letter[15]),
                .oOVERRIDE(override[15]),
                .oADDR(address[15])
                );

letter letter16(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[16]),
                .iletter_pos_y(letter_pos_y[16]),
                .iletter(letter[16]),
                .oOVERRIDE(override[16]),
                .oADDR(address[16])
                );

letter letter17(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[17]),
                .iletter_pos_y(letter_pos_y[17]),
                .iletter(letter[17]),
                .oOVERRIDE(override[17]),
                .oADDR(address[17])
                );

letter letter18(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[18]),
                .iletter_pos_y(letter_pos_y[18]),
```

```
                        .iletter(letter[18]),
                        .oOVERRIDE(override[18]),
                        .oADDR(address[18])
                        );

letter letter19(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[19]),
                        .iletter_pos_y(letter_pos_y[19]),
                        .iletter(letter[19]),
                        .oOVERRIDE(override[19]),
                        .oADDR(address[19])
                        );

letter letter20(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[20]),
                        .iletter_pos_y(letter_pos_y[20]),
                        .iletter(letter[20]),
                        .oOVERRIDE(override[20]),
                        .oADDR(address[20])
                        );


always@(posedge iclk or negedge ireset_n)
begin
      if(!ireset_n)
            oADDR <= 18'bZ;
      else
            case(override)
            21'b000000000000000000001 : oADDR <= address[0];
            21'b000000000000000000010 : oADDR <= address[1];
            21'b000000000000000000100 : oADDR <= address[2];
            21'b000000000000000001000 : oADDR <= address[3];
            21'b000000000000000010000 : oADDR <= address[4];
            21'b000000000000000100000 : oADDR <= address[5];
            21'b000000000000001000000 : oADDR <= address[6];
            21'b000000000000010000000 : oADDR <= address[7];
            21'b000000000000100000000 : oADDR <= address[8];
            21'b000000000001000000000 : oADDR <= address[9];
            21'b000000000010000000000 : oADDR <= address[10];
            21'b000000000100000000000 : oADDR <= address[11];
            21'b000000001000000000000 : oADDR <= address[12];
            21'b000000010000000000000 : oADDR <= address[13];
            21'b000000100000000000000 : oADDR <= address[14];
            21'b000001000000000000000 : oADDR <= address[15];
            21'b000010000000000000000 : oADDR <= address[16];
            21'b000100000000000000000 : oADDR <= address[17];
            21'b001000000000000000000 : oADDR <= address[18];
            21'b010000000000000000000 : oADDR <= address[19];
            21'b100000000000000000000 : oADDR <= address[20];
            default: oADDR <= 18'd89025;
```

```
            endcase;
end

endmodule
```

mole_controller.v

```
module mole_ctrl        (
                        iCLK,                   // LCD display clock
                        iRST_n,             // systen reset
                       x_cnt,
                y_cnt,
                        moleControl,
                        moleExist,
                        mole_address,
                        );
//////=================================================
// for every mole control
//    001 mole up, 000 mole down due to time out
//    101 bottle up, 100 bottle down
//    011 evil up, 010 evil down
//    110 mole hit!!!!
//////=================================================

//=============================================================================
// PARAMETER declarations
//=============================================================================
parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
parameter moleBase = 48000;

parameter bottleOffset = 103439-48000;
parameter evilOffset = 117839-48000;
//=============================================================================
// PORT declarations
//=============================================================================
input                   iCLK;
input                   iRST_n;
input [10:0] x_cnt;
input [9:0] y_cnt;
input [31:0]        moleControl;
output          moleExist;
output[17:0] mole_address;

//=============================================================================
// REG/WIRE declarations
//=============================================================================
wire [15:0] area;
wire [1:0] mole0;
wire [1:0] mole1;
wire [1:0] mole2;
```

```
wire [1:0] mole3;
wire [1:0] mole4;
wire [1:0] mole5;
wire [1:0] mole6;
wire [1:0] mole7;
wire [1:0] mole8;
wire updn0;
wire updn1;
wire updn2;
wire updn3;
wire updn4;
wire updn5;
wire updn6;
wire updn7;
wire updn8;

reg [9:0]x_offset;
reg [9:0]y_offset;
reg [9:0]abs_x;
reg [9:0]abs_y;
reg [9:0]bound_x;
reg [6:0]hole [0:14];
reg [6:0]moleLeftBound [0:119];
reg [6:0]moleRightBound [0:119];

reg [6:0]bottleLeftBound [0:119];
reg [6:0]bottleRightBound [0:119];

reg [6:0]evilLeftBound [0:119];
reg [6:0]evilRightBound [0:119];

reg [6:0]moleHeight[0:8];
reg [33:0]counter00;
reg [33:0]counter11;
reg [33:0]counter22;
reg [33:0]counter33;
reg [33:0]counter44;
reg [33:0]counter55;
reg [33:0]counter66;
reg [33:0]counter77;
reg [33:0]counter88;

reg [33:0]counter2;
reg [17:0] mole_address;//32000+1
reg [1:0] moleExist ;

//=============================================================================
// Structural coding
//=============================================================================



//generate the bound of the holes
always@(posedge iCLK    or negedge iRST_n)
      begin
            if (!iRST_n)
                  begin
                      hole[0] <= 0;
```

```
                    hole[1] <= 0;
                    hole[2] <= 0;
                    hole[3] <= 0;
                    hole[4] <= 0;
                    hole[5] <= 0;
                    hole[6] <= 0;
                    hole[7] <= 0;
                    hole[8] <= 0;
                    hole[9] <= 0;
                    hole[10] <= 0;
                    hole[11] <= 0;
                    hole[12] <= 0;
                    hole[13] <= 0;
                    hole[14] <= 0;
                end
        else
            begin
                hole[0] <= 80;
                    hole[1] <= 80;
                    hole[2] <= 80;
                    hole[3] <= 75;
                    hole[4] <= 75;
                    hole[5] <= 75;
                    hole[6] <= 75;
                    hole[7] <= 75;
                    hole[8] <= 75;
                    hole[9] <= 75;
                    hole[10] <= 70;
                    hole[11] <= 70;
                    hole[12] <= 70;
                    hole[13] <= 65;
                    hole[14] <= 65;
            end
    end



//generate the boundmoleHeight of the mole
always@(posedge iCLK or negedge iRST_n)
    begin
        if (!iRST_n)
                begin
                    moleLeftBound[      0     ] <=        0     ;
                      moleLeftBound[    1     ] <=        0     ;
                      moleLeftBound[    2     ] <=        0     ;
                      moleLeftBound[    3     ] <=        0     ;
                      moleLeftBound[    4     ] <=        0     ;
                      moleLeftBound[    5     ] <=        0     ;
                      moleLeftBound[    6     ] <=        0     ;
                      moleLeftBound[    7     ] <=        0     ;
                      moleLeftBound[    8     ] <=        0     ;
                      moleLeftBound[    9     ] <=        0     ;
                      moleLeftBound[    10    ] <=        0     ;
                      moleLeftBound[    11    ] <=        0     ;
                      moleLeftBound[    12    ] <=        0     ;
                      moleLeftBound[    13    ] <=        0     ;
                      moleLeftBound[    14    ] <=        0     ;
                      moleLeftBound[    15    ] <=        0     ;
```

```
moleLeftBound[  16  ] <=  0  ;
moleLeftBound[  17  ] <=  0  ;
moleLeftBound[  18  ] <=  0  ;
moleLeftBound[  19  ] <=  0  ;
moleLeftBound[  20  ] <=  0  ;
moleLeftBound[  21  ] <=  0  ;
moleLeftBound[  22  ] <=  0  ;
moleLeftBound[  23  ] <=  0  ;
moleLeftBound[  24  ] <=  0  ;
moleLeftBound[  25  ] <=  0  ;
moleLeftBound[  26  ] <=  0  ;
moleLeftBound[  27  ] <=  0  ;
moleLeftBound[  28  ] <=  0  ;
moleLeftBound[  29  ] <=  0  ;
moleLeftBound[  30  ] <=  0  ;
moleLeftBound[  31  ] <=  0  ;
moleLeftBound[  32  ] <=  0  ;
moleLeftBound[  33  ] <=  0  ;
moleLeftBound[  34  ] <=  0  ;
moleLeftBound[  35  ] <=  0  ;
moleLeftBound[  36  ] <=  0  ;
moleLeftBound[  37  ] <=  0  ;
moleLeftBound[  38  ] <=  0  ;
moleLeftBound[  39  ] <=  0  ;
moleLeftBound[  40  ] <=  0  ;
moleLeftBound[  41  ] <=  0  ;
moleLeftBound[  42  ] <=  0  ;
moleLeftBound[  43  ] <=  0  ;
moleLeftBound[  44  ] <=  0  ;
moleLeftBound[  45  ] <=  0  ;
moleLeftBound[  46  ] <=  0  ;
moleLeftBound[  47  ] <=  0  ;
moleLeftBound[  48  ] <=  0  ;
moleLeftBound[  49  ] <=  0  ;
moleLeftBound[  50  ] <=  0  ;
moleLeftBound[  51  ] <=  0  ;
moleLeftBound[  52  ] <=  0  ;
moleLeftBound[  53  ] <=  0  ;
moleLeftBound[  54  ] <=  0  ;
moleLeftBound[  55  ] <=  0  ;
moleLeftBound[  56  ] <=  0  ;
moleLeftBound[  57  ] <=  0  ;
moleLeftBound[  58  ] <=  0  ;
moleLeftBound[  59  ] <=  0  ;
moleLeftBound[  60  ] <=  0  ;
moleLeftBound[  61  ] <=  0  ;
moleLeftBound[  62  ] <=  0  ;
moleLeftBound[  63  ] <=  0  ;
moleLeftBound[  64  ] <=  0  ;
moleLeftBound[  65  ] <=  0  ;
moleLeftBound[  66  ] <=  0  ;
moleLeftBound[  67  ] <=  0  ;
moleLeftBound[  68  ] <=  0  ;
moleLeftBound[  69  ] <=  0  ;
moleLeftBound[  70  ] <=  0  ;
moleLeftBound[  71  ] <=  0  ;
moleLeftBound[  72  ] <=  0  ;
moleLeftBound[  73  ] <=  0  ;
```

```
moleLeftBound[   74  ] <=      0   ;
moleLeftBound[   75  ] <=      0   ;
moleLeftBound[   76  ] <=      0   ;
moleLeftBound[   77  ] <=      0   ;
moleLeftBound[   78  ] <=      0   ;
moleLeftBound[   79  ] <=      0   ;
moleLeftBound[   80  ] <=      0   ;
moleLeftBound[   81  ] <=      0   ;
moleLeftBound[   82  ] <=      0   ;
moleLeftBound[   83  ] <=      0   ;
moleLeftBound[   84  ] <=      0   ;
moleLeftBound[   85  ] <=      0   ;
moleLeftBound[   86  ] <=      0   ;
moleLeftBound[   87  ] <=      0   ;
moleLeftBound[   88  ] <=      0   ;
moleLeftBound[   89  ] <=      0   ;
moleLeftBound[   90  ] <=      0   ;
moleLeftBound[   91  ] <=      0   ;
moleLeftBound[   92  ] <=      0   ;
moleLeftBound[   93  ] <=      0   ;
moleLeftBound[   94  ] <=      0   ;
moleLeftBound[   95  ] <=      0   ;
moleLeftBound[   96  ] <=      0   ;
moleLeftBound[   97  ] <=      0   ;
moleLeftBound[   98  ] <=      0   ;
moleLeftBound[   99  ] <=      0   ;
moleLeftBound[   100 ] <=      0   ;
moleLeftBound[   101 ] <=      0   ;
moleLeftBound[   102 ] <=      0   ;
moleLeftBound[   103 ] <=      0   ;
moleLeftBound[   104 ] <=      0   ;
moleLeftBound[   105 ] <=      0   ;
moleLeftBound[   106 ] <=      0   ;
moleLeftBound[   107 ] <=      0   ;
moleLeftBound[   108 ] <=      0   ;
moleLeftBound[   109 ] <=      0   ;
moleLeftBound[   110 ] <=      0   ;
moleLeftBound[   111 ] <=      0   ;
moleLeftBound[   112 ] <=      0   ;
moleLeftBound[   113 ] <=      0   ;
moleLeftBound[   114 ] <=      0   ;
moleLeftBound[   115 ] <=      0   ;
moleLeftBound[   116 ] <=      0   ;
moleLeftBound[   117 ] <=      0   ;
moleLeftBound[   118 ] <=      0   ;
moleLeftBound[   119 ] <=      0   ;

moleRightBound[  0   ] <=      0   ;
moleRightBound[  1   ] <=      0   ;
moleRightBound[  2   ] <=      0   ;
moleRightBound[  3   ] <=      0   ;
moleRightBound[  4   ] <=      0   ;
moleRightBound[  5   ] <=      0   ;
moleRightBound[  6   ] <=      0   ;
moleRightBound[  7   ] <=      0   ;
moleRightBound[  8   ] <=      0   ;
moleRightBound[  9   ] <=      0   ;
moleRightBound[  10  ] <=      0   ;
```

```
moleRightBound[  11  ] <=      0    ;
moleRightBound[  12  ] <=      0    ;
moleRightBound[  13  ] <=      0    ;
moleRightBound[  14  ] <=      0    ;
moleRightBound[  15  ] <=      0    ;
moleRightBound[  16  ] <=      0    ;
moleRightBound[  17  ] <=      0    ;
moleRightBound[  18  ] <=      0    ;
moleRightBound[  19  ] <=      0    ;
moleRightBound[  20  ] <=      0    ;
moleRightBound[  21  ] <=      0    ;
moleRightBound[  22  ] <=      0    ;
moleRightBound[  23  ] <=      0    ;
moleRightBound[  24  ] <=      0    ;
moleRightBound[  25  ] <=      0    ;
moleRightBound[  26  ] <=      0    ;
moleRightBound[  27  ] <=      0    ;
moleRightBound[  28  ] <=      0    ;
moleRightBound[  29  ] <=      0    ;
moleRightBound[  30  ] <=      0    ;
moleRightBound[  31  ] <=      0    ;
moleRightBound[  32  ] <=      0    ;
moleRightBound[  33  ] <=      0    ;
moleRightBound[  34  ] <=      0    ;
moleRightBound[  35  ] <=      0    ;
moleRightBound[  36  ] <=      0    ;
moleRightBound[  37  ] <=      0    ;
moleRightBound[  38  ] <=      0    ;
moleRightBound[  39  ] <=      0    ;
moleRightBound[  40  ] <=      0    ;
moleRightBound[  41  ] <=      0    ;
moleRightBound[  42  ] <=      0    ;
moleRightBound[  43  ] <=      0    ;
moleRightBound[  44  ] <=      0    ;
moleRightBound[  45  ] <=      0    ;
moleRightBound[  46  ] <=      0    ;
moleRightBound[  47  ] <=      0    ;
moleRightBound[  48  ] <=      0    ;
moleRightBound[  49  ] <=      0    ;
moleRightBound[  50  ] <=      0    ;
moleRightBound[  51  ] <=      0    ;
moleRightBound[  52  ] <=      0    ;
moleRightBound[  53  ] <=      0    ;
moleRightBound[  54  ] <=      0    ;
moleRightBound[  55  ] <=      0    ;
moleRightBound[  56  ] <=      0    ;
moleRightBound[  57  ] <=      0    ;
moleRightBound[  58  ] <=      0    ;
moleRightBound[  59  ] <=      0    ;
moleRightBound[  60  ] <=      0    ;
moleRightBound[  61  ] <=      0    ;
moleRightBound[  62  ] <=      0    ;
moleRightBound[  63  ] <=      0    ;
moleRightBound[  64  ] <=      0    ;
moleRightBound[  65  ] <=      0    ;
moleRightBound[  66  ] <=      0    ;
moleRightBound[  67  ] <=      0    ;
moleRightBound[  68  ] <=      0    ;
```

```
                moleRightBound[  69  ] <=      0   ;
                moleRightBound[  70  ] <=      0   ;
                moleRightBound[  71  ] <=      0   ;
                moleRightBound[  72  ] <=      0   ;
                moleRightBound[  73  ] <=      0   ;
                moleRightBound[  74  ] <=      0   ;
                moleRightBound[  75  ] <=      0   ;
                moleRightBound[  76  ] <=      0   ;
                moleRightBound[  77  ] <=      0   ;
                moleRightBound[  78  ] <=      0   ;
                moleRightBound[  79  ] <=      0   ;
                moleRightBound[  80  ] <=      0   ;
                moleRightBound[  81  ] <=      0   ;
                moleRightBound[  82  ] <=      0   ;
                moleRightBound[  83  ] <=      0   ;
                moleRightBound[  84  ] <=      0   ;
                moleRightBound[  85  ] <=      0   ;
                moleRightBound[  86  ] <=      0   ;
                moleRightBound[  87  ] <=      0   ;
                moleRightBound[  88  ] <=      0   ;
                moleRightBound[  89  ] <=      0   ;
                moleRightBound[  90  ] <=      0   ;
                moleRightBound[  91  ] <=      0   ;
                moleRightBound[  92  ] <=      0   ;
                moleRightBound[  93  ] <=      0   ;
                moleRightBound[  94  ] <=      0   ;
                moleRightBound[  95  ] <=      0   ;
                moleRightBound[  96  ] <=      0   ;
                moleRightBound[  97  ] <=      0   ;
                moleRightBound[  98  ] <=      0   ;
                moleRightBound[  99  ] <=      0   ;
                moleRightBound[  100 ] <=      0   ;
                moleRightBound[  101 ] <=      0   ;
                moleRightBound[  102 ] <=      0   ;
                moleRightBound[  103 ] <=      0   ;
                moleRightBound[  104 ] <=      0   ;
                moleRightBound[  105 ] <=      0   ;
                moleRightBound[  106 ] <=      0   ;
                moleRightBound[  107 ] <=      0   ;
                moleRightBound[  108 ] <=      0   ;
                moleRightBound[  109 ] <=      0   ;
                moleRightBound[  110 ] <=      0   ;
                moleRightBound[  111 ] <=      0   ;
                moleRightBound[  112 ] <=      0   ;
                moleRightBound[  113 ] <=      0   ;
                moleRightBound[  114 ] <=      0   ;
                moleRightBound[  115 ] <=      0   ;
                moleRightBound[  116 ] <=      0   ;
                moleRightBound[  117 ] <=      0   ;
                moleRightBound[  118 ] <=      0   ;
                moleRightBound[  119 ] <=      0   ;

            end
    else
        begin
             moleLeftBound[   0    ] <=      119 ;
                moleLeftBound[    1    ] <=      119 ;
                moleLeftBound[    2    ] <=      54  ;
```

```
moleLeftBound[  3  ] <=  50  ;
moleLeftBound[  4  ] <=  47  ;
moleLeftBound[  5  ] <=  45  ;
moleLeftBound[  6  ] <=  43  ;
moleLeftBound[  7  ] <=  41  ;
moleLeftBound[  8  ] <=  40  ;
moleLeftBound[  9  ] <=  38  ;
moleLeftBound[  10 ] <=  37  ;
moleLeftBound[  11 ] <=  36  ;
moleLeftBound[  12 ] <=  35  ;
moleLeftBound[  13 ] <=  34  ;
moleLeftBound[  14 ] <=  33  ;
moleLeftBound[  15 ] <=  32  ;
moleLeftBound[  16 ] <=  32  ;
moleLeftBound[  17 ] <=  31  ;
moleLeftBound[  18 ] <=  30  ;
moleLeftBound[  19 ] <=  30  ;
moleLeftBound[  20 ] <=  29  ;
moleLeftBound[  21 ] <=  28  ;
moleLeftBound[  22 ] <=  28  ;
moleLeftBound[  23 ] <=  28  ;
moleLeftBound[  24 ] <=  27  ;
moleLeftBound[  25 ] <=  27  ;
moleLeftBound[  26 ] <=  26  ;
moleLeftBound[  27 ] <=  26  ;
moleLeftBound[  28 ] <=  26  ;
moleLeftBound[  29 ] <=  25  ;
moleLeftBound[  30 ] <=  25  ;
moleLeftBound[  31 ] <=  25  ;
moleLeftBound[  32 ] <=  25  ;
moleLeftBound[  33 ] <=  24  ;
moleLeftBound[  34 ] <=  23  ;
moleLeftBound[  35 ] <=  22  ;
moleLeftBound[  36 ] <=  21  ;
moleLeftBound[  37 ] <=  21  ;
moleLeftBound[  38 ] <=  20  ;
moleLeftBound[  39 ] <=  20  ;
moleLeftBound[  40 ] <=  18  ;
moleLeftBound[  41 ] <=  16  ;
moleLeftBound[  42 ] <=  14  ;
moleLeftBound[  43 ] <=  13  ;
moleLeftBound[  44 ] <=  12  ;
moleLeftBound[  45 ] <=  11  ;
moleLeftBound[  46 ] <=  10  ;
moleLeftBound[  47 ] <=  10  ;
moleLeftBound[  48 ] <=  10  ;
moleLeftBound[  49 ] <=  10  ;
moleLeftBound[  50 ] <=  9   ;
moleLeftBound[  51 ] <=  9   ;
moleLeftBound[  52 ] <=  10  ;
moleLeftBound[  53 ] <=  10  ;
moleLeftBound[  54 ] <=  10  ;
moleLeftBound[  55 ] <=  11  ;
moleLeftBound[  56 ] <=  11  ;
moleLeftBound[  57 ] <=  13  ;
moleLeftBound[  58 ] <=  15  ;
moleLeftBound[  59 ] <=  18  ;
moleLeftBound[  60 ] <=  18  ;
```

```
moleLeftBound[  61 ] <=  18 ;
moleLeftBound[  62 ] <=  18 ;
moleLeftBound[  63 ] <=  17 ;
moleLeftBound[  64 ] <=  17 ;
moleLeftBound[  65 ] <=  17 ;
moleLeftBound[  66 ] <=  17 ;
moleLeftBound[  67 ] <=  17 ;
moleLeftBound[  68 ] <=  17 ;
moleLeftBound[  69 ] <=  16 ;
moleLeftBound[  70 ] <=  16 ;
moleLeftBound[  71 ] <=  16 ;
moleLeftBound[  72 ] <=  16 ;
moleLeftBound[  73 ] <=  16 ;
moleLeftBound[  74 ] <=  16 ;
moleLeftBound[  75 ] <=  16 ;
moleLeftBound[  76 ] <=  15 ;
moleLeftBound[  77 ] <=  15 ;
moleLeftBound[  78 ] <=  15 ;
moleLeftBound[  79 ] <=  15 ;
moleLeftBound[  80 ] <=  15 ;
moleLeftBound[  81 ] <=  15 ;
moleLeftBound[  82 ] <=  15 ;
moleLeftBound[  83 ] <=  15 ;
moleLeftBound[  84 ] <=  15 ;
moleLeftBound[  85 ] <=  14 ;
moleLeftBound[  86 ] <=  14 ;
moleLeftBound[  87 ] <=  14 ;
moleLeftBound[  88 ] <=  14 ;
moleLeftBound[  89 ] <=  14 ;
moleLeftBound[  90 ] <=  14 ;
moleLeftBound[  91 ] <=  14 ;
moleLeftBound[  92 ] <=  14 ;
moleLeftBound[  93 ] <=  14 ;
moleLeftBound[  94 ] <=  14 ;
moleLeftBound[  95 ] <=  14 ;
moleLeftBound[  96 ] <=  14 ;
moleLeftBound[  97 ] <=  14 ;
moleLeftBound[  98 ] <=  14 ;
moleLeftBound[  99 ] <=  14 ;
moleLeftBound[ 100 ] <=  14 ;
moleLeftBound[ 101 ] <=  14 ;
moleLeftBound[ 102 ] <=  14 ;
moleLeftBound[ 103 ] <=  14 ;
moleLeftBound[ 104 ] <=  14 ;
moleLeftBound[ 105 ] <=  14 ;
moleLeftBound[ 106 ] <=  14 ;
moleLeftBound[ 107 ] <=  14 ;
moleLeftBound[ 108 ] <=  14 ;
moleLeftBound[ 109 ] <=  14 ;
moleLeftBound[ 110 ] <=  14 ;
moleLeftBound[ 111 ] <=  14 ;
moleLeftBound[ 112 ] <=  14 ;
moleLeftBound[ 113 ] <=  14 ;
moleLeftBound[ 114 ] <=  14 ;
moleLeftBound[ 115 ] <=  14 ;
moleLeftBound[ 116 ] <=  14 ;
moleLeftBound[ 117 ] <=  15 ;
moleLeftBound[ 118 ] <=  15 ;
```

```
moleLeftBound[   119 ] <=      10  ;

moleRightBound[  0   ] <=      119 ;
moleRightBound[  1   ] <=      119 ;
moleRightBound[  2   ] <=      67  ;
moleRightBound[  3   ] <=      71  ;
moleRightBound[  4   ] <=      74  ;
moleRightBound[  5   ] <=      76  ;
moleRightBound[  6   ] <=      78  ;
moleRightBound[  7   ] <=      80  ;
moleRightBound[  8   ] <=      81  ;
moleRightBound[  9   ] <=      82  ;
moleRightBound[  10  ] <=      83  ;
moleRightBound[  11  ] <=      85  ;
moleRightBound[  12  ] <=      86  ;
moleRightBound[  13  ] <=      87  ;
moleRightBound[  14  ] <=      88  ;
moleRightBound[  15  ] <=      89  ;
moleRightBound[  16  ] <=      90  ;
moleRightBound[  17  ] <=      90  ;
moleRightBound[  18  ] <=      91  ;
moleRightBound[  19  ] <=      92  ;
moleRightBound[  20  ] <=      92  ;
moleRightBound[  21  ] <=      93  ;
moleRightBound[  22  ] <=      93  ;
moleRightBound[  23  ] <=      94  ;
moleRightBound[  24  ] <=      94  ;
moleRightBound[  25  ] <=      95  ;
moleRightBound[  26  ] <=      95  ;
moleRightBound[  27  ] <=      95  ;
moleRightBound[  28  ] <=      96  ;
moleRightBound[  29  ] <=      96  ;
moleRightBound[  30  ] <=      96  ;
moleRightBound[  31  ] <=      96  ;
moleRightBound[  32  ] <=      96  ;
moleRightBound[  33  ] <=      97  ;
moleRightBound[  34  ] <=      98  ;
moleRightBound[  35  ] <=      99  ;
moleRightBound[  36  ] <=      100 ;
moleRightBound[  37  ] <=      101 ;
moleRightBound[  38  ] <=      101 ;
moleRightBound[  39  ] <=      101 ;
moleRightBound[  40  ] <=      103 ;
moleRightBound[  41  ] <=      105 ;
moleRightBound[  42  ] <=      107 ;
moleRightBound[  43  ] <=      108 ;
moleRightBound[  44  ] <=      109 ;
moleRightBound[  45  ] <=      110 ;
moleRightBound[  46  ] <=      111 ;
moleRightBound[  47  ] <=      111 ;
moleRightBound[  48  ] <=      111 ;
moleRightBound[  49  ] <=      111 ;
moleRightBound[  50  ] <=      112 ;
moleRightBound[  51  ] <=      112 ;
moleRightBound[  52  ] <=      111 ;
moleRightBound[  53  ] <=      111 ;
moleRightBound[  54  ] <=      111 ;
moleRightBound[  55  ] <=      110 ;
```

```
moleRightBound[  56  ] <=      110 ;
moleRightBound[  57  ] <=      108 ;
moleRightBound[  58  ] <=      106 ;
moleRightBound[  59  ] <=      104 ;
moleRightBound[  60  ] <=      104 ;
moleRightBound[  61  ] <=      104 ;
moleRightBound[  62  ] <=      104 ;
moleRightBound[  63  ] <=      105 ;
moleRightBound[  64  ] <=      105 ;
moleRightBound[  65  ] <=      105 ;
moleRightBound[  66  ] <=      105 ;
moleRightBound[  67  ] <=      105 ;
moleRightBound[  68  ] <=      105 ;
moleRightBound[  69  ] <=      106 ;
moleRightBound[  70  ] <=      106 ;
moleRightBound[  71  ] <=      106 ;
moleRightBound[  72  ] <=      106 ;
moleRightBound[  73  ] <=      106 ;
moleRightBound[  74  ] <=      106 ;
moleRightBound[  75  ] <=      106 ;
moleRightBound[  76  ] <=      107 ;
moleRightBound[  77  ] <=      107 ;
moleRightBound[  78  ] <=      107 ;
moleRightBound[  79  ] <=      107 ;
moleRightBound[  80  ] <=      107 ;
moleRightBound[  81  ] <=      107 ;
moleRightBound[  82  ] <=      107 ;
moleRightBound[  83  ] <=      107 ;
moleRightBound[  84  ] <=      108 ;
moleRightBound[  85  ] <=      108 ;
moleRightBound[  86  ] <=      108 ;
moleRightBound[  87  ] <=      108 ;
moleRightBound[  88  ] <=      108 ;
moleRightBound[  89  ] <=      108 ;
moleRightBound[  90  ] <=      108 ;
moleRightBound[  91  ] <=      108 ;
moleRightBound[  92  ] <=      108 ;
moleRightBound[  93  ] <=      108 ;
moleRightBound[  94  ] <=      108 ;
moleRightBound[  95  ] <=      108 ;
moleRightBound[  96  ] <=      108 ;
moleRightBound[  97  ] <=      108 ;
moleRightBound[  98  ] <=      108 ;
moleRightBound[  99  ] <=      108 ;
moleRightBound[  100 ] <=      108 ;
moleRightBound[  101 ] <=      108 ;
moleRightBound[  102 ] <=      108 ;
moleRightBound[  103 ] <=      108 ;
moleRightBound[  104 ] <=      108 ;
moleRightBound[  105 ] <=      108 ;
moleRightBound[  106 ] <=      108 ;
moleRightBound[  107 ] <=      108 ;
moleRightBound[  108 ] <=      108 ;
moleRightBound[  109 ] <=      108 ;
moleRightBound[  110 ] <=      108 ;
moleRightBound[  111 ] <=      108 ;
moleRightBound[  112 ] <=      108 ;
moleRightBound[  113 ] <=      108 ;
```

```
                moleRightBound[  114 ] <=      108 ;
                moleRightBound[  115 ] <=      108 ;
                moleRightBound[  116 ] <=      108 ;
                moleRightBound[  117 ] <=      108 ;
                moleRightBound[  118 ] <=      108 ;
                moleRightBound[  119 ] <=      108 ;
        end
    end


//generate the boundbottleHeight of the bottle
always@(posedge iCLK or negedge iRST_n)
    begin
        if (!iRST_n)
            begin
                bottleLeftBound[    0    ] <=      0    ;
                bottleLeftBound[    1    ] <=      0    ;
                bottleLeftBound[    2    ] <=      0    ;
                bottleLeftBound[    3    ] <=      0    ;
                bottleLeftBound[    4    ] <=      0    ;
                bottleLeftBound[    5    ] <=      0    ;
                bottleLeftBound[    6    ] <=      0    ;
                bottleLeftBound[    7    ] <=      0    ;
                bottleLeftBound[    8    ] <=      0    ;
                bottleLeftBound[    9    ] <=      0    ;
                bottleLeftBound[    10   ] <=      0    ;
                bottleLeftBound[    11   ] <=      0    ;
                bottleLeftBound[    12   ] <=      0    ;
                bottleLeftBound[    13   ] <=      0    ;
                bottleLeftBound[    14   ] <=      0    ;
                bottleLeftBound[    15   ] <=      0    ;
                bottleLeftBound[    16   ] <=      0    ;
                bottleLeftBound[    17   ] <=      0    ;
                bottleLeftBound[    18   ] <=      0    ;
                bottleLeftBound[    19   ] <=      0    ;
                bottleLeftBound[    20   ] <=      0    ;
                bottleLeftBound[    21   ] <=      0    ;
                bottleLeftBound[    22   ] <=      0    ;
                bottleLeftBound[    23   ] <=      0    ;
                bottleLeftBound[    24   ] <=      0    ;
                bottleLeftBound[    25   ] <=      0    ;
                bottleLeftBound[    26   ] <=      0    ;
                bottleLeftBound[    27   ] <=      0    ;
                bottleLeftBound[    28   ] <=      0    ;
                bottleLeftBound[    29   ] <=      0    ;
                bottleLeftBound[    30   ] <=      0    ;
                bottleLeftBound[    31   ] <=      0    ;
                bottleLeftBound[    32   ] <=      0    ;
                bottleLeftBound[    33   ] <=      0    ;
                bottleLeftBound[    34   ] <=      0    ;
                bottleLeftBound[    35   ] <=      0    ;
                bottleLeftBound[    36   ] <=      0    ;
                bottleLeftBound[    37   ] <=      0    ;
                bottleLeftBound[    38   ] <=      0    ;
                bottleLeftBound[    39   ] <=      0    ;
                bottleLeftBound[    40   ] <=      0    ;
                bottleLeftBound[    41   ] <=      0    ;
                bottleLeftBound[    42   ] <=      0    ;
```

```
bottleLeftBound[  43  ] <=      0   ;
bottleLeftBound[  44  ] <=      0   ;
bottleLeftBound[  45  ] <=      0   ;
bottleLeftBound[  46  ] <=      0   ;
bottleLeftBound[  47  ] <=      0   ;
bottleLeftBound[  48  ] <=      0   ;
bottleLeftBound[  49  ] <=      0   ;
bottleLeftBound[  50  ] <=      0   ;
bottleLeftBound[  51  ] <=      0   ;
bottleLeftBound[  52  ] <=      0   ;
bottleLeftBound[  53  ] <=      0   ;
bottleLeftBound[  54  ] <=      0   ;
bottleLeftBound[  55  ] <=      0   ;
bottleLeftBound[  56  ] <=      0   ;
bottleLeftBound[  57  ] <=      0   ;
bottleLeftBound[  58  ] <=      0   ;
bottleLeftBound[  59  ] <=      0   ;
bottleLeftBound[  60  ] <=      0   ;
bottleLeftBound[  61  ] <=      0   ;
bottleLeftBound[  62  ] <=      0   ;
bottleLeftBound[  63  ] <=      0   ;
bottleLeftBound[  64  ] <=      0   ;
bottleLeftBound[  65  ] <=      0   ;
bottleLeftBound[  66  ] <=      0   ;
bottleLeftBound[  67  ] <=      0   ;
bottleLeftBound[  68  ] <=      0   ;
bottleLeftBound[  69  ] <=      0   ;
bottleLeftBound[  70  ] <=      0   ;
bottleLeftBound[  71  ] <=      0   ;
bottleLeftBound[  72  ] <=      0   ;
bottleLeftBound[  73  ] <=      0   ;
bottleLeftBound[  74  ] <=      0   ;
bottleLeftBound[  75  ] <=      0   ;
bottleLeftBound[  76  ] <=      0   ;
bottleLeftBound[  77  ] <=      0   ;
bottleLeftBound[  78  ] <=      0   ;
bottleLeftBound[  79  ] <=      0   ;
bottleLeftBound[  80  ] <=      0   ;
bottleLeftBound[  81  ] <=      0   ;
bottleLeftBound[  82  ] <=      0   ;
bottleLeftBound[  83  ] <=      0   ;
bottleLeftBound[  84  ] <=      0   ;
bottleLeftBound[  85  ] <=      0   ;
bottleLeftBound[  86  ] <=      0   ;
bottleLeftBound[  87  ] <=      0   ;
bottleLeftBound[  88  ] <=      0   ;
bottleLeftBound[  89  ] <=      0   ;
bottleLeftBound[  90  ] <=      0   ;
bottleLeftBound[  91  ] <=      0   ;
bottleLeftBound[  92  ] <=      0   ;
bottleLeftBound[  93  ] <=      0   ;
bottleLeftBound[  94  ] <=      0   ;
bottleLeftBound[  95  ] <=      0   ;
bottleLeftBound[  96  ] <=      0   ;
bottleLeftBound[  97  ] <=      0   ;
bottleLeftBound[  98  ] <=      0   ;
bottleLeftBound[  99  ] <=      0   ;
bottleLeftBound[  100 ] <=      0   ;
```

```
bottleLeftBound[   101  ] <=       0    ;
bottleLeftBound[   102  ] <=       0    ;
bottleLeftBound[   103  ] <=       0    ;
bottleLeftBound[   104  ] <=       0    ;
bottleLeftBound[   105  ] <=       0    ;
bottleLeftBound[   106  ] <=       0    ;
bottleLeftBound[   107  ] <=       0    ;
bottleLeftBound[   108  ] <=       0    ;
bottleLeftBound[   109  ] <=       0    ;
bottleLeftBound[   110  ] <=       0    ;
bottleLeftBound[   111  ] <=       0    ;
bottleLeftBound[   112  ] <=       0    ;
bottleLeftBound[   113  ] <=       0    ;
bottleLeftBound[   114  ] <=       0    ;
bottleLeftBound[   115  ] <=       0    ;
bottleLeftBound[   116  ] <=       0    ;
bottleLeftBound[   117  ] <=       0    ;
bottleLeftBound[   118  ] <=       0    ;
bottleLeftBound[   119  ] <=       0    ;

bottleRightBound[  0    ] <=       0    ;
bottleRightBound[  1    ] <=       0    ;
bottleRightBound[  2    ] <=       0    ;
bottleRightBound[  3    ] <=       0    ;
bottleRightBound[  4    ] <=       0    ;
bottleRightBound[  5    ] <=       0    ;
bottleRightBound[  6    ] <=       0    ;
bottleRightBound[  7    ] <=       0    ;
bottleRightBound[  8    ] <=       0    ;
bottleRightBound[  9    ] <=       0    ;
bottleRightBound[  10   ] <=       0    ;
bottleRightBound[  11   ] <=       0    ;
bottleRightBound[  12   ] <=       0    ;
bottleRightBound[  13   ] <=       0    ;
bottleRightBound[  14   ] <=       0    ;
bottleRightBound[  15   ] <=       0    ;
bottleRightBound[  16   ] <=       0    ;
bottleRightBound[  17   ] <=       0    ;
bottleRightBound[  18   ] <=       0    ;
bottleRightBound[  19   ] <=       0    ;
bottleRightBound[  20   ] <=       0    ;
bottleRightBound[  21   ] <=       0    ;
bottleRightBound[  22   ] <=       0    ;
bottleRightBound[  23   ] <=       0    ;
bottleRightBound[  24   ] <=       0    ;
bottleRightBound[  25   ] <=       0    ;
bottleRightBound[  26   ] <=       0    ;
bottleRightBound[  27   ] <=       0    ;
bottleRightBound[  28   ] <=       0    ;
bottleRightBound[  29   ] <=       0    ;
bottleRightBound[  30   ] <=       0    ;
bottleRightBound[  31   ] <=       0    ;
bottleRightBound[  32   ] <=       0    ;
bottleRightBound[  33   ] <=       0    ;
bottleRightBound[  34   ] <=       0    ;
bottleRightBound[  35   ] <=       0    ;
bottleRightBound[  36   ] <=       0    ;
bottleRightBound[  37   ] <=       0    ;
```

```
bottleRightBound[ 38 ] <= 0 ;
bottleRightBound[ 39 ] <= 0 ;
bottleRightBound[ 40 ] <= 0 ;
bottleRightBound[ 41 ] <= 0 ;
bottleRightBound[ 42 ] <= 0 ;
bottleRightBound[ 43 ] <= 0 ;
bottleRightBound[ 44 ] <= 0 ;
bottleRightBound[ 45 ] <= 0 ;
bottleRightBound[ 46 ] <= 0 ;
bottleRightBound[ 47 ] <= 0 ;
bottleRightBound[ 48 ] <= 0 ;
bottleRightBound[ 49 ] <= 0 ;
bottleRightBound[ 50 ] <= 0 ;
bottleRightBound[ 51 ] <= 0 ;
bottleRightBound[ 52 ] <= 0 ;
bottleRightBound[ 53 ] <= 0 ;
bottleRightBound[ 54 ] <= 0 ;
bottleRightBound[ 55 ] <= 0 ;
bottleRightBound[ 56 ] <= 0 ;
bottleRightBound[ 57 ] <= 0 ;
bottleRightBound[ 58 ] <= 0 ;
bottleRightBound[ 59 ] <= 0 ;
bottleRightBound[ 60 ] <= 0 ;
bottleRightBound[ 61 ] <= 0 ;
bottleRightBound[ 62 ] <= 0 ;
bottleRightBound[ 63 ] <= 0 ;
bottleRightBound[ 64 ] <= 0 ;
bottleRightBound[ 65 ] <= 0 ;
bottleRightBound[ 66 ] <= 0 ;
bottleRightBound[ 67 ] <= 0 ;
bottleRightBound[ 68 ] <= 0 ;
bottleRightBound[ 69 ] <= 0 ;
bottleRightBound[ 70 ] <= 0 ;
bottleRightBound[ 71 ] <= 0 ;
bottleRightBound[ 72 ] <= 0 ;
bottleRightBound[ 73 ] <= 0 ;
bottleRightBound[ 74 ] <= 0 ;
bottleRightBound[ 75 ] <= 0 ;
bottleRightBound[ 76 ] <= 0 ;
bottleRightBound[ 77 ] <= 0 ;
bottleRightBound[ 78 ] <= 0 ;
bottleRightBound[ 79 ] <= 0 ;
bottleRightBound[ 80 ] <= 0 ;
bottleRightBound[ 81 ] <= 0 ;
bottleRightBound[ 82 ] <= 0 ;
bottleRightBound[ 83 ] <= 0 ;
bottleRightBound[ 84 ] <= 0 ;
bottleRightBound[ 85 ] <= 0 ;
bottleRightBound[ 86 ] <= 0 ;
bottleRightBound[ 87 ] <= 0 ;
bottleRightBound[ 88 ] <= 0 ;
bottleRightBound[ 89 ] <= 0 ;
bottleRightBound[ 90 ] <= 0 ;
bottleRightBound[ 91 ] <= 0 ;
bottleRightBound[ 92 ] <= 0 ;
bottleRightBound[ 93 ] <= 0 ;
bottleRightBound[ 94 ] <= 0 ;
bottleRightBound[ 95 ] <= 0 ;
```

```
                    bottleRightBound[  96   ] <=      0    ;
                    bottleRightBound[  97   ] <=      0    ;
                    bottleRightBound[  98   ] <=      0    ;
                    bottleRightBound[  99   ] <=      0    ;
                    bottleRightBound[  100  ] <=      0    ;
                    bottleRightBound[  101  ] <=      0    ;
                    bottleRightBound[  102  ] <=      0    ;
                    bottleRightBound[  103  ] <=      0    ;
                    bottleRightBound[  104  ] <=      0    ;
                    bottleRightBound[  105  ] <=      0    ;
                    bottleRightBound[  106  ] <=      0    ;
                    bottleRightBound[  107  ] <=      0    ;
                    bottleRightBound[  108  ] <=      0    ;
                    bottleRightBound[  109  ] <=      0    ;
                    bottleRightBound[  110  ] <=      0    ;
                    bottleRightBound[  111  ] <=      0    ;
                    bottleRightBound[  112  ] <=      0    ;
                    bottleRightBound[  113  ] <=      0    ;
                    bottleRightBound[  114  ] <=      0    ;
                    bottleRightBound[  115  ] <=      0    ;
                    bottleRightBound[  116  ] <=      0    ;
                    bottleRightBound[  117  ] <=      0    ;
                    bottleRightBound[  118  ] <=      0    ;
                    bottleRightBound[  119  ] <=      0    ;

                end
        else
            begin
                                                    bottleLeftBound[       0  ]                <=
    52          ;
                                                                    bottleLeftBound[       1
    ] <=       50   ;
                                                                    bottleLeftBound[       2
    ] <=       46   ;
                                                                    bottleLeftBound[       3
    ] <=       46   ;
                                                                    bottleLeftBound[       4
    ] <=       44   ;
                                                                    bottleLeftBound[       5
    ] <=       43   ;
                                                                    bottleLeftBound[       6
    ] <=       42   ;
                                                                    bottleLeftBound[       7
    ] <=       41   ;
                                                                    bottleLeftBound[       8
    ] <=       41   ;
                                                                    bottleLeftBound[       9
    ] <=       40   ;
                                                                    bottleLeftBound[
10  ] <=       40   ;
                                                                    bottleLeftBound[
11  ] <=       40   ;
                                                                    bottleLeftBound[
12  ] <=       40   ;
                                                                    bottleLeftBound[
13  ] <=       39   ;
                                                                    bottleLeftBound[
14  ] <=       38   ;
```

```
bottleLeftBound[ 15 ] <= 37 ;
bottleLeftBound[ 16 ] <= 36 ;
bottleLeftBound[ 17 ] <= 34 ;
bottleLeftBound[ 18 ] <= 33 ;
bottleLeftBound[ 19 ] <= 33 ;
bottleLeftBound[ 20 ] <= 30 ;
bottleLeftBound[ 21 ] <= 31 ;
bottleLeftBound[ 22 ] <= 30 ;
bottleLeftBound[ 23 ] <= 29 ;
bottleLeftBound[ 24 ] <= 28 ;
bottleLeftBound[ 25 ] <= 28 ;
bottleLeftBound[ 26 ] <= 28 ;
bottleLeftBound[ 27 ] <= 28 ;
bottleLeftBound[ 28 ] <= 28 ;
bottleLeftBound[ 29 ] <= 27 ;
bottleLeftBound[ 30 ] <= 27 ;
bottleLeftBound[ 31 ] <= 25 ;
bottleLeftBound[ 32 ] <= 23 ;
bottleLeftBound[ 33 ] <= 22 ;
bottleLeftBound[ 34 ] <= 21 ;
bottleLeftBound[ 35 ] <= 21 ;
bottleLeftBound[ 36 ] <= 20 ;
bottleLeftBound[ 37 ] <= 20 ;
bottleLeftBound[ 38 ] <= 20 ;
bottleLeftBound[ 39 ] <= 20 ;
bottleLeftBound[ 40 ] <= 20 ;
bottleLeftBound[ 41 ] <= 20 ;
bottleLeftBound[ 42 ] <= 20 ;
bottleLeftBound[ 43 ] <= 20 ;
```

```
                                                                bottleLeftBound[
44  ] <=        18  ;
                                                                bottleLeftBound[
45  ] <=        16  ;
                                                                bottleLeftBound[
46  ] <=        15  ;
                                                                bottleLeftBound[
47  ] <=        14  ;
                                                                bottleLeftBound[
48  ] <=        13  ;
                                                                bottleLeftBound[
49  ] <=        13  ;
                                                                bottleLeftBound[
50  ] <=        13  ;
                                                                bottleLeftBound[
51  ] <=        13  ;
                                                                bottleLeftBound[
52  ] <=        13  ;
                                                                bottleLeftBound[
53  ] <=        13  ;
                                                                bottleLeftBound[
54  ] <=        13  ;
                                                                bottleLeftBound[
55  ] <=        13  ;
                                                                bottleLeftBound[
56  ] <=        13  ;
                                                                bottleLeftBound[
57  ] <=        13  ;
                                                                bottleLeftBound[
58  ] <=        13  ;
                                                                bottleLeftBound[
59  ] <=        13  ;
                                                                bottleLeftBound[
60  ] <=        13  ;
                                                                bottleLeftBound[
61  ] <=        13  ;
                                                                bottleLeftBound[
62  ] <=        13  ;
                                                                bottleLeftBound[
63  ] <=        13  ;
                                                                bottleLeftBound[
64  ] <=        13  ;
                                                                bottleLeftBound[
65  ] <=        13  ;
                                                                bottleLeftBound[
66  ] <=        13  ;
                                                                bottleLeftBound[
67  ] <=        13  ;
                                                                bottleLeftBound[
68  ] <=        13  ;
                                                                bottleLeftBound[
69  ] <=        13  ;
                                                                bottleLeftBound[
70  ] <=        13  ;
                                                                bottleLeftBound[
71  ] <=        13  ;
                                                                bottleLeftBound[
72  ] <=        13  ;
```

```
                                                                bottleLeftBound[
73   ] <=      13   ;
                                                                bottleLeftBound[
74   ] <=      13   ;
                                                                bottleLeftBound[
75   ] <=      13   ;
                                                                bottleLeftBound[
76   ] <=      13   ;
                                                                bottleLeftBound[
77   ] <=      13   ;
                                                                bottleLeftBound[
78   ] <=      13   ;
                                                                bottleLeftBound[
79   ] <=      13   ;
                                                                bottleLeftBound[
80   ] <=      13   ;
                                                                bottleLeftBound[
81   ] <=      13   ;
                                                                bottleLeftBound[
82   ] <=      13   ;
                                                                bottleLeftBound[
83   ] <=      13   ;
                                                                bottleLeftBound[
84   ] <=      13   ;
                                                                bottleLeftBound[
85   ] <=      13   ;
                                                                bottleLeftBound[
86   ] <=      13   ;
                                                                bottleLeftBound[
87   ] <=      13   ;
                                                                bottleLeftBound[
88   ] <=      13   ;
                                                                bottleLeftBound[
89   ] <=      13   ;
                                                                bottleLeftBound[
90   ] <=      13   ;
                                                                bottleLeftBound[
91   ] <=      13   ;
                                                                bottleLeftBound[
92   ] <=      13   ;
                                                                bottleLeftBound[
93   ] <=      13   ;
                                                                bottleLeftBound[
94   ] <=      13   ;
                                                                bottleLeftBound[
95   ] <=      13   ;
                                                                bottleLeftBound[
96   ] <=      13   ;
                                                                bottleLeftBound[
97   ] <=      13   ;
                                                                bottleLeftBound[
98   ] <=      13   ;
                                                                bottleLeftBound[
99   ] <=      13   ;
                                                                bottleLeftBound[
100  ] <=      13   ;
                                                                bottleLeftBound[
101  ] <=      13   ;
```

```
                                                    bottleLeftBound[
102 ] <=      13  ;
                                                    bottleLeftBound[
103 ] <=      13  ;
                                                    bottleLeftBound[
104 ] <=      13  ;
                                                    bottleLeftBound[
105 ] <=      13  ;
                                                    bottleLeftBound[
106 ] <=      13  ;
                                                    bottleLeftBound[
107 ] <=      13  ;
                                                    bottleLeftBound[
108 ] <=      13  ;
                                                    bottleLeftBound[
109 ] <=      13  ;
                                                    bottleLeftBound[
110 ] <=      13  ;
                                                    bottleLeftBound[
111 ] <=      13  ;
                                                    bottleLeftBound[
112 ] <=      13  ;
                                                    bottleLeftBound[
113 ] <=      13  ;
                                                    bottleLeftBound[
114 ] <=      14  ;
                                                    bottleLeftBound[
115 ] <=      14  ;
                                                    bottleLeftBound[
116 ] <=      15  ;
                                                    bottleLeftBound[
117 ] <=      17  ;
                                                    bottleLeftBound[
118 ] <=      18  ;
                                                    bottleLeftBound[
119 ] <=      23  ;

                                                     bottleRightBound[   0
    ] <=      68  ;
                                                     bottleRightBound[   1
    ] <=      72  ;
                                                     bottleRightBound[   2
    ] <=      75  ;
                                                     bottleRightBound[   3
    ] <=      76  ;
                                                     bottleRightBound[   4
    ] <=      77  ;
                                                     bottleRightBound[   5
    ] <=      78  ;
                                                     bottleRightBound[   6
    ] <=      79  ;
                                                     bottleRightBound[   7
    ] <=      80  ;
                                                     bottleRightBound[   8
    ] <=      81  ;
                                                     bottleRightBound[   9
    ] <=      81  ;
                                                     bottleRightBound[
```

```
10  ] <=    82  ;
                                                    bottleRightBound[
11  ] <=    82  ;
                                                    bottleRightBound[
12  ] <=    82  ;
                                                    bottleRightBound[
13  ] <=    83  ;
                                                    bottleRightBound[
14  ] <=    84  ;
                                                    bottleRightBound[
15  ] <=    86  ;
                                                    bottleRightBound[
16  ] <=    87  ;
                                                    bottleRightBound[
17  ] <=    88  ;
                                                    bottleRightBound[
18  ] <=    90  ;
                                                    bottleRightBound[
19  ] <=    90  ;
                                                    bottleRightBound[
20  ] <=    92  ;
                                                    bottleRightBound[
21  ] <=    93  ;
                                                    bottleRightBound[
22  ] <=    93  ;
                                                    bottleRightBound[
23  ] <=    94  ;
                                                    bottleRightBound[
24  ] <=    94  ;
                                                    bottleRightBound[
25  ] <=    94  ;
                                                    bottleRightBound[
26  ] <=    95  ;
                                                    bottleRightBound[
27  ] <=    95  ;
                                                    bottleRightBound[
28  ] <=    95  ;
                                                    bottleRightBound[
29  ] <=    95  ;
                                                    bottleRightBound[
30  ] <=    95  ;
                                                    bottleRightBound[
31  ] <=    98  ;
                                                    bottleRightBound[
32  ] <=    100 ;
                                                    bottleRightBound[
33  ] <=    101 ;
                                                    bottleRightBound[
34  ] <=    102 ;
                                                    bottleRightBound[
35  ] <=    102 ;
                                                    bottleRightBound[
36  ] <=    103 ;
                                                    bottleRightBound[
37  ] <=    103 ;
                                                    bottleRightBound[
38  ] <=    103 ;
                                                    bottleRightBound[
```

```
39  ] <=      103 ;  bottleRightBound[
40  ] <=      103 ;  bottleRightBound[
41  ] <=      103 ;  bottleRightBound[
42  ] <=      103 ;  bottleRightBound[
43  ] <=      102 ;  bottleRightBound[
44  ] <=      103 ;  bottleRightBound[
45  ] <=      106 ;  bottleRightBound[
46  ] <=      106 ;  bottleRightBound[
47  ] <=      105 ;  bottleRightBound[
48  ] <=      107 ;  bottleRightBound[
49  ] <=      107 ;  bottleRightBound[
50  ] <=      107 ;  bottleRightBound[
51  ] <=      107 ;  bottleRightBound[
52  ] <=      107 ;  bottleRightBound[
53  ] <=      107 ;  bottleRightBound[
54  ] <=      107 ;  bottleRightBound[
55  ] <=      107 ;  bottleRightBound[
56  ] <=      107 ;  bottleRightBound[
57  ] <=      107 ;  bottleRightBound[
58  ] <=      107 ;  bottleRightBound[
59  ] <=      107 ;  bottleRightBound[
60  ] <=      107 ;  bottleRightBound[
61  ] <=      107 ;  bottleRightBound[
62  ] <=      107 ;  bottleRightBound[
63  ] <=      107 ;  bottleRightBound[
64  ] <=      107 ;  bottleRightBound[
65  ] <=      107 ;  bottleRightBound[
66  ] <=      107 ;  bottleRightBound[
67  ] <=      107 ;  bottleRightBound[
```

```
68  ] <=    107 ;
                                                    bottleRightBound[
69  ] <=    107 ;
                                                    bottleRightBound[
70  ] <=    107 ;
                                                    bottleRightBound[
71  ] <=    107 ;
                                                    bottleRightBound[
72  ] <=    107 ;
                                                    bottleRightBound[
73  ] <=    107 ;
                                                    bottleRightBound[
74  ] <=    107 ;
                                                    bottleRightBound[
75  ] <=    107 ;
                                                    bottleRightBound[
76  ] <=    107 ;
                                                    bottleRightBound[
77  ] <=    107 ;
                                                    bottleRightBound[
78  ] <=    107 ;
                                                    bottleRightBound[
79  ] <=    107 ;
                                                    bottleRightBound[
80  ] <=    107 ;
                                                    bottleRightBound[
81  ] <=    107 ;
                                                    bottleRightBound[
82  ] <=    107 ;
                                                    bottleRightBound[
83  ] <=    107 ;
                                                    bottleRightBound[
84  ] <=    107 ;
                                                    bottleRightBound[
85  ] <=    107 ;
                                                    bottleRightBound[
86  ] <=    107 ;
                                                    bottleRightBound[
87  ] <=    107 ;
                                                    bottleRightBound[
88  ] <=    107 ;
                                                    bottleRightBound[
89  ] <=    107 ;
                                                    bottleRightBound[
90  ] <=    107 ;
                                                    bottleRightBound[
91  ] <=    107 ;
                                                    bottleRightBound[
92  ] <=    107 ;
                                                    bottleRightBound[
93  ] <=    107 ;
                                                    bottleRightBound[
94  ] <=    107 ;
                                                    bottleRightBound[
95  ] <=    107 ;
                                                    bottleRightBound[
96  ] <=    107 ;
                                                    bottleRightBound[
```

```
97   ] <=       107 ;
                                                                bottleRightBound[
98   ] <=       107 ;
                                                                bottleRightBound[
99   ] <=       107 ;
                                                                bottleRightBound[
100  ] <=       107 ;
                                                                bottleRightBound[
101  ] <=       107 ;
                                                                bottleRightBound[
102  ] <=       107 ;
                                                                bottleRightBound[
103  ] <=       107 ;
                                                                bottleRightBound[
104  ] <=       107 ;
                                                                bottleRightBound[
105  ] <=       107 ;
                                                                bottleRightBound[
106  ] <=       107 ;
                                                                bottleRightBound[
107  ] <=       107 ;
                                                                bottleRightBound[
108  ] <=       107 ;
                                                                bottleRightBound[
109  ] <=       107 ;
                                                                bottleRightBound[
110  ] <=       107 ;
                                                                bottleRightBound[
111  ] <=       107 ;
                                                                bottleRightBound[
112  ] <=       107 ;
                                                                bottleRightBound[
113  ] <=       107 ;
                                                                bottleRightBound[
114  ] <=       106 ;
                                                                bottleRightBound[
115  ] <=       106 ;
                                                                bottleRightBound[
116  ] <=       105 ;
                                                                bottleRightBound[
117  ] <=       104 ;
                                                                bottleRightBound[
118  ] <=       103 ;
                                                                bottleRightBound[
119  ] <=       98   ;
            end
      end


//generate the boundevilHeight of the evil
always@(posedge iCLK or negedge iRST_n)
      begin
            if (!iRST_n)
                  begin
                        evilLeftBound[  0     ] <=       0     ;
                          evilLeftBound[ 1     ] <=       0     ;
                          evilLeftBound[ 2     ] <=       0     ;
                          evilLeftBound[ 3     ] <=       0     ;
```

```
evilLeftBound[4  ] <=  0  ;
evilLeftBound[5  ] <=  0  ;
evilLeftBound[6  ] <=  0  ;
evilLeftBound[7  ] <=  0  ;
evilLeftBound[8  ] <=  0  ;
evilLeftBound[9  ] <=  0  ;
evilLeftBound[10 ] <=  0  ;
evilLeftBound[11 ] <=  0  ;
evilLeftBound[12 ] <=  0  ;
evilLeftBound[13 ] <=  0  ;
evilLeftBound[14 ] <=  0  ;
evilLeftBound[15 ] <=  0  ;
evilLeftBound[16 ] <=  0  ;
evilLeftBound[17 ] <=  0  ;
evilLeftBound[18 ] <=  0  ;
evilLeftBound[19 ] <=  0  ;
evilLeftBound[20 ] <=  0  ;
evilLeftBound[21 ] <=  0  ;
evilLeftBound[22 ] <=  0  ;
evilLeftBound[23 ] <=  0  ;
evilLeftBound[24 ] <=  0  ;
evilLeftBound[25 ] <=  0  ;
evilLeftBound[26 ] <=  0  ;
evilLeftBound[27 ] <=  0  ;
evilLeftBound[28 ] <=  0  ;
evilLeftBound[29 ] <=  0  ;
evilLeftBound[30 ] <=  0  ;
evilLeftBound[31 ] <=  0  ;
evilLeftBound[32 ] <=  0  ;
evilLeftBound[33 ] <=  0  ;
evilLeftBound[34 ] <=  0  ;
evilLeftBound[35 ] <=  0  ;
evilLeftBound[36 ] <=  0  ;
evilLeftBound[37 ] <=  0  ;
evilLeftBound[38 ] <=  0  ;
evilLeftBound[39 ] <=  0  ;
evilLeftBound[40 ] <=  0  ;
evilLeftBound[41 ] <=  0  ;
evilLeftBound[42 ] <=  0  ;
evilLeftBound[43 ] <=  0  ;
evilLeftBound[44 ] <=  0  ;
evilLeftBound[45 ] <=  0  ;
evilLeftBound[46 ] <=  0  ;
evilLeftBound[47 ] <=  0  ;
evilLeftBound[48 ] <=  0  ;
evilLeftBound[49 ] <=  0  ;
evilLeftBound[50 ] <=  0  ;
evilLeftBound[51 ] <=  0  ;
evilLeftBound[52 ] <=  0  ;
evilLeftBound[53 ] <=  0  ;
evilLeftBound[54 ] <=  0  ;
evilLeftBound[55 ] <=  0  ;
evilLeftBound[56 ] <=  0  ;
evilLeftBound[57 ] <=  0  ;
evilLeftBound[58 ] <=  0  ;
evilLeftBound[59 ] <=  0  ;
evilLeftBound[60 ] <=  0  ;
evilLeftBound[61 ] <=  0  ;
```

```
evilLeftBound[62  ] <=      0   ;
evilLeftBound[63  ] <=      0   ;
evilLeftBound[64  ] <=      0   ;
evilLeftBound[65  ] <=      0   ;
evilLeftBound[66  ] <=      0   ;
evilLeftBound[67  ] <=      0   ;
evilLeftBound[68  ] <=      0   ;
evilLeftBound[69  ] <=      0   ;
evilLeftBound[70  ] <=      0   ;
evilLeftBound[71  ] <=      0   ;
evilLeftBound[72  ] <=      0   ;
evilLeftBound[73  ] <=      0   ;
evilLeftBound[74  ] <=      0   ;
evilLeftBound[75  ] <=      0   ;
evilLeftBound[76  ] <=      0   ;
evilLeftBound[77  ] <=      0   ;
evilLeftBound[78  ] <=      0   ;
evilLeftBound[79  ] <=      0   ;
evilLeftBound[80  ] <=      0   ;
evilLeftBound[81  ] <=      0   ;
evilLeftBound[82  ] <=      0   ;
evilLeftBound[83  ] <=      0   ;
evilLeftBound[84  ] <=      0   ;
evilLeftBound[85  ] <=      0   ;
evilLeftBound[86  ] <=      0   ;
evilLeftBound[87  ] <=      0   ;
evilLeftBound[88  ] <=      0   ;
evilLeftBound[89  ] <=      0   ;
evilLeftBound[90  ] <=      0   ;
evilLeftBound[91  ] <=      0   ;
evilLeftBound[92  ] <=      0   ;
evilLeftBound[93  ] <=      0   ;
evilLeftBound[94  ] <=      0   ;
evilLeftBound[95  ] <=      0   ;
evilLeftBound[96  ] <=      0   ;
evilLeftBound[97  ] <=      0   ;
evilLeftBound[98  ] <=      0   ;
evilLeftBound[99  ] <=      0   ;
evilLeftBound[100 ] <=      0   ;
evilLeftBound[101 ] <=      0   ;
evilLeftBound[102 ] <=      0   ;
evilLeftBound[103 ] <=      0   ;
evilLeftBound[104 ] <=      0   ;
evilLeftBound[105 ] <=      0   ;
evilLeftBound[106 ] <=      0   ;
evilLeftBound[107 ] <=      0   ;
evilLeftBound[108 ] <=      0   ;
evilLeftBound[109 ] <=      0   ;
evilLeftBound[110 ] <=      0   ;
evilLeftBound[111 ] <=      0   ;
evilLeftBound[112 ] <=      0   ;
evilLeftBound[113 ] <=      0   ;
evilLeftBound[114 ] <=      0   ;
evilLeftBound[115 ] <=      0   ;
evilLeftBound[116 ] <=      0   ;
evilLeftBound[117 ] <=      0   ;
evilLeftBound[118 ] <=      0   ;
evilLeftBound[119 ] <=      0   ;
```

```
evilRightBound[  0  ] <=  0  ;
evilRightBound[  1  ] <=  0  ;
evilRightBound[  2  ] <=  0  ;
evilRightBound[  3  ] <=  0  ;
evilRightBound[  4  ] <=  0  ;
evilRightBound[  5  ] <=  0  ;
evilRightBound[  6  ] <=  0  ;
evilRightBound[  7  ] <=  0  ;
evilRightBound[  8  ] <=  0  ;
evilRightBound[  9  ] <=  0  ;
evilRightBound[  10  ] <=  0  ;
evilRightBound[  11  ] <=  0  ;
evilRightBound[  12  ] <=  0  ;
evilRightBound[  13  ] <=  0  ;
evilRightBound[  14  ] <=  0  ;
evilRightBound[  15  ] <=  0  ;
evilRightBound[  16  ] <=  0  ;
evilRightBound[  17  ] <=  0  ;
evilRightBound[  18  ] <=  0  ;
evilRightBound[  19  ] <=  0  ;
evilRightBound[  20  ] <=  0  ;
evilRightBound[  21  ] <=  0  ;
evilRightBound[  22  ] <=  0  ;
evilRightBound[  23  ] <=  0  ;
evilRightBound[  24  ] <=  0  ;
evilRightBound[  25  ] <=  0  ;
evilRightBound[  26  ] <=  0  ;
evilRightBound[  27  ] <=  0  ;
evilRightBound[  28  ] <=  0  ;
evilRightBound[  29  ] <=  0  ;
evilRightBound[  30  ] <=  0  ;
evilRightBound[  31  ] <=  0  ;
evilRightBound[  32  ] <=  0  ;
evilRightBound[  33  ] <=  0  ;
evilRightBound[  34  ] <=  0  ;
evilRightBound[  35  ] <=  0  ;
evilRightBound[  36  ] <=  0  ;
evilRightBound[  37  ] <=  0  ;
evilRightBound[  38  ] <=  0  ;
evilRightBound[  39  ] <=  0  ;
evilRightBound[  40  ] <=  0  ;
evilRightBound[  41  ] <=  0  ;
evilRightBound[  42  ] <=  0  ;
evilRightBound[  43  ] <=  0  ;
evilRightBound[  44  ] <=  0  ;
evilRightBound[  45  ] <=  0  ;
evilRightBound[  46  ] <=  0  ;
evilRightBound[  47  ] <=  0  ;
evilRightBound[  48  ] <=  0  ;
evilRightBound[  49  ] <=  0  ;
evilRightBound[  50  ] <=  0  ;
evilRightBound[  51  ] <=  0  ;
evilRightBound[  52  ] <=  0  ;
evilRightBound[  53  ] <=  0  ;
evilRightBound[  54  ] <=  0  ;
evilRightBound[  55  ] <=  0  ;
evilRightBound[  56  ] <=  0  ;
```

```
evilRightBound[  57 ] <=  0 ;
evilRightBound[  58 ] <=  0 ;
evilRightBound[  59 ] <=  0 ;
evilRightBound[  60 ] <=  0 ;
evilRightBound[  61 ] <=  0 ;
evilRightBound[  62 ] <=  0 ;
evilRightBound[  63 ] <=  0 ;
evilRightBound[  64 ] <=  0 ;
evilRightBound[  65 ] <=  0 ;
evilRightBound[  66 ] <=  0 ;
evilRightBound[  67 ] <=  0 ;
evilRightBound[  68 ] <=  0 ;
evilRightBound[  69 ] <=  0 ;
evilRightBound[  70 ] <=  0 ;
evilRightBound[  71 ] <=  0 ;
evilRightBound[  72 ] <=  0 ;
evilRightBound[  73 ] <=  0 ;
evilRightBound[  74 ] <=  0 ;
evilRightBound[  75 ] <=  0 ;
evilRightBound[  76 ] <=  0 ;
evilRightBound[  77 ] <=  0 ;
evilRightBound[  78 ] <=  0 ;
evilRightBound[  79 ] <=  0 ;
evilRightBound[  80 ] <=  0 ;
evilRightBound[  81 ] <=  0 ;
evilRightBound[  82 ] <=  0 ;
evilRightBound[  83 ] <=  0 ;
evilRightBound[  84 ] <=  0 ;
evilRightBound[  85 ] <=  0 ;
evilRightBound[  86 ] <=  0 ;
evilRightBound[  87 ] <=  0 ;
evilRightBound[  88 ] <=  0 ;
evilRightBound[  89 ] <=  0 ;
evilRightBound[  90 ] <=  0 ;
evilRightBound[  91 ] <=  0 ;
evilRightBound[  92 ] <=  0 ;
evilRightBound[  93 ] <=  0 ;
evilRightBound[  94 ] <=  0 ;
evilRightBound[  95 ] <=  0 ;
evilRightBound[  96 ] <=  0 ;
evilRightBound[  97 ] <=  0 ;
evilRightBound[  98 ] <=  0 ;
evilRightBound[  99 ] <=  0 ;
evilRightBound[ 100 ] <=  0 ;
evilRightBound[ 101 ] <=  0 ;
evilRightBound[ 102 ] <=  0 ;
evilRightBound[ 103 ] <=  0 ;
evilRightBound[ 104 ] <=  0 ;
evilRightBound[ 105 ] <=  0 ;
evilRightBound[ 106 ] <=  0 ;
evilRightBound[ 107 ] <=  0 ;
evilRightBound[ 108 ] <=  0 ;
evilRightBound[ 109 ] <=  0 ;
evilRightBound[ 110 ] <=  0 ;
evilRightBound[ 111 ] <=  0 ;
evilRightBound[ 112 ] <=  0 ;
evilRightBound[ 113 ] <=  0 ;
evilRightBound[ 114 ] <=  0 ;
```

```
                evilRightBound[   115 ] <=      0   ;
                evilRightBound[   116 ] <=      0   ;
                evilRightBound[   117 ] <=      0   ;
                evilRightBound[   118 ] <=      0   ;
                evilRightBound[   119 ] <=      0   ;

        end
else
    begin
                                        evilLeftBound[0   ] <=      57  ;
                evilLeftBound[  1   ] <=      51  ;
                evilLeftBound[  2   ] <=      47  ;
                evilLeftBound[  3   ] <=      44  ;
                evilLeftBound[  4   ] <=      42  ;
                evilLeftBound[  5   ] <=      38  ;
                evilLeftBound[  6   ] <=      36  ;
                evilLeftBound[  7   ] <=      36  ;
                evilLeftBound[  8   ] <=      33  ;
                evilLeftBound[  9   ] <=      32  ;
                evilLeftBound[  10  ] <=      31  ;
                evilLeftBound[  11  ] <=      29  ;
                evilLeftBound[  12  ] <=      28  ;
                evilLeftBound[  13  ] <=      26  ;
                evilLeftBound[  14  ] <=      24  ;
                evilLeftBound[  15  ] <=      24  ;
                evilLeftBound[  16  ] <=      20  ;
                evilLeftBound[  17  ] <=      20  ;
                evilLeftBound[  18  ] <=      20  ;
                evilLeftBound[  19  ] <=      20  ;
                evilLeftBound[  20  ] <=      19  ;
                evilLeftBound[  21  ] <=      18  ;
                evilLeftBound[  22  ] <=      18  ;
                evilLeftBound[  23  ] <=      16  ;
                evilLeftBound[  24  ] <=      16  ;
                evilLeftBound[  25  ] <=      16  ;
                evilLeftBound[  26  ] <=      13  ;
                evilLeftBound[  27  ] <=      13  ;
                evilLeftBound[  28  ] <=      13  ;
                evilLeftBound[  29  ] <=      13  ;
                evilLeftBound[  30  ] <=      12  ;
                evilLeftBound[  31  ] <=      10  ;
                evilLeftBound[  32  ] <=      11  ;
                evilLeftBound[  33  ] <=      10  ;
                evilLeftBound[  34  ] <=      9   ;
                evilLeftBound[  35  ] <=      10  ;
                evilLeftBound[  36  ] <=      9   ;
                evilLeftBound[  37  ] <=      9   ;
                evilLeftBound[  38  ] <=      9   ;
                evilLeftBound[  39  ] <=      9   ;
                evilLeftBound[  40  ] <=      9   ;
                evilLeftBound[  41  ] <=      7   ;
                evilLeftBound[  42  ] <=      7   ;
                evilLeftBound[  43  ] <=      7   ;
                evilLeftBound[  44  ] <=      7   ;
                evilLeftBound[  45  ] <=      7   ;
                evilLeftBound[  46  ] <=      7   ;
                evilLeftBound[  47  ] <=      7   ;
                evilLeftBound[  48  ] <=      6   ;
```

```
evilLeftBound[  49  ] <=    7  ;
evilLeftBound[  50  ] <=    7  ;
evilLeftBound[  51  ] <=    6  ;
evilLeftBound[  52  ] <=    6  ;
evilLeftBound[  53  ] <=    6  ;
evilLeftBound[  54  ] <=    6  ;
evilLeftBound[  55  ] <=    6  ;
evilLeftBound[  56  ] <=    6  ;
evilLeftBound[  57  ] <=    6  ;
evilLeftBound[  58  ] <=    6  ;
evilLeftBound[  59  ] <=    6  ;
evilLeftBound[  60  ] <=    7  ;
evilLeftBound[  61  ] <=    7  ;
evilLeftBound[  62  ] <=    7  ;
evilLeftBound[  63  ] <=    7  ;
evilLeftBound[  64  ] <=    7  ;
evilLeftBound[  65  ] <=    7  ;
evilLeftBound[  66  ] <=    7  ;
evilLeftBound[  67  ] <=    8  ;
evilLeftBound[  68  ] <=    8  ;
evilLeftBound[  69  ] <=    9  ;
evilLeftBound[  70  ] <=    9  ;
evilLeftBound[  71  ] <=    10 ;
evilLeftBound[  72  ] <=    10 ;
evilLeftBound[  73  ] <=    10 ;
evilLeftBound[  74  ] <=    11 ;
evilLeftBound[  75  ] <=    11 ;
evilLeftBound[  76  ] <=    11 ;
evilLeftBound[  77  ] <=    12 ;
evilLeftBound[  78  ] <=    14 ;
evilLeftBound[  79  ] <=    14 ;
evilLeftBound[  80  ] <=    15 ;
evilLeftBound[  81  ] <=    15 ;
evilLeftBound[  82  ] <=    16 ;
evilLeftBound[  83  ] <=    17 ;
evilLeftBound[  84  ] <=    18 ;
evilLeftBound[  85  ] <=    19 ;
evilLeftBound[  86  ] <=    20 ;
evilLeftBound[  87  ] <=    19 ;
evilLeftBound[  88  ] <=    21 ;
evilLeftBound[  89  ] <=    23 ;
evilLeftBound[  90  ] <=    24 ;
evilLeftBound[  91  ] <=    25 ;
evilLeftBound[  92  ] <=    23 ;
evilLeftBound[  93  ] <=    27 ;
evilLeftBound[  94  ] <=    27 ;
evilLeftBound[  95  ] <=    28 ;
evilLeftBound[  96  ] <=    28 ;
evilLeftBound[  97  ] <=    28 ;
evilLeftBound[  98  ] <=    28 ;
evilLeftBound[  99  ] <=    27 ;
evilLeftBound[  100 ] <=    27 ;
evilLeftBound[  101 ] <=    27 ;
evilLeftBound[  102 ] <=    27 ;
evilLeftBound[  103 ] <=    27 ;
evilLeftBound[  104 ] <=    27 ;
evilLeftBound[  105 ] <=    27 ;
evilLeftBound[  106 ] <=    27 ;
```

```
evilLeftBound[  107  ] <=      27   ;
evilLeftBound[  108  ] <=      28   ;
evilLeftBound[  109  ] <=      27   ;
evilLeftBound[  110  ] <=      27   ;
evilLeftBound[  111  ] <=      27   ;
evilLeftBound[  112  ] <=      27   ;
evilLeftBound[  113  ] <=      28   ;
evilLeftBound[  114  ] <=      28   ;
evilLeftBound[  115  ] <=      28   ;
evilLeftBound[  116  ] <=      29   ;
evilLeftBound[  117  ] <=      29   ;
evilLeftBound[  118  ] <=      29   ;
evilLeftBound[  119  ] <=      30   ;

evilRightBound[0     ] <=      58   ;
evilRightBound[1     ] <=      70   ;
evilRightBound[2     ] <=      74   ;
evilRightBound[3     ] <=      77   ;
evilRightBound[4     ] <=      82   ;
evilRightBound[5     ] <=      81   ;
evilRightBound[6     ] <=      84   ;
evilRightBound[7     ] <=      85   ;
evilRightBound[8     ] <=      87   ;
evilRightBound[9     ] <=      90   ;
evilRightBound[10    ] <=      90   ;
evilRightBound[11    ] <=      91   ;
evilRightBound[12    ] <=      93   ;
evilRightBound[13    ] <=      94   ;
evilRightBound[14    ] <=      95   ;
evilRightBound[15    ] <=      96   ;
evilRightBound[16    ] <=      97   ;
evilRightBound[17    ] <=      21   ;
evilRightBound[18    ] <=      99   ;
evilRightBound[19    ] <=      100  ;
evilRightBound[20    ] <=      101  ;
evilRightBound[21    ] <=      101  ;
evilRightBound[22    ] <=      102  ;
evilRightBound[23    ] <=      103  ;
evilRightBound[24    ] <=      103  ;
evilRightBound[25    ] <=      104  ;
evilRightBound[26    ] <=      104  ;
evilRightBound[27    ] <=      106  ;
evilRightBound[28    ] <=      107  ;
evilRightBound[29    ] <=      107  ;
evilRightBound[30    ] <=      107  ;
evilRightBound[31    ] <=      107  ;
evilRightBound[32    ] <=      108  ;
evilRightBound[33    ] <=      109  ;
evilRightBound[34    ] <=      109  ;
evilRightBound[35    ] <=      109  ;
evilRightBound[36    ] <=      110  ;
evilRightBound[37    ] <=      110  ;
evilRightBound[38    ] <=      110  ;
evilRightBound[39    ] <=      111  ;
evilRightBound[40    ] <=      111  ;
evilRightBound[41    ] <=      112  ;
evilRightBound[42    ] <=      112  ;
evilRightBound[43    ] <=      112  ;
```

```
evilRightBound[44  ] <=      112 ;
evilRightBound[45  ] <=      112 ;
evilRightBound[46  ] <=      113 ;
evilRightBound[47  ] <=      113 ;
evilRightBound[48  ] <=      113 ;
evilRightBound[49  ] <=      114 ;
evilRightBound[50  ] <=      114 ;
evilRightBound[51  ] <=      114 ;
evilRightBound[52  ] <=      114 ;
evilRightBound[53  ] <=      114 ;
evilRightBound[54  ] <=      114 ;
evilRightBound[55  ] <=      114 ;
evilRightBound[56  ] <=      114 ;
evilRightBound[57  ] <=      114 ;
evilRightBound[58  ] <=      114 ;
evilRightBound[59  ] <=      114 ;
evilRightBound[60  ] <=      114 ;
evilRightBound[61  ] <=      114 ;
evilRightBound[62  ] <=      114 ;
evilRightBound[63  ] <=      114 ;
evilRightBound[64  ] <=      113 ;
evilRightBound[65  ] <=      113 ;
evilRightBound[66  ] <=      113 ;
evilRightBound[67  ] <=      113 ;
evilRightBound[68  ] <=      112 ;
evilRightBound[69  ] <=      112 ;
evilRightBound[70  ] <=      112 ;
evilRightBound[71  ] <=      111 ;
evilRightBound[72  ] <=      112 ;
evilRightBound[73  ] <=      110 ;
evilRightBound[74  ] <=      110 ;
evilRightBound[75  ] <=      110 ;
evilRightBound[76  ] <=      109 ;
evilRightBound[77  ] <=      109 ;
evilRightBound[78  ] <=      108 ;
evilRightBound[79  ] <=      108 ;
evilRightBound[80  ] <=      108 ;
evilRightBound[81  ] <=      108 ;
evilRightBound[82  ] <=      106 ;
evilRightBound[83  ] <=      105 ;
evilRightBound[84  ] <=      104 ;
evilRightBound[85  ] <=      104 ;
evilRightBound[86  ] <=      104 ;
evilRightBound[87  ] <=      102 ;
evilRightBound[88  ] <=      103 ;
evilRightBound[89  ] <=      100 ;
evilRightBound[90  ] <=      99  ;
evilRightBound[91  ] <=      98  ;
evilRightBound[92  ] <=      97  ;
evilRightBound[93  ] <=      95  ;
evilRightBound[94  ] <=      95  ;
evilRightBound[95  ] <=      94  ;
evilRightBound[96  ] <=      94  ;
evilRightBound[97  ] <=      94  ;
evilRightBound[98  ] <=      94  ;
evilRightBound[99  ] <=      94  ;
evilRightBound[100 ] <=      94  ;
evilRightBound[101 ] <=      94  ;
```

```
                                evilRightBound[102  ] <=      94   ;
                                evilRightBound[103  ] <=      94   ;
                                evilRightBound[104  ] <=      94   ;
                                evilRightBound[105  ] <=      94   ;
                                evilRightBound[106  ] <=      94   ;
                                evilRightBound[107  ] <=      94   ;
                                evilRightBound[108  ] <=      94   ;
                                evilRightBound[109  ] <=      94   ;
                                evilRightBound[110  ] <=      94   ;
                                evilRightBound[111  ] <=      94   ;
                                evilRightBound[112  ] <=      94   ;
                                evilRightBound[113  ] <=      94   ;
                                evilRightBound[114  ] <=      93   ;
                                evilRightBound[115  ] <=      94   ;
                                evilRightBound[116  ] <=      93   ;
                                evilRightBound[117  ] <=      93   ;
                                evilRightBound[118  ] <=      93   ;
                                evilRightBound[119  ] <=      92   ;
        end
end

assign mole0[0] = moleControl[0] ? 1: 0;
    assign mole0[1] = moleControl[1] ? 1: 0;
    assign updn0 = moleControl[2] ? 1: 0;

    assign mole1[0] = moleControl[3] ? 1: 0;
    assign mole1[1] = moleControl[4] ? 1: 0;
    assign updn1 = moleControl[5] ? 1: 0;

    assign mole2[0] = moleControl[6] ? 1: 0;
    assign mole2[1] = moleControl[7] ? 1: 0;
    assign updn2 = moleControl[8] ? 1: 0;


    assign mole3[0] = moleControl[9] ? 1: 0;
    assign mole3[1] = moleControl[10] ? 1: 0;
    assign updn3 = moleControl[11] ? 1: 0;

    assign mole4[0] = moleControl[12] ? 1: 0;
    assign mole4[1] = moleControl[13] ? 1: 0;
    assign updn4 = moleControl[14] ? 1: 0;

    assign mole5[0] = moleControl[15] ? 1: 0;
    assign mole5[1] = moleControl[16] ? 1: 0;
    assign updn5 = moleControl[17] ? 1: 0;

    assign mole6[0] = moleControl[18] ? 1: 0;
    assign mole6[1] = moleControl[19] ? 1: 0;
    assign updn6 = moleControl[20] ? 1: 0;

    assign mole7[0] = moleControl[21] ? 1: 0;
    assign mole7[1] = moleControl[22] ? 1: 0;
    assign updn7 = moleControl[23] ? 1: 0;

    assign mole8[0] = moleControl[24] ? 1: 0;
    assign mole8[1] = moleControl[25] ? 1: 0;
    assign updn8 = moleControl[26] ? 1: 0;
```

```
//Calculate the height of the mole0
always@(posedge iCLK or negedge iRST_n)
	begin
		if(!iRST_n)
			begin
			  moleHeight[0] <= 0;
			  counter00 <=0;
			 end
		 else if(updn0==1 && moleHeight[0]<118)
		    begin
				counter00 <= counter00 +1;
				 if(counter00 == 30000)
				    begin
				    moleHeight[0] <= moleHeight[0] +1;
					counter00 <=0;
					 end
		  end
		 else if(updn0==0)
			begin
				moleHeight[0] <= 0;
				 counter00 <=0;
			end
		 else
			begin
				moleHeight[0] <= moleHeight[0];
				 counter00 <=0;
			end
	  end

	  //Calculate the height of the mole1
always@(posedge iCLK or negedge iRST_n)
	begin
		if(!iRST_n)
			begin
			  moleHeight[1] <= 0;
			  counter11 <=0;
			 end
		 else if(updn1==1 && moleHeight[1]<118)
		    begin
				counter11 <= counter11 +1;
				 if(counter11 == 30000)
				    begin
				    moleHeight[1] <= moleHeight[1] +1;
					counter11 <=0;
					 end
		  end
		 else if(updn1==0)
			begin
				moleHeight[1] <= 0;
				 counter11 <=0;
			end
		 else
			begin
				moleHeight[1] <= moleHeight[1];
				 counter11 <=0;
			end
	  end
```

```
        //Calculate the height of the mole2
always@(posedge iCLK or negedge iRST_n)
    begin
        if(!iRST_n)
            begin
              moleHeight[2] <= 0;
              counter22 <=0;
             end
         else if(updn2==1 && moleHeight[2]<118)
            begin
                counter22 <= counter22 +1;
                 if(counter22 == 30000)
                    begin
                    moleHeight[2] <= moleHeight[2] +1;
                        counter22 <=0;
                     end
          end
         else if(updn2==0)
            begin
                moleHeight[2] <= 0;
                 counter22 <=0;
            end
         else
            begin
                moleHeight[2] <= moleHeight[2];
                 counter22 <=0;
            end
      end


        //Calculate the height of the mole3
always@(posedge iCLK or negedge iRST_n)
    begin
        if(!iRST_n)
            begin
              moleHeight[3] <= 0;
              counter33 <=0;
             end
         else if(updn3==1 && moleHeight[3]<118)
            begin
                counter33 <= counter33 +1;
                 if(counter33 == 30000)
                    begin
                    moleHeight[3] <= moleHeight[3] +1;
                        counter33 <=0;
                     end
          end
         else if(updn3==0)
            begin
                moleHeight[3] <= 0;
                 counter33 <=0;
            end
         else
            begin
                moleHeight[3] <= moleHeight[3];
                 counter33 <=0;
```

```
            end
        end


        //Calculate the height of the mole4
always@(posedge iCLK or negedge iRST_n)
    begin
        if(!iRST_n)
            begin
              moleHeight[4] <= 0;
              counter44 <=0;
             end
          else if(updn4==1 && moleHeight[4]<118)
             begin
                  counter44 <= counter44 +1;
                   if(counter44 == 30000)
                      begin
                      moleHeight[4] <= moleHeight[4] +1;
                          counter44 <=0;
                       end
           end
          else if(updn4==0)
              begin
                  moleHeight[4] <= 0;
                   counter44 <=0;
              end
          else
              begin
                  moleHeight[4] <= moleHeight[4];
                   counter44 <=0;
              end
        end


        //Calculate the height of the mole5
always@(posedge iCLK or negedge iRST_n)
    begin
        if(!iRST_n)
            begin
              moleHeight[5] <= 0;
              counter55 <=0;
             end
          else if(updn5==1 && moleHeight[5]<118)
             begin
                  counter55 <= counter55 +1;
                   if(counter55 == 30000)
                      begin
                      moleHeight[5] <= moleHeight[5] +1;
                          counter55 <=0;
                       end
           end
          else if(updn5==0)
              begin
                  moleHeight[5] <= 0;
                   counter55 <=0;
              end
          else
              begin
```

```
                    moleHeight[5] <= moleHeight[5];
                     counter55 <=0;
                end
        end

        //Calculate the height of the mole6
always@(posedge iCLK or negedge iRST_n)
    begin
        if(!iRST_n)
            begin
               moleHeight[6] <= 0;
               counter66 <=0;
             end
          else if(updn6==1 && moleHeight[6]<118)
             begin
                    counter66 <= counter66 +1;
                     if(counter66 == 30000)
                        begin
                        moleHeight[6] <= moleHeight[6] +1;
                            counter66 <=0;
                         end
           end
          else if(updn6==0)
            begin
                    moleHeight[6] <= 0;
                     counter66 <=0;
                end
          else
            begin
                    moleHeight[6] <= moleHeight[6];
                     counter66 <=0;
                end
        end

        //Calculate the height of the mole7
always@(posedge iCLK or negedge iRST_n)
    begin
        if(!iRST_n)
            begin
               moleHeight[7] <= 0;
               counter77 <=0;
             end
          else if(updn7==1 && moleHeight[7]<118)
             begin
                    counter77 <= counter77 +1;
                     if(counter77 == 30000)
                        begin
                        moleHeight[7] <= moleHeight[7] +1;
                            counter77 <=0;
                         end
           end
          else if(updn7==0)
            begin
                    moleHeight[7] <= 0;
                     counter77 <=0;
                end
          else
            begin
```

```
                    moleHeight[7] <= moleHeight[7];
                     counter77 <=0;
                end
        end

        //Calculate the height of the mole8
always@(posedge iCLK or negedge iRST_n)
      begin
          if(!iRST_n)
               begin
                  moleHeight[8] <= 0;
                  counter88 <=0;
                end
            else if(updn8==1 && moleHeight[8]<118)
               begin
                    counter88 <= counter88 +1;
                     if(counter88 == 30000)
                        begin
                        moleHeight[8] <= moleHeight[8] +1;
                            counter88 <=0;
                         end
             end
            else if(updn8==0)
               begin
                    moleHeight[8] <= 0;
                     counter88 <=0;
               end
            else
               begin
                    moleHeight[8] <= moleHeight[8];
                     counter88 <=0;
               end
        end




//Detect which area you are in// 0~14 different areas
assign area[0] = ((x_cnt>(Hsync_Blank-1+400)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-200))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0 ; //mole0
assign area[1] = ((x_cnt>(Hsync_Blank-1+200)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0;//mole1
assign area[2] = ((x_cnt>(Hsync_Blank-1)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0 ;//mole2
assign area[3] = ((x_cnt>(Hsync_Blank-1+400)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-200))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
```

```
                                ))    ? 1'b1 : 1'b0 ;    //mole3
assign area[4] = ((x_cnt>(Hsync_Blank-1+200)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ; //mole4
assign area[5] = ((x_cnt>(Hsync_Blank-1)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ;//mole5
assign area[6] = ((x_cnt>(Hsync_Blank-1+400)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-200))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;    //mole6
assign area[7] = ((x_cnt>(Hsync_Blank-1+200)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0; //mole7
assign area[8] = ((x_cnt>(Hsync_Blank-1)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;//mole8
assign area[9] = ((x_cnt>(Hsync_Blank-1+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-400))
                                ))    ? 1'b1 : 1'b0 ;//area 9 Time
assign area[10] = ((x_cnt>(Hsync_Blank-1+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+80))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0 ;//area 10 life
assign area[11] = ((x_cnt>(Hsync_Blank-1+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-240))
                                ))    ? 1'b1 : 1'b0 ;//area 11 score
assign area[12] = ((x_cnt>(Hsync_Blank-1+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+240))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ;//area 12 start/pause
assign area[13] = ((x_cnt>(Hsync_Blank-1+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-80))
                                ))    ? 1'b1 : 1'b0 ;//area 13 exit
assign area[14] = ((x_cnt>(Hsync_Blank-1+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+400))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;//area 14 label
assign area[15] = 1'b0;
//area 15 is the logo of the team defalt black
```

```
//Test --Allocate color to each block
always@(posedge iCLK or negedge iRST_n)
	begin
	if (!iRST_n)
		begin
		mole_address <= moleBase;
		end
	else
		case(area)
				16'b0000000000000001: begin
					x_offset <= Hsync_Blank + 400;
					y_offset <= Vertical_Back_Porch;
				//get the diff between the current x_cnt and the left boundary of the hole
					bound_x <=	-x_cnt+x_offset+200-40;
				//get the diff between the current y_cnt and the horizontal center of the hole
						if(	y_cnt>120+y_offset)
							begin
							 abs_y		<=	y_cnt-120-y_offset;
							 end
						else
							begin
							  abs_y		<=	-y_cnt+120+y_offset;
						    end

						//get the diff between the current x_cnt and the vertical center of the hole
						if(x_cnt>100+x_offset)
							begin
							  abs_x		<=	x_cnt-100-x_offset;
							 end
						else
							begin
							  abs_x		<=	-x_cnt+100+x_offset;
							 end

						//mole out of the hole
						if(moleHeight[0]>0)
							begin
							  case(mole0)
							    2'b00:begin	//mole
								//mole out
								// inside mole area
							      if (y_cnt < y_offset + 135 && y_cnt > y_offset + 135 - moleHeight[0]&& bound_x > moleLeftBound[y_cnt-135-y_offset+moleHeight[0]]+1 &&bound_x < moleRightBound[y_cnt-135-y_offset+moleHeight[0]]-1 )
									begin
									//update the
```

```
mole_address
                                                    mole_address <=
moleBase + (y_cnt-135-y_offset+moleHeight[0])*120 + x_cnt-x_offset-40;
                                                        moleExist
<= 1;
                                                    end
                                                     //hole area
                                                else    if(    abs_y<15    &&
abs_x<hole[abs_y] )
                                                    begin
                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                    end
                                                else
                                                    begin
                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                    end
                                                end
                                                2'b10:begin    //bottle
                                                      // bottle out
                                                     // inside bottle area
                                                  if    (y_cnt    <    y_offset    +
135    &&    y_cnt    >    y_offset    +    135    -    moleHeight[0]&&    bound_x    >
bottleLeftBound[y_cnt-135-y_offset+moleHeight[0]]+1              &&bound_x              <
bottleRightBound[y_cnt-135-y_offset+moleHeight[0]]-1    )
                                                     begin
                                                           //update    the
mole_address
                                                        mole_address <=
moleBase+bottleOffset + (y_cnt-135-y_offset+moleHeight[0])*120 + x_cnt-x_offset-40;
                                                        moleExist
<= 1;
                                                    end
                                                     //hole area
                                                else    if(    abs_y<15    &&
abs_x<hole[abs_y] )
                                                    begin
                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                    end
                                                else
                                                    begin
                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                    end
                                                end
                                                2'b01:begin    //evil
                                                      // bottle out
                                                     // inside bottle area
```

```
                                                    if (y_cnt < y_offset + 135 && y_cnt > y_offset + 135 - moleHeight[0]&& bound_x > evilLeftBound[y_cnt-135-y_offset+moleHeight[0]]+1 &&bound_x < evilRightBound[y_cnt-135-y_offset+moleHeight[0]]-1 )
                                                        begin
                                                            //update the mole_address
                                                            mole_address <= moleBase+evilOffset + (y_cnt-135-y_offset+moleHeight[0])*120 + x_cnt-x_offset-40;
                                                            moleExist <= 1;
                                                        end
                                                        //hole area
                                                    else if( abs_y<15 && abs_x<hole[abs_y] )
                                                        begin
                                                            moleExist <= 1;

mole_address <= moleBase + 120*120;
                                                        end
                                                    else
                                                        begin
                                                            moleExist <= 0;

mole_address <= 18'bZ ;
                                                        end
                                                end

                                            endcase
                                                end
                                        //no mole
                                        else
                                            if( abs_y<15 && abs_x<hole[abs_y] )
                                                begin
                                                    moleExist <= 1;
                                                    mole_address <= moleBase + 120*120;
                                                end
                                        else
                                                begin
                                                    moleExist <= 0;
                                                    mole_address <= 18'bZ ;
                                                end
                                    end



            16'b0000000000000010: begin x_offset <= Hsync_Blank + 200;
                                        y_offset <= Vertical_Back_Porch;
                                        //get the diff between the current x_cnt and the left boundary of the hole
                                        bound_x <= -x_cnt+x_offset+200-40;
                                        //get the diff between the current
```

```
y_cnt and the horizontal center of the hole
                                                    if( y_cnt>120+y_offset)
                                                      begin
                                                       abs_y                    <=
      y_cnt-120-y_offset;
                                                       end
                                                   else
                                                      begin
                                                         abs_y                  <=
      -y_cnt+120+y_offset;
                                                     end
                                                   //get the diff between the current
x_cnt and the vertical center of the hole
                                                   if(x_cnt>100+x_offset)
                                                      begin
                                                        abs_x                   <=
      x_cnt-100-x_offset;
                                                       end
                                                   else
                                                      begin
                                                         abs_x                  <=
      -x_cnt+100+x_offset;
                                                       end


                                                   //mole out of the hole
                                                   if(moleHeight[1]>0)
                                                     begin
                                                           case(mole1)
                                                              2'b00:begin     //mole

                                                                   //mole out
                                                                   // inside mole area
                                                                if    (y_cnt    <    y_offset    +
135    &&    y_cnt    >    y_offset    +    135    -    moleHeight[1]&&    bound_x    >
moleLeftBound[y_cnt-135-y_offset+moleHeight[1]]+1          &&bound_x                <
moleRightBound[y_cnt-135-y_offset+moleHeight[1]]-1    )
                                                                   begin
                                                                          //update the
mole_address
                                                                      mole_address <=
moleBase + (y_cnt-135-y_offset+moleHeight[1])*120 + x_cnt-x_offset-40;
                                                                          moleExist
<= 1;
                                                                  end
                                                                   //hole area
                                                               else    if(    abs_y<15    &&
abs_x<hole[abs_y] )
                                                                  begin
                                                                          moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                                   end
                                                               else
                                                                  begin
                                                                          moleExist
<= 0;
```

```
mole_address <= 18'bZ ;
                                                                    end

                                                                end

                                                                    2'b10:begin
//bottle

                                                                    //bottle out
                                                                    // inside bottle area
                                                                if (y_cnt < y_offset +
135 && y_cnt > y_offset + 135 - moleHeight[1]&& bound_x >
bottleLeftBound[y_cnt-135-y_offset+moleHeight[1]]+1 &&bound_x <
bottleRightBound[y_cnt-135-y_offset+moleHeight[1]]-1 )
                                                                    begin
                                                                        //update the
mole_address
                                                                    mole_address <=
bottleOffset + moleBase + (y_cnt-135-y_offset+moleHeight[1])*120 + x_cnt-x_offset-40;
                                                                        moleExist
<= 1;
                                                                    end
                                                                    //hole area
                                                                else if( abs_y<15 &&
abs_x<hole[abs_y] )
                                                                    begin
                                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                                    end
                                                                else
                                                                    begin
                                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                                    end

                                                                end

                                                                    2'b01:begin     //evil

                                                                    //evil out
                                                                    // inside evil area
                                                                if (y_cnt < y_offset +
135 && y_cnt > y_offset + 135 - moleHeight[1]&& bound_x >
evilLeftBound[y_cnt-135-y_offset+moleHeight[1]]+1 &&bound_x <
evilRightBound[y_cnt-135-y_offset+moleHeight[1]]-1 )
                                                                    begin
                                                                        //update the
mole_address
                                                                    mole_address <=
evilOffset + moleBase + (y_cnt-135-y_offset+moleHeight[1])*120 + x_cnt-x_offset-40;
                                                                        moleExist
<= 1;
                                                                    end
```

```
                                                        //hole area
                                                    else    if(    abs_y<15    &&
abs_x<hole[abs_y] )
                                                        begin
                                                                moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                        end
                                                    else
                                                        begin
                                                                moleExist
<= 0;

mole_address <= 18'bZ ;
                                                        end

                                                        end


                                                endcase
                                                    end
                                            //no mole
                                             else
                                                if(             abs_y<15             &&
abs_x<hole[abs_y] )
                                                    begin
                                                            moleExist <= 1;
                                                            mole_address        <=
moleBase + 120*120;
                                                     end
                                                else
                                                    begin
                                                             moleExist <= 0;
                                                             mole_address      <=
18'bZ ;
                                                     end
                                        end


                16'b0000000000000100: begin    x_offset <= Hsync_Blank;
                                                y_offset <= Vertical_Back_Porch;
                                                    //get the diff between the current x_cnt
and the left boundary of the hole
                                                 bound_x <=    -x_cnt+x_offset+200-40;
                                                       //get the diff between the current
y_cnt and the horizontal center of the hole
                                                    if( y_cnt>120+y_offset)
                                                       begin
                                                         abs_y                          <=
      y_cnt-120-y_offset;
                                                        end
                                                   else
                                                        begin
                                                           abs_y                        <=
      -y_cnt+120+y_offset;
                                                       end
                                                   //get the diff between the current
```

```
x_cnt and the vertical center of the hole
                                                    if(x_cnt>100+x_offset)
                                                        begin
                                                          abs_x                    <=
        x_cnt-100-x_offset;
                                                         end
                                                    else
                                                        begin
                                                          abs_x                    <=
        -x_cnt+100+x_offset;
                                                         end


                                                    //mole out of the hole
                                                    if(moleHeight[2]>0)
                                                        begin
                                                            case(mole2)
                                                              2'b00:begin     //mole
                                                                    //mole out
                                                                    // inside mole area
                                                                if    (y_cnt    <    y_offset    +
135    &&    y_cnt    >    y_offset    +    135    -    moleHeight[2]&&    bound_x    >
moleLeftBound[y_cnt-135-y_offset+moleHeight[2]]+1            &&bound_x                <
moleRightBound[y_cnt-135-y_offset+moleHeight[2]]-1    )
                                                                    begin
                                                                          //update    the
mole_address
                                                                       mole_address <=
moleBase + (y_cnt-135-y_offset+moleHeight[2])*120 + x_cnt-x_offset-40;
                                                                          moleExist
<= 1;
                                                                    end
                                                                    //hole area
                                                                else    if(    abs_y<15    &&
abs_x<hole[abs_y] )
                                                                    begin
                                                                          moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                                    end
                                                                else
                                                                    begin
                                                                          moleExist
<= 0;

mole_address <= 18'bZ ;
                                                                    end

                                                                    end

                                                                  2'b10:begin
//bottle
                                                                    //mole out
                                                                    // inside mole area
                                                                if    (y_cnt    <    y_offset    +
135    &&    y_cnt    >    y_offset    +    135    -    moleHeight[2]&&    bound_x    >
bottleLeftBound[y_cnt-135-y_offset+moleHeight[2]]+1            &&bound_x                <
```

```
bottleRightBound[y_cnt-135-y_offset+moleHeight[2]]-1    )
                                                                    begin
                                                                        //update the
mole_address
                                                                    mole_address <=
bottleOffset+moleBase + (y_cnt-135-y_offset+moleHeight[2])*120 + x_cnt-x_offset-40;
                                                                    moleExist
<= 1;
                                                                end
                                                                //hole area
                                                            else if( abs_y<15 &&
abs_x<hole[abs_y] )
                                                                begin
                                                                    moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                                end
                                                            else
                                                                begin
                                                                    moleExist
<= 0;

mole_address <= 18'bZ ;
                                                                end

                                                                end
                                                                2'b01:begin
//evil
                                                                //mole out
                                                                // inside mole area
                                                            if (y_cnt < y_offset +
135 && y_cnt > y_offset + 135 - moleHeight[2]&& bound_x >
evilLeftBound[y_cnt-135-y_offset+moleHeight[2]]+1 &&bound_x <
evilRightBound[y_cnt-135-y_offset+moleHeight[2]]-1    )
                                                                    begin
                                                                        //update the
mole_address
                                                                    mole_address <=
evilOffset + moleBase + (y_cnt-135-y_offset+moleHeight[2])*120 + x_cnt-x_offset-40;
                                                                    moleExist
<= 1;
                                                                end
                                                                //hole area
                                                            else if( abs_y<15 &&
abs_x<hole[abs_y] )
                                                                begin
                                                                    moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                                end
                                                            else
                                                                begin
                                                                    moleExist
<= 0;

mole_address <= 18'bZ ;
```

```
                                                    end

                                                    end

                                                endcase
                                                    end
                                                //no mole
                                                 else
                                                   if(              abs_y<15              &&
abs_x<hole[abs_y] )
                                                       begin
                                                            moleExist <= 1;
                                                            mole_address        <=
moleBase + 120*120;
                                                        end
                                                 else
                                                       begin
                                                            moleExist <= 0;
                                                            mole_address       <=
18'bZ ;
                                                        end
                                             end
                    16'b0000000000001000: begin    x_offset <= Hsync_Blank + 400;
                                                    y_offset <= Vertical_Back_Porch + 160;
                                                    //get the diff between the current x_cnt
and the left boundary of the hole
                                                    bound_x <=    -x_cnt+x_offset+200-40;
                                                    //get the diff between the current
y_cnt and the horizontal center of the hole
                                                    if( y_cnt>120+y_offset)
                                                       begin
                                                         abs_y                          <=
      y_cnt-120-y_offset;
                                                        end
                                                    else
                                                       begin
                                                          abs_y                         <=
      -y_cnt+120+y_offset;
                                                       end
                                                    //get the diff between the current
x_cnt and the vertical center of the hole
                                                    if(x_cnt>100+x_offset)
                                                       begin
                                                         abs_x                          <=
      x_cnt-100-x_offset;
                                                        end
                                                    else
                                                       begin
                                                          abs_x                         <=
      -x_cnt+100+x_offset;
                                                        end


                                                    //mole out of the hole
                                                    if(moleHeight[3]>0)
                                                       begin
                                                            case(mole3)
                                                              2'b00:begin      //mole
```

```
                                                                    //mole out
                                                                    // inside mole area
                                                                if (y_cnt < y_offset + 135 && y_cnt > y_offset + 135 - moleHeight[3]&& bound_x > moleLeftBound[y_cnt-135-y_offset+moleHeight[3]]+1 &&bound_x < moleRightBound[y_cnt-135-y_offset+moleHeight[3]]-1 )
                                                                    begin
                                                                        //update the mole_address
                                                                        mole_address <= moleBase + (y_cnt-135-y_offset+moleHeight[3])*120 + x_cnt-x_offset-40;
                                                                        moleExist <= 1;
                                                                    end
                                                                    //hole area
                                                                else if( abs_y<15 && abs_x<hole[abs_y] )
                                                                    begin
                                                                        moleExist <= 1;

mole_address <= moleBase + 120*120;
                                                                    end
                                                                else
                                                                    begin
                                                                        moleExist <= 0;

mole_address <= 18'bZ ;
                                                                    end
                                                                end
                                                                2'b10:begin //bottle
                                                                    //mole out
                                                                    // inside mole area
                                                                if (y_cnt < y_offset + 135 && y_cnt > y_offset + 135 - moleHeight[3]&& bound_x > bottleLeftBound[y_cnt-135-y_offset+moleHeight[3]]+1 &&bound_x < bottleRightBound[y_cnt-135-y_offset+moleHeight[3]]-1 )
                                                                    begin
                                                                        //update the mole_address
                                                                        mole_address <= bottleOffset + moleBase + (y_cnt-135-y_offset+moleHeight[3])*120 + x_cnt-x_offset-40;
                                                                        moleExist <= 1;
                                                                    end
                                                                    //hole area
                                                                else if( abs_y<15 && abs_x<hole[abs_y] )
                                                                    begin
                                                                        moleExist <= 1;

mole_address <= moleBase + 120*120;
                                                                    end
                                                                else
                                                                    begin
                                                                        moleExist
```

```
<= 0;

mole_address <= 18'bZ ;
                                                                    end
                                                                end

                                                                2'b01:begin     //evil
                                                                    //mole out
                                                                    // inside mole area
                                                                if (y_cnt < y_offset + 135 && y_cnt > y_offset + 135 - moleHeight[3]&& bound_x > evilLeftBound[y_cnt-135-y_offset+moleHeight[3]]+1 &&bound_x < evilRightBound[y_cnt-135-y_offset+moleHeight[3]]-1 )
                                                                    begin
                                                                        //update the mole_address
                                                                        mole_address <= evilOffset + moleBase + (y_cnt-135-y_offset+moleHeight[3])*120 + x_cnt-x_offset-40;
                                                                        moleExist <= 1;
                                                                    end
                                                                    //hole area
                                                                else if( abs_y<15 && abs_x<hole[abs_y] )
                                                                    begin
                                                                        moleExist <= 1;

mole_address <= moleBase + 120*120;
                                                                    end
                                                                else
                                                                    begin
                                                                        moleExist <= 0;

mole_address <= 18'bZ ;
                                                                    end
                                                                end




                                                            endcase
                                                        end
                                                    //no mole
                                                    else
                                                        if( abs_y<15 && abs_x<hole[abs_y] )
                                                            begin
                                                                moleExist <= 1;
                                                                mole_address <= moleBase + 120*120;
                                                            end
                                                    else
                                                            begin
                                                                moleExist <= 0;
```

```
                                                                mole_address      <=
18'bZ ;
                                                            end
                                                    end
                    16'b0000000000010000: begin    x_offset <= Hsync_Blank + 200;
                                                        y_offset <= Vertical_Back_Porch + 160;
                                                        //get the diff between the current x_cnt
and the left boundary of the hole
                                                     bound_x <=    -x_cnt+x_offset+200-40;
                                                        //get the diff between the current
y_cnt and the horizontal center of the hole
                                                      if( y_cnt>120+y_offset)
                                                        begin
                                                          abs_y                          <=
      y_cnt-120-y_offset;
                                                          end
                                                    else
                                                        begin
                                                          abs_y                          <=
      -y_cnt+120+y_offset;
                                                      end
                                                    //get the diff between the current
x_cnt and the vertical center of the hole
                                                    if(x_cnt>100+x_offset)
                                                        begin
                                                          abs_x                          <=
      x_cnt-100-x_offset;
                                                          end
                                                    else
                                                        begin
                                                          abs_x                          <=
      -x_cnt+100+x_offset;
                                                          end


                                                    //mole out of the hole
                                                    if(moleHeight[4]>0)
                                                        begin
                                                            case(mole4)
                                                              2'b00:begin     //mole
                                                                    //mole out
                                                                    // inside mole area
                                                                if    (y_cnt  <  y_offset  +
135    &&    y_cnt    >    y_offset    +    135    -    moleHeight[4]&&    bound_x    >
moleLeftBound[y_cnt-135-y_offset+moleHeight[4]]+1                &&bound_x                <
moleRightBound[y_cnt-135-y_offset+moleHeight[4]]-1    )
                                                                    begin
                                                                        //update the
mole_address
                                                                    mole_address <=
moleBase + (y_cnt-135-y_offset+moleHeight[4])*120 + x_cnt-x_offset-40;
                                                                    moleExist
<= 1;
                                                                end
                                                                //hole area
                                                            else    if(    abs_y<15    &&
abs_x<hole[abs_y] )
                                                                begin
```

```
                                                                moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                            end
                                                        else
                                                            begin
                                                                moleExist
<= 0;

mole_address <= 18'bZ ;
                                                            end
                                                            end

                                                             2'b10:begin
//bottle
                                                             //mole out
                                                             // inside mole area
                                                        if   (y_cnt  <  y_offset  +
135  &&  y_cnt  >  y_offset  +  135  -  moleHeight[4]&&  bound_x  >
bottleLeftBound[y_cnt-135-y_offset+moleHeight[4]]+1          &&bound_x          <
bottleRightBound[y_cnt-135-y_offset+moleHeight[4]]-1   )
                                                             begin
                                                                    //update  the
mole_address
                                                                mole_address <=
bottleOffset + moleBase + (y_cnt-135-y_offset+moleHeight[4])*120 + x_cnt-x_offset-40;
                                                                moleExist
<= 1;
                                                            end
                                                             //hole area
                                                        else   if(   abs_y<15   &&
abs_x<hole[abs_y] )
                                                            begin
                                                                moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                            end
                                                        else
                                                            begin
                                                                moleExist
<= 0;

mole_address <= 18'bZ ;
                                                            end
                                                            end

                                                            2'b01:begin     //evil
                                                             //mole out
                                                             // inside mole area
                                                        if   (y_cnt  <  y_offset  +
135  &&  y_cnt  >  y_offset  +  135  -  moleHeight[4]&&  bound_x  >
evilLeftBound[y_cnt-135-y_offset+moleHeight[4]]+1          &&bound_x          <
evilRightBound[y_cnt-135-y_offset+moleHeight[4]]-1   )
                                                             begin
                                                                    //update  the
mole_address
```

```
                                                                    mole_address <=
evilOffset + moleBase + (y_cnt-135-y_offset+moleHeight[4])*120 + x_cnt-x_offset-40;
                                                                        moleExist
<= 1;
                                                                    end
                                                                     //hole area
                                                                else    if(    abs_y<15    &&
abs_x<hole[abs_y] )
                                                                    begin
                                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                                    end
                                                                else
                                                                    begin
                                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                                    end
                                                                    end




                                                                endcase
                                                                    end
                                                            //no mole
                                                             else
                                                                if(              abs_y<15              &&
abs_x<hole[abs_y] )
                                                                    begin
                                                                        moleExist <= 1;
                                                                        mole_address        <=
moleBase + 120*120;
                                                                     end
                                                             else
                                                                    begin
                                                                        moleExist <= 0;
                                                                        mole_address      <=
18'bZ ;
                                                                     end
                                                          end
                    16'b0000000000100000: begin    x_offset <= Hsync_Blank ;
                                                        y_offset <= Vertical_Back_Porch +160;
                                                            //get the diff between the current x_cnt
and the left boundary of the hole
                                                         bound_x <=     -x_cnt+x_offset+200-40;
                                                               //get the diff between the current
y_cnt and the horizontal center of the hole
                                                                 if( y_cnt>120+y_offset)
                                                                    begin
                                                                      abs_y                          <=
       y_cnt-120-y_offset;
                                                                     end
                                                                else
```

```
                                                    begin
                                                        abs_y                    <=
    -y_cnt+120+y_offset;
                                                    end
                                                //get the diff between the current
x_cnt and the vertical center of the hole
                                                if(x_cnt>100+x_offset)
                                                    begin
                                                      abs_x                      <=
    x_cnt-100-x_offset;
                                                     end
                                                else
                                                    begin
                                                        abs_x                    <=
    -x_cnt+100+x_offset;
                                                     end


                                                //mole out of the hole
                                                if(moleHeight[5]>0)
                                                    begin
                                                        case(mole5)
                                                           2'b00:begin          //mole


                                                                  //mole out
                                                                  // inside mole area
                                                               if    (y_cnt    <    y_offset    +
135    &&    y_cnt    >    y_offset    +    135    -    moleHeight[5]&&    bound_x    >
moleLeftBound[y_cnt-135-y_offset+moleHeight[5]]+1          &&bound_x          <
moleRightBound[y_cnt-135-y_offset+moleHeight[5]]-1    )
                                                                  begin
                                                                        //update the
mole_address
                                                                     mole_address <=
moleBase + (y_cnt-135-y_offset+moleHeight[5])*120 + x_cnt-x_offset-40;
                                                                        moleExist
<= 1;
                                                                  end
                                                                   //hole area
                                                              else    if(    abs_y<15    &&
abs_x<hole[abs_y] )
                                                                  begin
                                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                                  end
                                                              else
                                                                  begin
                                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                                  end
                                                                  end

                                                                  2'b10:begin
```

```
//bottle

                                                                    //mole out
                                                                    // inside mole area
                                                                if (y_cnt < y_offset + 135 && y_cnt > y_offset + 135 - moleHeight[5]&& bound_x > bottleLeftBound[y_cnt-135-y_offset+moleHeight[5]]+1 &&bound_x < bottleRightBound[y_cnt-135-y_offset+moleHeight[5]]-1 )
                                                                    begin
                                                                        //update the mole_address
                                                                        mole_address <= bottleOffset + moleBase + (y_cnt-135-y_offset+moleHeight[5])*120 + x_cnt-x_offset-40;
                                                                        moleExist <= 1;
                                                                    end
                                                                    //hole area
                                                                else if( abs_y<15 && abs_x<hole[abs_y] )
                                                                    begin
                                                                        moleExist <= 1;

mole_address <= moleBase + 120*120;
                                                                    end
                                                                else
                                                                    begin
                                                                        moleExist <= 0;

mole_address <= 18'bZ ;
                                                                    end
                                                                    end

                                                                    2'b01:begin     //evil

                                                                    //mole out
                                                                    // inside mole area
                                                                if (y_cnt < y_offset + 135 && y_cnt > y_offset + 135 - moleHeight[5]&& bound_x > evilLeftBound[y_cnt-135-y_offset+moleHeight[5]]+1 &&bound_x < evilRightBound[y_cnt-135-y_offset+moleHeight[5]]-1 )
                                                                    begin
                                                                        //update the mole_address
                                                                        mole_address <= evilOffset + moleBase + (y_cnt-135-y_offset+moleHeight[5])*120 + x_cnt-x_offset-40;
                                                                        moleExist <= 1;
                                                                    end
                                                                    //hole area
                                                                else if( abs_y<15 && abs_x<hole[abs_y] )
                                                                    begin
                                                                        moleExist <= 1;

mole_address <= moleBase + 120*120;
```

```
                                                    end
                                                else
                                                    begin
                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                    end
                                                    end



                                                endcase
                                                    end
                                            //no mole
                                              else
                                                if(          abs_y<15          &&
abs_x<hole[abs_y] )
                                                    begin
                                                        moleExist <= 1;
                                                        mole_address      <=
moleBase + 120*120;
                                                     end
                                                else
                                                    begin
                                                        moleExist <= 0;
                                                        mole_address      <=
18'bZ ;
                                                     end
                                            end
                16'b0000000001000000: begin    x_offset <= Hsync_Blank + 400;
                                                y_offset <= Vertical_Back_Porch +320;
                                                    //get the diff between the current x_cnt
and the left boundary of the hole
                                                 bound_x <=    -x_cnt+x_offset+200-40;
                                                    //get the diff between the current
y_cnt and the horizontal center of the hole
                                                  if( y_cnt>120+y_offset)
                                                    begin
                                                      abs_y                      <=
      y_cnt-120-y_offset;
                                                      end
                                                else
                                                    begin
                                                       abs_y                      <=
      -y_cnt+120+y_offset;
                                                   end
                                                //get the diff between the current
x_cnt and the vertical center of the hole
                                                if(x_cnt>100+x_offset)
                                                    begin
                                                      abs_x                      <=
      x_cnt-100-x_offset;
                                                     end
                                                else
                                                    begin
                                                       abs_x                      <=
      -x_cnt+100+x_offset;
```

```
                                                    end


                                                //mole out of the hole
                                                if(moleHeight[6]>0)
                                                    begin
                                                        case(mole6)
                                                            2'b00:begin          //mole


                                                                //mole out
                                                                // inside mole area
                                                            if    (y_cnt  <  y_offset  +
135    &&    y_cnt    >    y_offset    +    135    -    moleHeight[6]&&    bound_x    >
moleLeftBound[y_cnt-135-y_offset+moleHeight[6]]+1              &&bound_x              <
moleRightBound[y_cnt-135-y_offset+moleHeight[6]]-1    )
                                                                begin
                                                                        //update  the
mole_address
                                                                    mole_address <=
moleBase + (y_cnt-135-y_offset+moleHeight[6])*120 + x_cnt-x_offset-40;
                                                                        moleExist
<= 1;
                                                                end
                                                                //hole area
                                                            else    if(    abs_y<15    &&
abs_x<hole[abs_y] )
                                                                begin
                                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                                end
                                                            else
                                                                begin
                                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                                end
                                                                end


                                                                2'b10:begin
//mole

                                                                //mole out
                                                                // inside mole area
                                                            if    (y_cnt  <  y_offset  +
135    &&    y_cnt    >    y_offset    +    135    -    moleHeight[6]&&    bound_x    >
bottleLeftBound[y_cnt-135-y_offset+moleHeight[6]]+1              &&bound_x              <
bottleRightBound[y_cnt-135-y_offset+moleHeight[6]]-1    )
                                                                begin
                                                                        //update  the
mole_address
                                                                    mole_address <=
bottleOffset + moleBase + (y_cnt-135-y_offset+moleHeight[6])*120 + x_cnt-x_offset-40;
                                                                        moleExist
<= 1;
```

```
                                                    end
                                                    //hole area
                                                else if( abs_y<15 &&
abs_x<hole[abs_y] )
                                                    begin
                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                    end
                                                else
                                                    begin
                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                    end
                                                    end

                                                    2'b01:begin
//mole

                                                    //mole out
                                                    // inside mole area
                                                if (y_cnt < y_offset +
135 && y_cnt > y_offset + 135 - moleHeight[6]&& bound_x >
evilLeftBound[y_cnt-135-y_offset+moleHeight[6]]+1 &&bound_x <
evilRightBound[y_cnt-135-y_offset+moleHeight[6]]-1 )
                                                    begin
                                                        //update the
mole_address
                                                    mole_address <=
evilOffset + moleBase + (y_cnt-135-y_offset+moleHeight[6])*120 + x_cnt-x_offset-40;
                                                        moleExist
<= 1;
                                                    end
                                                    //hole area
                                                else if( abs_y<15 &&
abs_x<hole[abs_y] )
                                                    begin
                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                    end
                                                else
                                                    begin
                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                    end
                                                    end
```

```
                                        endcase
                                            end
                                    //no mole
                                     else
                                       if(          abs_y<15          &&
abs_x<hole[abs_y] )
                                           begin
                                                   moleExist <= 1;
                                                   mole_address      <=
moleBase + 120*120;
                                            end
                                     else
                                           begin
                                                    moleExist <= 0;
                                                    mole_address     <=
18'bZ ;
                                            end

                                  end
                16'b0000000010000000: begin   x_offset <= Hsync_Blank + 200;
                                        y_offset <= Vertical_Back_Porch +320;
                                          //get the diff between the current x_cnt
and the left boundary of the hole
                                         bound_x <=    -x_cnt+x_offset+200-40;
                                          //get the diff between the current
y_cnt and the horizontal center of the hole
                                           if( y_cnt>120+y_offset)
                                              begin
                                                abs_y                       <=
      y_cnt-120-y_offset;
                                               end
                                    else
                                          begin
                                              abs_y                         <=
      -y_cnt+120+y_offset;
                                           end
                                    //get the diff between the current
x_cnt and the vertical center of the hole
                                    if(x_cnt>100+x_offset)
                                          begin
                                            abs_x                         <=
      x_cnt-100-x_offset;
                                           end
                                    else
                                          begin
                                              abs_x                       <=
      -x_cnt+100+x_offset;
                                           end


                                    //mole out of the hole
                                    if(moleHeight[7]>0)
                                           begin
                                               case(mole7)
                                                  2'b00:begin          //mole


                                                          //mole out
```

```
                                                    // inside mole area
                                                if (y_cnt < y_offset +
135 && y_cnt > y_offset + 135 - moleHeight[7]&& bound_x >
moleLeftBound[y_cnt-135-y_offset+moleHeight[7]]+1 &&bound_x <
moleRightBound[y_cnt-135-y_offset+moleHeight[7]]-1 )
                                                    begin
                                                        //update the
mole_address
                                                    mole_address <=
moleBase + (y_cnt-135-y_offset+moleHeight[7])*120 + x_cnt-x_offset-40;
                                                        moleExist
<= 1;
                                                    end
                                                    //hole area
                                                else if( abs_y<15 &&
abs_x<hole[abs_y] )
                                                    begin
                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                    end
                                                else
                                                    begin
                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                    end
                                                    end

                                                    2'b10:begin
//mole

                                                    //mole out
                                                    // inside mole area
                                                if (y_cnt < y_offset +
135 && y_cnt > y_offset + 135 - moleHeight[7]&& bound_x >
bottleLeftBound[y_cnt-135-y_offset+moleHeight[7]]+1 &&bound_x <
bottleRightBound[y_cnt-135-y_offset+moleHeight[7]]-1 )
                                                    begin
                                                        //update the
mole_address
                                                    mole_address <=
bottleOffset + moleBase + (y_cnt-135-y_offset+moleHeight[7])*120 + x_cnt-x_offset-40;
                                                        moleExist
<= 1;
                                                    end
                                                    //hole area
                                                else if( abs_y<15 &&
abs_x<hole[abs_y] )
                                                    begin
                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                    end
                                                else
```

```
                                                                    begin
                                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                                    end
                                                                    end

                                                                    2'b01:begin
//mole

                                                                     //mole out
                                                                     // inside mole area
                                                                  if  (y_cnt  <  y_offset  +
135  &&  y_cnt  >  y_offset  +  135  -  moleHeight[7]&&  bound_x  >
evilLeftBound[y_cnt-135-y_offset+moleHeight[7]]+1          &&bound_x          <
evilRightBound[y_cnt-135-y_offset+moleHeight[7]]-1   )
                                                                     begin
                                                                         //update  the
mole_address
                                                                        mole_address <=
evilOffset + moleBase + (y_cnt-135-y_offset+moleHeight[7])*120 + x_cnt-x_offset-40;
                                                                        moleExist
<= 1;
                                                                    end
                                                                     //hole area
                                                                 else  if(  abs_y<15  &&
abs_x<hole[abs_y] )
                                                                    begin
                                                                        moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                                    end
                                                                 else
                                                                    begin
                                                                        moleExist
<= 0;

mole_address <= 18'bZ ;
                                                                    end
                                                                    end




                                                               endcase
                                                                   end
                                                            //no mole
                                                              else
                                                                if(          abs_y<15          &&
abs_x<hole[abs_y] )
                                                                   begin
                                                                        moleExist <= 1;
                                                                        mole_address      <=
moleBase + 120*120;
                                                                    end
                                                              else
```

```
                                                    begin
                                                        moleExist <= 0;
                                                        mole_address      <=
18'bZ ;
                                                    end
                                                end
                    16'b0000000100000000: begin x_offset <= Hsync_Blank;
                                                y_offset <= Vertical_Back_Porch+320;
                                                //get the diff between the current x_cnt
and the left boundary of the hole
                                                bound_x <=    -x_cnt+x_offset+200-40;
                                                //get the diff between the current
y_cnt and the horizontal center of the hole
                                                if( y_cnt>120+y_offset)
                                                    begin
                                                        abs_y                    <=
    y_cnt-120-y_offset;
                                                    end
                                                else
                                                    begin
                                                        abs_y                    <=
    -y_cnt+120+y_offset;
                                                    end
                                                //get the diff between the current
x_cnt and the vertical center of the hole
                                                if(x_cnt>100+x_offset)
                                                    begin
                                                        abs_x                    <=
    x_cnt-100-x_offset;
                                                    end
                                                else
                                                    begin
                                                        abs_x                    <=
    -x_cnt+100+x_offset;
                                                    end


                                                //mole out of the hole
                                                if(moleHeight[8]>0)
                                                    begin
                                                        case(mole8)
                                                            2'b00:begin          //mole


                                                                //mole out
                                                                // inside mole area
                                                            if    (y_cnt    <    y_offset    +
135    &&    y_cnt    >    y_offset    +    135    -    moleHeight[8]&&    bound_x    >
moleLeftBound[y_cnt-135-y_offset+moleHeight[8]]+1          &&bound_x          <
moleRightBound[y_cnt-135-y_offset+moleHeight[8]]-1    )
                                                                begin
                                                                    //update the
mole_address
                                                                    mole_address <=
moleBase + (y_cnt-135-y_offset+moleHeight[8])*120 + x_cnt-x_offset-40;
                                                                    moleExist
<= 1;
                                                                end
```

```
                                                        //hole area
                                                    else if( abs_y<15 &&
abs_x<hole[abs_y] )
                                                        begin
                                                            moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                        end
                                                    else
                                                        begin
                                                            moleExist
<= 0;

mole_address <= 18'bZ ;
                                                        end
                                                    end

                                                    2'b10:begin     //mole


                                                        //mole out
                                                        // inside mole area
                                                    if (y_cnt < y_offset +
135 && y_cnt > y_offset + 135 - moleHeight[8]&& bound_x >
bottleLeftBound[y_cnt-135-y_offset+moleHeight[8]]+1 &&bound_x <
bottleRightBound[y_cnt-135-y_offset+moleHeight[8]]-1 )
                                                        begin
                                                            //update the
mole_address
                                                            mole_address <=
bottleOffset+ moleBase + (y_cnt-135-y_offset+moleHeight[8])*120 + x_cnt-x_offset-40;
                                                            moleExist
<= 1;
                                                        end
                                                        //hole area
                                                    else if( abs_y<15 &&
abs_x<hole[abs_y] )
                                                        begin
                                                            moleExist
<= 1;

mole_address <= moleBase + 120*120;
                                                        end
                                                    else
                                                        begin
                                                            moleExist
<= 0;

mole_address <= 18'bZ ;
                                                        end
                                                    end

                                                    2'b01:begin     //mole


                                                        //mole out
                                                        // inside mole area
```

```
                                                    if (y_cnt < y_offset + 135 && y_cnt > y_offset + 135 - moleHeight[8]&& bound_x > evilLeftBound[y_cnt-135-y_offset+moleHeight[8]]+1 &&bound_x < evilRightBound[y_cnt-135-y_offset+moleHeight[8]]-1 )
                                                        begin
                                                            //update the mole_address
                                                            mole_address <= evilOffset+ moleBase + (y_cnt-135-y_offset+moleHeight[8])*120 + x_cnt-x_offset-40;
                                                            moleExist <= 1;
                                                        end
                                                    //hole area
                                                    else if( abs_y<15 && abs_x<hole[abs_y] )
                                                        begin
                                                            moleExist <= 1;

mole_address <= moleBase + 120*120;
                                                        end
                                                    else
                                                        begin
                                                            moleExist <= 0;

mole_address <= 18'bZ ;
                                                        end
                                                end



                                            endcase
                                        end
                                    //no mole
                                    else
                                        if( abs_y<15 && abs_x<hole[abs_y] )
                                            begin
                                                moleExist <= 1;
                                                mole_address <= moleBase + 120*120;
                                            end
                                        else
                                            begin
                                                moleExist <= 0;
                                                mole_address <= 18'bZ ;
                                            end
                                    end
                16'b0000001000000000: begin moleExist <= 0;
                                                mole_address <= 18'bZ ;end
                16'b0000010000000000: begin moleExist <= 0;
                                                mole_address <= 18'bZ ;end
                16'b0000100000000000: begin moleExist <= 0;
```

```
                                                                    mole_address      <=
18'bZ ;end
                16'b0001000000000000: begin moleExist <= 0;
                                                                    mole_address      <=
18'bZ ;end
                16'b0010000000000000: begin moleExist <= 0;
                                                                    mole_address      <=
18'bZ ;end
                16'b0100000000000000: begin moleExist <= 0;
                                                                    mole_address      <=
18'bZ ;end
                16'b1000000000000000: begin moleExist <= 0;
                                                                    mole_address      <=
18'bZ ;end
                default: begin moleExist <= 0;
                                                                    mole_address      <=
18'bZ ;     end
            endcase
        end


endmodule
```

musiscontroller.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity musiccontroller is
port (

        clk             : in    std_logic;
        reset_n         : in    std_logic;
        read             : in    std_logic;
        write           : in    std_logic;
        chipselect : in    std_logic;
        address         : in    std_logic_vector(16 downto 0);
        readdata        : out std_logic_vector(15 downto 0);
        writedata     : in    std_logic_vector(15 downto 0);
        irq             : out std_logic;


        AUD_ADCLRCK    : out    std_logic;      --       Audio CODEC ADC LR Clock
        AUD_ADCDAT      : in      std_logic;      --       Audio CODEC ADC Data
        AUD_DACLRCK    : out    std_logic;      --       Audio CODEC DAC LR Clock
        AUD_DACDAT      : out    std_logic;      --       Audio CODEC DAC Data
        AUD_BCLK         : inout std_logic; --       Audio CODEC Bit-Stream Clock
        AUD_XCK          : out std_logic;
        clk_18            : in std_logic
    );

    end musiccontroller;

    architecture mc of musiccontroller is
```

```
   signal audio_request : std_logic;
   signal audio_clock : unsigned(1 downto 0) := "00";
   signal store : std_logic;
--    signal clk_18:std_logic;



     component de2_wm8731_audio is port(
                clk_chip : in std_logic;
                  clk : in std_logic;              --    Audio CODEC Chip Clock AUD_XCK (18.43 MHz)
                    reset_n : in std_logic;
                  test_mode : in std_logic;              --       Audio CODEC controller test mode
                  audio_request  :  out  std_logic_vector(15  downto  0);  --       Audio  controller
request new data
                  data             : in std_logic_vector(15 downto 0);
                    address        : in    std_logic_vector(16 downto 0);
                    write          : in std_logic;
                    chipselect : in std_logic;
              irq               : out std_logic;

                  -- Audio interface signals
                    AUD_ADCLRCK    : out    std_logic;      --       Audio CODEC ADC LR Clock
                  AUD_ADCDAT      : in      std_logic;      --       Audio CODEC ADC Data
                  AUD_DACLRCK    : out    std_logic;      --       Audio CODEC DAC LR Clock
                  AUD_DACDAT      : out    std_logic;      --       Audio CODEC DAC Data
                  AUD_BCLK         : inout std_logic--           Audio CODEC Bit-Stream Clock

              );
              end component de2_wm8731_audio;



              component de2_audio_pll is
PORT
(
inclk0             : IN STD_LOGIC    := '0';
c0          : OUT STD_LOGIC
);end component de2_audio_pll;



begin

  AUD_XCK <= clk_18;

process (clk)
   begin
     if rising_edge(clk) then
           audio_clock <= audio_clock + "1";
       end if;
end process;

--     process (clk)
--     begin
--           if rising_edge(clk) then
--                 if reset_n = '0' then
--                           store <= '0';
--                 else
```

```
--                          if chipSelect = '1' and write = '1'    then
--                                store <= '1';
--                          end if;
--                    end if;
--              end if;
--      end process;


V1: de2_wm8731_audio port map (
        clk_chip => clk,
        clk => clk_18,
        reset_n      => '1',
        test_mode => '0',
        audio_request => readdata,
        data =>    writedata ,
          address => address,
          write      => write,
          chipselect => chipselect,
          irq          =>    irq,

        -- Audio interface signals
        AUD_ADCLRCK    => AUD_ADCLRCK,
        AUD_ADCDAT      => AUD_ADCDAT,
        AUD_DACLRCK    => AUD_DACLRCK,
        AUD_DACDAT      => AUD_DACDAT,
        AUD_BCLK          => AUD_BCLK
    );


--    V2: de2_audio_pll port map(
--    inclk0 => clk_27,
--    c0          => clk_18
--    );
    end architecture;
```

No_GEN.v

```
//////////////////////Number Display Module//////////////////////////
module No_GEN(
                          iclk,
                          iReset_n,
                          ixcount,
                          iycount,
                          iled_pos_x,
                          iled_pos_y,
                          inumber,
                          oOVERRIDE,
                          oR,
                          oG,
                          oB
                          );

//Port IO
input iclk;
input iReset_n;
input [3:0]inumber;
```

```
input [10:0] ixcount;
input [9:0]    iycount;
input [10:0] iled_pos_x;
input [10:0] iled_pos_y;
output oOVERRIDE;
output [7:0] oR;
output [7:0] oG;
output [7:0] oB;

//Wire & Reg
reg [7:0]    oR;
reg [7:0]    oG;
reg [7:0]    oB;
reg oOVERRIDE;
wire display_area;
wire number_area;
wire a;
wire b;
wire c;
wire d;
wire e;
wire f;
wire g;
wire a_en;
wire b_en;
wire c_en;
wire d_en;
wire e_en;
wire f_en;
wire g_en;

//Structure
assign display_area = ((ixcount > (iled_pos_x -1))
                                        && (ixcount < (iled_pos_x+ 60) )
                                        && (iycount > (iled_pos_y -1))
                                        && (iycount < (iled_pos_y +80))) ? 1'b1:1'b0;

assign number_area = ((ixcount > (iled_pos_x +4))
                                        && (ixcount < (iled_pos_x+ 35) )
                                        && (iycount > (iled_pos_y + 9))
                                        && (iycount < (iled_pos_y +70))) ? 1'b1:1'b0;

assign a = ((ixcount > (iled_pos_x + 4))
                        && (ixcount < (iled_pos_x + 35))
                        &&(iycount > (iled_pos_y + 9))
                        &&(iycount < (iled_pos_y + 18))) ? 1'b1 : 1'b0;
assign b =        ((ixcount > (iled_pos_x + 4))
                        && (ixcount < (iled_pos_x + 13))
                        &&(iycount > (iled_pos_y + 9))
                        &&(iycount < (iled_pos_y + 40))) ? 1'b1 : 1'b0;
assign c =    ((ixcount > (iled_pos_x + 4))
                        && (ixcount < (iled_pos_x + 13))
                        &&(iycount > (iled_pos_y + 39))
                        &&(iycount < (iled_pos_y + 70))) ? 1'b1 : 1'b0;
assign d =          ((ixcount > (iled_pos_x + 4))
                        && (ixcount < (iled_pos_x + 35))
                        &&(iycount > (iled_pos_y + 61))
                        &&(iycount < (iled_pos_y + 70))) ? 1'b1 : 1'b0;
```

```
assign e =      ((ixcount > (iled_pos_x + 26))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 39))
                    &&(iycount < (iled_pos_y + 70))) ? 1'b1 : 1'b0;
assign f =      ((ixcount > (iled_pos_x + 26))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 9))
                    &&(iycount < (iled_pos_y + 40))) ? 1'b1 : 1'b0;
assign g =      ((ixcount > (iled_pos_x + 4))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 35))
                    &&(iycount < (iled_pos_y + 44))) ? 1'b1 : 1'b0;

//assign a_en = ((inumber != 1) && (inumber != 4)) ? 1'b1 : 1'b0;
//assign b_en = ((inumber != 5) && (inumber != 6)) ? 1'b1 : 1'b0;
//assign c_en = ((inumber != 2)) ? 1'b1 : 1'b0;
//assign d_en = ((inumber != 1) && (inumber != 4) && (inumber != 7)) ? 1'b1 : 1'b0;
//assign e_en = ((inumber == 2) || (inumber == 6) || (inumber == 8) || (inumber == 0)) ? 1'b1:
1'b0;
//assign f_en = ((inumber == 4) || (inumber == 5) || (inumber == 6) || (inumber == 8)
//                              || (inumber == 9) || (inumber == 0)) ? 1'b1: 1'b0;
//assign g_en = ((inumber != 1) && (inumber != 7) && (inumber != 0)) ? 1'b1 : 1'b0;

always @(posedge iclk or negedge iReset_n)
begin
    if(!iReset_n)
            oOVERRIDE <= 1'b0;
    else
    case(inumber)
        0: oOVERRIDE <= (a||b||c||d||e||f);
        1: oOVERRIDE <= (b||c);
        2: oOVERRIDE <= (a||b||g||e||d);
        3: oOVERRIDE <= (a||b||c||d||g);
        4: oOVERRIDE <= (b||c||f||g);
        5: oOVERRIDE <= (a||c||d||f||g);
        6: oOVERRIDE <= (a||c||d||e||f||g);
        7: oOVERRIDE <= (a||b||c);
        8: oOVERRIDE <= (a||b||c||d||e||f||g);
        9: oOVERRIDE <= (a||b||c||d||f||g);
        default : oOVERRIDE <= 1'b0;
    endcase
    oR <= 8'd255;
    oG <= 8'd255;
    oB <= 8'd255;
end
endmodule
```

number.v

```
//////////////////////Number Display Module//////////////////////////
module number(
                    iclk,
                    iReset_n,
                    ixcount,
                    iycount,
                    iled_pos_x,
```

```
                        iled_pos_y,
                        inumber,
                        oOVERRIDE,
                        );

//Port IO
input iclk;
input iReset_n;
input [3:0]inumber;
input [10:0] ixcount;
input [9:0]    iycount;
input [10:0] iled_pos_x;
input [10:0] iled_pos_y;
output oOVERRIDE;

//Wire & Reg
reg oOVERRIDE;
wire display_area;
wire number_area;
wire a;
wire b;
wire c;
wire d;
wire e;
wire f;
wire g;
wire a_en;
wire b_en;
wire c_en;
wire d_en;
wire e_en;
wire f_en;
wire g_en;

//Structure
assign display_area = ((ixcount > (iled_pos_x -1))
                                        && (ixcount < (iled_pos_x+ 60) )
                                        && (iycount > (iled_pos_y -1))
                                        && (iycount < (iled_pos_y +80))) ? 1'b1:1'b0;

assign number_area = ((ixcount > (iled_pos_x +4))
                                        && (ixcount < (iled_pos_x+ 35) )
                                        && (iycount > (iled_pos_y + 9))
                                        && (iycount < (iled_pos_y +70))) ? 1'b1:1'b0;

assign a = ((ixcount > (iled_pos_x + 4))
                        && (ixcount < (iled_pos_x + 35))
                        &&(iycount > (iled_pos_y + 9))
                        &&(iycount < (iled_pos_y + 18))) ? 1'b1 : 1'b0;
assign b =        ((ixcount > (iled_pos_x + 4))
                        && (ixcount < (iled_pos_x + 13))
                        &&(iycount > (iled_pos_y + 9))
                        &&(iycount < (iled_pos_y + 40))) ? 1'b1 : 1'b0;
assign c =    ((ixcount > (iled_pos_x + 4))
                        && (ixcount < (iled_pos_x + 13))
                        &&(iycount > (iled_pos_y + 39))
                        &&(iycount < (iled_pos_y + 70))) ? 1'b1 : 1'b0;
assign d =          ((ixcount > (iled_pos_x + 4))
```

```
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 61))
                    &&(iycount < (iled_pos_y + 70))) ? 1'b1 : 1'b0;
assign e =      ((ixcount > (iled_pos_x + 26))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 39))
                    &&(iycount < (iled_pos_y + 70))) ? 1'b1 : 1'b0;
assign f =      ((ixcount > (iled_pos_x + 26))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 9))
                    &&(iycount < (iled_pos_y + 40))) ? 1'b1 : 1'b0;
assign g =      ((ixcount > (iled_pos_x + 4))
                    && (ixcount < (iled_pos_x + 35))
                    &&(iycount > (iled_pos_y + 35))
                    &&(iycount < (iled_pos_y + 44))) ? 1'b1 : 1'b0;

//assign a_en = ((inumber != 1) && (inumber != 4)) ? 1'b1 : 1'b0;
//assign b_en = ((inumber != 5) && (inumber != 6)) ? 1'b1 : 1'b0;
//assign c_en = ((inumber != 2)) ? 1'b1 : 1'b0;
//assign d_en = ((inumber != 1) && (inumber != 4) && (inumber != 7)) ? 1'b1 : 1'b0;
//assign e_en = ((inumber == 2) || (inumber == 6) || (inumber == 8) || (inumber == 0)) ? 1'b1:
1'b0;
//assign f_en = ((inumber == 4) || (inumber == 5) || (inumber == 6) || (inumber == 8)
//                    || (inumber == 9) || (inumber == 0)) ? 1'b1: 1'b0;
//assign g_en = ((inumber != 1) && (inumber != 7) && (inumber != 0)) ? 1'b1 : 1'b0;

always @(posedge iclk or negedge iReset_n)
begin
    if(!iReset_n)
        oOVERRIDE <= 1'b0;
    else
    case(inumber)
        0: oOVERRIDE <= (a||b||c||d||e||f);
        1: oOVERRIDE <= (b||c);
        2: oOVERRIDE <= (a||b||g||e||d);
        3: oOVERRIDE <= (a||b||c||d||g);
        4: oOVERRIDE <= (b||c||f||g);
        5: oOVERRIDE <= (a||c||d||f||g);
        6: oOVERRIDE <= (a||c||d||e||f||g);
        7: oOVERRIDE <= (a||b||c);
        8: oOVERRIDE <= (a||b||c||d||e||f||g);
        9: oOVERRIDE <= (a||b||c||d||f||g);
        default : oOVERRIDE <= 1'b0;
    endcase
end
endmodule
```

panel_control.v

```
module panel_ctrl(
                    clk_ltm,
                    reset_n, //top level connection
                    x_cnt, // input from lcd controlelr
                    y_cnt, // input from lcd controller
                    iPANELCTRL, // command from nios containing 32 bits
                    oOVERRIDE,//override output to the lcd_controller
```

```
                        oADDR, //the output address to the lcd controller
                        //ostate
                        );
//===============
//PARAMETER
//==================
parameter BASE_ADDR = 89024;
parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
//=================================
//IO PORT
//=================================
input clk_ltm;
input reset_n;
input [10:0] x_cnt;
input [9:0] y_cnt;
input [31:0] iPANELCTRL;
output oOVERRIDE;
output [17:0] oADDR;
//output [3:0] ostate;
//=================================
//REG and WIRE
//=================================
wire oOVERRIDE;
wire [17:0] oADDR; // output address to lcd controller
wire [15:0] area; // 15 areas in total
wire [17:0] iADDR [0:5]; // address from the 6 peripheral areas
wire [5:0] override; //6 bit override signal from 6 area
wire [3:0] life;
wire [3:0] score0;
wire [3:0] score1;
wire [3:0] score2;
wire [3:0] state; // init->start-> pause

assign          state = iPANELCTRL[3:0];
assign          life =    iPANELCTRL[7:4];
assign          score0 = iPANELCTRL[11:8];
assign          score1 = iPANELCTRL[15:12];
assign          score2 = iPANELCTRL[19:16];
assign          ostate = state;

Timmer Timmer1(
                .clk_ltm(clk_ltm),
                .reset_n(reset_n),
                .x_cnt(x_cnt),
                .y_cnt(y_cnt),
                .state(state),
                .oOVERRIDE(override[0]),
                .oADDR(iADDR[0])
                );

Life Life1(
                .clk_ltm(clk_ltm),
                .reset_n(reset_n),
```

```
                .x_cnt(x_cnt),
                .y_cnt(y_cnt),
                .life(life),
                .oOVERRIDE(override[1]),
                .oADDR(iADDR[1])
                );

Score Score1(
                .clk_ltm(clk_ltm),
                .reset_n(reset_n),
                .x_cnt(x_cnt),
                .y_cnt(y_cnt),
                .iScore0(score0),
                .iScore1(score1),
                .iScore2(score2),
                .oOVERRIDE(override[2]),
                .oADDR(iADDR[2])
                );

quit quit1 (
                .iclk(clk_ltm),
                .ireset_n(reset_n),
                .ix_cnt(x_cnt),
                .iy_cnt(y_cnt),
                .oOVERRIDE(override[5]),
                .oADDR(iADDR[5]));

start start1 (
                .iclk(clk_ltm),
                .ireset_n(reset_n),
                .ix_cnt(x_cnt),
                .iy_cnt(y_cnt),
                .oOVERRIDE(override[3]),
                .oADDR(iADDR[3])
                );

pause pause1 (
                .iclk(clk_ltm),
                .ireset_n(reset_n),
                .ix_cnt(x_cnt),
                .iy_cnt(y_cnt),
                .oOVERRIDE(override[4]),
                .oADDR(iADDR[4])
                );



//Detect which area you are in// 0~14 different areas//offset corrected
assign area[0] = ((x_cnt>(Hsync_Blank-2+400)&& //>214
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-200))&& //< 1015
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0 ; //mole0
assign area[1] = ((x_cnt>(Hsync_Blank-2+200)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0;//mole1
```

```
assign area[2] = ((x_cnt>(Hsync_Blank-2)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0 ;//mole2
assign area[3] = ((x_cnt>(Hsync_Blank-2+400)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-200))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ;    //mole3
assign area[4] = ((x_cnt>(Hsync_Blank-2+200)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ; //mole4
assign area[5] = ((x_cnt>(Hsync_Blank-2)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ;//mole5
assign area[6] = ((x_cnt>(Hsync_Blank-2+400)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-200))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;    //mole6
assign area[7] = ((x_cnt>(Hsync_Blank-2+200)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0; //mole7
assign area[8] = ((x_cnt>(Hsync_Blank-2)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;//mole8
assign area[9] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-400))
                                ))    ? 1'b1 : 1'b0 ;//area 9 Time
assign area[10] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+80))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0 ;//area 10 life
assign area[11] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-240))
                                ))    ? 1'b1 : 1'b0 ;//area 11 score
assign area[12] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+240))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ;//area 12 start/pause
assign area[13] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
```

```
                                (y_cnt<(V_LINE - Vertical_Front_Porch-80))
                                ))    ? 1'b1 : 1'b0 ;//area 13 exit
assign area[14] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+400))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;//area 14 label


assign oOVERRIDE = override[0] || override[1] || override[2] || override[3] || override[4] ||
override[5];
assign oADDR = (area[9] && override[0]) ? iADDR [0] :
                                (area[10] && override[1]) ? iADDR [1] :
                                (area[11] && override[2]) ? iADDR [2] :
                                (area[12] && override[3]) ? iADDR [3] :
                                (area[13] && override[4]) ? iADDR [4] :
                                (area[14] && override[5]) ? iADDR [5] : BASE_ADDR;
endmodule
```

pause.v

```
module pause(
                iclk,
                ireset_n,
                ix_cnt,
                iy_cnt,
                oOVERRIDE,
                oADDR
                        );

//================================================
//I/O DECALARATION==============================
//================================================
input iclk;
input ireset_n;
input [10:0] ix_cnt;
input [9:0] iy_cnt;
output oOVERRIDE;
output [17:0] oADDR;

//================================================
//WIRE & REG====================================
//================================================
wire [10:0]iletter_pos_x;
wire [9:0] iletter_pos_y;
wire iletter;
wire [4:0]override;
wire [17:0] address [0:4];

//================================================
//PARAMETER=====================================
//================================================
parameter A = 5'd1;
parameter B = 5'd2;
parameter C = 5'd3;
parameter D = 5'd4;
```

```
parameter E = 5'd5;
parameter F = 5'd6;
parameter G = 5'd7;
parameter H = 5'd8;
parameter I = 5'd9;
parameter J = 5'd10;
parameter K = 5'd11;
parameter L = 5'd12;
parameter M = 5'd13;
parameter N = 5'd14;
parameter O = 5'd15;
parameter P = 5'd16;
parameter Q = 5'd17;
parameter R = 5'd18;
parameter S = 5'd19;
parameter T = 5'd20;
parameter U = 5'd21;
parameter V = 5'd22;
parameter W = 5'd23;
parameter X = 5'd24;
parameter Y = 5'd25;
parameter Z = 5'd26;
parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
//=============================================
//Structured logci==============================
//=============================================

letter letter0(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+620),
                        .iletter_pos_y(Vertical_Back_Porch-1+344),
                        .iletter(E),
                        .oOVERRIDE(override[0]),
                        .oADDR(address[0]));

letter letter1(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+652),
                        .iletter_pos_y(Vertical_Back_Porch-1+344),
                        .iletter(S),
                        .oOVERRIDE(override[1]),
                        .oADDR(address[1]));

letter letter2(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
```

```
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+684),
                        .iletter_pos_y(Vertical_Back_Porch-1+344),
                        .iletter(U),
                        .oOVERRIDE(override[2]),
                        .oADDR(address[2]));

letter letter3(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+716),
                        .iletter_pos_y(Vertical_Back_Porch-1+344),
                        .iletter(A),
                        .oOVERRIDE(override[3]),
                        .oADDR(address[3]));

letter letter4(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+748),
                        .iletter_pos_y(Vertical_Back_Porch-1+344),
                        .iletter(P),
                        .oOVERRIDE(override[4]),
                        .oADDR(address[4]));

assign oOVERRIDE = (override[0] || override[1] || override [2] || override[3]|| override[4]);
assign oADDR = override[0] ? address[0] :
                                override[1] ? address[1] :
                                override[2] ? address[2] :
                                override[3] ? address[3] :
                                override[4] ? address[4] : 18'bZ;
endmodule
```

play.v

```
module play(
                        iclk,
                        ireset_n,
                        ix_cnt, // input from lcd controlelr
                        iy_cnt, // input from lcd controller
                        iPANELCTRL, // command from nios containing 32 bits
                        iMOLECTRL,
                        oADDR, //the output address to the lcd controller
                                );
//=============================================================================
// PARAMETER declarations
//=============================================================================
parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
```

```
parameter Vertical_Front_Porch = 10;
parameter offset = 32000; //control panel image address offset

//=============================================
//I/O DECALARATION=============================
//=============================================
input iclk;
input ireset_n;
input [10:0] ix_cnt;
input [9:0] iy_cnt;
input [31:0] iPANELCTRL;
input [31:0] iMOLECTRL;
output [17:0] oADDR;

//=============================================
//WIRE & REG=============================
//=============================================
wire [1:0] override;
wire [17:0] MOLE_ADDR;
wire [17:0] PANEL_ADDR;
reg [17:0] BACK_ADDR;
wire [15:0] area;
wire [10:0] x_cnt;
wire [9:0] y_cnt;
reg [15:0] area_counter[0:15];
wire state = iPANELCTRL[3:0];

assign oADDR = (override[1]) ? MOLE_ADDR : (override[0]) ? PANEL_ADDR : BACK_ADDR;
assign x_cnt = ix_cnt;
assign y_cnt = iy_cnt;

mole_ctrl mole_ctrl1(
                        .iCLK(iclk),                    // LCD display clock
                        .iRST_n(ireset_n),              // systen reset
                       .x_cnt(x_cnt),
                  .y_cnt(y_cnt),
                        .moleControl(iMOLECTRL),
                        .moleExist(override[1]),
                        .mole_address(MOLE_ADDR),
                        );

panel_ctrl panel_ctrl1(
                      .clk_ltm(iclk),
                      .reset_n(ireset_n), //top level connection
                      .x_cnt(x_cnt), // input from lcd controlelr
                      .y_cnt(y_cnt), // input from lcd controller
                      .iPANELCTRL(iPANELCTRL), // command from nios containing 32 bits
                      .oOVERRIDE(override[0]),//override output to the lcd_controller
                      .oADDR(PANEL_ADDR) //the output address to the lcd controller
                      );

//Detect which area you are in// 0~14 different areas//offset corrected
assign area[0] = ((x_cnt>(Hsync_Blank-2+400)&& //>214
                        (x_cnt<(H_LINE-Hsync_Front_Porch-1-200))&& //< 1015
                        (y_cnt>(Vertical_Back_Porch-1))&&
                        (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                        ))    ? 1'b1 : 1'b0 ; //mole0
assign area[1] = ((x_cnt>(Hsync_Blank-2+200)&& //>215
```

```
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0;//mole1
assign area[2] = ((x_cnt>(Hsync_Blank-2)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0 ;//mole2
assign area[3] = ((x_cnt>(Hsync_Blank-2+400)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-200))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ;    //mole3
assign area[4] = ((x_cnt>(Hsync_Blank-2+200)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ; //mole4
assign area[5] = ((x_cnt>(Hsync_Blank-2)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
                                ))    ? 1'b1 : 1'b0 ;//mole5
assign area[6] = ((x_cnt>(Hsync_Blank-2+400)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-200))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;    //mole6
assign area[7] = ((x_cnt>(Hsync_Blank-2+200)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-400))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0; //mole7
assign area[8] = ((x_cnt>(Hsync_Blank-2)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1-600))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;//mole8
assign area[9] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-400))
                                ))    ? 1'b1 : 1'b0 ;//area 9 Time
assign area[10] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+80))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-320))
                                ))    ? 1'b1 : 1'b0 ;//area 10 life
assign area[11] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+160))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-240))
                                ))    ? 1'b1 : 1'b0 ;//area 11 score
assign area[12] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+240))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-160))
```

```
                                ))    ? 1'b1 : 1'b0 ;//area 12 start/pause
assign area[13] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+320))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch-80))
                                ))    ? 1'b1 : 1'b0 ;//area 13 exit
assign area[14] = ((x_cnt>(Hsync_Blank-2+600)&& //>215
                                (x_cnt<(H_LINE-Hsync_Front_Porch-1))&& //< 1016
                                (y_cnt>(Vertical_Back_Porch-1+400))&&
                                (y_cnt<(V_LINE - Vertical_Front_Porch))
                                ))    ? 1'b1 : 1'b0 ;//area 14 label


always@(posedge iclk or negedge ireset_n)
begin
      if(!ireset_n)
            BACK_ADDR <= 18'bZ;
      else
         BACK_ADDR <=                area[0] ? area_counter[0] :
                                             area[1] ? area_counter[1] :
                                             area[2] ? area_counter[2] :
                                             area[3] ? area_counter[3] :
                                             area[4] ? area_counter[4] :
                                             area[5] ? area_counter[5] :
                                             area[6] ? area_counter[6] :
                                             area[7] ? area_counter[7] :
                                             area[8] ? area_counter[8] :
                                             area[9] ? (area_counter[9] + offset):
                                             area[10] ? (area_counter[10] + offset):
                                             area[11] ? (area_counter[11] + offset):
                                             area[12] ? (area_counter[12] + offset):
                                             area[13] ? (area_counter[13] + offset):
                                             area[14] ? (area_counter[14] + offset): 18'bZ;
end

always@(posedge iclk or negedge ireset_n)
      begin
      if (!ireset_n)
            begin
            area_counter[0] <= 0;
            area_counter[1] <= 0;
            area_counter[2] <= 0;
            area_counter[3] <= 0;
            area_counter[4] <= 0;
            area_counter[5] <= 0;
            area_counter[6] <= 0;
            area_counter[7] <= 0;
            area_counter[8] <= 0;
            area_counter[9] <= 0;
            area_counter[10] <= 0;
            area_counter[11] <= 0;
            area_counter[12] <= 0;
            area_counter[13] <= 0;
            area_counter[14] <= 0;
            end
      else
            case(area)
                  16'b0000000000000001:
```

```
begin
if(area_counter[0] == 15'd31999)
	area_counter[0] <= 0;
else
	area_counter[0] <= area_counter[0] +1;
end//area 0

16'b0000000000000010:
begin
if(area_counter[1] == 15'd31999)
	area_counter[1] <= 0;
else
	area_counter[1] <= area_counter[1] +1;
end//area 1

16'b0000000000000100:
begin
if(area_counter[2] == 15'd31999)
	area_counter[2] <= 0;
else
	area_counter[2] <= area_counter[2] +1;
end//area 2

16'b0000000000001000:
begin
if(area_counter[3] == 15'd31999)
	area_counter[3] <= 0;
else
	area_counter[3] <= area_counter[3] +1;
end//area 3

16'b0000000000010000:
begin
if(area_counter[4] == 15'd31999)
	area_counter[4] <= 0;
else
	area_counter[4] <= area_counter[4] +1;
end//area 4

16'b0000000000100000:
begin
if(area_counter[5] == 15'd31999)
	area_counter[5] <= 0;
else
	area_counter[5] <= area_counter[5] +1;
end//area 5

16'b0000000001000000:
begin
if(area_counter[6] == 15'd31999)
	area_counter[6] <= 0;
else
	area_counter[6] <= area_counter[6] +1;
end//area 6

16'b0000000010000000:
begin
if(area_counter[7] == 15'd31999)
```

```
        area_counter[7] <= 0;
else
        area_counter[7] <= area_counter[7] +1;
end//area 7

16'b0000000100000000:
begin
if(area_counter[8] == 15'd31999)
        area_counter[8] <= 0;
else
        area_counter[8] <= area_counter[8] +1;
end//area 8

16'b0000001000000000:
begin
if(area_counter[9] == 15'd15999)
        area_counter[9] <= 0;
else
        area_counter[9] <= area_counter[9] +1;
end//area 9

16'b0000010000000000:
        begin
if(area_counter[10] == 15'd15999)
        area_counter[10] <= 0;
else
        area_counter[10] <= area_counter[10] +1;
end//area 10

16'b0000100000000000:
begin
if(area_counter[11] == 15'd15999)
        area_counter[11] <= 0;
else
        area_counter[11] <= area_counter[11] +1;
end//area 11

16'b0001000000000000:
begin
if(area_counter[12] == 15'd15999)
        area_counter[12] <= 0;
else
        area_counter[12] <= area_counter[12] +1;
end//area 12

16'b0010000000000000:
begin
if(area_counter[13] == 15'd15999)
        area_counter[13] <= 0;
else
        area_counter[13] <= area_counter[13] +1;
end//area 13

16'b0100000000000000:
begin
if(area_counter[14] == 15'd15999)
        area_counter[14] <= 0;
else
```

```
                    area_counter[14] <= area_counter[14] +1;
                end//area 14

                //default: begin red_1 <= 8'd255;green_1 <= 8'd255;blue_1 <= 8'd255;     end
            endcase
        end
endmodule
```

quit.v

```
module quit(
                iclk,
                ireset_n,
                ix_cnt,
                iy_cnt,
                oOVERRIDE,
                oADDR);

//================================================
//I/O DECALARATION================================
//================================================
input iclk;
input ireset_n;
input [10:0] ix_cnt;
input [9:0] iy_cnt;
output oOVERRIDE;
output [17:0] oADDR;

//================================================
//WIRE & REG======================================
//================================================
wire [10:0]iletter_pos_x;
wire [9:0] iletter_pos_y;
wire iletter;
wire [3:0]override;
wire [17:0] address [0:3];

//================================================
//PARAMETER=======================================
//================================================
parameter A = 5'd1;
parameter B = 5'd2;
parameter C = 5'd3;
parameter D = 5'd4;
parameter E = 5'd5;
parameter F = 5'd6;
parameter G = 5'd7;
parameter H = 5'd8;
parameter I = 5'd9;
parameter J = 5'd10;
parameter K = 5'd11;
parameter L = 5'd12;
parameter M = 5'd13;
parameter N = 5'd14;
parameter O = 5'd15;
parameter P = 5'd16;
```

```
parameter Q = 5'd17;
parameter R = 5'd18;
parameter S = 5'd19;
parameter T = 5'd20;
parameter U = 5'd21;
parameter V = 5'd22;
parameter W = 5'd23;
parameter X = 5'd24;
parameter Y = 5'd25;
parameter Z = 5'd26;
parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
//=============================================
//Structured logci==============================
//=============================================

letter letter0(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+636),
                        .iletter_pos_y(Vertical_Back_Porch-1+424),
                        .iletter(T),
                        .oOVERRIDE(override[0]),
                        .oADDR(address[0]));

letter letter1(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+668),
                        .iletter_pos_y(Vertical_Back_Porch-1+424),
                        .iletter(I),
                        .oOVERRIDE(override[1]),
                        .oADDR(address[1]));

letter letter2(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+700),
                        .iletter_pos_y(Vertical_Back_Porch-1+424),
                        .iletter(U),
                        .oOVERRIDE(override[2]),
                        .oADDR(address[2]));

letter letter3(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
```

```
                    .iletter_pos_x(Hsync_Blank-2+732),
                    .iletter_pos_y(Vertical_Back_Porch-1+424),
                    .iletter(Q),
                    .oOVERRIDE(override[3]),
                    .oADDR(address[3]));

assign oOVERRIDE = (override[0] || override[1] || override [2] || override[3]);
assign oADDR = override[0] ? address[0] :
                            override[1] ? address[1] :
                            override[2] ? address[2] :
                            override[3] ? address[3] : 18'bZ;
endmodule
```

score.v

```
module Score(
            clk_ltm,
            reset_n,
            x_cnt,
            y_cnt,
            iScore0,
            iScore1,
            iScore2,
            oOVERRIDE,
            oADDR
            );

input clk_ltm;
input reset_n;
input [10:0] x_cnt;
input [9:0] y_cnt;
input [4:0] iScore0;
input [4:0] iScore1;
input [4:0] iScore2;
output oOVERRIDE;
output [17:0] oADDR;
//PARAMETER
parameter BASE_ADDR = 89025;
parameter INIT = 2'b00;
parameter START = 2'b01;
parameter PAUSE = 2'b10;
parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
parameter coin_base = 18'd96239;
//WIRE & REG

wire [3:0] OVERRIDE;
wire [4:0] score[0:2];
wire [17:0] address;
assign score[0] = iScore0;
assign score[1] = iScore1;
assign score[2] = iScore2;
```

```
symbol coin(
                        .iclk(clk_ltm),
                        .ireset_n(reset_n),
                        .ix_cnt(x_cnt),
                        .iy_cnt(y_cnt),
                        .isymbol_pos_x(Hsync_Blank-1+730),
                        .isymbol_pos_y(Vertical_Back_Porch+170),
                        .iBASE_ADDR(coin_base),
                        .oOVERRIDE(OVERRIDE[3]),
                        .oADDR(address)
);

number score0(
                        .iclk(clk_ltm),
                        .iReset_n(reset_n),
                        .ixcount(x_cnt),
                        .iycount(y_cnt),
                        .iled_pos_x(Hsync_Blank-1+600),
                        .iled_pos_y(Vertical_Back_Porch+160),
                        .inumber(score[0]),
                        .oOVERRIDE(OVERRIDE[0])
);

number score1(
                        .iclk(clk_ltm),
                        .iReset_n(reset_n),
                        .ixcount(x_cnt),
                        .iycount(y_cnt),
                        .iled_pos_x(Hsync_Blank-1+640),
                        .iled_pos_y(Vertical_Back_Porch+160),
                        .inumber(score[1]),
                        .oOVERRIDE(OVERRIDE[1])
);

number score2(
                        .iclk(clk_ltm),
                        .iReset_n(reset_n),
                        .ixcount(x_cnt),
                        .iycount(y_cnt),
                        .iled_pos_x(Hsync_Blank-1+680),
                        .iled_pos_y(Vertical_Back_Porch+160),
                        .inumber(score[2]),
                        .oOVERRIDE(OVERRIDE[2])
);
////OUTPUT
assign oOVERRIDE = OVERRIDE[0] || OVERRIDE [1] || OVERRIDE[2] || OVERRIDE[3];
assign oADDR = (OVERRIDE[0] || OVERRIDE [1] || OVERRIDE[2]) ? BASE_ADDR :
                                OVERRIDE[3] ? address : 18'bZ;

endmodule
```

reset_delay.v

```
module    Reset_Delay(iCLK,iRST,oRST_0,oRST_1,oRST_2);

input             iCLK;
```

```
input           iRST;
output reg      oRST_0;
output reg      oRST_1;
output reg      oRST_2;

reg  [21:0]     Cont;

always@(posedge iCLK or negedge iRST)
begin
    if(!iRST)
    begin
        Cont<=  0;
        oRST_0  <=  0;
        oRST_1  <=  0;
        oRST_2  <=  0;
    end
    else
    begin
        if(Cont!=22'h3FFFFF)
        Cont<=  Cont+1;
        if(Cont>=22'h1FFFFF)
        oRST_0  <=  1;
        if(Cont>=22'h2FFFFF)
        oRST_1  <=  1;
        if(Cont>=22'h3FFFFF)
        oRST_2  <=  1;
    end
end
```

```
endmodule
```

SEG7_LUT_8.v

```
//>>
module SEG7_LUT_8 (
	output	[6:0]oSEG0,
	output	[6:0]oSEG1,
	output	[6:0]oSEG2,
	output	[6:0]oSEG3,
	output	[6:0]oSEG4,
	output	[6:0]oSEG5,
	output	[6:0]oSEG6,
	output	[6:0]oSEG7,
	input	[31:0]iDIG,
	input	[7:0]ON_OFF
  );
//7-Seg 0//
	SEG7_LUT u0(
		.iDIG	(iDIG[3:0]),
		.oSEG	(oSEG0),
		.ON_OFF(ON_OFF[0])
	  );
//7-Seg 1//
	SEG7_LUT u1(
		.iDIG	(iDIG[7:4]),
		.oSEG	(oSEG1),
		.ON_OFF(ON_OFF[1])
	  );
//7-Seg 2//
```

```
	SEG7_LUT u2(
		.iDIG	(iDIG[11:8]),
		.oSEG	(oSEG2),
		.ON_OFF(ON_OFF[2])
	 );
//7-Seg 3//
	SEG7_LUT u3(
		.iDIG	(iDIG[15:12]),
		.oSEG	(oSEG3),
		.ON_OFF(ON_OFF[3])
	 );
//7-Seg 4//
	SEG7_LUT u4(
		.iDIG	(iDIG[19:16]),
		.oSEG	(oSEG4),
		.ON_OFF(ON_OFF[4])
	 );
//7-Seg 5//
	SEG7_LUT u5(
		.iDIG	(iDIG[23:20]),
		.oSEG	(oSEG5),
		.ON_OFF(ON_OFF[5])
	 );
//7-Seg 6//
	SEG7_LUT u6(
		.iDIG	(iDIG[27:24]),
		.oSEG	(oSEG6),
		.ON_OFF(ON_OFF[6])
	 );
```

```
//7-Seg 7//
	SEG7_LUT u7(
		.iDIG	(iDIG[31:28]),
		.oSEG	(oSEG7),
		.ON_OFF(ON_OFF[7])
	 );
endmodule
```

SEG_LUT.v

```
module SEG7_LUT	(
	input		[3:0]iDIG,
	output	reg [6:0]oSEG,
	input			ON_OFF
);

	always @({iDIG,ON_OFF})begin
		if (!ON_OFF)
			oSEG=7'b1111111;
		else begin
		case(iDIG)
		4'h1: oSEG = 7'b1111001;	// ---t----
		4'h2: oSEG = 7'b0100100;	// |	|
		4'h3: oSEG = 7'b0110000;	// lt	rt
		4'h4: oSEG = 7'b0011001;	// |	|
		4'h5: oSEG = 7'b0010010;	// ---m----
		4'h6: oSEG = 7'b0000010;	// |	|
		4'h7: oSEG = 7'b1111000;	// lb	rb
		4'h8: oSEG = 7'b0000000;	// |	|
		4'h9: oSEG = 7'b0011000;	// ---b----
```

```
            4'ha: oSEG = 7'b0001000;

            4'hb: oSEG = 7'b0000011;

            4'hc: oSEG = 7'b1000110;

            4'hd: oSEG = 7'b0100001;

            4'he: oSEG = 7'b0000110;

            4'hf: oSEG = 7'b0001110;

            4'h0: oSEG = 7'b1000000;

            endcase

            end

      end


endmodule
```

single_clk_irq.v

```
module single_clk_irq(
                        clk,
                        reset_n,
                        idisplay_mode,
                        irq
);

input clk;
input reset_n;
input idisplay_mode;
output irq;

reg irq;
reg [4:0]flag;

always@(posedge clk or negedge reset_n)
begin
        if (!reset_n)
            begin
                irq <= 0;
                flag <= 5'b0;
            end
        else  if(idisplay_mode == 0)
            begin
                irq <= 0;
                flag <= 5'b0;
            end
        else if(idisplay_mode == 1 && flag < 16)
            begin
```

```
                        irq <= 1;
                        flag <= flag +1;
                end
        else
                        irq <= 0;
end

endmodule
```

sram_mux.v

```
//library ieee;
//use ieee.std_logic_1164.all;
module sram_mux(
//              chipselect,
                write,
                read,
                address,
                writedata,
                readdata,
//              byteenable,
                SRAM_DQ,
                SRAM_ADDR,
                SRAM_UB_N,
                SRAM_LB_N,
                SRAM_WE_N,
                SRAM_CE_N,
                SRAM_OE_N,
                VGA_DQ,
                VGA_ADDR,
//              VGA_UB_N,
//              VGA_LB_N,
                VGA_WE_N,
//              VGA_CE_N,
                VGA_OE_N
                );

//input chipselect;
input write;
input read;
input [17:0] address;
input [15:0] writedata;
output [15:0] readdata;
//input [1:0] byteenable;
inout [15:0] SRAM_DQ;
output [17:0] SRAM_ADDR;
output SRAM_UB_N;
output SRAM_LB_N;
output SRAM_WE_N;
output SRAM_CE_N;
output SRAM_OE_N;
output [15:0] VGA_DQ;
input [17:0] VGA_ADDR;
//input VGA_UB_N;
//input VGA_LB_N;
input VGA_WE_N;
```

```
//input VGA_CE_N;
input VGA_OE_N;


assign	SRAM_DQ = (write || read) ? writedata : 16'bZ;
assign	readdata = SRAM_DQ;
assign	VGA_DQ = SRAM_DQ;
assign	SRAM_ADDR = (write || read) ? address : VGA_ADDR;
assign	SRAM_UB_N = 0;
assign	SRAM_LB_N = 0;
assign	SRAM_WE_N = (write || read) ? !write : !VGA_WE_N;
assign	SRAM_CE_N = 0;
assign	SRAM_OE_N = (write || read) ? !read : !VGA_OE_N;

endmodule
```

sram.v

```
// sram.v

// Generated using ACDS version 12.1 177 at 2013.03.31.17:14:42

`timescale 1 ps / 1 ps
module sram (
		input	wire		chipselect, // avalon_slave_0.chipselect
		input	wire		write,		//			.write
		input	wire		read,		//			.read
		input	wire [17:0] address,	//			.address
		output wire [15:0] readdata,	//			.readdata
		input	wire [15:0] writedata,	//			.writedata
		input	wire [1:0]	byteenable, //			.byteenable
		inout	wire [15:0] SRAM_DQ,	//	conduit_end.export
		output wire [17:0] SRAM_ADDR,	//			.export
		output wire		SRAM_UB_N,	//			.export
		output wire		SRAM_LB_N,	//			.export
		output wire		SRAM_WE_N,	//			.export
		output wire		SRAM_CE_N,	//			.export
		output wire		SRAM_OE_N	//			.export
	);

	de2_sram_controller sram_inst (
		.chipselect (chipselect), // avalon_slave_0.chipselect
		.write		(write),	//			.write
		.read		(read),		//			.read
		.address	(address),	//			.address
		.readdata	(readdata),	//			.readdata
		.writedata	(writedata),	//			.writedata
		.byteenable (byteenable), //			.byteenable
		.SRAM_DQ	(SRAM_DQ),	//	conduit_end.export
		.SRAM_ADDR	(SRAM_ADDR),	//			.export
		.SRAM_UB_N	(SRAM_UB_N),	//			.export
		.SRAM_LB_N	(SRAM_LB_N),	//			.export
		.SRAM_WE_N	(SRAM_WE_N),	//			.export
		.SRAM_CE_N	(SRAM_CE_N),	//			.export
		.SRAM_OE_N	(SRAM_OE_N)	//			.export
	);
```

```
endmodule
```

start.v

```
module start(
                    iclk,
                    ireset_n,
                    ix_cnt,
                    iy_cnt,
                    oOVERRIDE,
                    oADDR
                        );

//================================================
//I/O DECALARATION==============================
//================================================
input iclk;
input ireset_n;
input [10:0] ix_cnt;
input [9:0] iy_cnt;
output oOVERRIDE;
output [17:0] oADDR;

//================================================
//WIRE & REG====================================
//================================================
wire [10:0]iletter_pos_x;
wire [9:0] iletter_pos_y;
wire iletter;
wire [4:0]override;
wire [17:0] address [0:4];

//================================================
//PARAMETER=====================================
//================================================
parameter A = 5'd1;
parameter B = 5'd2;
parameter C = 5'd3;
parameter D = 5'd4;
parameter E = 5'd5;
parameter F = 5'd6;
parameter G = 5'd7;
parameter H = 5'd8;
parameter I = 5'd9;
parameter J = 5'd10;
parameter K = 5'd11;
parameter L = 5'd12;
parameter M = 5'd13;
parameter N = 5'd14;
parameter O = 5'd15;
parameter P = 5'd16;
parameter Q = 5'd17;
parameter R = 5'd18;
parameter S = 5'd19;
parameter T = 5'd20;
```

```
parameter U = 5'd21;
parameter V = 5'd22;
parameter W = 5'd23;
parameter X = 5'd24;
parameter Y = 5'd25;
parameter Z = 5'd26;
parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
//=============================================
//Structured logci==============================
//=============================================

letter letter0(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+620),
                        .iletter_pos_y(Vertical_Back_Porch-1+264),
                        .iletter(T),
                        .oOVERRIDE(override[0]),
                        .oADDR(address[0]));

letter letter1(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+652),
                        .iletter_pos_y(Vertical_Back_Porch-1+264),
                        .iletter(R),
                        .oOVERRIDE(override[1]),
                        .oADDR(address[1]));

letter letter2(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+684),
                        .iletter_pos_y(Vertical_Back_Porch-1+264),
                        .iletter(A),
                        .oOVERRIDE(override[2]),
                        .oADDR(address[2]));

letter letter3(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+716),
                        .iletter_pos_y(Vertical_Back_Porch-1+264),
                        .iletter(T),
                        .oOVERRIDE(override[3]),
```

```
                        .oADDR(address[3]));

letter letter4(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(Hsync_Blank-2+748),
                        .iletter_pos_y(Vertical_Back_Porch-1+264),
                        .iletter(S),
                        .oOVERRIDE(override[4]),
                        .oADDR(address[4]));

assign oOVERRIDE = (override[0] || override[1] || override [2] || override[3] || override[4]);
assign oADDR = override[0] ? address[0] :
                                override[1] ? address[1] :
                                override[2] ? address[2] :
                                override[3] ? address[3] :
                                override[4] ? address[4] : 18'bZ;
endmodule
```

symbol.v

```
module symbol(
                        iclk,
                        ireset_n,
                        ix_cnt,
                        iy_cnt,
                        isymbol_pos_x,
                        isymbol_pos_y,
                        iBASE_ADDR,
                        oOVERRIDE,
                        oADDR
                        );

//=============================================
//I/O DECALARATION=============================
//=============================================
input iclk;
input ireset_n;
input [10:0] ix_cnt;
input [9:0] iy_cnt;
input [10:0]isymbol_pos_x;
input [9:0]isymbol_pos_y;
input [17:0] iBASE_ADDR;
output oOVERRIDE;
output [17:0] oADDR;


//=============================================
//WIRE & REG==================================
//=============================================
reg    oOVERRIDE;
reg [17:0] oADDR;
reg [5:0] counter;
wire display_area;
```

```
reg white[0:3599];

//================================================
//Structured logic================================
//================================================
assign display_area = ((ix_cnt > isymbol_pos_x-1) && (ix_cnt < isymbol_pos_x + 60)
                                        && (iy_cnt > isymbol_pos_y -1) && (iy_cnt <
isymbol_pos_y + 60)) ? 1'b1:1'b0;


always@(negedge ireset_n)
begin
      if(!ireset_n)
            begin
            case(iBASE_ADDR)
            //clock
            18'd89039: begin
            white[ 0 ]<=0;
            white[ 1 ]<=0;
            white[ 2 ]<=0;
            white[ 3 ]<=0;
            white[ 4 ]<=0;
            white[ 5 ]<=0;
            white[ 6 ]<=0;
            white[ 7 ]<=0;
            white[ 8 ]<=0;
            white[ 9 ]<=0;
            white[ 10 ]<=0;
            white[ 11 ]<=0;
            white[ 12 ]<=0;
            white[ 13 ]<=0;
            white[ 14 ]<=0;
            white[ 15 ]<=1;
            white[ 16 ]<=1;
            white[ 17 ]<=1;
            white[ 18 ]<=1;
            white[ 19 ]<=1;
            white[ 20 ]<=1;
            white[ 21 ]<=1;
            white[ 22 ]<=0;
            white[ 23 ]<=0;
            white[ 24 ]<=0;
            white[ 25 ]<=0;
            white[ 26 ]<=0;
            white[ 27 ]<=0;
            white[ 28 ]<=0;
            white[ 29 ]<=0;
            white[ 30 ]<=0;
            white[ 31 ]<=0;
            white[ 32 ]<=0;
            white[ 33 ]<=0;
            white[ 34 ]<=0;
            white[ 35 ]<=0;
            white[ 36 ]<=0;
            white[ 37 ]<=0;
            white[ 38 ]<=1;
            white[ 39 ]<=1;
            white[ 40 ]<=1;
```

```
white[ 41 ]<=1;
white[ 42 ]<=1;
white[ 43 ]<=1;
white[ 44 ]<=1;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=1;
white[ 74 ]<=1;
white[ 75 ]<=1;
white[ 76 ]<=1;
white[ 77 ]<=1;
white[ 78 ]<=1;
white[ 79 ]<=1;
white[ 80 ]<=1;
white[ 81 ]<=1;
white[ 82 ]<=1;
white[ 83 ]<=1;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=1;
white[ 97 ]<=1;
white[ 98 ]<=1;
```

```
white[ 99 ]<=1;
white[ 100 ]<=1;
white[ 101 ]<=1;
white[ 102 ]<=1;
white[ 103 ]<=1;
white[ 104 ]<=1;
white[ 105 ]<=1;
white[ 106 ]<=1;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=1;
white[ 133 ]<=1;
white[ 134 ]<=1;
white[ 135 ]<=1;
white[ 136 ]<=1;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
white[ 143 ]<=1;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=0;
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=0;
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=1;
```

```
white[ 157 ]<=1;
white[ 158 ]<=1;
white[ 159 ]<=1;
white[ 160 ]<=1;
white[ 161 ]<=1;
white[ 162 ]<=1;
white[ 163 ]<=1;
white[ 164 ]<=1;
white[ 165 ]<=1;
white[ 166 ]<=1;
white[ 167 ]<=1;
white[ 168 ]<=0;
white[ 169 ]<=0;
white[ 170 ]<=0;
white[ 171 ]<=0;
white[ 172 ]<=0;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=0;
white[ 180 ]<=0;
white[ 181 ]<=0;
white[ 182 ]<=0;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=1;
white[ 192 ]<=1;
white[ 193 ]<=1;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=0;
white[ 206 ]<=0;
white[ 207 ]<=0;
white[ 208 ]<=0;
white[ 209 ]<=0;
white[ 210 ]<=0;
white[ 211 ]<=0;
white[ 212 ]<=0;
white[ 213 ]<=0;
white[ 214 ]<=0;
```

```
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=1;
white[ 222 ]<=1;
white[ 223 ]<=1;
white[ 224 ]<=1;
white[ 225 ]<=1;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=0;
white[ 230 ]<=0;
white[ 231 ]<=0;
white[ 232 ]<=0;
white[ 233 ]<=0;
white[ 234 ]<=0;
white[ 235 ]<=0;
white[ 236 ]<=0;
white[ 237 ]<=0;
white[ 238 ]<=0;
white[ 239 ]<=0;
white[ 240 ]<=0;
white[ 241 ]<=0;
white[ 242 ]<=0;
white[ 243 ]<=0;
white[ 244 ]<=0;
white[ 245 ]<=0;
white[ 246 ]<=0;
white[ 247 ]<=0;
white[ 248 ]<=0;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=1;
white[ 255 ]<=1;
white[ 256 ]<=1;
white[ 257 ]<=1;
white[ 258 ]<=1;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=0;
white[ 267 ]<=0;
white[ 268 ]<=0;
white[ 269 ]<=0;
white[ 270 ]<=0;
white[ 271 ]<=0;
white[ 272 ]<=0;
```

```
white[ 273 ]<=0;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=1;
white[ 287 ]<=1;
white[ 288 ]<=1;
white[ 289 ]<=1;
white[ 290 ]<=1;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=0;
white[ 294 ]<=0;
white[ 295 ]<=0;
white[ 296 ]<=0;
white[ 297 ]<=0;
white[ 298 ]<=0;
white[ 299 ]<=0;
white[ 300 ]<=0;
white[ 301 ]<=0;
white[ 302 ]<=0;
white[ 303 ]<=0;
white[ 304 ]<=0;
white[ 305 ]<=0;
white[ 306 ]<=0;
white[ 307 ]<=0;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=1;
white[ 318 ]<=1;
white[ 319 ]<=1;
white[ 320 ]<=1;
white[ 321 ]<=1;
white[ 322 ]<=1;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=0;
white[ 327 ]<=0;
white[ 328 ]<=0;
white[ 329 ]<=0;
white[ 330 ]<=0;
```

```
white[ 331 ]<=0;
white[ 332 ]<=0;
white[ 333 ]<=0;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=1;
white[ 351 ]<=1;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=0;
white[ 358 ]<=0;
white[ 359 ]<=0;
white[ 360 ]<=0;
white[ 361 ]<=0;
white[ 362 ]<=0;
white[ 363 ]<=0;
white[ 364 ]<=0;
white[ 365 ]<=0;
white[ 366 ]<=0;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=0;
white[ 383 ]<=0;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
```

```
white[ 389 ]<=0;
white[ 390 ]<=0;
white[ 391 ]<=0;
white[ 392 ]<=0;
white[ 393 ]<=0;
white[ 394 ]<=0;
white[ 395 ]<=0;
white[ 396 ]<=0;
white[ 397 ]<=0;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=0;
white[ 423 ]<=0;
white[ 424 ]<=0;
white[ 425 ]<=0;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=0;
white[ 440 ]<=0;
white[ 441 ]<=0;
white[ 442 ]<=0;
white[ 443 ]<=0;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=1;
```

```
white[ 447 ]<=1;
white[ 448 ]<=1;
white[ 449 ]<=1;
white[ 450 ]<=1;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=0;
white[ 457 ]<=0;
white[ 458 ]<=0;
white[ 459 ]<=0;
white[ 460 ]<=0;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=0;
white[ 498 ]<=0;
white[ 499 ]<=0;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
```

```
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=1;
white[ 511 ]<=1;
white[ 512 ]<=1;
white[ 513 ]<=1;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=0;
white[ 521 ]<=0;
white[ 522 ]<=0;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=0;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=1;
white[ 546 ]<=1;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=0;
white[ 556 ]<=0;
white[ 557 ]<=0;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
```

```
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=1;
white[ 576 ]<=1;
white[ 577 ]<=1;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=0;
white[ 583 ]<=0;
white[ 584 ]<=0;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=0;
white[ 596 ]<=0;
white[ 597 ]<=0;
white[ 598 ]<=0;
white[ 599 ]<=0;
white[ 600 ]<=0;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=1;
white[ 606 ]<=1;
white[ 607 ]<=1;
white[ 608 ]<=1;
white[ 609 ]<=1;
white[ 610 ]<=1;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=0;
white[ 615 ]<=0;
white[ 616 ]<=0;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
```

```
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=1;
white[ 640 ]<=1;
white[ 641 ]<=1;
white[ 642 ]<=1;
white[ 643 ]<=0;
white[ 644 ]<=0;
white[ 645 ]<=0;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=0;
white[ 656 ]<=0;
white[ 657 ]<=0;
white[ 658 ]<=0;
white[ 659 ]<=0;
white[ 660 ]<=0;
white[ 661 ]<=0;
white[ 662 ]<=0;
white[ 663 ]<=0;
white[ 664 ]<=0;
white[ 665 ]<=0;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=1;
white[ 671 ]<=1;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=0;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
```

```
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=0;
white[ 687 ]<=0;
white[ 688 ]<=0;
white[ 689 ]<=0;
white[ 690 ]<=0;
white[ 691 ]<=0;
white[ 692 ]<=0;
white[ 693 ]<=0;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=1;
white[ 703 ]<=1;
white[ 704 ]<=1;
white[ 705 ]<=0;
white[ 706 ]<=0;
white[ 707 ]<=0;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=0;
white[ 715 ]<=0;
white[ 716 ]<=0;
white[ 717 ]<=0;
white[ 718 ]<=0;
white[ 719 ]<=0;
white[ 720 ]<=0;
white[ 721 ]<=0;
white[ 722 ]<=0;
white[ 723 ]<=0;
white[ 724 ]<=0;
white[ 725 ]<=0;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=1;
white[ 734 ]<=1;
white[ 735 ]<=1;
white[ 736 ]<=1;
```

```
white[ 737 ]<=1;
white[ 738 ]<=1;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=0;
white[ 744 ]<=0;
white[ 745 ]<=0;
white[ 746 ]<=0;
white[ 747 ]<=0;
white[ 748 ]<=0;
white[ 749 ]<=0;
white[ 750 ]<=0;
white[ 751 ]<=0;
white[ 752 ]<=0;
white[ 753 ]<=0;
white[ 754 ]<=0;
white[ 755 ]<=0;
white[ 756 ]<=0;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=0;
white[ 768 ]<=0;
white[ 769 ]<=1;
white[ 770 ]<=1;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=0;
white[ 775 ]<=0;
white[ 776 ]<=0;
white[ 777 ]<=0;
white[ 778 ]<=0;
white[ 779 ]<=0;
white[ 780 ]<=0;
white[ 781 ]<=0;
white[ 782 ]<=0;
white[ 783 ]<=0;
white[ 784 ]<=0;
white[ 785 ]<=0;
white[ 786 ]<=0;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=0;
white[ 791 ]<=0;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
```

```
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=1;
white[ 800 ]<=1;
white[ 801 ]<=0;
white[ 802 ]<=0;
white[ 803 ]<=0;
white[ 804 ]<=0;
white[ 805 ]<=0;
white[ 806 ]<=0;
white[ 807 ]<=0;
white[ 808 ]<=0;
white[ 809 ]<=0;
white[ 810 ]<=0;
white[ 811 ]<=0;
white[ 812 ]<=0;
white[ 813 ]<=0;
white[ 814 ]<=0;
white[ 815 ]<=0;
white[ 816 ]<=0;
white[ 817 ]<=0;
white[ 818 ]<=0;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=1;
white[ 831 ]<=1;
white[ 832 ]<=1;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=0;
white[ 839 ]<=0;
white[ 840 ]<=0;
white[ 841 ]<=0;
white[ 842 ]<=0;
white[ 843 ]<=0;
white[ 844 ]<=0;
white[ 845 ]<=0;
white[ 846 ]<=0;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=0;
white[ 850 ]<=0;
white[ 851 ]<=1;
white[ 852 ]<=1;
```

```
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=0;
white[ 860 ]<=0;
white[ 861 ]<=0;
white[ 862 ]<=0;
white[ 863 ]<=0;
white[ 864 ]<=0;
white[ 865 ]<=0;
white[ 866 ]<=0;
white[ 867 ]<=0;
white[ 868 ]<=0;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=0;
white[ 872 ]<=0;
white[ 873 ]<=0;
white[ 874 ]<=0;
white[ 875 ]<=0;
white[ 876 ]<=0;
white[ 877 ]<=0;
white[ 878 ]<=0;
white[ 879 ]<=0;
white[ 880 ]<=0;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=0;
white[ 905 ]<=0;
white[ 906 ]<=0;
white[ 907 ]<=0;
white[ 908 ]<=1;
white[ 909 ]<=0;
white[ 910 ]<=0;
```

```
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=0;
white[ 918 ]<=0;
white[ 919 ]<=0;
white[ 920 ]<=0;
white[ 921 ]<=0;
white[ 922 ]<=0;
white[ 923 ]<=0;
white[ 924 ]<=0;
white[ 925 ]<=0;
white[ 926 ]<=0;
white[ 927 ]<=0;
white[ 928 ]<=1;
white[ 929 ]<=1;
white[ 930 ]<=1;
white[ 931 ]<=1;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=0;
white[ 936 ]<=0;
white[ 937 ]<=0;
white[ 938 ]<=0;
white[ 939 ]<=0;
white[ 940 ]<=0;
white[ 941 ]<=0;
white[ 942 ]<=0;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=1;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
```

```
white[ 969 ]<=1;
white[ 970 ]<=1;
white[ 971 ]<=1;
white[ 972 ]<=1;
white[ 973 ]<=1;
white[ 974 ]<=1;
white[ 975 ]<=1;
white[ 976 ]<=0;
white[ 977 ]<=0;
white[ 978 ]<=0;
white[ 979 ]<=0;
white[ 980 ]<=0;
white[ 981 ]<=0;
white[ 982 ]<=0;
white[ 983 ]<=0;
white[ 984 ]<=0;
white[ 985 ]<=0;
white[ 986 ]<=0;
white[ 987 ]<=0;
white[ 988 ]<=1;
white[ 989 ]<=1;
white[ 990 ]<=1;
white[ 991 ]<=1;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=0;
white[ 996 ]<=0;
white[ 997 ]<=0;
white[ 998 ]<=0;
white[ 999 ]<=0;
white[ 1000 ]<=0;
white[ 1001 ]<=0;
white[ 1002 ]<=0;
white[ 1003 ]<=0;
white[ 1004 ]<=1;
white[ 1005 ]<=1;
white[ 1006 ]<=1;
white[ 1007 ]<=1;
white[ 1008 ]<=1;
white[ 1009 ]<=1;
white[ 1010 ]<=1;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
white[ 1024 ]<=0;
white[ 1025 ]<=0;
white[ 1026 ]<=0;
```

```
white[ 1027 ]<=0;
white[ 1028 ]<=0;
white[ 1029 ]<=1;
white[ 1030 ]<=1;
white[ 1031 ]<=1;
white[ 1032 ]<=1;
white[ 1033 ]<=1;
white[ 1034 ]<=1;
white[ 1035 ]<=0;
white[ 1036 ]<=0;
white[ 1037 ]<=0;
white[ 1038 ]<=0;
white[ 1039 ]<=0;
white[ 1040 ]<=0;
white[ 1041 ]<=0;
white[ 1042 ]<=0;
white[ 1043 ]<=0;
white[ 1044 ]<=0;
white[ 1045 ]<=0;
white[ 1046 ]<=0;
white[ 1047 ]<=0;
white[ 1048 ]<=1;
white[ 1049 ]<=1;
white[ 1050 ]<=1;
white[ 1051 ]<=1;
white[ 1052 ]<=0;
white[ 1053 ]<=0;
white[ 1054 ]<=0;
white[ 1055 ]<=0;
white[ 1056 ]<=0;
white[ 1057 ]<=0;
white[ 1058 ]<=0;
white[ 1059 ]<=0;
white[ 1060 ]<=0;
white[ 1061 ]<=0;
white[ 1062 ]<=0;
white[ 1063 ]<=0;
white[ 1064 ]<=0;
white[ 1065 ]<=1;
white[ 1066 ]<=1;
white[ 1067 ]<=1;
white[ 1068 ]<=1;
white[ 1069 ]<=1;
white[ 1070 ]<=1;
white[ 1071 ]<=0;
white[ 1072 ]<=0;
white[ 1073 ]<=0;
white[ 1074 ]<=0;
white[ 1075 ]<=0;
white[ 1076 ]<=0;
white[ 1077 ]<=0;
white[ 1078 ]<=0;
white[ 1079 ]<=0;
white[ 1080 ]<=0;
white[ 1081 ]<=0;
white[ 1082 ]<=0;
white[ 1083 ]<=0;
white[ 1084 ]<=0;
```

```
white[ 1085 ]<=0;
white[ 1086 ]<=0;
white[ 1087 ]<=0;
white[ 1088 ]<=1;
white[ 1089 ]<=1;
white[ 1090 ]<=1;
white[ 1091 ]<=1;
white[ 1092 ]<=1;
white[ 1093 ]<=1;
white[ 1094 ]<=0;
white[ 1095 ]<=0;
white[ 1096 ]<=0;
white[ 1097 ]<=0;
white[ 1098 ]<=0;
white[ 1099 ]<=0;
white[ 1100 ]<=0;
white[ 1101 ]<=0;
white[ 1102 ]<=0;
white[ 1103 ]<=0;
white[ 1104 ]<=0;
white[ 1105 ]<=0;
white[ 1106 ]<=0;
white[ 1107 ]<=0;
white[ 1108 ]<=1;
white[ 1109 ]<=1;
white[ 1110 ]<=1;
white[ 1111 ]<=1;
white[ 1112 ]<=0;
white[ 1113 ]<=0;
white[ 1114 ]<=0;
white[ 1115 ]<=0;
white[ 1116 ]<=0;
white[ 1117 ]<=0;
white[ 1118 ]<=0;
white[ 1119 ]<=0;
white[ 1120 ]<=0;
white[ 1121 ]<=0;
white[ 1122 ]<=0;
white[ 1123 ]<=0;
white[ 1124 ]<=0;
white[ 1125 ]<=0;
white[ 1126 ]<=1;
white[ 1127 ]<=1;
white[ 1128 ]<=1;
white[ 1129 ]<=1;
white[ 1130 ]<=1;
white[ 1131 ]<=1;
white[ 1132 ]<=0;
white[ 1133 ]<=0;
white[ 1134 ]<=0;
white[ 1135 ]<=0;
white[ 1136 ]<=0;
white[ 1137 ]<=0;
white[ 1138 ]<=0;
white[ 1139 ]<=0;
white[ 1140 ]<=0;
white[ 1141 ]<=0;
white[ 1142 ]<=0;
```

```
white[ 1143 ]<=0;
white[ 1144 ]<=0;
white[ 1145 ]<=0;
white[ 1146 ]<=0;
white[ 1147 ]<=1;
white[ 1148 ]<=1;
white[ 1149 ]<=1;
white[ 1150 ]<=1;
white[ 1151 ]<=1;
white[ 1152 ]<=1;
white[ 1153 ]<=0;
white[ 1154 ]<=0;
white[ 1155 ]<=0;
white[ 1156 ]<=0;
white[ 1157 ]<=0;
white[ 1158 ]<=0;
white[ 1159 ]<=0;
white[ 1160 ]<=0;
white[ 1161 ]<=0;
white[ 1162 ]<=0;
white[ 1163 ]<=0;
white[ 1164 ]<=0;
white[ 1165 ]<=0;
white[ 1166 ]<=0;
white[ 1167 ]<=0;
white[ 1168 ]<=1;
white[ 1169 ]<=1;
white[ 1170 ]<=1;
white[ 1171 ]<=1;
white[ 1172 ]<=0;
white[ 1173 ]<=0;
white[ 1174 ]<=0;
white[ 1175 ]<=0;
white[ 1176 ]<=0;
white[ 1177 ]<=0;
white[ 1178 ]<=0;
white[ 1179 ]<=0;
white[ 1180 ]<=0;
white[ 1181 ]<=0;
white[ 1182 ]<=0;
white[ 1183 ]<=0;
white[ 1184 ]<=0;
white[ 1185 ]<=0;
white[ 1186 ]<=0;
white[ 1187 ]<=1;
white[ 1188 ]<=1;
white[ 1189 ]<=1;
white[ 1190 ]<=1;
white[ 1191 ]<=1;
white[ 1192 ]<=1;
white[ 1193 ]<=0;
white[ 1194 ]<=0;
white[ 1195 ]<=0;
white[ 1196 ]<=0;
white[ 1197 ]<=0;
white[ 1198 ]<=0;
white[ 1199 ]<=0;
white[ 1200 ]<=0;
```

```
white[ 1201 ]<=0;
white[ 1202 ]<=0;
white[ 1203 ]<=0;
white[ 1204 ]<=0;
white[ 1205 ]<=0;
white[ 1206 ]<=0;
white[ 1207 ]<=1;
white[ 1208 ]<=1;
white[ 1209 ]<=1;
white[ 1210 ]<=1;
white[ 1211 ]<=1;
white[ 1212 ]<=0;
white[ 1213 ]<=0;
white[ 1214 ]<=0;
white[ 1215 ]<=0;
white[ 1216 ]<=0;
white[ 1217 ]<=0;
white[ 1218 ]<=0;
white[ 1219 ]<=0;
white[ 1220 ]<=0;
white[ 1221 ]<=0;
white[ 1222 ]<=0;
white[ 1223 ]<=0;
white[ 1224 ]<=0;
white[ 1225 ]<=0;
white[ 1226 ]<=0;
white[ 1227 ]<=0;
white[ 1228 ]<=1;
white[ 1229 ]<=1;
white[ 1230 ]<=1;
white[ 1231 ]<=1;
white[ 1232 ]<=0;
white[ 1233 ]<=0;
white[ 1234 ]<=0;
white[ 1235 ]<=0;
white[ 1236 ]<=0;
white[ 1237 ]<=0;
white[ 1238 ]<=0;
white[ 1239 ]<=0;
white[ 1240 ]<=0;
white[ 1241 ]<=0;
white[ 1242 ]<=0;
white[ 1243 ]<=0;
white[ 1244 ]<=0;
white[ 1245 ]<=0;
white[ 1246 ]<=0;
white[ 1247 ]<=0;
white[ 1248 ]<=1;
white[ 1249 ]<=1;
white[ 1250 ]<=1;
white[ 1251 ]<=1;
white[ 1252 ]<=1;
white[ 1253 ]<=0;
white[ 1254 ]<=0;
white[ 1255 ]<=0;
white[ 1256 ]<=0;
white[ 1257 ]<=0;
white[ 1258 ]<=0;
```

```
white[ 1259 ]<=0;
white[ 1260 ]<=0;
white[ 1261 ]<=0;
white[ 1262 ]<=0;
white[ 1263 ]<=0;
white[ 1264 ]<=0;
white[ 1265 ]<=0;
white[ 1266 ]<=1;
white[ 1267 ]<=1;
white[ 1268 ]<=1;
white[ 1269 ]<=1;
white[ 1270 ]<=1;
white[ 1271 ]<=0;
white[ 1272 ]<=0;
white[ 1273 ]<=0;
white[ 1274 ]<=0;
white[ 1275 ]<=0;
white[ 1276 ]<=0;
white[ 1277 ]<=0;
white[ 1278 ]<=0;
white[ 1279 ]<=0;
white[ 1280 ]<=0;
white[ 1281 ]<=0;
white[ 1282 ]<=0;
white[ 1283 ]<=0;
white[ 1284 ]<=0;
white[ 1285 ]<=0;
white[ 1286 ]<=0;
white[ 1287 ]<=0;
white[ 1288 ]<=1;
white[ 1289 ]<=1;
white[ 1290 ]<=1;
white[ 1291 ]<=1;
white[ 1292 ]<=0;
white[ 1293 ]<=0;
white[ 1294 ]<=0;
white[ 1295 ]<=0;
white[ 1296 ]<=0;
white[ 1297 ]<=0;
white[ 1298 ]<=0;
white[ 1299 ]<=0;
white[ 1300 ]<=0;
white[ 1301 ]<=0;
white[ 1302 ]<=0;
white[ 1303 ]<=0;
white[ 1304 ]<=0;
white[ 1305 ]<=0;
white[ 1306 ]<=0;
white[ 1307 ]<=0;
white[ 1308 ]<=0;
white[ 1309 ]<=1;
white[ 1310 ]<=1;
white[ 1311 ]<=1;
white[ 1312 ]<=1;
white[ 1313 ]<=1;
white[ 1314 ]<=0;
white[ 1315 ]<=0;
white[ 1316 ]<=0;
```

```
white[ 1317 ]<=0;
white[ 1318 ]<=0;
white[ 1319 ]<=0;
white[ 1320 ]<=0;
white[ 1321 ]<=0;
white[ 1322 ]<=0;
white[ 1323 ]<=0;
white[ 1324 ]<=0;
white[ 1325 ]<=0;
white[ 1326 ]<=1;
white[ 1327 ]<=1;
white[ 1328 ]<=1;
white[ 1329 ]<=1;
white[ 1330 ]<=1;
white[ 1331 ]<=0;
white[ 1332 ]<=0;
white[ 1333 ]<=0;
white[ 1334 ]<=0;
white[ 1335 ]<=0;
white[ 1336 ]<=0;
white[ 1337 ]<=0;
white[ 1338 ]<=0;
white[ 1339 ]<=0;
white[ 1340 ]<=0;
white[ 1341 ]<=0;
white[ 1342 ]<=0;
white[ 1343 ]<=0;
white[ 1344 ]<=0;
white[ 1345 ]<=0;
white[ 1346 ]<=0;
white[ 1347 ]<=0;
white[ 1348 ]<=1;
white[ 1349 ]<=1;
white[ 1350 ]<=1;
white[ 1351 ]<=1;
white[ 1352 ]<=0;
white[ 1353 ]<=0;
white[ 1354 ]<=0;
white[ 1355 ]<=0;
white[ 1356 ]<=0;
white[ 1357 ]<=0;
white[ 1358 ]<=0;
white[ 1359 ]<=0;
white[ 1360 ]<=0;
white[ 1361 ]<=0;
white[ 1362 ]<=0;
white[ 1363 ]<=0;
white[ 1364 ]<=0;
white[ 1365 ]<=0;
white[ 1366 ]<=0;
white[ 1367 ]<=0;
white[ 1368 ]<=0;
white[ 1369 ]<=1;
white[ 1370 ]<=1;
white[ 1371 ]<=1;
white[ 1372 ]<=1;
white[ 1373 ]<=1;
white[ 1374 ]<=0;
```

```
white[ 1375 ]<=0;
white[ 1376 ]<=0;
white[ 1377 ]<=0;
white[ 1378 ]<=0;
white[ 1379 ]<=0;
white[ 1380 ]<=0;
white[ 1381 ]<=0;
white[ 1382 ]<=0;
white[ 1383 ]<=0;
white[ 1384 ]<=0;
white[ 1385 ]<=1;
white[ 1386 ]<=1;
white[ 1387 ]<=1;
white[ 1388 ]<=1;
white[ 1389 ]<=1;
white[ 1390 ]<=1;
white[ 1391 ]<=0;
white[ 1392 ]<=0;
white[ 1393 ]<=0;
white[ 1394 ]<=0;
white[ 1395 ]<=0;
white[ 1396 ]<=0;
white[ 1397 ]<=0;
white[ 1398 ]<=0;
white[ 1399 ]<=0;
white[ 1400 ]<=0;
white[ 1401 ]<=0;
white[ 1402 ]<=0;
white[ 1403 ]<=0;
white[ 1404 ]<=0;
white[ 1405 ]<=0;
white[ 1406 ]<=0;
white[ 1407 ]<=0;
white[ 1408 ]<=1;
white[ 1409 ]<=1;
white[ 1410 ]<=1;
white[ 1411 ]<=1;
white[ 1412 ]<=0;
white[ 1413 ]<=0;
white[ 1414 ]<=0;
white[ 1415 ]<=0;
white[ 1416 ]<=0;
white[ 1417 ]<=0;
white[ 1418 ]<=0;
white[ 1419 ]<=0;
white[ 1420 ]<=0;
white[ 1421 ]<=0;
white[ 1422 ]<=0;
white[ 1423 ]<=0;
white[ 1424 ]<=0;
white[ 1425 ]<=0;
white[ 1426 ]<=0;
white[ 1427 ]<=0;
white[ 1428 ]<=0;
white[ 1429 ]<=1;
white[ 1430 ]<=1;
white[ 1431 ]<=1;
white[ 1432 ]<=1;
```

white[1433]<=1;
white[1434]<=1;
white[1435]<=0;
white[1436]<=0;
white[1437]<=0;
white[1438]<=0;
white[1439]<=0;
white[1440]<=0;
white[1441]<=0;
white[1442]<=0;
white[1443]<=0;
white[1444]<=0;
white[1445]<=1;
white[1446]<=1;
white[1447]<=1;
white[1448]<=1;
white[1449]<=1;
white[1450]<=0;
white[1451]<=0;
white[1452]<=0;
white[1453]<=0;
white[1454]<=0;
white[1455]<=0;
white[1456]<=0;
white[1457]<=0;
white[1458]<=0;
white[1459]<=0;
white[1460]<=0;
white[1461]<=0;
white[1462]<=0;
white[1463]<=0;
white[1464]<=0;
white[1465]<=0;
white[1466]<=0;
white[1467]<=0;
white[1468]<=1;
white[1469]<=1;
white[1470]<=1;
white[1471]<=1;
white[1472]<=0;
white[1473]<=0;
white[1474]<=0;
white[1475]<=0;
white[1476]<=0;
white[1477]<=0;
white[1478]<=0;
white[1479]<=0;
white[1480]<=0;
white[1481]<=0;
white[1482]<=0;
white[1483]<=0;
white[1484]<=0;
white[1485]<=0;
white[1486]<=0;
white[1487]<=0;
white[1488]<=0;
white[1489]<=0;
white[1490]<=1;

```
white[ 1491 ]<=1;
white[ 1492 ]<=1;
white[ 1493 ]<=1;
white[ 1494 ]<=1;
white[ 1495 ]<=0;
white[ 1496 ]<=0;
white[ 1497 ]<=0;
white[ 1498 ]<=0;
white[ 1499 ]<=0;
white[ 1500 ]<=0;
white[ 1501 ]<=0;
white[ 1502 ]<=0;
white[ 1503 ]<=0;
white[ 1504 ]<=0;
white[ 1505 ]<=1;
white[ 1506 ]<=1;
white[ 1507 ]<=1;
white[ 1508 ]<=1;
white[ 1509 ]<=1;
white[ 1510 ]<=0;
white[ 1511 ]<=0;
white[ 1512 ]<=0;
white[ 1513 ]<=0;
white[ 1514 ]<=0;
white[ 1515 ]<=0;
white[ 1516 ]<=0;
white[ 1517 ]<=0;
white[ 1518 ]<=0;
white[ 1519 ]<=0;
white[ 1520 ]<=0;
white[ 1521 ]<=0;
white[ 1522 ]<=0;
white[ 1523 ]<=0;
white[ 1524 ]<=0;
white[ 1525 ]<=0;
white[ 1526 ]<=0;
white[ 1527 ]<=0;
white[ 1528 ]<=1;
white[ 1529 ]<=1;
white[ 1530 ]<=1;
white[ 1531 ]<=1;
white[ 1532 ]<=0;
white[ 1533 ]<=0;
white[ 1534 ]<=0;
white[ 1535 ]<=0;
white[ 1536 ]<=0;
white[ 1537 ]<=0;
white[ 1538 ]<=0;
white[ 1539 ]<=0;
white[ 1540 ]<=0;
white[ 1541 ]<=0;
white[ 1542 ]<=0;
white[ 1543 ]<=0;
white[ 1544 ]<=0;
white[ 1545 ]<=0;
white[ 1546 ]<=0;
white[ 1547 ]<=0;
white[ 1548 ]<=0;
```

```
white[ 1549 ]<=0;
white[ 1550 ]<=1;
white[ 1551 ]<=1;
white[ 1552 ]<=1;
white[ 1553 ]<=1;
white[ 1554 ]<=1;
white[ 1555 ]<=0;
white[ 1556 ]<=0;
white[ 1557 ]<=0;
white[ 1558 ]<=0;
white[ 1559 ]<=0;
white[ 1560 ]<=0;
white[ 1561 ]<=0;
white[ 1562 ]<=0;
white[ 1563 ]<=0;
white[ 1564 ]<=1;
white[ 1565 ]<=1;
white[ 1566 ]<=1;
white[ 1567 ]<=1;
white[ 1568 ]<=1;
white[ 1569 ]<=0;
white[ 1570 ]<=0;
white[ 1571 ]<=0;
white[ 1572 ]<=0;
white[ 1573 ]<=0;
white[ 1574 ]<=0;
white[ 1575 ]<=0;
white[ 1576 ]<=0;
white[ 1577 ]<=0;
white[ 1578 ]<=0;
white[ 1579 ]<=0;
white[ 1580 ]<=0;
white[ 1581 ]<=0;
white[ 1582 ]<=0;
white[ 1583 ]<=0;
white[ 1584 ]<=0;
white[ 1585 ]<=0;
white[ 1586 ]<=0;
white[ 1587 ]<=0;
white[ 1588 ]<=1;
white[ 1589 ]<=1;
white[ 1590 ]<=1;
white[ 1591 ]<=1;
white[ 1592 ]<=0;
white[ 1593 ]<=0;
white[ 1594 ]<=0;
white[ 1595 ]<=0;
white[ 1596 ]<=0;
white[ 1597 ]<=0;
white[ 1598 ]<=0;
white[ 1599 ]<=0;
white[ 1600 ]<=0;
white[ 1601 ]<=0;
white[ 1602 ]<=0;
white[ 1603 ]<=0;
white[ 1604 ]<=0;
white[ 1605 ]<=0;
white[ 1606 ]<=0;
```

```
white[ 1607 ]<=0;
white[ 1608 ]<=0;
white[ 1609 ]<=0;
white[ 1610 ]<=0;
white[ 1611 ]<=1;
white[ 1612 ]<=1;
white[ 1613 ]<=1;
white[ 1614 ]<=1;
white[ 1615 ]<=1;
white[ 1616 ]<=0;
white[ 1617 ]<=0;
white[ 1618 ]<=0;
white[ 1619 ]<=0;
white[ 1620 ]<=0;
white[ 1621 ]<=0;
white[ 1622 ]<=0;
white[ 1623 ]<=0;
white[ 1624 ]<=1;
white[ 1625 ]<=1;
white[ 1626 ]<=1;
white[ 1627 ]<=1;
white[ 1628 ]<=1;
white[ 1629 ]<=0;
white[ 1630 ]<=0;
white[ 1631 ]<=0;
white[ 1632 ]<=0;
white[ 1633 ]<=0;
white[ 1634 ]<=0;
white[ 1635 ]<=0;
white[ 1636 ]<=0;
white[ 1637 ]<=0;
white[ 1638 ]<=0;
white[ 1639 ]<=0;
white[ 1640 ]<=0;
white[ 1641 ]<=0;
white[ 1642 ]<=0;
white[ 1643 ]<=0;
white[ 1644 ]<=0;
white[ 1645 ]<=0;
white[ 1646 ]<=0;
white[ 1647 ]<=0;
white[ 1648 ]<=1;
white[ 1649 ]<=1;
white[ 1650 ]<=1;
white[ 1651 ]<=1;
white[ 1652 ]<=0;
white[ 1653 ]<=0;
white[ 1654 ]<=0;
white[ 1655 ]<=0;
white[ 1656 ]<=0;
white[ 1657 ]<=0;
white[ 1658 ]<=0;
white[ 1659 ]<=0;
white[ 1660 ]<=0;
white[ 1661 ]<=0;
white[ 1662 ]<=0;
white[ 1663 ]<=0;
white[ 1664 ]<=0;
```

```
white[ 1665 ]<=0;
white[ 1666 ]<=0;
white[ 1667 ]<=0;
white[ 1668 ]<=0;
white[ 1669 ]<=0;
white[ 1670 ]<=0;
white[ 1671 ]<=1;
white[ 1672 ]<=1;
white[ 1673 ]<=1;
white[ 1674 ]<=1;
white[ 1675 ]<=1;
white[ 1676 ]<=0;
white[ 1677 ]<=0;
white[ 1678 ]<=0;
white[ 1679 ]<=0;
white[ 1680 ]<=0;
white[ 1681 ]<=0;
white[ 1682 ]<=0;
white[ 1683 ]<=0;
white[ 1684 ]<=1;
white[ 1685 ]<=1;
white[ 1686 ]<=1;
white[ 1687 ]<=1;
white[ 1688 ]<=1;
white[ 1689 ]<=0;
white[ 1690 ]<=0;
white[ 1691 ]<=0;
white[ 1692 ]<=0;
white[ 1693 ]<=0;
white[ 1694 ]<=0;
white[ 1695 ]<=0;
white[ 1696 ]<=0;
white[ 1697 ]<=0;
white[ 1698 ]<=0;
white[ 1699 ]<=0;
white[ 1700 ]<=0;
white[ 1701 ]<=0;
white[ 1702 ]<=0;
white[ 1703 ]<=0;
white[ 1704 ]<=0;
white[ 1705 ]<=0;
white[ 1706 ]<=0;
white[ 1707 ]<=0;
white[ 1708 ]<=1;
white[ 1709 ]<=1;
white[ 1710 ]<=1;
white[ 1711 ]<=1;
white[ 1712 ]<=0;
white[ 1713 ]<=0;
white[ 1714 ]<=0;
white[ 1715 ]<=0;
white[ 1716 ]<=0;
white[ 1717 ]<=0;
white[ 1718 ]<=0;
white[ 1719 ]<=0;
white[ 1720 ]<=0;
white[ 1721 ]<=0;
white[ 1722 ]<=0;
```

```
white[ 1723 ]<=0;
white[ 1724 ]<=0;
white[ 1725 ]<=0;
white[ 1726 ]<=0;
white[ 1727 ]<=0;
white[ 1728 ]<=0;
white[ 1729 ]<=0;
white[ 1730 ]<=0;
white[ 1731 ]<=1;
white[ 1732 ]<=1;
white[ 1733 ]<=1;
white[ 1734 ]<=1;
white[ 1735 ]<=1;
white[ 1736 ]<=0;
white[ 1737 ]<=0;
white[ 1738 ]<=0;
white[ 1739 ]<=0;
white[ 1740 ]<=0;
white[ 1741 ]<=0;
white[ 1742 ]<=0;
white[ 1743 ]<=0;
white[ 1744 ]<=1;
white[ 1745 ]<=1;
white[ 1746 ]<=1;
white[ 1747 ]<=1;
white[ 1748 ]<=0;
white[ 1749 ]<=0;
white[ 1750 ]<=0;
white[ 1751 ]<=0;
white[ 1752 ]<=0;
white[ 1753 ]<=0;
white[ 1754 ]<=0;
white[ 1755 ]<=0;
white[ 1756 ]<=0;
white[ 1757 ]<=0;
white[ 1758 ]<=0;
white[ 1759 ]<=0;
white[ 1760 ]<=0;
white[ 1761 ]<=0;
white[ 1762 ]<=0;
white[ 1763 ]<=0;
white[ 1764 ]<=0;
white[ 1765 ]<=0;
white[ 1766 ]<=0;
white[ 1767 ]<=0;
white[ 1768 ]<=1;
white[ 1769 ]<=1;
white[ 1770 ]<=1;
white[ 1771 ]<=1;
white[ 1772 ]<=0;
white[ 1773 ]<=0;
white[ 1774 ]<=0;
white[ 1775 ]<=0;
white[ 1776 ]<=0;
white[ 1777 ]<=0;
white[ 1778 ]<=0;
white[ 1779 ]<=0;
white[ 1780 ]<=0;
```

```
white[ 1781 ]<=0;
white[ 1782 ]<=0;
white[ 1783 ]<=0;
white[ 1784 ]<=0;
white[ 1785 ]<=0;
white[ 1786 ]<=0;
white[ 1787 ]<=0;
white[ 1788 ]<=0;
white[ 1789 ]<=0;
white[ 1790 ]<=0;
white[ 1791 ]<=0;
white[ 1792 ]<=1;
white[ 1793 ]<=1;
white[ 1794 ]<=1;
white[ 1795 ]<=1;
white[ 1796 ]<=0;
white[ 1797 ]<=0;
white[ 1798 ]<=0;
white[ 1799 ]<=0;
white[ 1800 ]<=0;
white[ 1801 ]<=0;
white[ 1802 ]<=0;
white[ 1803 ]<=0;
white[ 1804 ]<=1;
white[ 1805 ]<=1;
white[ 1806 ]<=1;
white[ 1807 ]<=1;
white[ 1808 ]<=0;
white[ 1809 ]<=0;
white[ 1810 ]<=0;
white[ 1811 ]<=0;
white[ 1812 ]<=0;
white[ 1813 ]<=0;
white[ 1814 ]<=0;
white[ 1815 ]<=0;
white[ 1816 ]<=0;
white[ 1817 ]<=0;
white[ 1818 ]<=0;
white[ 1819 ]<=0;
white[ 1820 ]<=0;
white[ 1821 ]<=0;
white[ 1822 ]<=0;
white[ 1823 ]<=0;
white[ 1824 ]<=0;
white[ 1825 ]<=0;
white[ 1826 ]<=0;
white[ 1827 ]<=0;
white[ 1828 ]<=1;
white[ 1829 ]<=1;
white[ 1830 ]<=1;
white[ 1831 ]<=1;
white[ 1832 ]<=0;
white[ 1833 ]<=0;
white[ 1834 ]<=0;
white[ 1835 ]<=0;
white[ 1836 ]<=0;
white[ 1837 ]<=0;
white[ 1838 ]<=0;
```

```
white[ 1839 ]<=0;
white[ 1840 ]<=0;
white[ 1841 ]<=0;
white[ 1842 ]<=0;
white[ 1843 ]<=0;
white[ 1844 ]<=0;
white[ 1845 ]<=0;
white[ 1846 ]<=0;
white[ 1847 ]<=0;
white[ 1848 ]<=0;
white[ 1849 ]<=0;
white[ 1850 ]<=0;
white[ 1851 ]<=0;
white[ 1852 ]<=1;
white[ 1853 ]<=1;
white[ 1854 ]<=1;
white[ 1855 ]<=1;
white[ 1856 ]<=0;
white[ 1857 ]<=0;
white[ 1858 ]<=0;
white[ 1859 ]<=0;
white[ 1860 ]<=0;
white[ 1861 ]<=0;
white[ 1862 ]<=0;
white[ 1863 ]<=0;
white[ 1864 ]<=1;
white[ 1865 ]<=1;
white[ 1866 ]<=1;
white[ 1867 ]<=1;
white[ 1868 ]<=0;
white[ 1869 ]<=0;
white[ 1870 ]<=0;
white[ 1871 ]<=0;
white[ 1872 ]<=1;
white[ 1873 ]<=1;
white[ 1874 ]<=1;
white[ 1875 ]<=1;
white[ 1876 ]<=1;
white[ 1877 ]<=1;
white[ 1878 ]<=1;
white[ 1879 ]<=1;
white[ 1880 ]<=1;
white[ 1881 ]<=1;
white[ 1882 ]<=1;
white[ 1883 ]<=1;
white[ 1884 ]<=1;
white[ 1885 ]<=1;
white[ 1886 ]<=1;
white[ 1887 ]<=1;
white[ 1888 ]<=1;
white[ 1889 ]<=1;
white[ 1890 ]<=1;
white[ 1891 ]<=1;
white[ 1892 ]<=0;
white[ 1893 ]<=0;
white[ 1894 ]<=0;
white[ 1895 ]<=0;
white[ 1896 ]<=0;
```

```
white[ 1897 ]<=0;
white[ 1898 ]<=0;
white[ 1899 ]<=0;
white[ 1900 ]<=0;
white[ 1901 ]<=0;
white[ 1902 ]<=0;
white[ 1903 ]<=0;
white[ 1904 ]<=0;
white[ 1905 ]<=0;
white[ 1906 ]<=0;
white[ 1907 ]<=0;
white[ 1908 ]<=0;
white[ 1909 ]<=0;
white[ 1910 ]<=0;
white[ 1911 ]<=0;
white[ 1912 ]<=1;
white[ 1913 ]<=1;
white[ 1914 ]<=1;
white[ 1915 ]<=1;
white[ 1916 ]<=0;
white[ 1917 ]<=0;
white[ 1918 ]<=0;
white[ 1919 ]<=0;
white[ 1920 ]<=0;
white[ 1921 ]<=0;
white[ 1922 ]<=0;
white[ 1923 ]<=0;
white[ 1924 ]<=1;
white[ 1925 ]<=1;
white[ 1926 ]<=1;
white[ 1927 ]<=1;
white[ 1928 ]<=0;
white[ 1929 ]<=0;
white[ 1930 ]<=0;
white[ 1931 ]<=1;
white[ 1932 ]<=1;
white[ 1933 ]<=1;
white[ 1934 ]<=1;
white[ 1935 ]<=1;
white[ 1936 ]<=1;
white[ 1937 ]<=1;
white[ 1938 ]<=1;
white[ 1939 ]<=1;
white[ 1940 ]<=1;
white[ 1941 ]<=1;
white[ 1942 ]<=1;
white[ 1943 ]<=1;
white[ 1944 ]<=1;
white[ 1945 ]<=1;
white[ 1946 ]<=1;
white[ 1947 ]<=1;
white[ 1948 ]<=1;
white[ 1949 ]<=1;
white[ 1950 ]<=1;
white[ 1951 ]<=1;
white[ 1952 ]<=0;
white[ 1953 ]<=0;
white[ 1954 ]<=0;
```

```
white[ 1955 ]<=0;
white[ 1956 ]<=0;
white[ 1957 ]<=0;
white[ 1958 ]<=0;
white[ 1959 ]<=0;
white[ 1960 ]<=0;
white[ 1961 ]<=0;
white[ 1962 ]<=0;
white[ 1963 ]<=0;
white[ 1964 ]<=0;
white[ 1965 ]<=0;
white[ 1966 ]<=0;
white[ 1967 ]<=0;
white[ 1968 ]<=0;
white[ 1969 ]<=0;
white[ 1970 ]<=0;
white[ 1971 ]<=0;
white[ 1972 ]<=1;
white[ 1973 ]<=1;
white[ 1974 ]<=1;
white[ 1975 ]<=1;
white[ 1976 ]<=0;
white[ 1977 ]<=0;
white[ 1978 ]<=0;
white[ 1979 ]<=0;
white[ 1980 ]<=0;
white[ 1981 ]<=0;
white[ 1982 ]<=0;
white[ 1983 ]<=0;
white[ 1984 ]<=1;
white[ 1985 ]<=1;
white[ 1986 ]<=1;
white[ 1987 ]<=1;
white[ 1988 ]<=0;
white[ 1989 ]<=0;
white[ 1990 ]<=0;
white[ 1991 ]<=1;
white[ 1992 ]<=1;
white[ 1993 ]<=1;
white[ 1994 ]<=1;
white[ 1995 ]<=1;
white[ 1996 ]<=1;
white[ 1997 ]<=1;
white[ 1998 ]<=1;
white[ 1999 ]<=1;
white[ 2000 ]<=1;
white[ 2001 ]<=1;
white[ 2002 ]<=1;
white[ 2003 ]<=1;
white[ 2004 ]<=1;
white[ 2005 ]<=1;
white[ 2006 ]<=1;
white[ 2007 ]<=1;
white[ 2008 ]<=1;
white[ 2009 ]<=1;
white[ 2010 ]<=1;
white[ 2011 ]<=1;
white[ 2012 ]<=0;
```

```
white[ 2013 ]<=0;
white[ 2014 ]<=0;
white[ 2015 ]<=0;
white[ 2016 ]<=0;
white[ 2017 ]<=0;
white[ 2018 ]<=0;
white[ 2019 ]<=0;
white[ 2020 ]<=0;
white[ 2021 ]<=0;
white[ 2022 ]<=0;
white[ 2023 ]<=0;
white[ 2024 ]<=0;
white[ 2025 ]<=0;
white[ 2026 ]<=0;
white[ 2027 ]<=0;
white[ 2028 ]<=0;
white[ 2029 ]<=0;
white[ 2030 ]<=0;
white[ 2031 ]<=0;
white[ 2032 ]<=1;
white[ 2033 ]<=1;
white[ 2034 ]<=1;
white[ 2035 ]<=1;
white[ 2036 ]<=0;
white[ 2037 ]<=0;
white[ 2038 ]<=0;
white[ 2039 ]<=0;
white[ 2040 ]<=0;
white[ 2041 ]<=0;
white[ 2042 ]<=0;
white[ 2043 ]<=0;
white[ 2044 ]<=1;
white[ 2045 ]<=1;
white[ 2046 ]<=1;
white[ 2047 ]<=1;
white[ 2048 ]<=0;
white[ 2049 ]<=0;
white[ 2050 ]<=0;
white[ 2051 ]<=1;
white[ 2052 ]<=1;
white[ 2053 ]<=1;
white[ 2054 ]<=1;
white[ 2055 ]<=1;
white[ 2056 ]<=1;
white[ 2057 ]<=1;
white[ 2058 ]<=1;
white[ 2059 ]<=1;
white[ 2060 ]<=1;
white[ 2061 ]<=1;
white[ 2062 ]<=1;
white[ 2063 ]<=1;
white[ 2064 ]<=1;
white[ 2065 ]<=1;
white[ 2066 ]<=1;
white[ 2067 ]<=1;
white[ 2068 ]<=1;
white[ 2069 ]<=1;
white[ 2070 ]<=1;
```

```
white[ 2071 ]<=1;
white[ 2072 ]<=0;
white[ 2073 ]<=0;
white[ 2074 ]<=0;
white[ 2075 ]<=0;
white[ 2076 ]<=0;
white[ 2077 ]<=0;
white[ 2078 ]<=0;
white[ 2079 ]<=0;
white[ 2080 ]<=0;
white[ 2081 ]<=0;
white[ 2082 ]<=0;
white[ 2083 ]<=0;
white[ 2084 ]<=0;
white[ 2085 ]<=0;
white[ 2086 ]<=0;
white[ 2087 ]<=0;
white[ 2088 ]<=0;
white[ 2089 ]<=0;
white[ 2090 ]<=0;
white[ 2091 ]<=0;
white[ 2092 ]<=1;
white[ 2093 ]<=1;
white[ 2094 ]<=1;
white[ 2095 ]<=1;
white[ 2096 ]<=0;
white[ 2097 ]<=0;
white[ 2098 ]<=0;
white[ 2099 ]<=0;
white[ 2100 ]<=0;
white[ 2101 ]<=0;
white[ 2102 ]<=0;
white[ 2103 ]<=0;
white[ 2104 ]<=1;
white[ 2105 ]<=1;
white[ 2106 ]<=1;
white[ 2107 ]<=1;
white[ 2108 ]<=0;
white[ 2109 ]<=0;
white[ 2110 ]<=0;
white[ 2111 ]<=0;
white[ 2112 ]<=1;
white[ 2113 ]<=1;
white[ 2114 ]<=1;
white[ 2115 ]<=1;
white[ 2116 ]<=1;
white[ 2117 ]<=1;
white[ 2118 ]<=1;
white[ 2119 ]<=1;
white[ 2120 ]<=1;
white[ 2121 ]<=1;
white[ 2122 ]<=1;
white[ 2123 ]<=1;
white[ 2124 ]<=1;
white[ 2125 ]<=1;
white[ 2126 ]<=1;
white[ 2127 ]<=1;
white[ 2128 ]<=1;
```

```
white[ 2129 ]<=1;
white[ 2130 ]<=1;
white[ 2131 ]<=0;
white[ 2132 ]<=0;
white[ 2133 ]<=0;
white[ 2134 ]<=0;
white[ 2135 ]<=0;
white[ 2136 ]<=0;
white[ 2137 ]<=0;
white[ 2138 ]<=0;
white[ 2139 ]<=0;
white[ 2140 ]<=0;
white[ 2141 ]<=0;
white[ 2142 ]<=0;
white[ 2143 ]<=0;
white[ 2144 ]<=0;
white[ 2145 ]<=0;
white[ 2146 ]<=0;
white[ 2147 ]<=0;
white[ 2148 ]<=0;
white[ 2149 ]<=0;
white[ 2150 ]<=0;
white[ 2151 ]<=0;
white[ 2152 ]<=1;
white[ 2153 ]<=1;
white[ 2154 ]<=1;
white[ 2155 ]<=1;
white[ 2156 ]<=0;
white[ 2157 ]<=0;
white[ 2158 ]<=0;
white[ 2159 ]<=0;
white[ 2160 ]<=0;
white[ 2161 ]<=0;
white[ 2162 ]<=0;
white[ 2163 ]<=0;
white[ 2164 ]<=1;
white[ 2165 ]<=1;
white[ 2166 ]<=1;
white[ 2167 ]<=1;
white[ 2168 ]<=0;
white[ 2169 ]<=0;
white[ 2170 ]<=0;
white[ 2171 ]<=0;
white[ 2172 ]<=0;
white[ 2173 ]<=0;
white[ 2174 ]<=0;
white[ 2175 ]<=0;
white[ 2176 ]<=0;
white[ 2177 ]<=0;
white[ 2178 ]<=0;
white[ 2179 ]<=0;
white[ 2180 ]<=0;
white[ 2181 ]<=0;
white[ 2182 ]<=0;
white[ 2183 ]<=0;
white[ 2184 ]<=0;
white[ 2185 ]<=0;
white[ 2186 ]<=0;
```

```
white[ 2187 ]<=0;
white[ 2188 ]<=0;
white[ 2189 ]<=0;
white[ 2190 ]<=0;
white[ 2191 ]<=0;
white[ 2192 ]<=0;
white[ 2193 ]<=0;
white[ 2194 ]<=0;
white[ 2195 ]<=0;
white[ 2196 ]<=0;
white[ 2197 ]<=0;
white[ 2198 ]<=0;
white[ 2199 ]<=0;
white[ 2200 ]<=0;
white[ 2201 ]<=0;
white[ 2202 ]<=0;
white[ 2203 ]<=0;
white[ 2204 ]<=0;
white[ 2205 ]<=0;
white[ 2206 ]<=0;
white[ 2207 ]<=0;
white[ 2208 ]<=0;
white[ 2209 ]<=0;
white[ 2210 ]<=0;
white[ 2211 ]<=0;
white[ 2212 ]<=1;
white[ 2213 ]<=1;
white[ 2214 ]<=1;
white[ 2215 ]<=1;
white[ 2216 ]<=0;
white[ 2217 ]<=0;
white[ 2218 ]<=0;
white[ 2219 ]<=0;
white[ 2220 ]<=0;
white[ 2221 ]<=0;
white[ 2222 ]<=0;
white[ 2223 ]<=0;
white[ 2224 ]<=1;
white[ 2225 ]<=1;
white[ 2226 ]<=1;
white[ 2227 ]<=1;
white[ 2228 ]<=0;
white[ 2229 ]<=0;
white[ 2230 ]<=0;
white[ 2231 ]<=0;
white[ 2232 ]<=0;
white[ 2233 ]<=0;
white[ 2234 ]<=0;
white[ 2235 ]<=0;
white[ 2236 ]<=0;
white[ 2237 ]<=0;
white[ 2238 ]<=0;
white[ 2239 ]<=0;
white[ 2240 ]<=0;
white[ 2241 ]<=0;
white[ 2242 ]<=0;
white[ 2243 ]<=0;
white[ 2244 ]<=0;
```

```
white[ 2245 ]<=0;
white[ 2246 ]<=0;
white[ 2247 ]<=0;
white[ 2248 ]<=0;
white[ 2249 ]<=0;
white[ 2250 ]<=0;
white[ 2251 ]<=0;
white[ 2252 ]<=0;
white[ 2253 ]<=0;
white[ 2254 ]<=0;
white[ 2255 ]<=0;
white[ 2256 ]<=0;
white[ 2257 ]<=0;
white[ 2258 ]<=0;
white[ 2259 ]<=0;
white[ 2260 ]<=0;
white[ 2261 ]<=0;
white[ 2262 ]<=0;
white[ 2263 ]<=0;
white[ 2264 ]<=0;
white[ 2265 ]<=0;
white[ 2266 ]<=0;
white[ 2267 ]<=0;
white[ 2268 ]<=0;
white[ 2269 ]<=0;
white[ 2270 ]<=0;
white[ 2271 ]<=0;
white[ 2272 ]<=1;
white[ 2273 ]<=1;
white[ 2274 ]<=1;
white[ 2275 ]<=1;
white[ 2276 ]<=0;
white[ 2277 ]<=0;
white[ 2278 ]<=0;
white[ 2279 ]<=0;
white[ 2280 ]<=0;
white[ 2281 ]<=0;
white[ 2282 ]<=0;
white[ 2283 ]<=0;
white[ 2284 ]<=1;
white[ 2285 ]<=1;
white[ 2286 ]<=1;
white[ 2287 ]<=1;
white[ 2288 ]<=1;
white[ 2289 ]<=0;
white[ 2290 ]<=0;
white[ 2291 ]<=0;
white[ 2292 ]<=0;
white[ 2293 ]<=0;
white[ 2294 ]<=0;
white[ 2295 ]<=0;
white[ 2296 ]<=0;
white[ 2297 ]<=0;
white[ 2298 ]<=0;
white[ 2299 ]<=0;
white[ 2300 ]<=0;
white[ 2301 ]<=0;
white[ 2302 ]<=0;
```

```
white[ 2303 ]<=0;
white[ 2304 ]<=0;
white[ 2305 ]<=0;
white[ 2306 ]<=0;
white[ 2307 ]<=0;
white[ 2308 ]<=0;
white[ 2309 ]<=0;
white[ 2310 ]<=0;
white[ 2311 ]<=0;
white[ 2312 ]<=0;
white[ 2313 ]<=0;
white[ 2314 ]<=0;
white[ 2315 ]<=0;
white[ 2316 ]<=0;
white[ 2317 ]<=0;
white[ 2318 ]<=0;
white[ 2319 ]<=0;
white[ 2320 ]<=0;
white[ 2321 ]<=0;
white[ 2322 ]<=0;
white[ 2323 ]<=0;
white[ 2324 ]<=0;
white[ 2325 ]<=0;
white[ 2326 ]<=0;
white[ 2327 ]<=0;
white[ 2328 ]<=0;
white[ 2329 ]<=0;
white[ 2330 ]<=0;
white[ 2331 ]<=1;
white[ 2332 ]<=1;
white[ 2333 ]<=1;
white[ 2334 ]<=1;
white[ 2335 ]<=1;
white[ 2336 ]<=0;
white[ 2337 ]<=0;
white[ 2338 ]<=0;
white[ 2339 ]<=0;
white[ 2340 ]<=0;
white[ 2341 ]<=0;
white[ 2342 ]<=0;
white[ 2343 ]<=0;
white[ 2344 ]<=1;
white[ 2345 ]<=1;
white[ 2346 ]<=1;
white[ 2347 ]<=1;
white[ 2348 ]<=1;
white[ 2349 ]<=0;
white[ 2350 ]<=0;
white[ 2351 ]<=0;
white[ 2352 ]<=0;
white[ 2353 ]<=0;
white[ 2354 ]<=0;
white[ 2355 ]<=0;
white[ 2356 ]<=0;
white[ 2357 ]<=0;
white[ 2358 ]<=0;
white[ 2359 ]<=0;
white[ 2360 ]<=0;
```

```
white[ 2361 ]<=0;
white[ 2362 ]<=0;
white[ 2363 ]<=0;
white[ 2364 ]<=0;
white[ 2365 ]<=0;
white[ 2366 ]<=0;
white[ 2367 ]<=0;
white[ 2368 ]<=0;
white[ 2369 ]<=0;
white[ 2370 ]<=0;
white[ 2371 ]<=0;
white[ 2372 ]<=0;
white[ 2373 ]<=0;
white[ 2374 ]<=0;
white[ 2375 ]<=0;
white[ 2376 ]<=0;
white[ 2377 ]<=0;
white[ 2378 ]<=0;
white[ 2379 ]<=0;
white[ 2380 ]<=0;
white[ 2381 ]<=0;
white[ 2382 ]<=0;
white[ 2383 ]<=0;
white[ 2384 ]<=0;
white[ 2385 ]<=0;
white[ 2386 ]<=0;
white[ 2387 ]<=0;
white[ 2388 ]<=0;
white[ 2389 ]<=0;
white[ 2390 ]<=0;
white[ 2391 ]<=1;
white[ 2392 ]<=1;
white[ 2393 ]<=1;
white[ 2394 ]<=1;
white[ 2395 ]<=1;
white[ 2396 ]<=0;
white[ 2397 ]<=0;
white[ 2398 ]<=0;
white[ 2399 ]<=0;
white[ 2400 ]<=0;
white[ 2401 ]<=0;
white[ 2402 ]<=0;
white[ 2403 ]<=0;
white[ 2404 ]<=1;
white[ 2405 ]<=1;
white[ 2406 ]<=1;
white[ 2407 ]<=1;
white[ 2408 ]<=1;
white[ 2409 ]<=0;
white[ 2410 ]<=0;
white[ 2411 ]<=0;
white[ 2412 ]<=0;
white[ 2413 ]<=0;
white[ 2414 ]<=0;
white[ 2415 ]<=0;
white[ 2416 ]<=0;
white[ 2417 ]<=0;
white[ 2418 ]<=0;
```

```
white[ 2419 ]<=0;
white[ 2420 ]<=0;
white[ 2421 ]<=0;
white[ 2422 ]<=0;
white[ 2423 ]<=0;
white[ 2424 ]<=0;
white[ 2425 ]<=0;
white[ 2426 ]<=0;
white[ 2427 ]<=0;
white[ 2428 ]<=0;
white[ 2429 ]<=0;
white[ 2430 ]<=0;
white[ 2431 ]<=0;
white[ 2432 ]<=0;
white[ 2433 ]<=0;
white[ 2434 ]<=0;
white[ 2435 ]<=0;
white[ 2436 ]<=0;
white[ 2437 ]<=0;
white[ 2438 ]<=0;
white[ 2439 ]<=0;
white[ 2440 ]<=0;
white[ 2441 ]<=0;
white[ 2442 ]<=0;
white[ 2443 ]<=0;
white[ 2444 ]<=0;
white[ 2445 ]<=0;
white[ 2446 ]<=0;
white[ 2447 ]<=0;
white[ 2448 ]<=0;
white[ 2449 ]<=0;
white[ 2450 ]<=0;
white[ 2451 ]<=1;
white[ 2452 ]<=1;
white[ 2453 ]<=1;
white[ 2454 ]<=1;
white[ 2455 ]<=1;
white[ 2456 ]<=0;
white[ 2457 ]<=0;
white[ 2458 ]<=0;
white[ 2459 ]<=0;
white[ 2460 ]<=0;
white[ 2461 ]<=0;
white[ 2462 ]<=0;
white[ 2463 ]<=0;
white[ 2464 ]<=1;
white[ 2465 ]<=1;
white[ 2466 ]<=1;
white[ 2467 ]<=1;
white[ 2468 ]<=1;
white[ 2469 ]<=1;
white[ 2470 ]<=0;
white[ 2471 ]<=0;
white[ 2472 ]<=0;
white[ 2473 ]<=0;
white[ 2474 ]<=0;
white[ 2475 ]<=0;
white[ 2476 ]<=0;
```

```
white[ 2477 ]<=0;
white[ 2478 ]<=0;
white[ 2479 ]<=0;
white[ 2480 ]<=0;
white[ 2481 ]<=0;
white[ 2482 ]<=0;
white[ 2483 ]<=0;
white[ 2484 ]<=0;
white[ 2485 ]<=0;
white[ 2486 ]<=0;
white[ 2487 ]<=0;
white[ 2488 ]<=0;
white[ 2489 ]<=0;
white[ 2490 ]<=0;
white[ 2491 ]<=0;
white[ 2492 ]<=0;
white[ 2493 ]<=0;
white[ 2494 ]<=0;
white[ 2495 ]<=0;
white[ 2496 ]<=0;
white[ 2497 ]<=0;
white[ 2498 ]<=0;
white[ 2499 ]<=0;
white[ 2500 ]<=0;
white[ 2501 ]<=0;
white[ 2502 ]<=0;
white[ 2503 ]<=0;
white[ 2504 ]<=0;
white[ 2505 ]<=0;
white[ 2506 ]<=0;
white[ 2507 ]<=0;
white[ 2508 ]<=0;
white[ 2509 ]<=0;
white[ 2510 ]<=1;
white[ 2511 ]<=1;
white[ 2512 ]<=1;
white[ 2513 ]<=1;
white[ 2514 ]<=1;
white[ 2515 ]<=1;
white[ 2516 ]<=0;
white[ 2517 ]<=0;
white[ 2518 ]<=0;
white[ 2519 ]<=0;
white[ 2520 ]<=0;
white[ 2521 ]<=0;
white[ 2522 ]<=0;
white[ 2523 ]<=0;
white[ 2524 ]<=0;
white[ 2525 ]<=1;
white[ 2526 ]<=1;
white[ 2527 ]<=1;
white[ 2528 ]<=1;
white[ 2529 ]<=1;
white[ 2530 ]<=0;
white[ 2531 ]<=0;
white[ 2532 ]<=0;
white[ 2533 ]<=0;
white[ 2534 ]<=0;
```

```
white[ 2535 ]<=0;
white[ 2536 ]<=0;
white[ 2537 ]<=0;
white[ 2538 ]<=0;
white[ 2539 ]<=0;
white[ 2540 ]<=0;
white[ 2541 ]<=0;
white[ 2542 ]<=0;
white[ 2543 ]<=0;
white[ 2544 ]<=0;
white[ 2545 ]<=0;
white[ 2546 ]<=0;
white[ 2547 ]<=0;
white[ 2548 ]<=0;
white[ 2549 ]<=0;
white[ 2550 ]<=0;
white[ 2551 ]<=0;
white[ 2552 ]<=0;
white[ 2553 ]<=0;
white[ 2554 ]<=0;
white[ 2555 ]<=0;
white[ 2556 ]<=0;
white[ 2557 ]<=0;
white[ 2558 ]<=0;
white[ 2559 ]<=0;
white[ 2560 ]<=0;
white[ 2561 ]<=0;
white[ 2562 ]<=0;
white[ 2563 ]<=0;
white[ 2564 ]<=0;
white[ 2565 ]<=0;
white[ 2566 ]<=0;
white[ 2567 ]<=0;
white[ 2568 ]<=0;
white[ 2569 ]<=0;
white[ 2570 ]<=1;
white[ 2571 ]<=1;
white[ 2572 ]<=1;
white[ 2573 ]<=1;
white[ 2574 ]<=1;
white[ 2575 ]<=0;
white[ 2576 ]<=0;
white[ 2577 ]<=0;
white[ 2578 ]<=0;
white[ 2579 ]<=0;
white[ 2580 ]<=0;
white[ 2581 ]<=0;
white[ 2582 ]<=0;
white[ 2583 ]<=0;
white[ 2584 ]<=0;
white[ 2585 ]<=1;
white[ 2586 ]<=1;
white[ 2587 ]<=1;
white[ 2588 ]<=1;
white[ 2589 ]<=1;
white[ 2590 ]<=1;
white[ 2591 ]<=0;
white[ 2592 ]<=0;
```

```
white[ 2593 ]<=0;
white[ 2594 ]<=0;
white[ 2595 ]<=0;
white[ 2596 ]<=0;
white[ 2597 ]<=0;
white[ 2598 ]<=0;
white[ 2599 ]<=0;
white[ 2600 ]<=0;
white[ 2601 ]<=0;
white[ 2602 ]<=0;
white[ 2603 ]<=0;
white[ 2604 ]<=0;
white[ 2605 ]<=0;
white[ 2606 ]<=0;
white[ 2607 ]<=0;
white[ 2608 ]<=0;
white[ 2609 ]<=0;
white[ 2610 ]<=0;
white[ 2611 ]<=0;
white[ 2612 ]<=0;
white[ 2613 ]<=0;
white[ 2614 ]<=0;
white[ 2615 ]<=0;
white[ 2616 ]<=0;
white[ 2617 ]<=0;
white[ 2618 ]<=0;
white[ 2619 ]<=0;
white[ 2620 ]<=0;
white[ 2621 ]<=0;
white[ 2622 ]<=0;
white[ 2623 ]<=0;
white[ 2624 ]<=0;
white[ 2625 ]<=0;
white[ 2626 ]<=0;
white[ 2627 ]<=0;
white[ 2628 ]<=0;
white[ 2629 ]<=1;
white[ 2630 ]<=1;
white[ 2631 ]<=1;
white[ 2632 ]<=1;
white[ 2633 ]<=1;
white[ 2634 ]<=1;
white[ 2635 ]<=0;
white[ 2636 ]<=0;
white[ 2637 ]<=0;
white[ 2638 ]<=0;
white[ 2639 ]<=0;
white[ 2640 ]<=0;
white[ 2641 ]<=0;
white[ 2642 ]<=0;
white[ 2643 ]<=0;
white[ 2644 ]<=0;
white[ 2645 ]<=0;
white[ 2646 ]<=1;
white[ 2647 ]<=1;
white[ 2648 ]<=1;
white[ 2649 ]<=1;
white[ 2650 ]<=1;
```

```
white[ 2651 ]<=0;
white[ 2652 ]<=0;
white[ 2653 ]<=0;
white[ 2654 ]<=0;
white[ 2655 ]<=0;
white[ 2656 ]<=0;
white[ 2657 ]<=0;
white[ 2658 ]<=0;
white[ 2659 ]<=0;
white[ 2660 ]<=0;
white[ 2661 ]<=0;
white[ 2662 ]<=0;
white[ 2663 ]<=0;
white[ 2664 ]<=0;
white[ 2665 ]<=0;
white[ 2666 ]<=0;
white[ 2667 ]<=0;
white[ 2668 ]<=0;
white[ 2669 ]<=0;
white[ 2670 ]<=0;
white[ 2671 ]<=0;
white[ 2672 ]<=0;
white[ 2673 ]<=0;
white[ 2674 ]<=0;
white[ 2675 ]<=0;
white[ 2676 ]<=0;
white[ 2677 ]<=0;
white[ 2678 ]<=0;
white[ 2679 ]<=0;
white[ 2680 ]<=0;
white[ 2681 ]<=0;
white[ 2682 ]<=0;
white[ 2683 ]<=0;
white[ 2684 ]<=0;
white[ 2685 ]<=0;
white[ 2686 ]<=0;
white[ 2687 ]<=0;
white[ 2688 ]<=0;
white[ 2689 ]<=1;
white[ 2690 ]<=1;
white[ 2691 ]<=1;
white[ 2692 ]<=1;
white[ 2693 ]<=1;
white[ 2694 ]<=0;
white[ 2695 ]<=0;
white[ 2696 ]<=0;
white[ 2697 ]<=0;
white[ 2698 ]<=0;
white[ 2699 ]<=0;
white[ 2700 ]<=0;
white[ 2701 ]<=0;
white[ 2702 ]<=0;
white[ 2703 ]<=0;
white[ 2704 ]<=0;
white[ 2705 ]<=0;
white[ 2706 ]<=1;
white[ 2707 ]<=1;
white[ 2708 ]<=1;
```

```
white[ 2709 ]<=1;
white[ 2710 ]<=1;
white[ 2711 ]<=0;
white[ 2712 ]<=0;
white[ 2713 ]<=0;
white[ 2714 ]<=0;
white[ 2715 ]<=0;
white[ 2716 ]<=0;
white[ 2717 ]<=0;
white[ 2718 ]<=0;
white[ 2719 ]<=0;
white[ 2720 ]<=0;
white[ 2721 ]<=0;
white[ 2722 ]<=0;
white[ 2723 ]<=0;
white[ 2724 ]<=0;
white[ 2725 ]<=0;
white[ 2726 ]<=0;
white[ 2727 ]<=0;
white[ 2728 ]<=0;
white[ 2729 ]<=0;
white[ 2730 ]<=0;
white[ 2731 ]<=0;
white[ 2732 ]<=0;
white[ 2733 ]<=0;
white[ 2734 ]<=0;
white[ 2735 ]<=0;
white[ 2736 ]<=0;
white[ 2737 ]<=0;
white[ 2738 ]<=0;
white[ 2739 ]<=0;
white[ 2740 ]<=0;
white[ 2741 ]<=0;
white[ 2742 ]<=0;
white[ 2743 ]<=0;
white[ 2744 ]<=0;
white[ 2745 ]<=0;
white[ 2746 ]<=0;
white[ 2747 ]<=0;
white[ 2748 ]<=0;
white[ 2749 ]<=1;
white[ 2750 ]<=1;
white[ 2751 ]<=1;
white[ 2752 ]<=1;
white[ 2753 ]<=1;
white[ 2754 ]<=0;
white[ 2755 ]<=0;
white[ 2756 ]<=0;
white[ 2757 ]<=0;
white[ 2758 ]<=0;
white[ 2759 ]<=0;
white[ 2760 ]<=0;
white[ 2761 ]<=0;
white[ 2762 ]<=0;
white[ 2763 ]<=0;
white[ 2764 ]<=0;
white[ 2765 ]<=0;
white[ 2766 ]<=0;
```

```
white[ 2767 ]<=1;
white[ 2768 ]<=1;
white[ 2769 ]<=1;
white[ 2770 ]<=1;
white[ 2771 ]<=1;
white[ 2772 ]<=0;
white[ 2773 ]<=0;
white[ 2774 ]<=0;
white[ 2775 ]<=0;
white[ 2776 ]<=0;
white[ 2777 ]<=0;
white[ 2778 ]<=0;
white[ 2779 ]<=0;
white[ 2780 ]<=0;
white[ 2781 ]<=0;
white[ 2782 ]<=0;
white[ 2783 ]<=0;
white[ 2784 ]<=0;
white[ 2785 ]<=0;
white[ 2786 ]<=0;
white[ 2787 ]<=0;
white[ 2788 ]<=0;
white[ 2789 ]<=0;
white[ 2790 ]<=0;
white[ 2791 ]<=0;
white[ 2792 ]<=0;
white[ 2793 ]<=0;
white[ 2794 ]<=0;
white[ 2795 ]<=0;
white[ 2796 ]<=0;
white[ 2797 ]<=0;
white[ 2798 ]<=0;
white[ 2799 ]<=0;
white[ 2800 ]<=0;
white[ 2801 ]<=0;
white[ 2802 ]<=0;
white[ 2803 ]<=0;
white[ 2804 ]<=0;
white[ 2805 ]<=0;
white[ 2806 ]<=0;
white[ 2807 ]<=0;
white[ 2808 ]<=1;
white[ 2809 ]<=1;
white[ 2810 ]<=1;
white[ 2811 ]<=1;
white[ 2812 ]<=1;
white[ 2813 ]<=0;
white[ 2814 ]<=0;
white[ 2815 ]<=0;
white[ 2816 ]<=0;
white[ 2817 ]<=0;
white[ 2818 ]<=0;
white[ 2819 ]<=0;
white[ 2820 ]<=0;
white[ 2821 ]<=0;
white[ 2822 ]<=0;
white[ 2823 ]<=0;
white[ 2824 ]<=0;
```

```
white[ 2825 ]<=0;
white[ 2826 ]<=0;
white[ 2827 ]<=1;
white[ 2828 ]<=1;
white[ 2829 ]<=1;
white[ 2830 ]<=1;
white[ 2831 ]<=1;
white[ 2832 ]<=1;
white[ 2833 ]<=0;
white[ 2834 ]<=0;
white[ 2835 ]<=0;
white[ 2836 ]<=0;
white[ 2837 ]<=0;
white[ 2838 ]<=0;
white[ 2839 ]<=0;
white[ 2840 ]<=0;
white[ 2841 ]<=0;
white[ 2842 ]<=0;
white[ 2843 ]<=0;
white[ 2844 ]<=0;
white[ 2845 ]<=0;
white[ 2846 ]<=0;
white[ 2847 ]<=0;
white[ 2848 ]<=0;
white[ 2849 ]<=0;
white[ 2850 ]<=0;
white[ 2851 ]<=0;
white[ 2852 ]<=0;
white[ 2853 ]<=0;
white[ 2854 ]<=0;
white[ 2855 ]<=0;
white[ 2856 ]<=0;
white[ 2857 ]<=0;
white[ 2858 ]<=0;
white[ 2859 ]<=0;
white[ 2860 ]<=0;
white[ 2861 ]<=0;
white[ 2862 ]<=0;
white[ 2863 ]<=0;
white[ 2864 ]<=0;
white[ 2865 ]<=0;
white[ 2866 ]<=0;
white[ 2867 ]<=1;
white[ 2868 ]<=1;
white[ 2869 ]<=1;
white[ 2870 ]<=1;
white[ 2871 ]<=1;
white[ 2872 ]<=1;
white[ 2873 ]<=0;
white[ 2874 ]<=0;
white[ 2875 ]<=0;
white[ 2876 ]<=0;
white[ 2877 ]<=0;
white[ 2878 ]<=0;
white[ 2879 ]<=0;
white[ 2880 ]<=0;
white[ 2881 ]<=0;
white[ 2882 ]<=0;
```

```
white[ 2883 ]<=0;
white[ 2884 ]<=0;
white[ 2885 ]<=0;
white[ 2886 ]<=0;
white[ 2887 ]<=0;
white[ 2888 ]<=1;
white[ 2889 ]<=1;
white[ 2890 ]<=1;
white[ 2891 ]<=1;
white[ 2892 ]<=1;
white[ 2893 ]<=1;
white[ 2894 ]<=0;
white[ 2895 ]<=0;
white[ 2896 ]<=0;
white[ 2897 ]<=0;
white[ 2898 ]<=0;
white[ 2899 ]<=0;
white[ 2900 ]<=0;
white[ 2901 ]<=0;
white[ 2902 ]<=0;
white[ 2903 ]<=0;
white[ 2904 ]<=0;
white[ 2905 ]<=0;
white[ 2906 ]<=0;
white[ 2907 ]<=0;
white[ 2908 ]<=0;
white[ 2909 ]<=0;
white[ 2910 ]<=0;
white[ 2911 ]<=0;
white[ 2912 ]<=0;
white[ 2913 ]<=0;
white[ 2914 ]<=0;
white[ 2915 ]<=0;
white[ 2916 ]<=0;
white[ 2917 ]<=0;
white[ 2918 ]<=0;
white[ 2919 ]<=0;
white[ 2920 ]<=0;
white[ 2921 ]<=0;
white[ 2922 ]<=0;
white[ 2923 ]<=0;
white[ 2924 ]<=0;
white[ 2925 ]<=0;
white[ 2926 ]<=1;
white[ 2927 ]<=1;
white[ 2928 ]<=1;
white[ 2929 ]<=1;
white[ 2930 ]<=1;
white[ 2931 ]<=1;
white[ 2932 ]<=0;
white[ 2933 ]<=0;
white[ 2934 ]<=0;
white[ 2935 ]<=0;
white[ 2936 ]<=0;
white[ 2937 ]<=0;
white[ 2938 ]<=0;
white[ 2939 ]<=0;
white[ 2940 ]<=0;
```

```
white[ 2941 ]<=0;
white[ 2942 ]<=0;
white[ 2943 ]<=0;
white[ 2944 ]<=0;
white[ 2945 ]<=0;
white[ 2946 ]<=0;
white[ 2947 ]<=0;
white[ 2948 ]<=0;
white[ 2949 ]<=1;
white[ 2950 ]<=1;
white[ 2951 ]<=1;
white[ 2952 ]<=1;
white[ 2953 ]<=1;
white[ 2954 ]<=1;
white[ 2955 ]<=0;
white[ 2956 ]<=0;
white[ 2957 ]<=0;
white[ 2958 ]<=0;
white[ 2959 ]<=0;
white[ 2960 ]<=0;
white[ 2961 ]<=0;
white[ 2962 ]<=0;
white[ 2963 ]<=0;
white[ 2964 ]<=0;
white[ 2965 ]<=0;
white[ 2966 ]<=0;
white[ 2967 ]<=0;
white[ 2968 ]<=0;
white[ 2969 ]<=0;
white[ 2970 ]<=0;
white[ 2971 ]<=0;
white[ 2972 ]<=0;
white[ 2973 ]<=0;
white[ 2974 ]<=0;
white[ 2975 ]<=0;
white[ 2976 ]<=0;
white[ 2977 ]<=0;
white[ 2978 ]<=0;
white[ 2979 ]<=0;
white[ 2980 ]<=0;
white[ 2981 ]<=0;
white[ 2982 ]<=0;
white[ 2983 ]<=0;
white[ 2984 ]<=0;
white[ 2985 ]<=1;
white[ 2986 ]<=1;
white[ 2987 ]<=1;
white[ 2988 ]<=1;
white[ 2989 ]<=1;
white[ 2990 ]<=1;
white[ 2991 ]<=0;
white[ 2992 ]<=0;
white[ 2993 ]<=0;
white[ 2994 ]<=0;
white[ 2995 ]<=0;
white[ 2996 ]<=0;
white[ 2997 ]<=0;
white[ 2998 ]<=0;
```

```
white[ 2999 ]<=0;
white[ 3000 ]<=0;
white[ 3001 ]<=0;
white[ 3002 ]<=0;
white[ 3003 ]<=0;
white[ 3004 ]<=0;
white[ 3005 ]<=0;
white[ 3006 ]<=0;
white[ 3007 ]<=0;
white[ 3008 ]<=0;
white[ 3009 ]<=1;
white[ 3010 ]<=1;
white[ 3011 ]<=1;
white[ 3012 ]<=1;
white[ 3013 ]<=1;
white[ 3014 ]<=1;
white[ 3015 ]<=1;
white[ 3016 ]<=0;
white[ 3017 ]<=0;
white[ 3018 ]<=0;
white[ 3019 ]<=0;
white[ 3020 ]<=0;
white[ 3021 ]<=0;
white[ 3022 ]<=0;
white[ 3023 ]<=0;
white[ 3024 ]<=0;
white[ 3025 ]<=0;
white[ 3026 ]<=0;
white[ 3027 ]<=0;
white[ 3028 ]<=0;
white[ 3029 ]<=0;
white[ 3030 ]<=0;
white[ 3031 ]<=0;
white[ 3032 ]<=0;
white[ 3033 ]<=0;
white[ 3034 ]<=0;
white[ 3035 ]<=0;
white[ 3036 ]<=0;
white[ 3037 ]<=0;
white[ 3038 ]<=0;
white[ 3039 ]<=0;
white[ 3040 ]<=0;
white[ 3041 ]<=0;
white[ 3042 ]<=0;
white[ 3043 ]<=0;
white[ 3044 ]<=1;
white[ 3045 ]<=1;
white[ 3046 ]<=1;
white[ 3047 ]<=1;
white[ 3048 ]<=1;
white[ 3049 ]<=1;
white[ 3050 ]<=1;
white[ 3051 ]<=0;
white[ 3052 ]<=0;
white[ 3053 ]<=0;
white[ 3054 ]<=0;
white[ 3055 ]<=0;
white[ 3056 ]<=0;
```

```
white[ 3057 ]<=0;
white[ 3058 ]<=0;
white[ 3059 ]<=0;
white[ 3060 ]<=0;
white[ 3061 ]<=0;
white[ 3062 ]<=0;
white[ 3063 ]<=0;
white[ 3064 ]<=0;
white[ 3065 ]<=0;
white[ 3066 ]<=0;
white[ 3067 ]<=0;
white[ 3068 ]<=0;
white[ 3069 ]<=0;
white[ 3070 ]<=1;
white[ 3071 ]<=1;
white[ 3072 ]<=1;
white[ 3073 ]<=1;
white[ 3074 ]<=1;
white[ 3075 ]<=1;
white[ 3076 ]<=1;
white[ 3077 ]<=0;
white[ 3078 ]<=0;
white[ 3079 ]<=0;
white[ 3080 ]<=0;
white[ 3081 ]<=0;
white[ 3082 ]<=0;
white[ 3083 ]<=0;
white[ 3084 ]<=0;
white[ 3085 ]<=0;
white[ 3086 ]<=0;
white[ 3087 ]<=0;
white[ 3088 ]<=0;
white[ 3089 ]<=0;
white[ 3090 ]<=0;
white[ 3091 ]<=0;
white[ 3092 ]<=0;
white[ 3093 ]<=0;
white[ 3094 ]<=0;
white[ 3095 ]<=0;
white[ 3096 ]<=0;
white[ 3097 ]<=0;
white[ 3098 ]<=0;
white[ 3099 ]<=0;
white[ 3100 ]<=0;
white[ 3101 ]<=0;
white[ 3102 ]<=0;
white[ 3103 ]<=1;
white[ 3104 ]<=1;
white[ 3105 ]<=1;
white[ 3106 ]<=1;
white[ 3107 ]<=1;
white[ 3108 ]<=1;
white[ 3109 ]<=1;
white[ 3110 ]<=0;
white[ 3111 ]<=0;
white[ 3112 ]<=0;
white[ 3113 ]<=0;
white[ 3114 ]<=0;
```

```
white[ 3115 ]<=0;
white[ 3116 ]<=0;
white[ 3117 ]<=0;
white[ 3118 ]<=0;
white[ 3119 ]<=0;
white[ 3120 ]<=0;
white[ 3121 ]<=0;
white[ 3122 ]<=0;
white[ 3123 ]<=0;
white[ 3124 ]<=0;
white[ 3125 ]<=0;
white[ 3126 ]<=0;
white[ 3127 ]<=0;
white[ 3128 ]<=0;
white[ 3129 ]<=0;
white[ 3130 ]<=0;
white[ 3131 ]<=1;
white[ 3132 ]<=1;
white[ 3133 ]<=1;
white[ 3134 ]<=1;
white[ 3135 ]<=1;
white[ 3136 ]<=1;
white[ 3137 ]<=1;
white[ 3138 ]<=0;
white[ 3139 ]<=0;
white[ 3140 ]<=0;
white[ 3141 ]<=0;
white[ 3142 ]<=0;
white[ 3143 ]<=0;
white[ 3144 ]<=0;
white[ 3145 ]<=0;
white[ 3146 ]<=0;
white[ 3147 ]<=0;
white[ 3148 ]<=0;
white[ 3149 ]<=0;
white[ 3150 ]<=0;
white[ 3151 ]<=0;
white[ 3152 ]<=0;
white[ 3153 ]<=0;
white[ 3154 ]<=0;
white[ 3155 ]<=0;
white[ 3156 ]<=0;
white[ 3157 ]<=0;
white[ 3158 ]<=0;
white[ 3159 ]<=0;
white[ 3160 ]<=0;
white[ 3161 ]<=0;
white[ 3162 ]<=1;
white[ 3163 ]<=1;
white[ 3164 ]<=1;
white[ 3165 ]<=1;
white[ 3166 ]<=1;
white[ 3167 ]<=1;
white[ 3168 ]<=1;
white[ 3169 ]<=0;
white[ 3170 ]<=0;
white[ 3171 ]<=0;
white[ 3172 ]<=0;
```

```
white[ 3173 ]<=0;
white[ 3174 ]<=0;
white[ 3175 ]<=0;
white[ 3176 ]<=0;
white[ 3177 ]<=0;
white[ 3178 ]<=0;
white[ 3179 ]<=0;
white[ 3180 ]<=0;
white[ 3181 ]<=0;
white[ 3182 ]<=0;
white[ 3183 ]<=0;
white[ 3184 ]<=0;
white[ 3185 ]<=0;
white[ 3186 ]<=0;
white[ 3187 ]<=0;
white[ 3188 ]<=0;
white[ 3189 ]<=0;
white[ 3190 ]<=0;
white[ 3191 ]<=0;
white[ 3192 ]<=1;
white[ 3193 ]<=1;
white[ 3194 ]<=1;
white[ 3195 ]<=1;
white[ 3196 ]<=1;
white[ 3197 ]<=1;
white[ 3198 ]<=1;
white[ 3199 ]<=1;
white[ 3200 ]<=0;
white[ 3201 ]<=0;
white[ 3202 ]<=0;
white[ 3203 ]<=0;
white[ 3204 ]<=0;
white[ 3205 ]<=0;
white[ 3206 ]<=0;
white[ 3207 ]<=0;
white[ 3208 ]<=0;
white[ 3209 ]<=0;
white[ 3210 ]<=0;
white[ 3211 ]<=0;
white[ 3212 ]<=0;
white[ 3213 ]<=0;
white[ 3214 ]<=0;
white[ 3215 ]<=0;
white[ 3216 ]<=0;
white[ 3217 ]<=0;
white[ 3218 ]<=0;
white[ 3219 ]<=0;
white[ 3220 ]<=1;
white[ 3221 ]<=1;
white[ 3222 ]<=1;
white[ 3223 ]<=1;
white[ 3224 ]<=1;
white[ 3225 ]<=1;
white[ 3226 ]<=1;
white[ 3227 ]<=1;
white[ 3228 ]<=0;
white[ 3229 ]<=0;
white[ 3230 ]<=0;
```

```
white[ 3231 ]<=0;
white[ 3232 ]<=0;
white[ 3233 ]<=0;
white[ 3234 ]<=0;
white[ 3235 ]<=0;
white[ 3236 ]<=0;
white[ 3237 ]<=0;
white[ 3238 ]<=0;
white[ 3239 ]<=0;
white[ 3240 ]<=0;
white[ 3241 ]<=0;
white[ 3242 ]<=0;
white[ 3243 ]<=0;
white[ 3244 ]<=0;
white[ 3245 ]<=0;
white[ 3246 ]<=0;
white[ 3247 ]<=0;
white[ 3248 ]<=0;
white[ 3249 ]<=0;
white[ 3250 ]<=0;
white[ 3251 ]<=0;
white[ 3252 ]<=0;
white[ 3253 ]<=1;
white[ 3254 ]<=1;
white[ 3255 ]<=1;
white[ 3256 ]<=1;
white[ 3257 ]<=1;
white[ 3258 ]<=1;
white[ 3259 ]<=1;
white[ 3260 ]<=1;
white[ 3261 ]<=1;
white[ 3262 ]<=0;
white[ 3263 ]<=0;
white[ 3264 ]<=0;
white[ 3265 ]<=0;
white[ 3266 ]<=0;
white[ 3267 ]<=0;
white[ 3268 ]<=0;
white[ 3269 ]<=0;
white[ 3270 ]<=0;
white[ 3271 ]<=0;
white[ 3272 ]<=0;
white[ 3273 ]<=0;
white[ 3274 ]<=0;
white[ 3275 ]<=0;
white[ 3276 ]<=0;
white[ 3277 ]<=0;
white[ 3278 ]<=1;
white[ 3279 ]<=1;
white[ 3280 ]<=1;
white[ 3281 ]<=1;
white[ 3282 ]<=1;
white[ 3283 ]<=1;
white[ 3284 ]<=1;
white[ 3285 ]<=1;
white[ 3286 ]<=1;
white[ 3287 ]<=0;
white[ 3288 ]<=0;
```

```
white[ 3289 ]<=0;
white[ 3290 ]<=0;
white[ 3291 ]<=0;
white[ 3292 ]<=0;
white[ 3293 ]<=0;
white[ 3294 ]<=0;
white[ 3295 ]<=0;
white[ 3296 ]<=0;
white[ 3297 ]<=0;
white[ 3298 ]<=0;
white[ 3299 ]<=0;
white[ 3300 ]<=0;
white[ 3301 ]<=0;
white[ 3302 ]<=0;
white[ 3303 ]<=0;
white[ 3304 ]<=0;
white[ 3305 ]<=0;
white[ 3306 ]<=0;
white[ 3307 ]<=0;
white[ 3308 ]<=0;
white[ 3309 ]<=0;
white[ 3310 ]<=0;
white[ 3311 ]<=0;
white[ 3312 ]<=0;
white[ 3313 ]<=0;
white[ 3314 ]<=0;
white[ 3315 ]<=1;
white[ 3316 ]<=1;
white[ 3317 ]<=1;
white[ 3318 ]<=1;
white[ 3319 ]<=1;
white[ 3320 ]<=1;
white[ 3321 ]<=1;
white[ 3322 ]<=1;
white[ 3323 ]<=1;
white[ 3324 ]<=1;
white[ 3325 ]<=0;
white[ 3326 ]<=0;
white[ 3327 ]<=0;
white[ 3328 ]<=0;
white[ 3329 ]<=0;
white[ 3330 ]<=0;
white[ 3331 ]<=0;
white[ 3332 ]<=0;
white[ 3333 ]<=0;
white[ 3334 ]<=0;
white[ 3335 ]<=1;
white[ 3336 ]<=1;
white[ 3337 ]<=1;
white[ 3338 ]<=1;
white[ 3339 ]<=1;
white[ 3340 ]<=1;
white[ 3341 ]<=1;
white[ 3342 ]<=1;
white[ 3343 ]<=1;
white[ 3344 ]<=1;
white[ 3345 ]<=0;
white[ 3346 ]<=0;
```

```
white[ 3347 ]<=0;
white[ 3348 ]<=0;
white[ 3349 ]<=0;
white[ 3350 ]<=0;
white[ 3351 ]<=0;
white[ 3352 ]<=0;
white[ 3353 ]<=0;
white[ 3354 ]<=0;
white[ 3355 ]<=0;
white[ 3356 ]<=0;
white[ 3357 ]<=0;
white[ 3358 ]<=0;
white[ 3359 ]<=0;
white[ 3360 ]<=0;
white[ 3361 ]<=0;
white[ 3362 ]<=0;
white[ 3363 ]<=0;
white[ 3364 ]<=0;
white[ 3365 ]<=0;
white[ 3366 ]<=0;
white[ 3367 ]<=0;
white[ 3368 ]<=0;
white[ 3369 ]<=0;
white[ 3370 ]<=0;
white[ 3371 ]<=0;
white[ 3372 ]<=0;
white[ 3373 ]<=0;
white[ 3374 ]<=0;
white[ 3375 ]<=0;
white[ 3376 ]<=1;
white[ 3377 ]<=1;
white[ 3378 ]<=1;
white[ 3379 ]<=1;
white[ 3380 ]<=1;
white[ 3381 ]<=1;
white[ 3382 ]<=1;
white[ 3383 ]<=1;
white[ 3384 ]<=1;
white[ 3385 ]<=1;
white[ 3386 ]<=1;
white[ 3387 ]<=1;
white[ 3388 ]<=1;
white[ 3389 ]<=1;
white[ 3390 ]<=1;
white[ 3391 ]<=1;
white[ 3392 ]<=1;
white[ 3393 ]<=1;
white[ 3394 ]<=1;
white[ 3395 ]<=1;
white[ 3396 ]<=1;
white[ 3397 ]<=1;
white[ 3398 ]<=1;
white[ 3399 ]<=1;
white[ 3400 ]<=1;
white[ 3401 ]<=1;
white[ 3402 ]<=1;
white[ 3403 ]<=1;
white[ 3404 ]<=0;
```

```
white[ 3405 ]<=0;
white[ 3406 ]<=0;
white[ 3407 ]<=0;
white[ 3408 ]<=0;
white[ 3409 ]<=0;
white[ 3410 ]<=0;
white[ 3411 ]<=0;
white[ 3412 ]<=0;
white[ 3413 ]<=0;
white[ 3414 ]<=0;
white[ 3415 ]<=0;
white[ 3416 ]<=0;
white[ 3417 ]<=0;
white[ 3418 ]<=0;
white[ 3419 ]<=0;
white[ 3420 ]<=0;
white[ 3421 ]<=0;
white[ 3422 ]<=0;
white[ 3423 ]<=0;
white[ 3424 ]<=0;
white[ 3425 ]<=0;
white[ 3426 ]<=0;
white[ 3427 ]<=0;
white[ 3428 ]<=0;
white[ 3429 ]<=0;
white[ 3430 ]<=0;
white[ 3431 ]<=0;
white[ 3432 ]<=0;
white[ 3433 ]<=0;
white[ 3434 ]<=0;
white[ 3435 ]<=0;
white[ 3436 ]<=0;
white[ 3437 ]<=0;
white[ 3438 ]<=1;
white[ 3439 ]<=1;
white[ 3440 ]<=1;
white[ 3441 ]<=1;
white[ 3442 ]<=1;
white[ 3443 ]<=1;
white[ 3444 ]<=1;
white[ 3445 ]<=1;
white[ 3446 ]<=1;
white[ 3447 ]<=1;
white[ 3448 ]<=1;
white[ 3449 ]<=1;
white[ 3450 ]<=1;
white[ 3451 ]<=1;
white[ 3452 ]<=1;
white[ 3453 ]<=1;
white[ 3454 ]<=1;
white[ 3455 ]<=1;
white[ 3456 ]<=1;
white[ 3457 ]<=1;
white[ 3458 ]<=1;
white[ 3459 ]<=1;
white[ 3460 ]<=1;
white[ 3461 ]<=1;
white[ 3462 ]<=0;
```

```
white[ 3463 ]<=0;
white[ 3464 ]<=0;
white[ 3465 ]<=0;
white[ 3466 ]<=0;
white[ 3467 ]<=0;
white[ 3468 ]<=0;
white[ 3469 ]<=0;
white[ 3470 ]<=0;
white[ 3471 ]<=0;
white[ 3472 ]<=0;
white[ 3473 ]<=0;
white[ 3474 ]<=0;
white[ 3475 ]<=0;
white[ 3476 ]<=0;
white[ 3477 ]<=0;
white[ 3478 ]<=0;
white[ 3479 ]<=0;
white[ 3480 ]<=0;
white[ 3481 ]<=0;
white[ 3482 ]<=0;
white[ 3483 ]<=0;
white[ 3484 ]<=0;
white[ 3485 ]<=0;
white[ 3486 ]<=0;
white[ 3487 ]<=0;
white[ 3488 ]<=0;
white[ 3489 ]<=0;
white[ 3490 ]<=0;
white[ 3491 ]<=0;
white[ 3492 ]<=0;
white[ 3493 ]<=0;
white[ 3494 ]<=0;
white[ 3495 ]<=0;
white[ 3496 ]<=0;
white[ 3497 ]<=0;
white[ 3498 ]<=0;
white[ 3499 ]<=0;
white[ 3500 ]<=1;
white[ 3501 ]<=1;
white[ 3502 ]<=1;
white[ 3503 ]<=1;
white[ 3504 ]<=1;
white[ 3505 ]<=1;
white[ 3506 ]<=1;
white[ 3507 ]<=1;
white[ 3508 ]<=1;
white[ 3509 ]<=1;
white[ 3510 ]<=1;
white[ 3511 ]<=1;
white[ 3512 ]<=1;
white[ 3513 ]<=1;
white[ 3514 ]<=1;
white[ 3515 ]<=1;
white[ 3516 ]<=1;
white[ 3517 ]<=1;
white[ 3518 ]<=1;
white[ 3519 ]<=1;
white[ 3520 ]<=0;
```

```
white[ 3521 ]<=0;
white[ 3522 ]<=0;
white[ 3523 ]<=0;
white[ 3524 ]<=0;
white[ 3525 ]<=0;
white[ 3526 ]<=0;
white[ 3527 ]<=0;
white[ 3528 ]<=0;
white[ 3529 ]<=0;
white[ 3530 ]<=0;
white[ 3531 ]<=0;
white[ 3532 ]<=0;
white[ 3533 ]<=0;
white[ 3534 ]<=0;
white[ 3535 ]<=0;
white[ 3536 ]<=0;
white[ 3537 ]<=0;
white[ 3538 ]<=0;
white[ 3539 ]<=0;
white[ 3540 ]<=0;
white[ 3541 ]<=0;
white[ 3542 ]<=0;
white[ 3543 ]<=0;
white[ 3544 ]<=0;
white[ 3545 ]<=0;
white[ 3546 ]<=0;
white[ 3547 ]<=0;
white[ 3548 ]<=0;
white[ 3549 ]<=0;
white[ 3550 ]<=0;
white[ 3551 ]<=0;
white[ 3552 ]<=0;
white[ 3553 ]<=0;
white[ 3554 ]<=0;
white[ 3555 ]<=0;
white[ 3556 ]<=0;
white[ 3557 ]<=0;
white[ 3558 ]<=0;
white[ 3559 ]<=0;
white[ 3560 ]<=0;
white[ 3561 ]<=0;
white[ 3562 ]<=1;
white[ 3563 ]<=1;
white[ 3564 ]<=1;
white[ 3565 ]<=1;
white[ 3566 ]<=1;
white[ 3567 ]<=1;
white[ 3568 ]<=1;
white[ 3569 ]<=1;
white[ 3570 ]<=1;
white[ 3571 ]<=1;
white[ 3572 ]<=1;
white[ 3573 ]<=1;
white[ 3574 ]<=1;
white[ 3575 ]<=1;
white[ 3576 ]<=1;
white[ 3577 ]<=1;
white[ 3578 ]<=0;
```

```
white[ 3579 ]<=0;
white[ 3580 ]<=0;
white[ 3581 ]<=0;
white[ 3582 ]<=0;
white[ 3583 ]<=0;
white[ 3584 ]<=0;
white[ 3585 ]<=0;
white[ 3586 ]<=0;
white[ 3587 ]<=0;
white[ 3588 ]<=0;
white[ 3589 ]<=0;
white[ 3590 ]<=0;
white[ 3591 ]<=0;
white[ 3592 ]<=0;
white[ 3593 ]<=0;
white[ 3594 ]<=0;
white[ 3595 ]<=0;
white[ 3596 ]<=0;
white[ 3597 ]<=0;
white[ 3598 ]<=0;
white[ 3599 ]<=0;
end
//heart
18'd92639: begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
```

```
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=0;
white[ 86 ]<=0;
white[ 87 ]<=0;
white[ 88 ]<=0;
white[ 89 ]<=0;
white[ 90 ]<=0;
white[ 91 ]<=0;
```

```
white[ 92 ]<=0;
white[ 93 ]<=0;
white[ 94 ]<=0;
white[ 95 ]<=0;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=0;
white[ 144 ]<=0;
white[ 145 ]<=0;
white[ 146 ]<=0;
white[ 147 ]<=0;
white[ 148 ]<=0;
white[ 149 ]<=0;
```

```
white[ 150 ]<=0;
white[ 151 ]<=0;
white[ 152 ]<=0;
white[ 153 ]<=0;
white[ 154 ]<=0;
white[ 155 ]<=0;
white[ 156 ]<=0;
white[ 157 ]<=0;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=0;
white[ 168 ]<=0;
white[ 169 ]<=0;
white[ 170 ]<=0;
white[ 171 ]<=0;
white[ 172 ]<=0;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=0;
white[ 180 ]<=0;
white[ 181 ]<=0;
white[ 182 ]<=0;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=1;
white[ 192 ]<=1;
white[ 193 ]<=1;
white[ 194 ]<=1;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=0;
white[ 198 ]<=0;
white[ 199 ]<=0;
white[ 200 ]<=0;
white[ 201 ]<=0;
white[ 202 ]<=0;
white[ 203 ]<=0;
white[ 204 ]<=0;
white[ 205 ]<=0;
white[ 206 ]<=0;
white[ 207 ]<=0;
```

```
white[ 208 ]<=0;
white[ 209 ]<=0;
white[ 210 ]<=0;
white[ 211 ]<=0;
white[ 212 ]<=0;
white[ 213 ]<=0;
white[ 214 ]<=0;
white[ 215 ]<=0;
white[ 216 ]<=0;
white[ 217 ]<=0;
white[ 218 ]<=0;
white[ 219 ]<=0;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=1;
white[ 224 ]<=1;
white[ 225 ]<=1;
white[ 226 ]<=1;
white[ 227 ]<=1;
white[ 228 ]<=1;
white[ 229 ]<=0;
white[ 230 ]<=0;
white[ 231 ]<=0;
white[ 232 ]<=0;
white[ 233 ]<=0;
white[ 234 ]<=0;
white[ 235 ]<=0;
white[ 236 ]<=0;
white[ 237 ]<=0;
white[ 238 ]<=0;
white[ 239 ]<=0;
white[ 240 ]<=0;
white[ 241 ]<=0;
white[ 242 ]<=0;
white[ 243 ]<=0;
white[ 244 ]<=0;
white[ 245 ]<=0;
white[ 246 ]<=0;
white[ 247 ]<=0;
white[ 248 ]<=1;
white[ 249 ]<=1;
white[ 250 ]<=1;
white[ 251 ]<=1;
white[ 252 ]<=1;
white[ 253 ]<=1;
white[ 254 ]<=1;
white[ 255 ]<=1;
white[ 256 ]<=1;
white[ 257 ]<=1;
white[ 258 ]<=1;
white[ 259 ]<=1;
white[ 260 ]<=0;
white[ 261 ]<=0;
white[ 262 ]<=0;
white[ 263 ]<=0;
white[ 264 ]<=0;
white[ 265 ]<=0;
```

```
white[ 266 ]<=0;
white[ 267 ]<=0;
white[ 268 ]<=0;
white[ 269 ]<=0;
white[ 270 ]<=0;
white[ 271 ]<=0;
white[ 272 ]<=0;
white[ 273 ]<=0;
white[ 274 ]<=0;
white[ 275 ]<=0;
white[ 276 ]<=0;
white[ 277 ]<=0;
white[ 278 ]<=0;
white[ 279 ]<=0;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=1;
white[ 287 ]<=1;
white[ 288 ]<=1;
white[ 289 ]<=1;
white[ 290 ]<=1;
white[ 291 ]<=1;
white[ 292 ]<=0;
white[ 293 ]<=0;
white[ 294 ]<=0;
white[ 295 ]<=0;
white[ 296 ]<=0;
white[ 297 ]<=0;
white[ 298 ]<=0;
white[ 299 ]<=0;
white[ 300 ]<=0;
white[ 301 ]<=0;
white[ 302 ]<=0;
white[ 303 ]<=0;
white[ 304 ]<=0;
white[ 305 ]<=0;
white[ 306 ]<=1;
white[ 307 ]<=1;
white[ 308 ]<=1;
white[ 309 ]<=1;
white[ 310 ]<=1;
white[ 311 ]<=1;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
white[ 317 ]<=1;
white[ 318 ]<=1;
white[ 319 ]<=1;
white[ 320 ]<=1;
white[ 321 ]<=1;
white[ 322 ]<=0;
white[ 323 ]<=0;
```

```
white[ 324 ]<=0;
white[ 325 ]<=0;
white[ 326 ]<=0;
white[ 327 ]<=0;
white[ 328 ]<=0;
white[ 329 ]<=0;
white[ 330 ]<=0;
white[ 331 ]<=0;
white[ 332 ]<=0;
white[ 333 ]<=0;
white[ 334 ]<=0;
white[ 335 ]<=0;
white[ 336 ]<=0;
white[ 337 ]<=0;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=1;
white[ 348 ]<=1;
white[ 349 ]<=1;
white[ 350 ]<=1;
white[ 351 ]<=1;
white[ 352 ]<=1;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=0;
white[ 358 ]<=0;
white[ 359 ]<=0;
white[ 360 ]<=0;
white[ 361 ]<=0;
white[ 362 ]<=0;
white[ 363 ]<=0;
white[ 364 ]<=0;
white[ 365 ]<=1;
white[ 366 ]<=1;
white[ 367 ]<=1;
white[ 368 ]<=1;
white[ 369 ]<=1;
white[ 370 ]<=1;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
```

```
white[ 382 ]<=1;
white[ 383 ]<=1;
white[ 384 ]<=0;
white[ 385 ]<=0;
white[ 386 ]<=0;
white[ 387 ]<=0;
white[ 388 ]<=0;
white[ 389 ]<=0;
white[ 390 ]<=0;
white[ 391 ]<=0;
white[ 392 ]<=0;
white[ 393 ]<=0;
white[ 394 ]<=0;
white[ 395 ]<=0;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=1;
white[ 410 ]<=1;
white[ 411 ]<=1;
white[ 412 ]<=1;
white[ 413 ]<=1;
white[ 414 ]<=1;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=0;
white[ 423 ]<=0;
white[ 424 ]<=1;
white[ 425 ]<=1;
white[ 426 ]<=1;
white[ 427 ]<=1;
white[ 428 ]<=1;
white[ 429 ]<=1;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
```

```
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=0;
white[ 446 ]<=0;
white[ 447 ]<=0;
white[ 448 ]<=0;
white[ 449 ]<=0;
white[ 450 ]<=0;
white[ 451 ]<=0;
white[ 452 ]<=0;
white[ 453 ]<=0;
white[ 454 ]<=0;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=1;
white[ 471 ]<=1;
white[ 472 ]<=1;
white[ 473 ]<=1;
white[ 474 ]<=1;
white[ 475 ]<=1;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=1;
white[ 484 ]<=1;
white[ 485 ]<=1;
white[ 486 ]<=1;
white[ 487 ]<=1;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
```

```
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=0;
white[ 507 ]<=0;
white[ 508 ]<=0;
white[ 509 ]<=0;
white[ 510 ]<=0;
white[ 511 ]<=0;
white[ 512 ]<=0;
white[ 513 ]<=0;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=1;
white[ 532 ]<=1;
white[ 533 ]<=1;
white[ 534 ]<=1;
white[ 535 ]<=1;
white[ 536 ]<=1;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=1;
white[ 544 ]<=1;
white[ 545 ]<=1;
white[ 546 ]<=1;
white[ 547 ]<=1;
white[ 548 ]<=1;
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
```

```
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=0;
white[ 568 ]<=0;
white[ 569 ]<=0;
white[ 570 ]<=0;
white[ 571 ]<=0;
white[ 572 ]<=0;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=1;
white[ 576 ]<=1;
white[ 577 ]<=1;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=1;
white[ 593 ]<=1;
white[ 594 ]<=1;
white[ 595 ]<=1;
white[ 596 ]<=1;
white[ 597 ]<=0;
white[ 598 ]<=0;
white[ 599 ]<=0;
white[ 600 ]<=0;
white[ 601 ]<=0;
white[ 602 ]<=1;
white[ 603 ]<=1;
white[ 604 ]<=1;
white[ 605 ]<=1;
white[ 606 ]<=1;
white[ 607 ]<=1;
white[ 608 ]<=1;
white[ 609 ]<=1;
white[ 610 ]<=1;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
```

```
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=0;
white[ 629 ]<=0;
white[ 630 ]<=0;
white[ 631 ]<=0;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=1;
white[ 640 ]<=1;
white[ 641 ]<=1;
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=1;
white[ 654 ]<=1;
white[ 655 ]<=1;
white[ 656 ]<=1;
white[ 657 ]<=1;
white[ 658 ]<=0;
white[ 659 ]<=0;
white[ 660 ]<=0;
white[ 661 ]<=0;
white[ 662 ]<=1;
white[ 663 ]<=1;
white[ 664 ]<=1;
white[ 665 ]<=1;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=1;
white[ 671 ]<=1;
```

```
white[ 672 ]<=1;
white[ 673 ]<=1;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=0;
white[ 690 ]<=0;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=1;
white[ 703 ]<=1;
white[ 704 ]<=1;
white[ 705 ]<=1;
white[ 706 ]<=1;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=1;
white[ 715 ]<=1;
white[ 716 ]<=1;
white[ 717 ]<=1;
white[ 718 ]<=0;
white[ 719 ]<=0;
white[ 720 ]<=0;
white[ 721 ]<=0;
white[ 722 ]<=1;
white[ 723 ]<=1;
white[ 724 ]<=1;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
```

```
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=1;
white[ 735 ]<=1;
white[ 736 ]<=1;
white[ 737 ]<=1;
white[ 738 ]<=1;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=1;
white[ 768 ]<=1;
white[ 769 ]<=1;
white[ 770 ]<=1;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=1;
white[ 775 ]<=1;
white[ 776 ]<=1;
white[ 777 ]<=1;
white[ 778 ]<=0;
white[ 779 ]<=0;
white[ 780 ]<=0;
white[ 781 ]<=1;
white[ 782 ]<=1;
white[ 783 ]<=1;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
```

```
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=1;
white[ 800 ]<=1;
white[ 801 ]<=1;
white[ 802 ]<=1;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=1;
white[ 832 ]<=1;
white[ 833 ]<=1;
white[ 834 ]<=1;
white[ 835 ]<=1;
white[ 836 ]<=1;
white[ 837 ]<=1;
white[ 838 ]<=0;
white[ 839 ]<=0;
white[ 840 ]<=0;
white[ 841 ]<=1;
white[ 842 ]<=1;
white[ 843 ]<=1;
white[ 844 ]<=1;
white[ 845 ]<=1;
```

```
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=1;
white[ 863 ]<=1;
white[ 864 ]<=1;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=1;
white[ 895 ]<=1;
white[ 896 ]<=1;
white[ 897 ]<=1;
white[ 898 ]<=1;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=1;
white[ 902 ]<=1;
white[ 903 ]<=1;
```

```
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=1;
white[ 927 ]<=1;
white[ 928 ]<=1;
white[ 929 ]<=1;
white[ 930 ]<=1;
white[ 931 ]<=1;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=1;
white[ 955 ]<=1;
white[ 956 ]<=1;
white[ 957 ]<=1;
white[ 958 ]<=1;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=1;
```

```
white[ 962 ]<=1;
white[ 963 ]<=1;
white[ 964 ]<=1;
white[ 965 ]<=1;
white[ 966 ]<=1;
white[ 967 ]<=1;
white[ 968 ]<=1;
white[ 969 ]<=1;
white[ 970 ]<=1;
white[ 971 ]<=1;
white[ 972 ]<=1;
white[ 973 ]<=1;
white[ 974 ]<=1;
white[ 975 ]<=1;
white[ 976 ]<=1;
white[ 977 ]<=1;
white[ 978 ]<=1;
white[ 979 ]<=1;
white[ 980 ]<=1;
white[ 981 ]<=1;
white[ 982 ]<=1;
white[ 983 ]<=1;
white[ 984 ]<=1;
white[ 985 ]<=1;
white[ 986 ]<=1;
white[ 987 ]<=1;
white[ 988 ]<=1;
white[ 989 ]<=1;
white[ 990 ]<=1;
white[ 991 ]<=1;
white[ 992 ]<=1;
white[ 993 ]<=1;
white[ 994 ]<=1;
white[ 995 ]<=1;
white[ 996 ]<=1;
white[ 997 ]<=1;
white[ 998 ]<=1;
white[ 999 ]<=1;
white[ 1000 ]<=1;
white[ 1001 ]<=1;
white[ 1002 ]<=1;
white[ 1003 ]<=1;
white[ 1004 ]<=1;
white[ 1005 ]<=1;
white[ 1006 ]<=1;
white[ 1007 ]<=1;
white[ 1008 ]<=1;
white[ 1009 ]<=1;
white[ 1010 ]<=1;
white[ 1011 ]<=1;
white[ 1012 ]<=1;
white[ 1013 ]<=1;
white[ 1014 ]<=1;
white[ 1015 ]<=1;
white[ 1016 ]<=1;
white[ 1017 ]<=1;
white[ 1018 ]<=1;
white[ 1019 ]<=0;
```

```
white[ 1020 ]<=0;
white[ 1021 ]<=1;
white[ 1022 ]<=1;
white[ 1023 ]<=1;
white[ 1024 ]<=1;
white[ 1025 ]<=1;
white[ 1026 ]<=1;
white[ 1027 ]<=1;
white[ 1028 ]<=1;
white[ 1029 ]<=1;
white[ 1030 ]<=1;
white[ 1031 ]<=1;
white[ 1032 ]<=1;
white[ 1033 ]<=1;
white[ 1034 ]<=1;
white[ 1035 ]<=1;
white[ 1036 ]<=1;
white[ 1037 ]<=1;
white[ 1038 ]<=1;
white[ 1039 ]<=1;
white[ 1040 ]<=1;
white[ 1041 ]<=1;
white[ 1042 ]<=1;
white[ 1043 ]<=1;
white[ 1044 ]<=1;
white[ 1045 ]<=1;
white[ 1046 ]<=1;
white[ 1047 ]<=1;
white[ 1048 ]<=1;
white[ 1049 ]<=1;
white[ 1050 ]<=1;
white[ 1051 ]<=1;
white[ 1052 ]<=1;
white[ 1053 ]<=1;
white[ 1054 ]<=1;
white[ 1055 ]<=1;
white[ 1056 ]<=1;
white[ 1057 ]<=1;
white[ 1058 ]<=1;
white[ 1059 ]<=1;
white[ 1060 ]<=1;
white[ 1061 ]<=1;
white[ 1062 ]<=1;
white[ 1063 ]<=1;
white[ 1064 ]<=1;
white[ 1065 ]<=1;
white[ 1066 ]<=1;
white[ 1067 ]<=1;
white[ 1068 ]<=1;
white[ 1069 ]<=1;
white[ 1070 ]<=1;
white[ 1071 ]<=1;
white[ 1072 ]<=1;
white[ 1073 ]<=1;
white[ 1074 ]<=1;
white[ 1075 ]<=1;
white[ 1076 ]<=1;
white[ 1077 ]<=1;
```

```
white[ 1078 ]<=1;
white[ 1079 ]<=0;
white[ 1080 ]<=0;
white[ 1081 ]<=1;
white[ 1082 ]<=1;
white[ 1083 ]<=1;
white[ 1084 ]<=1;
white[ 1085 ]<=1;
white[ 1086 ]<=1;
white[ 1087 ]<=1;
white[ 1088 ]<=1;
white[ 1089 ]<=1;
white[ 1090 ]<=1;
white[ 1091 ]<=1;
white[ 1092 ]<=1;
white[ 1093 ]<=1;
white[ 1094 ]<=1;
white[ 1095 ]<=1;
white[ 1096 ]<=1;
white[ 1097 ]<=1;
white[ 1098 ]<=1;
white[ 1099 ]<=1;
white[ 1100 ]<=1;
white[ 1101 ]<=1;
white[ 1102 ]<=1;
white[ 1103 ]<=1;
white[ 1104 ]<=1;
white[ 1105 ]<=1;
white[ 1106 ]<=1;
white[ 1107 ]<=1;
white[ 1108 ]<=1;
white[ 1109 ]<=1;
white[ 1110 ]<=1;
white[ 1111 ]<=1;
white[ 1112 ]<=1;
white[ 1113 ]<=1;
white[ 1114 ]<=1;
white[ 1115 ]<=1;
white[ 1116 ]<=1;
white[ 1117 ]<=1;
white[ 1118 ]<=1;
white[ 1119 ]<=1;
white[ 1120 ]<=1;
white[ 1121 ]<=1;
white[ 1122 ]<=1;
white[ 1123 ]<=1;
white[ 1124 ]<=1;
white[ 1125 ]<=1;
white[ 1126 ]<=1;
white[ 1127 ]<=1;
white[ 1128 ]<=1;
white[ 1129 ]<=1;
white[ 1130 ]<=1;
white[ 1131 ]<=1;
white[ 1132 ]<=1;
white[ 1133 ]<=1;
white[ 1134 ]<=1;
white[ 1135 ]<=1;
```

```
white[ 1136 ]<=1;
white[ 1137 ]<=1;
white[ 1138 ]<=0;
white[ 1139 ]<=0;
white[ 1140 ]<=0;
white[ 1141 ]<=0;
white[ 1142 ]<=1;
white[ 1143 ]<=1;
white[ 1144 ]<=1;
white[ 1145 ]<=1;
white[ 1146 ]<=1;
white[ 1147 ]<=1;
white[ 1148 ]<=1;
white[ 1149 ]<=1;
white[ 1150 ]<=1;
white[ 1151 ]<=1;
white[ 1152 ]<=1;
white[ 1153 ]<=1;
white[ 1154 ]<=1;
white[ 1155 ]<=1;
white[ 1156 ]<=1;
white[ 1157 ]<=1;
white[ 1158 ]<=1;
white[ 1159 ]<=1;
white[ 1160 ]<=1;
white[ 1161 ]<=1;
white[ 1162 ]<=1;
white[ 1163 ]<=1;
white[ 1164 ]<=1;
white[ 1165 ]<=1;
white[ 1166 ]<=1;
white[ 1167 ]<=1;
white[ 1168 ]<=1;
white[ 1169 ]<=1;
white[ 1170 ]<=1;
white[ 1171 ]<=1;
white[ 1172 ]<=1;
white[ 1173 ]<=1;
white[ 1174 ]<=1;
white[ 1175 ]<=1;
white[ 1176 ]<=1;
white[ 1177 ]<=1;
white[ 1178 ]<=1;
white[ 1179 ]<=1;
white[ 1180 ]<=1;
white[ 1181 ]<=1;
white[ 1182 ]<=1;
white[ 1183 ]<=1;
white[ 1184 ]<=1;
white[ 1185 ]<=1;
white[ 1186 ]<=1;
white[ 1187 ]<=1;
white[ 1188 ]<=1;
white[ 1189 ]<=1;
white[ 1190 ]<=1;
white[ 1191 ]<=1;
white[ 1192 ]<=1;
white[ 1193 ]<=1;
```

```
white[ 1194 ]<=1;
white[ 1195 ]<=1;
white[ 1196 ]<=1;
white[ 1197 ]<=1;
white[ 1198 ]<=0;
white[ 1199 ]<=0;
white[ 1200 ]<=0;
white[ 1201 ]<=0;
white[ 1202 ]<=1;
white[ 1203 ]<=1;
white[ 1204 ]<=1;
white[ 1205 ]<=1;
white[ 1206 ]<=1;
white[ 1207 ]<=1;
white[ 1208 ]<=1;
white[ 1209 ]<=1;
white[ 1210 ]<=1;
white[ 1211 ]<=1;
white[ 1212 ]<=1;
white[ 1213 ]<=1;
white[ 1214 ]<=1;
white[ 1215 ]<=1;
white[ 1216 ]<=1;
white[ 1217 ]<=1;
white[ 1218 ]<=1;
white[ 1219 ]<=1;
white[ 1220 ]<=1;
white[ 1221 ]<=1;
white[ 1222 ]<=1;
white[ 1223 ]<=1;
white[ 1224 ]<=1;
white[ 1225 ]<=1;
white[ 1226 ]<=1;
white[ 1227 ]<=1;
white[ 1228 ]<=1;
white[ 1229 ]<=1;
white[ 1230 ]<=1;
white[ 1231 ]<=1;
white[ 1232 ]<=1;
white[ 1233 ]<=1;
white[ 1234 ]<=1;
white[ 1235 ]<=1;
white[ 1236 ]<=1;
white[ 1237 ]<=1;
white[ 1238 ]<=1;
white[ 1239 ]<=1;
white[ 1240 ]<=1;
white[ 1241 ]<=1;
white[ 1242 ]<=1;
white[ 1243 ]<=1;
white[ 1244 ]<=1;
white[ 1245 ]<=1;
white[ 1246 ]<=1;
white[ 1247 ]<=1;
white[ 1248 ]<=1;
white[ 1249 ]<=1;
white[ 1250 ]<=1;
white[ 1251 ]<=1;
```

```
white[ 1252 ]<=1;
white[ 1253 ]<=1;
white[ 1254 ]<=1;
white[ 1255 ]<=1;
white[ 1256 ]<=1;
white[ 1257 ]<=1;
white[ 1258 ]<=0;
white[ 1259 ]<=0;
white[ 1260 ]<=0;
white[ 1261 ]<=0;
white[ 1262 ]<=1;
white[ 1263 ]<=1;
white[ 1264 ]<=1;
white[ 1265 ]<=1;
white[ 1266 ]<=1;
white[ 1267 ]<=1;
white[ 1268 ]<=1;
white[ 1269 ]<=1;
white[ 1270 ]<=1;
white[ 1271 ]<=1;
white[ 1272 ]<=1;
white[ 1273 ]<=1;
white[ 1274 ]<=1;
white[ 1275 ]<=1;
white[ 1276 ]<=1;
white[ 1277 ]<=1;
white[ 1278 ]<=1;
white[ 1279 ]<=1;
white[ 1280 ]<=1;
white[ 1281 ]<=1;
white[ 1282 ]<=1;
white[ 1283 ]<=1;
white[ 1284 ]<=1;
white[ 1285 ]<=1;
white[ 1286 ]<=1;
white[ 1287 ]<=1;
white[ 1288 ]<=1;
white[ 1289 ]<=1;
white[ 1290 ]<=1;
white[ 1291 ]<=1;
white[ 1292 ]<=1;
white[ 1293 ]<=1;
white[ 1294 ]<=1;
white[ 1295 ]<=1;
white[ 1296 ]<=1;
white[ 1297 ]<=1;
white[ 1298 ]<=1;
white[ 1299 ]<=1;
white[ 1300 ]<=1;
white[ 1301 ]<=1;
white[ 1302 ]<=1;
white[ 1303 ]<=1;
white[ 1304 ]<=1;
white[ 1305 ]<=1;
white[ 1306 ]<=1;
white[ 1307 ]<=1;
white[ 1308 ]<=1;
white[ 1309 ]<=1;
```

```
white[ 1310 ]<=1;
white[ 1311 ]<=1;
white[ 1312 ]<=1;
white[ 1313 ]<=1;
white[ 1314 ]<=1;
white[ 1315 ]<=1;
white[ 1316 ]<=1;
white[ 1317 ]<=1;
white[ 1318 ]<=0;
white[ 1319 ]<=0;
white[ 1320 ]<=0;
white[ 1321 ]<=0;
white[ 1322 ]<=1;
white[ 1323 ]<=1;
white[ 1324 ]<=1;
white[ 1325 ]<=1;
white[ 1326 ]<=1;
white[ 1327 ]<=1;
white[ 1328 ]<=1;
white[ 1329 ]<=1;
white[ 1330 ]<=1;
white[ 1331 ]<=1;
white[ 1332 ]<=1;
white[ 1333 ]<=1;
white[ 1334 ]<=1;
white[ 1335 ]<=1;
white[ 1336 ]<=1;
white[ 1337 ]<=1;
white[ 1338 ]<=1;
white[ 1339 ]<=1;
white[ 1340 ]<=1;
white[ 1341 ]<=1;
white[ 1342 ]<=1;
white[ 1343 ]<=1;
white[ 1344 ]<=1;
white[ 1345 ]<=1;
white[ 1346 ]<=1;
white[ 1347 ]<=1;
white[ 1348 ]<=1;
white[ 1349 ]<=1;
white[ 1350 ]<=1;
white[ 1351 ]<=1;
white[ 1352 ]<=1;
white[ 1353 ]<=1;
white[ 1354 ]<=1;
white[ 1355 ]<=1;
white[ 1356 ]<=1;
white[ 1357 ]<=1;
white[ 1358 ]<=1;
white[ 1359 ]<=1;
white[ 1360 ]<=1;
white[ 1361 ]<=1;
white[ 1362 ]<=1;
white[ 1363 ]<=1;
white[ 1364 ]<=1;
white[ 1365 ]<=1;
white[ 1366 ]<=1;
white[ 1367 ]<=1;
```

```
white[ 1368 ]<=1;
white[ 1369 ]<=1;
white[ 1370 ]<=1;
white[ 1371 ]<=1;
white[ 1372 ]<=1;
white[ 1373 ]<=1;
white[ 1374 ]<=1;
white[ 1375 ]<=1;
white[ 1376 ]<=1;
white[ 1377 ]<=0;
white[ 1378 ]<=0;
white[ 1379 ]<=0;
white[ 1380 ]<=0;
white[ 1381 ]<=0;
white[ 1382 ]<=0;
white[ 1383 ]<=1;
white[ 1384 ]<=1;
white[ 1385 ]<=1;
white[ 1386 ]<=1;
white[ 1387 ]<=1;
white[ 1388 ]<=1;
white[ 1389 ]<=1;
white[ 1390 ]<=1;
white[ 1391 ]<=1;
white[ 1392 ]<=1;
white[ 1393 ]<=1;
white[ 1394 ]<=1;
white[ 1395 ]<=1;
white[ 1396 ]<=1;
white[ 1397 ]<=1;
white[ 1398 ]<=1;
white[ 1399 ]<=1;
white[ 1400 ]<=1;
white[ 1401 ]<=1;
white[ 1402 ]<=1;
white[ 1403 ]<=1;
white[ 1404 ]<=1;
white[ 1405 ]<=1;
white[ 1406 ]<=1;
white[ 1407 ]<=1;
white[ 1408 ]<=1;
white[ 1409 ]<=1;
white[ 1410 ]<=1;
white[ 1411 ]<=1;
white[ 1412 ]<=1;
white[ 1413 ]<=1;
white[ 1414 ]<=1;
white[ 1415 ]<=1;
white[ 1416 ]<=1;
white[ 1417 ]<=1;
white[ 1418 ]<=1;
white[ 1419 ]<=1;
white[ 1420 ]<=1;
white[ 1421 ]<=1;
white[ 1422 ]<=1;
white[ 1423 ]<=1;
white[ 1424 ]<=1;
white[ 1425 ]<=1;
```

```
white[ 1426 ]<=1;
white[ 1427 ]<=1;
white[ 1428 ]<=1;
white[ 1429 ]<=1;
white[ 1430 ]<=1;
white[ 1431 ]<=1;
white[ 1432 ]<=1;
white[ 1433 ]<=1;
white[ 1434 ]<=1;
white[ 1435 ]<=1;
white[ 1436 ]<=1;
white[ 1437 ]<=0;
white[ 1438 ]<=0;
white[ 1439 ]<=0;
white[ 1440 ]<=0;
white[ 1441 ]<=0;
white[ 1442 ]<=0;
white[ 1443 ]<=1;
white[ 1444 ]<=1;
white[ 1445 ]<=1;
white[ 1446 ]<=1;
white[ 1447 ]<=1;
white[ 1448 ]<=1;
white[ 1449 ]<=1;
white[ 1450 ]<=1;
white[ 1451 ]<=1;
white[ 1452 ]<=1;
white[ 1453 ]<=1;
white[ 1454 ]<=1;
white[ 1455 ]<=1;
white[ 1456 ]<=1;
white[ 1457 ]<=1;
white[ 1458 ]<=1;
white[ 1459 ]<=1;
white[ 1460 ]<=1;
white[ 1461 ]<=1;
white[ 1462 ]<=1;
white[ 1463 ]<=1;
white[ 1464 ]<=1;
white[ 1465 ]<=1;
white[ 1466 ]<=1;
white[ 1467 ]<=1;
white[ 1468 ]<=1;
white[ 1469 ]<=1;
white[ 1470 ]<=1;
white[ 1471 ]<=1;
white[ 1472 ]<=1;
white[ 1473 ]<=1;
white[ 1474 ]<=1;
white[ 1475 ]<=1;
white[ 1476 ]<=1;
white[ 1477 ]<=1;
white[ 1478 ]<=1;
white[ 1479 ]<=1;
white[ 1480 ]<=1;
white[ 1481 ]<=1;
white[ 1482 ]<=1;
white[ 1483 ]<=1;
```

```
white[ 1484 ]<=1;
white[ 1485 ]<=1;
white[ 1486 ]<=1;
white[ 1487 ]<=1;
white[ 1488 ]<=1;
white[ 1489 ]<=1;
white[ 1490 ]<=1;
white[ 1491 ]<=1;
white[ 1492 ]<=1;
white[ 1493 ]<=1;
white[ 1494 ]<=1;
white[ 1495 ]<=1;
white[ 1496 ]<=1;
white[ 1497 ]<=0;
white[ 1498 ]<=0;
white[ 1499 ]<=0;
white[ 1500 ]<=0;
white[ 1501 ]<=0;
white[ 1502 ]<=0;
white[ 1503 ]<=0;
white[ 1504 ]<=1;
white[ 1505 ]<=1;
white[ 1506 ]<=1;
white[ 1507 ]<=1;
white[ 1508 ]<=1;
white[ 1509 ]<=1;
white[ 1510 ]<=1;
white[ 1511 ]<=1;
white[ 1512 ]<=1;
white[ 1513 ]<=1;
white[ 1514 ]<=1;
white[ 1515 ]<=1;
white[ 1516 ]<=1;
white[ 1517 ]<=1;
white[ 1518 ]<=1;
white[ 1519 ]<=1;
white[ 1520 ]<=1;
white[ 1521 ]<=1;
white[ 1522 ]<=1;
white[ 1523 ]<=1;
white[ 1524 ]<=1;
white[ 1525 ]<=1;
white[ 1526 ]<=1;
white[ 1527 ]<=1;
white[ 1528 ]<=1;
white[ 1529 ]<=1;
white[ 1530 ]<=1;
white[ 1531 ]<=1;
white[ 1532 ]<=1;
white[ 1533 ]<=1;
white[ 1534 ]<=1;
white[ 1535 ]<=1;
white[ 1536 ]<=1;
white[ 1537 ]<=1;
white[ 1538 ]<=1;
white[ 1539 ]<=1;
white[ 1540 ]<=1;
white[ 1541 ]<=1;
```

```
white[ 1542 ]<=1;
white[ 1543 ]<=1;
white[ 1544 ]<=1;
white[ 1545 ]<=1;
white[ 1546 ]<=1;
white[ 1547 ]<=1;
white[ 1548 ]<=1;
white[ 1549 ]<=1;
white[ 1550 ]<=1;
white[ 1551 ]<=1;
white[ 1552 ]<=1;
white[ 1553 ]<=1;
white[ 1554 ]<=1;
white[ 1555 ]<=1;
white[ 1556 ]<=0;
white[ 1557 ]<=0;
white[ 1558 ]<=0;
white[ 1559 ]<=0;
white[ 1560 ]<=0;
white[ 1561 ]<=0;
white[ 1562 ]<=0;
white[ 1563 ]<=0;
white[ 1564 ]<=1;
white[ 1565 ]<=1;
white[ 1566 ]<=1;
white[ 1567 ]<=1;
white[ 1568 ]<=1;
white[ 1569 ]<=1;
white[ 1570 ]<=1;
white[ 1571 ]<=1;
white[ 1572 ]<=1;
white[ 1573 ]<=1;
white[ 1574 ]<=1;
white[ 1575 ]<=1;
white[ 1576 ]<=1;
white[ 1577 ]<=1;
white[ 1578 ]<=1;
white[ 1579 ]<=1;
white[ 1580 ]<=1;
white[ 1581 ]<=1;
white[ 1582 ]<=1;
white[ 1583 ]<=1;
white[ 1584 ]<=1;
white[ 1585 ]<=1;
white[ 1586 ]<=1;
white[ 1587 ]<=1;
white[ 1588 ]<=1;
white[ 1589 ]<=1;
white[ 1590 ]<=1;
white[ 1591 ]<=1;
white[ 1592 ]<=1;
white[ 1593 ]<=1;
white[ 1594 ]<=1;
white[ 1595 ]<=1;
white[ 1596 ]<=1;
white[ 1597 ]<=1;
white[ 1598 ]<=1;
white[ 1599 ]<=1;
```

```
white[ 1600 ]<=1;
white[ 1601 ]<=1;
white[ 1602 ]<=1;
white[ 1603 ]<=1;
white[ 1604 ]<=1;
white[ 1605 ]<=1;
white[ 1606 ]<=1;
white[ 1607 ]<=1;
white[ 1608 ]<=1;
white[ 1609 ]<=1;
white[ 1610 ]<=1;
white[ 1611 ]<=1;
white[ 1612 ]<=1;
white[ 1613 ]<=1;
white[ 1614 ]<=1;
white[ 1615 ]<=1;
white[ 1616 ]<=0;
white[ 1617 ]<=0;
white[ 1618 ]<=0;
white[ 1619 ]<=0;
white[ 1620 ]<=0;
white[ 1621 ]<=0;
white[ 1622 ]<=0;
white[ 1623 ]<=0;
white[ 1624 ]<=1;
white[ 1625 ]<=1;
white[ 1626 ]<=1;
white[ 1627 ]<=1;
white[ 1628 ]<=1;
white[ 1629 ]<=1;
white[ 1630 ]<=1;
white[ 1631 ]<=1;
white[ 1632 ]<=1;
white[ 1633 ]<=1;
white[ 1634 ]<=1;
white[ 1635 ]<=1;
white[ 1636 ]<=1;
white[ 1637 ]<=1;
white[ 1638 ]<=1;
white[ 1639 ]<=1;
white[ 1640 ]<=1;
white[ 1641 ]<=1;
white[ 1642 ]<=1;
white[ 1643 ]<=1;
white[ 1644 ]<=1;
white[ 1645 ]<=1;
white[ 1646 ]<=1;
white[ 1647 ]<=1;
white[ 1648 ]<=1;
white[ 1649 ]<=1;
white[ 1650 ]<=1;
white[ 1651 ]<=1;
white[ 1652 ]<=1;
white[ 1653 ]<=1;
white[ 1654 ]<=1;
white[ 1655 ]<=1;
white[ 1656 ]<=1;
white[ 1657 ]<=1;
```

```
white[ 1658 ]<=1;
white[ 1659 ]<=1;
white[ 1660 ]<=1;
white[ 1661 ]<=1;
white[ 1662 ]<=1;
white[ 1663 ]<=1;
white[ 1664 ]<=1;
white[ 1665 ]<=1;
white[ 1666 ]<=1;
white[ 1667 ]<=1;
white[ 1668 ]<=1;
white[ 1669 ]<=1;
white[ 1670 ]<=1;
white[ 1671 ]<=1;
white[ 1672 ]<=1;
white[ 1673 ]<=1;
white[ 1674 ]<=1;
white[ 1675 ]<=0;
white[ 1676 ]<=0;
white[ 1677 ]<=0;
white[ 1678 ]<=0;
white[ 1679 ]<=0;
white[ 1680 ]<=0;
white[ 1681 ]<=0;
white[ 1682 ]<=0;
white[ 1683 ]<=0;
white[ 1684 ]<=0;
white[ 1685 ]<=1;
white[ 1686 ]<=1;
white[ 1687 ]<=1;
white[ 1688 ]<=1;
white[ 1689 ]<=1;
white[ 1690 ]<=1;
white[ 1691 ]<=1;
white[ 1692 ]<=1;
white[ 1693 ]<=1;
white[ 1694 ]<=1;
white[ 1695 ]<=1;
white[ 1696 ]<=1;
white[ 1697 ]<=1;
white[ 1698 ]<=1;
white[ 1699 ]<=1;
white[ 1700 ]<=1;
white[ 1701 ]<=1;
white[ 1702 ]<=1;
white[ 1703 ]<=1;
white[ 1704 ]<=1;
white[ 1705 ]<=1;
white[ 1706 ]<=1;
white[ 1707 ]<=1;
white[ 1708 ]<=1;
white[ 1709 ]<=1;
white[ 1710 ]<=1;
white[ 1711 ]<=1;
white[ 1712 ]<=1;
white[ 1713 ]<=1;
white[ 1714 ]<=1;
white[ 1715 ]<=1;
```

```
white[ 1716 ]<=1;
white[ 1717 ]<=1;
white[ 1718 ]<=1;
white[ 1719 ]<=1;
white[ 1720 ]<=1;
white[ 1721 ]<=1;
white[ 1722 ]<=1;
white[ 1723 ]<=1;
white[ 1724 ]<=1;
white[ 1725 ]<=1;
white[ 1726 ]<=1;
white[ 1727 ]<=1;
white[ 1728 ]<=1;
white[ 1729 ]<=1;
white[ 1730 ]<=1;
white[ 1731 ]<=1;
white[ 1732 ]<=1;
white[ 1733 ]<=1;
white[ 1734 ]<=1;
white[ 1735 ]<=0;
white[ 1736 ]<=0;
white[ 1737 ]<=0;
white[ 1738 ]<=0;
white[ 1739 ]<=0;
white[ 1740 ]<=0;
white[ 1741 ]<=0;
white[ 1742 ]<=0;
white[ 1743 ]<=0;
white[ 1744 ]<=0;
white[ 1745 ]<=1;
white[ 1746 ]<=1;
white[ 1747 ]<=1;
white[ 1748 ]<=1;
white[ 1749 ]<=1;
white[ 1750 ]<=1;
white[ 1751 ]<=1;
white[ 1752 ]<=1;
white[ 1753 ]<=1;
white[ 1754 ]<=1;
white[ 1755 ]<=1;
white[ 1756 ]<=1;
white[ 1757 ]<=1;
white[ 1758 ]<=1;
white[ 1759 ]<=1;
white[ 1760 ]<=1;
white[ 1761 ]<=1;
white[ 1762 ]<=1;
white[ 1763 ]<=1;
white[ 1764 ]<=1;
white[ 1765 ]<=1;
white[ 1766 ]<=1;
white[ 1767 ]<=1;
white[ 1768 ]<=1;
white[ 1769 ]<=1;
white[ 1770 ]<=1;
white[ 1771 ]<=1;
white[ 1772 ]<=1;
white[ 1773 ]<=1;
```

```
white[ 1774 ]<=1;
white[ 1775 ]<=1;
white[ 1776 ]<=1;
white[ 1777 ]<=1;
white[ 1778 ]<=1;
white[ 1779 ]<=1;
white[ 1780 ]<=1;
white[ 1781 ]<=1;
white[ 1782 ]<=1;
white[ 1783 ]<=1;
white[ 1784 ]<=1;
white[ 1785 ]<=1;
white[ 1786 ]<=1;
white[ 1787 ]<=1;
white[ 1788 ]<=1;
white[ 1789 ]<=1;
white[ 1790 ]<=1;
white[ 1791 ]<=1;
white[ 1792 ]<=1;
white[ 1793 ]<=1;
white[ 1794 ]<=0;
white[ 1795 ]<=0;
white[ 1796 ]<=0;
white[ 1797 ]<=0;
white[ 1798 ]<=0;
white[ 1799 ]<=0;
white[ 1800 ]<=0;
white[ 1801 ]<=0;
white[ 1802 ]<=0;
white[ 1803 ]<=0;
white[ 1804 ]<=0;
white[ 1805 ]<=0;
white[ 1806 ]<=1;
white[ 1807 ]<=1;
white[ 1808 ]<=1;
white[ 1809 ]<=1;
white[ 1810 ]<=1;
white[ 1811 ]<=1;
white[ 1812 ]<=1;
white[ 1813 ]<=1;
white[ 1814 ]<=1;
white[ 1815 ]<=1;
white[ 1816 ]<=1;
white[ 1817 ]<=1;
white[ 1818 ]<=1;
white[ 1819 ]<=1;
white[ 1820 ]<=1;
white[ 1821 ]<=1;
white[ 1822 ]<=1;
white[ 1823 ]<=1;
white[ 1824 ]<=1;
white[ 1825 ]<=1;
white[ 1826 ]<=1;
white[ 1827 ]<=1;
white[ 1828 ]<=1;
white[ 1829 ]<=1;
white[ 1830 ]<=1;
white[ 1831 ]<=1;
```

```
white[ 1832 ]<=1;
white[ 1833 ]<=1;
white[ 1834 ]<=1;
white[ 1835 ]<=1;
white[ 1836 ]<=1;
white[ 1837 ]<=1;
white[ 1838 ]<=1;
white[ 1839 ]<=1;
white[ 1840 ]<=1;
white[ 1841 ]<=1;
white[ 1842 ]<=1;
white[ 1843 ]<=1;
white[ 1844 ]<=1;
white[ 1845 ]<=1;
white[ 1846 ]<=1;
white[ 1847 ]<=1;
white[ 1848 ]<=1;
white[ 1849 ]<=1;
white[ 1850 ]<=1;
white[ 1851 ]<=1;
white[ 1852 ]<=1;
white[ 1853 ]<=1;
white[ 1854 ]<=0;
white[ 1855 ]<=0;
white[ 1856 ]<=0;
white[ 1857 ]<=0;
white[ 1858 ]<=0;
white[ 1859 ]<=0;
white[ 1860 ]<=0;
white[ 1861 ]<=0;
white[ 1862 ]<=0;
white[ 1863 ]<=0;
white[ 1864 ]<=0;
white[ 1865 ]<=0;
white[ 1866 ]<=0;
white[ 1867 ]<=1;
white[ 1868 ]<=1;
white[ 1869 ]<=1;
white[ 1870 ]<=1;
white[ 1871 ]<=1;
white[ 1872 ]<=1;
white[ 1873 ]<=1;
white[ 1874 ]<=1;
white[ 1875 ]<=1;
white[ 1876 ]<=1;
white[ 1877 ]<=1;
white[ 1878 ]<=1;
white[ 1879 ]<=1;
white[ 1880 ]<=1;
white[ 1881 ]<=1;
white[ 1882 ]<=1;
white[ 1883 ]<=1;
white[ 1884 ]<=1;
white[ 1885 ]<=1;
white[ 1886 ]<=1;
white[ 1887 ]<=1;
white[ 1888 ]<=1;
white[ 1889 ]<=1;
```

```
white[ 1890 ]<=1;
white[ 1891 ]<=1;
white[ 1892 ]<=1;
white[ 1893 ]<=1;
white[ 1894 ]<=1;
white[ 1895 ]<=1;
white[ 1896 ]<=1;
white[ 1897 ]<=1;
white[ 1898 ]<=1;
white[ 1899 ]<=1;
white[ 1900 ]<=1;
white[ 1901 ]<=1;
white[ 1902 ]<=1;
white[ 1903 ]<=1;
white[ 1904 ]<=1;
white[ 1905 ]<=1;
white[ 1906 ]<=1;
white[ 1907 ]<=1;
white[ 1908 ]<=1;
white[ 1909 ]<=1;
white[ 1910 ]<=1;
white[ 1911 ]<=1;
white[ 1912 ]<=1;
white[ 1913 ]<=0;
white[ 1914 ]<=0;
white[ 1915 ]<=0;
white[ 1916 ]<=0;
white[ 1917 ]<=0;
white[ 1918 ]<=0;
white[ 1919 ]<=0;
white[ 1920 ]<=0;
white[ 1921 ]<=0;
white[ 1922 ]<=0;
white[ 1923 ]<=0;
white[ 1924 ]<=0;
white[ 1925 ]<=0;
white[ 1926 ]<=0;
white[ 1927 ]<=1;
white[ 1928 ]<=1;
white[ 1929 ]<=1;
white[ 1930 ]<=1;
white[ 1931 ]<=1;
white[ 1932 ]<=1;
white[ 1933 ]<=1;
white[ 1934 ]<=1;
white[ 1935 ]<=1;
white[ 1936 ]<=1;
white[ 1937 ]<=1;
white[ 1938 ]<=1;
white[ 1939 ]<=1;
white[ 1940 ]<=1;
white[ 1941 ]<=1;
white[ 1942 ]<=1;
white[ 1943 ]<=1;
white[ 1944 ]<=1;
white[ 1945 ]<=1;
white[ 1946 ]<=1;
white[ 1947 ]<=1;
```

```
white[ 1948 ]<=1;
white[ 1949 ]<=1;
white[ 1950 ]<=1;
white[ 1951 ]<=1;
white[ 1952 ]<=1;
white[ 1953 ]<=1;
white[ 1954 ]<=1;
white[ 1955 ]<=1;
white[ 1956 ]<=1;
white[ 1957 ]<=1;
white[ 1958 ]<=1;
white[ 1959 ]<=1;
white[ 1960 ]<=1;
white[ 1961 ]<=1;
white[ 1962 ]<=1;
white[ 1963 ]<=1;
white[ 1964 ]<=1;
white[ 1965 ]<=1;
white[ 1966 ]<=1;
white[ 1967 ]<=1;
white[ 1968 ]<=1;
white[ 1969 ]<=1;
white[ 1970 ]<=1;
white[ 1971 ]<=1;
white[ 1972 ]<=1;
white[ 1973 ]<=0;
white[ 1974 ]<=0;
white[ 1975 ]<=0;
white[ 1976 ]<=0;
white[ 1977 ]<=0;
white[ 1978 ]<=0;
white[ 1979 ]<=0;
white[ 1980 ]<=0;
white[ 1981 ]<=0;
white[ 1982 ]<=0;
white[ 1983 ]<=0;
white[ 1984 ]<=0;
white[ 1985 ]<=0;
white[ 1986 ]<=0;
white[ 1987 ]<=0;
white[ 1988 ]<=1;
white[ 1989 ]<=1;
white[ 1990 ]<=1;
white[ 1991 ]<=1;
white[ 1992 ]<=1;
white[ 1993 ]<=1;
white[ 1994 ]<=1;
white[ 1995 ]<=1;
white[ 1996 ]<=1;
white[ 1997 ]<=1;
white[ 1998 ]<=1;
white[ 1999 ]<=1;
white[ 2000 ]<=1;
white[ 2001 ]<=1;
white[ 2002 ]<=1;
white[ 2003 ]<=1;
white[ 2004 ]<=1;
white[ 2005 ]<=1;
```

```
white[ 2006 ]<=1;
white[ 2007 ]<=1;
white[ 2008 ]<=1;
white[ 2009 ]<=1;
white[ 2010 ]<=1;
white[ 2011 ]<=1;
white[ 2012 ]<=1;
white[ 2013 ]<=1;
white[ 2014 ]<=1;
white[ 2015 ]<=1;
white[ 2016 ]<=1;
white[ 2017 ]<=1;
white[ 2018 ]<=1;
white[ 2019 ]<=1;
white[ 2020 ]<=1;
white[ 2021 ]<=1;
white[ 2022 ]<=1;
white[ 2023 ]<=1;
white[ 2024 ]<=1;
white[ 2025 ]<=1;
white[ 2026 ]<=1;
white[ 2027 ]<=1;
white[ 2028 ]<=1;
white[ 2029 ]<=1;
white[ 2030 ]<=1;
white[ 2031 ]<=1;
white[ 2032 ]<=0;
white[ 2033 ]<=0;
white[ 2034 ]<=0;
white[ 2035 ]<=0;
white[ 2036 ]<=0;
white[ 2037 ]<=0;
white[ 2038 ]<=0;
white[ 2039 ]<=0;
white[ 2040 ]<=0;
white[ 2041 ]<=0;
white[ 2042 ]<=0;
white[ 2043 ]<=0;
white[ 2044 ]<=0;
white[ 2045 ]<=0;
white[ 2046 ]<=0;
white[ 2047 ]<=0;
white[ 2048 ]<=1;
white[ 2049 ]<=1;
white[ 2050 ]<=1;
white[ 2051 ]<=1;
white[ 2052 ]<=1;
white[ 2053 ]<=1;
white[ 2054 ]<=1;
white[ 2055 ]<=1;
white[ 2056 ]<=1;
white[ 2057 ]<=1;
white[ 2058 ]<=1;
white[ 2059 ]<=1;
white[ 2060 ]<=1;
white[ 2061 ]<=1;
white[ 2062 ]<=1;
white[ 2063 ]<=1;
```

```
white[ 2064 ]<=1;
white[ 2065 ]<=1;
white[ 2066 ]<=1;
white[ 2067 ]<=1;
white[ 2068 ]<=1;
white[ 2069 ]<=1;
white[ 2070 ]<=1;
white[ 2071 ]<=1;
white[ 2072 ]<=1;
white[ 2073 ]<=1;
white[ 2074 ]<=1;
white[ 2075 ]<=1;
white[ 2076 ]<=1;
white[ 2077 ]<=1;
white[ 2078 ]<=1;
white[ 2079 ]<=1;
white[ 2080 ]<=1;
white[ 2081 ]<=1;
white[ 2082 ]<=1;
white[ 2083 ]<=1;
white[ 2084 ]<=1;
white[ 2085 ]<=1;
white[ 2086 ]<=1;
white[ 2087 ]<=1;
white[ 2088 ]<=1;
white[ 2089 ]<=1;
white[ 2090 ]<=1;
white[ 2091 ]<=0;
white[ 2092 ]<=0;
white[ 2093 ]<=0;
white[ 2094 ]<=0;
white[ 2095 ]<=0;
white[ 2096 ]<=0;
white[ 2097 ]<=0;
white[ 2098 ]<=0;
white[ 2099 ]<=0;
white[ 2100 ]<=0;
white[ 2101 ]<=0;
white[ 2102 ]<=0;
white[ 2103 ]<=0;
white[ 2104 ]<=0;
white[ 2105 ]<=0;
white[ 2106 ]<=0;
white[ 2107 ]<=0;
white[ 2108 ]<=0;
white[ 2109 ]<=1;
white[ 2110 ]<=1;
white[ 2111 ]<=1;
white[ 2112 ]<=1;
white[ 2113 ]<=1;
white[ 2114 ]<=1;
white[ 2115 ]<=1;
white[ 2116 ]<=1;
white[ 2117 ]<=1;
white[ 2118 ]<=1;
white[ 2119 ]<=1;
white[ 2120 ]<=1;
white[ 2121 ]<=1;
```

```
white[ 2122 ]<=1;
white[ 2123 ]<=1;
white[ 2124 ]<=1;
white[ 2125 ]<=1;
white[ 2126 ]<=1;
white[ 2127 ]<=1;
white[ 2128 ]<=1;
white[ 2129 ]<=1;
white[ 2130 ]<=1;
white[ 2131 ]<=1;
white[ 2132 ]<=1;
white[ 2133 ]<=1;
white[ 2134 ]<=1;
white[ 2135 ]<=1;
white[ 2136 ]<=1;
white[ 2137 ]<=1;
white[ 2138 ]<=1;
white[ 2139 ]<=1;
white[ 2140 ]<=1;
white[ 2141 ]<=1;
white[ 2142 ]<=1;
white[ 2143 ]<=1;
white[ 2144 ]<=1;
white[ 2145 ]<=1;
white[ 2146 ]<=1;
white[ 2147 ]<=1;
white[ 2148 ]<=1;
white[ 2149 ]<=1;
white[ 2150 ]<=1;
white[ 2151 ]<=0;
white[ 2152 ]<=0;
white[ 2153 ]<=0;
white[ 2154 ]<=0;
white[ 2155 ]<=0;
white[ 2156 ]<=0;
white[ 2157 ]<=0;
white[ 2158 ]<=0;
white[ 2159 ]<=0;
white[ 2160 ]<=0;
white[ 2161 ]<=0;
white[ 2162 ]<=0;
white[ 2163 ]<=0;
white[ 2164 ]<=0;
white[ 2165 ]<=0;
white[ 2166 ]<=0;
white[ 2167 ]<=0;
white[ 2168 ]<=0;
white[ 2169 ]<=0;
white[ 2170 ]<=1;
white[ 2171 ]<=1;
white[ 2172 ]<=1;
white[ 2173 ]<=1;
white[ 2174 ]<=1;
white[ 2175 ]<=1;
white[ 2176 ]<=1;
white[ 2177 ]<=1;
white[ 2178 ]<=1;
white[ 2179 ]<=1;
```

```
white[ 2180 ]<=1;
white[ 2181 ]<=1;
white[ 2182 ]<=1;
white[ 2183 ]<=1;
white[ 2184 ]<=1;
white[ 2185 ]<=1;
white[ 2186 ]<=1;
white[ 2187 ]<=1;
white[ 2188 ]<=1;
white[ 2189 ]<=1;
white[ 2190 ]<=1;
white[ 2191 ]<=1;
white[ 2192 ]<=1;
white[ 2193 ]<=1;
white[ 2194 ]<=1;
white[ 2195 ]<=1;
white[ 2196 ]<=1;
white[ 2197 ]<=1;
white[ 2198 ]<=1;
white[ 2199 ]<=1;
white[ 2200 ]<=1;
white[ 2201 ]<=1;
white[ 2202 ]<=1;
white[ 2203 ]<=1;
white[ 2204 ]<=1;
white[ 2205 ]<=1;
white[ 2206 ]<=1;
white[ 2207 ]<=1;
white[ 2208 ]<=1;
white[ 2209 ]<=1;
white[ 2210 ]<=0;
white[ 2211 ]<=0;
white[ 2212 ]<=0;
white[ 2213 ]<=0;
white[ 2214 ]<=0;
white[ 2215 ]<=0;
white[ 2216 ]<=0;
white[ 2217 ]<=0;
white[ 2218 ]<=0;
white[ 2219 ]<=0;
white[ 2220 ]<=0;
white[ 2221 ]<=0;
white[ 2222 ]<=0;
white[ 2223 ]<=0;
white[ 2224 ]<=0;
white[ 2225 ]<=0;
white[ 2226 ]<=0;
white[ 2227 ]<=0;
white[ 2228 ]<=0;
white[ 2229 ]<=0;
white[ 2230 ]<=0;
white[ 2231 ]<=1;
white[ 2232 ]<=1;
white[ 2233 ]<=1;
white[ 2234 ]<=1;
white[ 2235 ]<=1;
white[ 2236 ]<=1;
white[ 2237 ]<=1;
```

```
white[ 2238 ]<=1;
white[ 2239 ]<=1;
white[ 2240 ]<=1;
white[ 2241 ]<=1;
white[ 2242 ]<=1;
white[ 2243 ]<=1;
white[ 2244 ]<=1;
white[ 2245 ]<=1;
white[ 2246 ]<=1;
white[ 2247 ]<=1;
white[ 2248 ]<=1;
white[ 2249 ]<=1;
white[ 2250 ]<=1;
white[ 2251 ]<=1;
white[ 2252 ]<=1;
white[ 2253 ]<=1;
white[ 2254 ]<=1;
white[ 2255 ]<=1;
white[ 2256 ]<=1;
white[ 2257 ]<=1;
white[ 2258 ]<=1;
white[ 2259 ]<=1;
white[ 2260 ]<=1;
white[ 2261 ]<=1;
white[ 2262 ]<=1;
white[ 2263 ]<=1;
white[ 2264 ]<=1;
white[ 2265 ]<=1;
white[ 2266 ]<=1;
white[ 2267 ]<=1;
white[ 2268 ]<=1;
white[ 2269 ]<=0;
white[ 2270 ]<=0;
white[ 2271 ]<=0;
white[ 2272 ]<=0;
white[ 2273 ]<=0;
white[ 2274 ]<=0;
white[ 2275 ]<=0;
white[ 2276 ]<=0;
white[ 2277 ]<=0;
white[ 2278 ]<=0;
white[ 2279 ]<=0;
white[ 2280 ]<=0;
white[ 2281 ]<=0;
white[ 2282 ]<=0;
white[ 2283 ]<=0;
white[ 2284 ]<=0;
white[ 2285 ]<=0;
white[ 2286 ]<=0;
white[ 2287 ]<=0;
white[ 2288 ]<=0;
white[ 2289 ]<=0;
white[ 2290 ]<=0;
white[ 2291 ]<=1;
white[ 2292 ]<=1;
white[ 2293 ]<=1;
white[ 2294 ]<=1;
white[ 2295 ]<=1;
```

```
white[ 2296 ]<=1;
white[ 2297 ]<=1;
white[ 2298 ]<=1;
white[ 2299 ]<=1;
white[ 2300 ]<=1;
white[ 2301 ]<=1;
white[ 2302 ]<=1;
white[ 2303 ]<=1;
white[ 2304 ]<=1;
white[ 2305 ]<=1;
white[ 2306 ]<=1;
white[ 2307 ]<=1;
white[ 2308 ]<=1;
white[ 2309 ]<=1;
white[ 2310 ]<=1;
white[ 2311 ]<=1;
white[ 2312 ]<=1;
white[ 2313 ]<=1;
white[ 2314 ]<=1;
white[ 2315 ]<=1;
white[ 2316 ]<=1;
white[ 2317 ]<=1;
white[ 2318 ]<=1;
white[ 2319 ]<=1;
white[ 2320 ]<=1;
white[ 2321 ]<=1;
white[ 2322 ]<=1;
white[ 2323 ]<=1;
white[ 2324 ]<=1;
white[ 2325 ]<=1;
white[ 2326 ]<=1;
white[ 2327 ]<=1;
white[ 2328 ]<=0;
white[ 2329 ]<=0;
white[ 2330 ]<=0;
white[ 2331 ]<=0;
white[ 2332 ]<=0;
white[ 2333 ]<=0;
white[ 2334 ]<=0;
white[ 2335 ]<=0;
white[ 2336 ]<=0;
white[ 2337 ]<=0;
white[ 2338 ]<=0;
white[ 2339 ]<=0;
white[ 2340 ]<=0;
white[ 2341 ]<=0;
white[ 2342 ]<=0;
white[ 2343 ]<=0;
white[ 2344 ]<=0;
white[ 2345 ]<=0;
white[ 2346 ]<=0;
white[ 2347 ]<=0;
white[ 2348 ]<=0;
white[ 2349 ]<=0;
white[ 2350 ]<=0;
white[ 2351 ]<=0;
white[ 2352 ]<=1;
white[ 2353 ]<=1;
```

```
white[ 2354 ]<=1;
white[ 2355 ]<=1;
white[ 2356 ]<=1;
white[ 2357 ]<=1;
white[ 2358 ]<=1;
white[ 2359 ]<=1;
white[ 2360 ]<=1;
white[ 2361 ]<=1;
white[ 2362 ]<=1;
white[ 2363 ]<=1;
white[ 2364 ]<=1;
white[ 2365 ]<=1;
white[ 2366 ]<=1;
white[ 2367 ]<=1;
white[ 2368 ]<=1;
white[ 2369 ]<=1;
white[ 2370 ]<=1;
white[ 2371 ]<=1;
white[ 2372 ]<=1;
white[ 2373 ]<=1;
white[ 2374 ]<=1;
white[ 2375 ]<=1;
white[ 2376 ]<=1;
white[ 2377 ]<=1;
white[ 2378 ]<=1;
white[ 2379 ]<=1;
white[ 2380 ]<=1;
white[ 2381 ]<=1;
white[ 2382 ]<=1;
white[ 2383 ]<=1;
white[ 2384 ]<=1;
white[ 2385 ]<=1;
white[ 2386 ]<=1;
white[ 2387 ]<=1;
white[ 2388 ]<=0;
white[ 2389 ]<=0;
white[ 2390 ]<=0;
white[ 2391 ]<=0;
white[ 2392 ]<=0;
white[ 2393 ]<=0;
white[ 2394 ]<=0;
white[ 2395 ]<=0;
white[ 2396 ]<=0;
white[ 2397 ]<=0;
white[ 2398 ]<=0;
white[ 2399 ]<=0;
white[ 2400 ]<=0;
white[ 2401 ]<=0;
white[ 2402 ]<=0;
white[ 2403 ]<=0;
white[ 2404 ]<=0;
white[ 2405 ]<=0;
white[ 2406 ]<=0;
white[ 2407 ]<=0;
white[ 2408 ]<=0;
white[ 2409 ]<=0;
white[ 2410 ]<=0;
white[ 2411 ]<=0;
```

```
white[ 2412 ]<=0;
white[ 2413 ]<=1;
white[ 2414 ]<=1;
white[ 2415 ]<=1;
white[ 2416 ]<=1;
white[ 2417 ]<=1;
white[ 2418 ]<=1;
white[ 2419 ]<=1;
white[ 2420 ]<=1;
white[ 2421 ]<=1;
white[ 2422 ]<=1;
white[ 2423 ]<=1;
white[ 2424 ]<=1;
white[ 2425 ]<=1;
white[ 2426 ]<=1;
white[ 2427 ]<=1;
white[ 2428 ]<=1;
white[ 2429 ]<=1;
white[ 2430 ]<=1;
white[ 2431 ]<=1;
white[ 2432 ]<=1;
white[ 2433 ]<=1;
white[ 2434 ]<=1;
white[ 2435 ]<=1;
white[ 2436 ]<=1;
white[ 2437 ]<=1;
white[ 2438 ]<=1;
white[ 2439 ]<=1;
white[ 2440 ]<=1;
white[ 2441 ]<=1;
white[ 2442 ]<=1;
white[ 2443 ]<=1;
white[ 2444 ]<=1;
white[ 2445 ]<=1;
white[ 2446 ]<=1;
white[ 2447 ]<=0;
white[ 2448 ]<=0;
white[ 2449 ]<=0;
white[ 2450 ]<=0;
white[ 2451 ]<=0;
white[ 2452 ]<=0;
white[ 2453 ]<=0;
white[ 2454 ]<=0;
white[ 2455 ]<=0;
white[ 2456 ]<=0;
white[ 2457 ]<=0;
white[ 2458 ]<=0;
white[ 2459 ]<=0;
white[ 2460 ]<=0;
white[ 2461 ]<=0;
white[ 2462 ]<=0;
white[ 2463 ]<=0;
white[ 2464 ]<=0;
white[ 2465 ]<=0;
white[ 2466 ]<=0;
white[ 2467 ]<=0;
white[ 2468 ]<=0;
white[ 2469 ]<=0;
```

```
white[ 2470 ]<=0;
white[ 2471 ]<=0;
white[ 2472 ]<=0;
white[ 2473 ]<=0;
white[ 2474 ]<=1;
white[ 2475 ]<=1;
white[ 2476 ]<=1;
white[ 2477 ]<=1;
white[ 2478 ]<=1;
white[ 2479 ]<=1;
white[ 2480 ]<=1;
white[ 2481 ]<=1;
white[ 2482 ]<=1;
white[ 2483 ]<=1;
white[ 2484 ]<=1;
white[ 2485 ]<=1;
white[ 2486 ]<=1;
white[ 2487 ]<=1;
white[ 2488 ]<=1;
white[ 2489 ]<=1;
white[ 2490 ]<=1;
white[ 2491 ]<=1;
white[ 2492 ]<=1;
white[ 2493 ]<=1;
white[ 2494 ]<=1;
white[ 2495 ]<=1;
white[ 2496 ]<=1;
white[ 2497 ]<=1;
white[ 2498 ]<=1;
white[ 2499 ]<=1;
white[ 2500 ]<=1;
white[ 2501 ]<=1;
white[ 2502 ]<=1;
white[ 2503 ]<=1;
white[ 2504 ]<=1;
white[ 2505 ]<=1;
white[ 2506 ]<=0;
white[ 2507 ]<=0;
white[ 2508 ]<=0;
white[ 2509 ]<=0;
white[ 2510 ]<=0;
white[ 2511 ]<=0;
white[ 2512 ]<=0;
white[ 2513 ]<=0;
white[ 2514 ]<=0;
white[ 2515 ]<=0;
white[ 2516 ]<=0;
white[ 2517 ]<=0;
white[ 2518 ]<=0;
white[ 2519 ]<=0;
white[ 2520 ]<=0;
white[ 2521 ]<=0;
white[ 2522 ]<=0;
white[ 2523 ]<=0;
white[ 2524 ]<=0;
white[ 2525 ]<=0;
white[ 2526 ]<=0;
white[ 2527 ]<=0;
```

```
white[ 2528 ]<=0;
white[ 2529 ]<=0;
white[ 2530 ]<=0;
white[ 2531 ]<=0;
white[ 2532 ]<=0;
white[ 2533 ]<=0;
white[ 2534 ]<=0;
white[ 2535 ]<=1;
white[ 2536 ]<=1;
white[ 2537 ]<=1;
white[ 2538 ]<=1;
white[ 2539 ]<=1;
white[ 2540 ]<=1;
white[ 2541 ]<=1;
white[ 2542 ]<=1;
white[ 2543 ]<=1;
white[ 2544 ]<=1;
white[ 2545 ]<=1;
white[ 2546 ]<=1;
white[ 2547 ]<=1;
white[ 2548 ]<=1;
white[ 2549 ]<=1;
white[ 2550 ]<=1;
white[ 2551 ]<=1;
white[ 2552 ]<=1;
white[ 2553 ]<=1;
white[ 2554 ]<=1;
white[ 2555 ]<=1;
white[ 2556 ]<=1;
white[ 2557 ]<=1;
white[ 2558 ]<=1;
white[ 2559 ]<=1;
white[ 2560 ]<=1;
white[ 2561 ]<=1;
white[ 2562 ]<=1;
white[ 2563 ]<=1;
white[ 2564 ]<=1;
white[ 2565 ]<=0;
white[ 2566 ]<=0;
white[ 2567 ]<=0;
white[ 2568 ]<=0;
white[ 2569 ]<=0;
white[ 2570 ]<=0;
white[ 2571 ]<=0;
white[ 2572 ]<=0;
white[ 2573 ]<=0;
white[ 2574 ]<=0;
white[ 2575 ]<=0;
white[ 2576 ]<=0;
white[ 2577 ]<=0;
white[ 2578 ]<=0;
white[ 2579 ]<=0;
white[ 2580 ]<=0;
white[ 2581 ]<=0;
white[ 2582 ]<=0;
white[ 2583 ]<=0;
white[ 2584 ]<=0;
white[ 2585 ]<=0;
```

```
white[ 2586 ]<=0;
white[ 2587 ]<=0;
white[ 2588 ]<=0;
white[ 2589 ]<=0;
white[ 2590 ]<=0;
white[ 2591 ]<=0;
white[ 2592 ]<=0;
white[ 2593 ]<=0;
white[ 2594 ]<=0;
white[ 2595 ]<=1;
white[ 2596 ]<=1;
white[ 2597 ]<=1;
white[ 2598 ]<=1;
white[ 2599 ]<=1;
white[ 2600 ]<=1;
white[ 2601 ]<=1;
white[ 2602 ]<=1;
white[ 2603 ]<=1;
white[ 2604 ]<=1;
white[ 2605 ]<=1;
white[ 2606 ]<=1;
white[ 2607 ]<=1;
white[ 2608 ]<=1;
white[ 2609 ]<=1;
white[ 2610 ]<=1;
white[ 2611 ]<=1;
white[ 2612 ]<=1;
white[ 2613 ]<=1;
white[ 2614 ]<=1;
white[ 2615 ]<=1;
white[ 2616 ]<=1;
white[ 2617 ]<=1;
white[ 2618 ]<=1;
white[ 2619 ]<=1;
white[ 2620 ]<=1;
white[ 2621 ]<=1;
white[ 2622 ]<=1;
white[ 2623 ]<=1;
white[ 2624 ]<=0;
white[ 2625 ]<=0;
white[ 2626 ]<=0;
white[ 2627 ]<=0;
white[ 2628 ]<=0;
white[ 2629 ]<=0;
white[ 2630 ]<=0;
white[ 2631 ]<=0;
white[ 2632 ]<=0;
white[ 2633 ]<=0;
white[ 2634 ]<=0;
white[ 2635 ]<=0;
white[ 2636 ]<=0;
white[ 2637 ]<=0;
white[ 2638 ]<=0;
white[ 2639 ]<=0;
white[ 2640 ]<=0;
white[ 2641 ]<=0;
white[ 2642 ]<=0;
white[ 2643 ]<=0;
```

```
white[ 2644 ]<=0;
white[ 2645 ]<=0;
white[ 2646 ]<=0;
white[ 2647 ]<=0;
white[ 2648 ]<=0;
white[ 2649 ]<=0;
white[ 2650 ]<=0;
white[ 2651 ]<=0;
white[ 2652 ]<=0;
white[ 2653 ]<=0;
white[ 2654 ]<=0;
white[ 2655 ]<=0;
white[ 2656 ]<=1;
white[ 2657 ]<=1;
white[ 2658 ]<=1;
white[ 2659 ]<=1;
white[ 2660 ]<=1;
white[ 2661 ]<=1;
white[ 2662 ]<=1;
white[ 2663 ]<=1;
white[ 2664 ]<=1;
white[ 2665 ]<=1;
white[ 2666 ]<=1;
white[ 2667 ]<=1;
white[ 2668 ]<=1;
white[ 2669 ]<=1;
white[ 2670 ]<=1;
white[ 2671 ]<=1;
white[ 2672 ]<=1;
white[ 2673 ]<=1;
white[ 2674 ]<=1;
white[ 2675 ]<=1;
white[ 2676 ]<=1;
white[ 2677 ]<=1;
white[ 2678 ]<=1;
white[ 2679 ]<=1;
white[ 2680 ]<=1;
white[ 2681 ]<=1;
white[ 2682 ]<=1;
white[ 2683 ]<=1;
white[ 2684 ]<=0;
white[ 2685 ]<=0;
white[ 2686 ]<=0;
white[ 2687 ]<=0;
white[ 2688 ]<=0;
white[ 2689 ]<=0;
white[ 2690 ]<=0;
white[ 2691 ]<=0;
white[ 2692 ]<=0;
white[ 2693 ]<=0;
white[ 2694 ]<=0;
white[ 2695 ]<=0;
white[ 2696 ]<=0;
white[ 2697 ]<=0;
white[ 2698 ]<=0;
white[ 2699 ]<=0;
white[ 2700 ]<=0;
white[ 2701 ]<=0;
```

```
white[ 2702 ]<=0;
white[ 2703 ]<=0;
white[ 2704 ]<=0;
white[ 2705 ]<=0;
white[ 2706 ]<=0;
white[ 2707 ]<=0;
white[ 2708 ]<=0;
white[ 2709 ]<=0;
white[ 2710 ]<=0;
white[ 2711 ]<=0;
white[ 2712 ]<=0;
white[ 2713 ]<=0;
white[ 2714 ]<=0;
white[ 2715 ]<=0;
white[ 2716 ]<=0;
white[ 2717 ]<=1;
white[ 2718 ]<=1;
white[ 2719 ]<=1;
white[ 2720 ]<=1;
white[ 2721 ]<=1;
white[ 2722 ]<=1;
white[ 2723 ]<=1;
white[ 2724 ]<=1;
white[ 2725 ]<=1;
white[ 2726 ]<=1;
white[ 2727 ]<=1;
white[ 2728 ]<=1;
white[ 2729 ]<=1;
white[ 2730 ]<=1;
white[ 2731 ]<=1;
white[ 2732 ]<=1;
white[ 2733 ]<=1;
white[ 2734 ]<=1;
white[ 2735 ]<=1;
white[ 2736 ]<=1;
white[ 2737 ]<=1;
white[ 2738 ]<=1;
white[ 2739 ]<=1;
white[ 2740 ]<=1;
white[ 2741 ]<=1;
white[ 2742 ]<=1;
white[ 2743 ]<=0;
white[ 2744 ]<=0;
white[ 2745 ]<=0;
white[ 2746 ]<=0;
white[ 2747 ]<=0;
white[ 2748 ]<=0;
white[ 2749 ]<=0;
white[ 2750 ]<=0;
white[ 2751 ]<=0;
white[ 2752 ]<=0;
white[ 2753 ]<=0;
white[ 2754 ]<=0;
white[ 2755 ]<=0;
white[ 2756 ]<=0;
white[ 2757 ]<=0;
white[ 2758 ]<=0;
white[ 2759 ]<=0;
```

```
white[ 2760 ]<=0;
white[ 2761 ]<=0;
white[ 2762 ]<=0;
white[ 2763 ]<=0;
white[ 2764 ]<=0;
white[ 2765 ]<=0;
white[ 2766 ]<=0;
white[ 2767 ]<=0;
white[ 2768 ]<=0;
white[ 2769 ]<=0;
white[ 2770 ]<=0;
white[ 2771 ]<=0;
white[ 2772 ]<=0;
white[ 2773 ]<=0;
white[ 2774 ]<=0;
white[ 2775 ]<=0;
white[ 2776 ]<=0;
white[ 2777 ]<=0;
white[ 2778 ]<=1;
white[ 2779 ]<=1;
white[ 2780 ]<=1;
white[ 2781 ]<=1;
white[ 2782 ]<=1;
white[ 2783 ]<=1;
white[ 2784 ]<=1;
white[ 2785 ]<=1;
white[ 2786 ]<=1;
white[ 2787 ]<=1;
white[ 2788 ]<=1;
white[ 2789 ]<=1;
white[ 2790 ]<=1;
white[ 2791 ]<=1;
white[ 2792 ]<=1;
white[ 2793 ]<=1;
white[ 2794 ]<=1;
white[ 2795 ]<=1;
white[ 2796 ]<=1;
white[ 2797 ]<=1;
white[ 2798 ]<=1;
white[ 2799 ]<=1;
white[ 2800 ]<=1;
white[ 2801 ]<=1;
white[ 2802 ]<=0;
white[ 2803 ]<=0;
white[ 2804 ]<=0;
white[ 2805 ]<=0;
white[ 2806 ]<=0;
white[ 2807 ]<=0;
white[ 2808 ]<=0;
white[ 2809 ]<=0;
white[ 2810 ]<=0;
white[ 2811 ]<=0;
white[ 2812 ]<=0;
white[ 2813 ]<=0;
white[ 2814 ]<=0;
white[ 2815 ]<=0;
white[ 2816 ]<=0;
white[ 2817 ]<=0;
```

```
white[ 2818 ]<=0;
white[ 2819 ]<=0;
white[ 2820 ]<=0;
white[ 2821 ]<=0;
white[ 2822 ]<=0;
white[ 2823 ]<=0;
white[ 2824 ]<=0;
white[ 2825 ]<=0;
white[ 2826 ]<=0;
white[ 2827 ]<=0;
white[ 2828 ]<=0;
white[ 2829 ]<=0;
white[ 2830 ]<=0;
white[ 2831 ]<=0;
white[ 2832 ]<=0;
white[ 2833 ]<=0;
white[ 2834 ]<=0;
white[ 2835 ]<=0;
white[ 2836 ]<=0;
white[ 2837 ]<=0;
white[ 2838 ]<=0;
white[ 2839 ]<=1;
white[ 2840 ]<=1;
white[ 2841 ]<=1;
white[ 2842 ]<=1;
white[ 2843 ]<=1;
white[ 2844 ]<=1;
white[ 2845 ]<=1;
white[ 2846 ]<=1;
white[ 2847 ]<=1;
white[ 2848 ]<=1;
white[ 2849 ]<=1;
white[ 2850 ]<=1;
white[ 2851 ]<=1;
white[ 2852 ]<=1;
white[ 2853 ]<=1;
white[ 2854 ]<=1;
white[ 2855 ]<=1;
white[ 2856 ]<=1;
white[ 2857 ]<=1;
white[ 2858 ]<=1;
white[ 2859 ]<=1;
white[ 2860 ]<=1;
white[ 2861 ]<=0;
white[ 2862 ]<=0;
white[ 2863 ]<=0;
white[ 2864 ]<=0;
white[ 2865 ]<=0;
white[ 2866 ]<=0;
white[ 2867 ]<=0;
white[ 2868 ]<=0;
white[ 2869 ]<=0;
white[ 2870 ]<=0;
white[ 2871 ]<=0;
white[ 2872 ]<=0;
white[ 2873 ]<=0;
white[ 2874 ]<=0;
white[ 2875 ]<=0;
```

```
white[ 2876 ]<=0;
white[ 2877 ]<=0;
white[ 2878 ]<=0;
white[ 2879 ]<=0;
white[ 2880 ]<=0;
white[ 2881 ]<=0;
white[ 2882 ]<=0;
white[ 2883 ]<=0;
white[ 2884 ]<=0;
white[ 2885 ]<=0;
white[ 2886 ]<=0;
white[ 2887 ]<=0;
white[ 2888 ]<=0;
white[ 2889 ]<=0;
white[ 2890 ]<=0;
white[ 2891 ]<=0;
white[ 2892 ]<=0;
white[ 2893 ]<=0;
white[ 2894 ]<=0;
white[ 2895 ]<=0;
white[ 2896 ]<=0;
white[ 2897 ]<=0;
white[ 2898 ]<=0;
white[ 2899 ]<=0;
white[ 2900 ]<=1;
white[ 2901 ]<=1;
white[ 2902 ]<=1;
white[ 2903 ]<=1;
white[ 2904 ]<=1;
white[ 2905 ]<=1;
white[ 2906 ]<=1;
white[ 2907 ]<=1;
white[ 2908 ]<=1;
white[ 2909 ]<=1;
white[ 2910 ]<=1;
white[ 2911 ]<=1;
white[ 2912 ]<=1;
white[ 2913 ]<=1;
white[ 2914 ]<=1;
white[ 2915 ]<=1;
white[ 2916 ]<=1;
white[ 2917 ]<=1;
white[ 2918 ]<=1;
white[ 2919 ]<=1;
white[ 2920 ]<=0;
white[ 2921 ]<=0;
white[ 2922 ]<=0;
white[ 2923 ]<=0;
white[ 2924 ]<=0;
white[ 2925 ]<=0;
white[ 2926 ]<=0;
white[ 2927 ]<=0;
white[ 2928 ]<=0;
white[ 2929 ]<=0;
white[ 2930 ]<=0;
white[ 2931 ]<=0;
white[ 2932 ]<=0;
white[ 2933 ]<=0;
```

```
white[ 2934 ]<=0;
white[ 2935 ]<=0;
white[ 2936 ]<=0;
white[ 2937 ]<=0;
white[ 2938 ]<=0;
white[ 2939 ]<=0;
white[ 2940 ]<=0;
white[ 2941 ]<=0;
white[ 2942 ]<=0;
white[ 2943 ]<=0;
white[ 2944 ]<=0;
white[ 2945 ]<=0;
white[ 2946 ]<=0;
white[ 2947 ]<=0;
white[ 2948 ]<=0;
white[ 2949 ]<=0;
white[ 2950 ]<=0;
white[ 2951 ]<=0;
white[ 2952 ]<=0;
white[ 2953 ]<=0;
white[ 2954 ]<=0;
white[ 2955 ]<=0;
white[ 2956 ]<=0;
white[ 2957 ]<=0;
white[ 2958 ]<=0;
white[ 2959 ]<=0;
white[ 2960 ]<=0;
white[ 2961 ]<=1;
white[ 2962 ]<=1;
white[ 2963 ]<=1;
white[ 2964 ]<=1;
white[ 2965 ]<=1;
white[ 2966 ]<=1;
white[ 2967 ]<=1;
white[ 2968 ]<=1;
white[ 2969 ]<=1;
white[ 2970 ]<=1;
white[ 2971 ]<=1;
white[ 2972 ]<=1;
white[ 2973 ]<=1;
white[ 2974 ]<=1;
white[ 2975 ]<=1;
white[ 2976 ]<=1;
white[ 2977 ]<=1;
white[ 2978 ]<=1;
white[ 2979 ]<=0;
white[ 2980 ]<=0;
white[ 2981 ]<=0;
white[ 2982 ]<=0;
white[ 2983 ]<=0;
white[ 2984 ]<=0;
white[ 2985 ]<=0;
white[ 2986 ]<=0;
white[ 2987 ]<=0;
white[ 2988 ]<=0;
white[ 2989 ]<=0;
white[ 2990 ]<=0;
white[ 2991 ]<=0;
```

```
white[ 2992 ]<=0;
white[ 2993 ]<=0;
white[ 2994 ]<=0;
white[ 2995 ]<=0;
white[ 2996 ]<=0;
white[ 2997 ]<=0;
white[ 2998 ]<=0;
white[ 2999 ]<=0;
white[ 3000 ]<=0;
white[ 3001 ]<=0;
white[ 3002 ]<=0;
white[ 3003 ]<=0;
white[ 3004 ]<=0;
white[ 3005 ]<=0;
white[ 3006 ]<=0;
white[ 3007 ]<=0;
white[ 3008 ]<=0;
white[ 3009 ]<=0;
white[ 3010 ]<=0;
white[ 3011 ]<=0;
white[ 3012 ]<=0;
white[ 3013 ]<=0;
white[ 3014 ]<=0;
white[ 3015 ]<=0;
white[ 3016 ]<=0;
white[ 3017 ]<=0;
white[ 3018 ]<=0;
white[ 3019 ]<=0;
white[ 3020 ]<=0;
white[ 3021 ]<=0;
white[ 3022 ]<=1;
white[ 3023 ]<=1;
white[ 3024 ]<=1;
white[ 3025 ]<=1;
white[ 3026 ]<=1;
white[ 3027 ]<=1;
white[ 3028 ]<=1;
white[ 3029 ]<=1;
white[ 3030 ]<=1;
white[ 3031 ]<=1;
white[ 3032 ]<=1;
white[ 3033 ]<=1;
white[ 3034 ]<=1;
white[ 3035 ]<=1;
white[ 3036 ]<=1;
white[ 3037 ]<=1;
white[ 3038 ]<=0;
white[ 3039 ]<=0;
white[ 3040 ]<=0;
white[ 3041 ]<=0;
white[ 3042 ]<=0;
white[ 3043 ]<=0;
white[ 3044 ]<=0;
white[ 3045 ]<=0;
white[ 3046 ]<=0;
white[ 3047 ]<=0;
white[ 3048 ]<=0;
white[ 3049 ]<=0;
```

```
white[ 3050 ]<=0;
white[ 3051 ]<=0;
white[ 3052 ]<=0;
white[ 3053 ]<=0;
white[ 3054 ]<=0;
white[ 3055 ]<=0;
white[ 3056 ]<=0;
white[ 3057 ]<=0;
white[ 3058 ]<=0;
white[ 3059 ]<=0;
white[ 3060 ]<=0;
white[ 3061 ]<=0;
white[ 3062 ]<=0;
white[ 3063 ]<=0;
white[ 3064 ]<=0;
white[ 3065 ]<=0;
white[ 3066 ]<=0;
white[ 3067 ]<=0;
white[ 3068 ]<=0;
white[ 3069 ]<=0;
white[ 3070 ]<=0;
white[ 3071 ]<=0;
white[ 3072 ]<=0;
white[ 3073 ]<=0;
white[ 3074 ]<=0;
white[ 3075 ]<=0;
white[ 3076 ]<=0;
white[ 3077 ]<=0;
white[ 3078 ]<=0;
white[ 3079 ]<=0;
white[ 3080 ]<=0;
white[ 3081 ]<=0;
white[ 3082 ]<=0;
white[ 3083 ]<=1;
white[ 3084 ]<=1;
white[ 3085 ]<=1;
white[ 3086 ]<=1;
white[ 3087 ]<=1;
white[ 3088 ]<=1;
white[ 3089 ]<=1;
white[ 3090 ]<=1;
white[ 3091 ]<=1;
white[ 3092 ]<=1;
white[ 3093 ]<=1;
white[ 3094 ]<=1;
white[ 3095 ]<=1;
white[ 3096 ]<=1;
white[ 3097 ]<=0;
white[ 3098 ]<=0;
white[ 3099 ]<=0;
white[ 3100 ]<=0;
white[ 3101 ]<=0;
white[ 3102 ]<=0;
white[ 3103 ]<=0;
white[ 3104 ]<=0;
white[ 3105 ]<=0;
white[ 3106 ]<=0;
white[ 3107 ]<=0;
```

```
white[ 3108 ]<=0;
white[ 3109 ]<=0;
white[ 3110 ]<=0;
white[ 3111 ]<=0;
white[ 3112 ]<=0;
white[ 3113 ]<=0;
white[ 3114 ]<=0;
white[ 3115 ]<=0;
white[ 3116 ]<=0;
white[ 3117 ]<=0;
white[ 3118 ]<=0;
white[ 3119 ]<=0;
white[ 3120 ]<=0;
white[ 3121 ]<=0;
white[ 3122 ]<=0;
white[ 3123 ]<=0;
white[ 3124 ]<=0;
white[ 3125 ]<=0;
white[ 3126 ]<=0;
white[ 3127 ]<=0;
white[ 3128 ]<=0;
white[ 3129 ]<=0;
white[ 3130 ]<=0;
white[ 3131 ]<=0;
white[ 3132 ]<=0;
white[ 3133 ]<=0;
white[ 3134 ]<=0;
white[ 3135 ]<=0;
white[ 3136 ]<=0;
white[ 3137 ]<=0;
white[ 3138 ]<=0;
white[ 3139 ]<=0;
white[ 3140 ]<=0;
white[ 3141 ]<=0;
white[ 3142 ]<=0;
white[ 3143 ]<=0;
white[ 3144 ]<=1;
white[ 3145 ]<=1;
white[ 3146 ]<=1;
white[ 3147 ]<=1;
white[ 3148 ]<=1;
white[ 3149 ]<=1;
white[ 3150 ]<=1;
white[ 3151 ]<=1;
white[ 3152 ]<=1;
white[ 3153 ]<=1;
white[ 3154 ]<=1;
white[ 3155 ]<=1;
white[ 3156 ]<=0;
white[ 3157 ]<=0;
white[ 3158 ]<=0;
white[ 3159 ]<=0;
white[ 3160 ]<=0;
white[ 3161 ]<=0;
white[ 3162 ]<=0;
white[ 3163 ]<=0;
white[ 3164 ]<=0;
white[ 3165 ]<=0;
```

```
white[ 3166 ]<=0;
white[ 3167 ]<=0;
white[ 3168 ]<=0;
white[ 3169 ]<=0;
white[ 3170 ]<=0;
white[ 3171 ]<=0;
white[ 3172 ]<=0;
white[ 3173 ]<=0;
white[ 3174 ]<=0;
white[ 3175 ]<=0;
white[ 3176 ]<=0;
white[ 3177 ]<=0;
white[ 3178 ]<=0;
white[ 3179 ]<=0;
white[ 3180 ]<=0;
white[ 3181 ]<=0;
white[ 3182 ]<=0;
white[ 3183 ]<=0;
white[ 3184 ]<=0;
white[ 3185 ]<=0;
white[ 3186 ]<=0;
white[ 3187 ]<=0;
white[ 3188 ]<=0;
white[ 3189 ]<=0;
white[ 3190 ]<=0;
white[ 3191 ]<=0;
white[ 3192 ]<=0;
white[ 3193 ]<=0;
white[ 3194 ]<=0;
white[ 3195 ]<=0;
white[ 3196 ]<=0;
white[ 3197 ]<=0;
white[ 3198 ]<=0;
white[ 3199 ]<=0;
white[ 3200 ]<=0;
white[ 3201 ]<=0;
white[ 3202 ]<=0;
white[ 3203 ]<=0;
white[ 3204 ]<=0;
white[ 3205 ]<=1;
white[ 3206 ]<=1;
white[ 3207 ]<=1;
white[ 3208 ]<=1;
white[ 3209 ]<=1;
white[ 3210 ]<=1;
white[ 3211 ]<=1;
white[ 3212 ]<=1;
white[ 3213 ]<=1;
white[ 3214 ]<=0;
white[ 3215 ]<=0;
white[ 3216 ]<=0;
white[ 3217 ]<=0;
white[ 3218 ]<=0;
white[ 3219 ]<=0;
white[ 3220 ]<=0;
white[ 3221 ]<=0;
white[ 3222 ]<=0;
white[ 3223 ]<=0;
```

```
white[ 3224 ]<=0;
white[ 3225 ]<=0;
white[ 3226 ]<=0;
white[ 3227 ]<=0;
white[ 3228 ]<=0;
white[ 3229 ]<=0;
white[ 3230 ]<=0;
white[ 3231 ]<=0;
white[ 3232 ]<=0;
white[ 3233 ]<=0;
white[ 3234 ]<=0;
white[ 3235 ]<=0;
white[ 3236 ]<=0;
white[ 3237 ]<=0;
white[ 3238 ]<=0;
white[ 3239 ]<=0;
white[ 3240 ]<=0;
white[ 3241 ]<=0;
white[ 3242 ]<=0;
white[ 3243 ]<=0;
white[ 3244 ]<=0;
white[ 3245 ]<=0;
white[ 3246 ]<=0;
white[ 3247 ]<=0;
white[ 3248 ]<=0;
white[ 3249 ]<=0;
white[ 3250 ]<=0;
white[ 3251 ]<=0;
white[ 3252 ]<=0;
white[ 3253 ]<=0;
white[ 3254 ]<=0;
white[ 3255 ]<=0;
white[ 3256 ]<=0;
white[ 3257 ]<=0;
white[ 3258 ]<=0;
white[ 3259 ]<=0;
white[ 3260 ]<=0;
white[ 3261 ]<=0;
white[ 3262 ]<=0;
white[ 3263 ]<=0;
white[ 3264 ]<=0;
white[ 3265 ]<=0;
white[ 3266 ]<=1;
white[ 3267 ]<=1;
white[ 3268 ]<=1;
white[ 3269 ]<=1;
white[ 3270 ]<=1;
white[ 3271 ]<=1;
white[ 3272 ]<=1;
white[ 3273 ]<=0;
white[ 3274 ]<=0;
white[ 3275 ]<=0;
white[ 3276 ]<=0;
white[ 3277 ]<=0;
white[ 3278 ]<=0;
white[ 3279 ]<=0;
white[ 3280 ]<=0;
white[ 3281 ]<=0;
```

```
white[ 3282 ]<=0;
white[ 3283 ]<=0;
white[ 3284 ]<=0;
white[ 3285 ]<=0;
white[ 3286 ]<=0;
white[ 3287 ]<=0;
white[ 3288 ]<=0;
white[ 3289 ]<=0;
white[ 3290 ]<=0;
white[ 3291 ]<=0;
white[ 3292 ]<=0;
white[ 3293 ]<=0;
white[ 3294 ]<=0;
white[ 3295 ]<=0;
white[ 3296 ]<=0;
white[ 3297 ]<=0;
white[ 3298 ]<=0;
white[ 3299 ]<=0;
white[ 3300 ]<=0;
white[ 3301 ]<=0;
white[ 3302 ]<=0;
white[ 3303 ]<=0;
white[ 3304 ]<=0;
white[ 3305 ]<=0;
white[ 3306 ]<=0;
white[ 3307 ]<=0;
white[ 3308 ]<=0;
white[ 3309 ]<=0;
white[ 3310 ]<=0;
white[ 3311 ]<=0;
white[ 3312 ]<=0;
white[ 3313 ]<=0;
white[ 3314 ]<=0;
white[ 3315 ]<=0;
white[ 3316 ]<=0;
white[ 3317 ]<=0;
white[ 3318 ]<=0;
white[ 3319 ]<=0;
white[ 3320 ]<=0;
white[ 3321 ]<=0;
white[ 3322 ]<=0;
white[ 3323 ]<=0;
white[ 3324 ]<=0;
white[ 3325 ]<=0;
white[ 3326 ]<=0;
white[ 3327 ]<=0;
white[ 3328 ]<=1;
white[ 3329 ]<=1;
white[ 3330 ]<=1;
white[ 3331 ]<=1;
white[ 3332 ]<=0;
white[ 3333 ]<=0;
white[ 3334 ]<=0;
white[ 3335 ]<=0;
white[ 3336 ]<=0;
white[ 3337 ]<=0;
white[ 3338 ]<=0;
white[ 3339 ]<=0;
```

```
white[ 3340 ]<=0;
white[ 3341 ]<=0;
white[ 3342 ]<=0;
white[ 3343 ]<=0;
white[ 3344 ]<=0;
white[ 3345 ]<=0;
white[ 3346 ]<=0;
white[ 3347 ]<=0;
white[ 3348 ]<=0;
white[ 3349 ]<=0;
white[ 3350 ]<=0;
white[ 3351 ]<=0;
white[ 3352 ]<=0;
white[ 3353 ]<=0;
white[ 3354 ]<=0;
white[ 3355 ]<=0;
white[ 3356 ]<=0;
white[ 3357 ]<=0;
white[ 3358 ]<=0;
white[ 3359 ]<=0;
white[ 3360 ]<=0;
white[ 3361 ]<=0;
white[ 3362 ]<=0;
white[ 3363 ]<=0;
white[ 3364 ]<=0;
white[ 3365 ]<=0;
white[ 3366 ]<=0;
white[ 3367 ]<=0;
white[ 3368 ]<=0;
white[ 3369 ]<=0;
white[ 3370 ]<=0;
white[ 3371 ]<=0;
white[ 3372 ]<=0;
white[ 3373 ]<=0;
white[ 3374 ]<=0;
white[ 3375 ]<=0;
white[ 3376 ]<=0;
white[ 3377 ]<=0;
white[ 3378 ]<=0;
white[ 3379 ]<=0;
white[ 3380 ]<=0;
white[ 3381 ]<=0;
white[ 3382 ]<=0;
white[ 3383 ]<=0;
white[ 3384 ]<=0;
white[ 3385 ]<=0;
white[ 3386 ]<=0;
white[ 3387 ]<=0;
white[ 3388 ]<=0;
white[ 3389 ]<=1;
white[ 3390 ]<=1;
white[ 3391 ]<=0;
white[ 3392 ]<=0;
white[ 3393 ]<=0;
white[ 3394 ]<=0;
white[ 3395 ]<=0;
white[ 3396 ]<=0;
white[ 3397 ]<=0;
```

```
white[ 3398 ]<=0;
white[ 3399 ]<=0;
white[ 3400 ]<=0;
white[ 3401 ]<=0;
white[ 3402 ]<=0;
white[ 3403 ]<=0;
white[ 3404 ]<=0;
white[ 3405 ]<=0;
white[ 3406 ]<=0;
white[ 3407 ]<=0;
white[ 3408 ]<=0;
white[ 3409 ]<=0;
white[ 3410 ]<=0;
white[ 3411 ]<=0;
white[ 3412 ]<=0;
white[ 3413 ]<=0;
white[ 3414 ]<=0;
white[ 3415 ]<=0;
white[ 3416 ]<=0;
white[ 3417 ]<=0;
white[ 3418 ]<=0;
white[ 3419 ]<=0;
white[ 3420 ]<=0;
white[ 3421 ]<=0;
white[ 3422 ]<=0;
white[ 3423 ]<=0;
white[ 3424 ]<=0;
white[ 3425 ]<=0;
white[ 3426 ]<=0;
white[ 3427 ]<=0;
white[ 3428 ]<=0;
white[ 3429 ]<=0;
white[ 3430 ]<=0;
white[ 3431 ]<=0;
white[ 3432 ]<=0;
white[ 3433 ]<=0;
white[ 3434 ]<=0;
white[ 3435 ]<=0;
white[ 3436 ]<=0;
white[ 3437 ]<=0;
white[ 3438 ]<=0;
white[ 3439 ]<=0;
white[ 3440 ]<=0;
white[ 3441 ]<=0;
white[ 3442 ]<=0;
white[ 3443 ]<=0;
white[ 3444 ]<=0;
white[ 3445 ]<=0;
white[ 3446 ]<=0;
white[ 3447 ]<=0;
white[ 3448 ]<=0;
white[ 3449 ]<=0;
white[ 3450 ]<=0;
white[ 3451 ]<=0;
white[ 3452 ]<=0;
white[ 3453 ]<=0;
white[ 3454 ]<=0;
white[ 3455 ]<=0;
```

```
white[ 3456 ]<=0;
white[ 3457 ]<=0;
white[ 3458 ]<=0;
white[ 3459 ]<=0;
white[ 3460 ]<=0;
white[ 3461 ]<=0;
white[ 3462 ]<=0;
white[ 3463 ]<=0;
white[ 3464 ]<=0;
white[ 3465 ]<=0;
white[ 3466 ]<=0;
white[ 3467 ]<=0;
white[ 3468 ]<=0;
white[ 3469 ]<=0;
white[ 3470 ]<=0;
white[ 3471 ]<=0;
white[ 3472 ]<=0;
white[ 3473 ]<=0;
white[ 3474 ]<=0;
white[ 3475 ]<=0;
white[ 3476 ]<=0;
white[ 3477 ]<=0;
white[ 3478 ]<=0;
white[ 3479 ]<=0;
white[ 3480 ]<=0;
white[ 3481 ]<=0;
white[ 3482 ]<=0;
white[ 3483 ]<=0;
white[ 3484 ]<=0;
white[ 3485 ]<=0;
white[ 3486 ]<=0;
white[ 3487 ]<=0;
white[ 3488 ]<=0;
white[ 3489 ]<=0;
white[ 3490 ]<=0;
white[ 3491 ]<=0;
white[ 3492 ]<=0;
white[ 3493 ]<=0;
white[ 3494 ]<=0;
white[ 3495 ]<=0;
white[ 3496 ]<=0;
white[ 3497 ]<=0;
white[ 3498 ]<=0;
white[ 3499 ]<=0;
white[ 3500 ]<=0;
white[ 3501 ]<=0;
white[ 3502 ]<=0;
white[ 3503 ]<=0;
white[ 3504 ]<=0;
white[ 3505 ]<=0;
white[ 3506 ]<=0;
white[ 3507 ]<=0;
white[ 3508 ]<=0;
white[ 3509 ]<=0;
white[ 3510 ]<=0;
white[ 3511 ]<=0;
white[ 3512 ]<=0;
white[ 3513 ]<=0;
```

```
white[ 3514 ]<=0;
white[ 3515 ]<=0;
white[ 3516 ]<=0;
white[ 3517 ]<=0;
white[ 3518 ]<=0;
white[ 3519 ]<=0;
white[ 3520 ]<=0;
white[ 3521 ]<=0;
white[ 3522 ]<=0;
white[ 3523 ]<=0;
white[ 3524 ]<=0;
white[ 3525 ]<=0;
white[ 3526 ]<=0;
white[ 3527 ]<=0;
white[ 3528 ]<=0;
white[ 3529 ]<=0;
white[ 3530 ]<=0;
white[ 3531 ]<=0;
white[ 3532 ]<=0;
white[ 3533 ]<=0;
white[ 3534 ]<=0;
white[ 3535 ]<=0;
white[ 3536 ]<=0;
white[ 3537 ]<=0;
white[ 3538 ]<=0;
white[ 3539 ]<=0;
white[ 3540 ]<=0;
white[ 3541 ]<=0;
white[ 3542 ]<=0;
white[ 3543 ]<=0;
white[ 3544 ]<=0;
white[ 3545 ]<=0;
white[ 3546 ]<=0;
white[ 3547 ]<=0;
white[ 3548 ]<=0;
white[ 3549 ]<=0;
white[ 3550 ]<=0;
white[ 3551 ]<=0;
white[ 3552 ]<=0;
white[ 3553 ]<=0;
white[ 3554 ]<=0;
white[ 3555 ]<=0;
white[ 3556 ]<=0;
white[ 3557 ]<=0;
white[ 3558 ]<=0;
white[ 3559 ]<=0;
white[ 3560 ]<=0;
white[ 3561 ]<=0;
white[ 3562 ]<=0;
white[ 3563 ]<=0;
white[ 3564 ]<=0;
white[ 3565 ]<=0;
white[ 3566 ]<=0;
white[ 3567 ]<=0;
white[ 3568 ]<=0;
white[ 3569 ]<=0;
white[ 3570 ]<=0;
white[ 3571 ]<=0;
```

```
white[ 3572 ]<=0;
white[ 3573 ]<=0;
white[ 3574 ]<=0;
white[ 3575 ]<=0;
white[ 3576 ]<=0;
white[ 3577 ]<=0;
white[ 3578 ]<=0;
white[ 3579 ]<=0;
white[ 3580 ]<=0;
white[ 3581 ]<=0;
white[ 3582 ]<=0;
white[ 3583 ]<=0;
white[ 3584 ]<=0;
white[ 3585 ]<=0;
white[ 3586 ]<=0;
white[ 3587 ]<=0;
white[ 3588 ]<=0;
white[ 3589 ]<=0;
white[ 3590 ]<=0;
white[ 3591 ]<=0;
white[ 3592 ]<=0;
white[ 3593 ]<=0;
white[ 3594 ]<=0;
white[ 3595 ]<=0;
white[ 3596 ]<=0;
white[ 3597 ]<=0;
white[ 3598 ]<=0;
white[ 3599 ]<=0;
end
    //coin
    18'd96239: begin
    white[ 0 ]<=0;
    white[ 1 ]<=0;
    white[ 2 ]<=0;
    white[ 3 ]<=0;
    white[ 4 ]<=0;
    white[ 5 ]<=0;
    white[ 6 ]<=0;
    white[ 7 ]<=0;
    white[ 8 ]<=0;
    white[ 9 ]<=0;
    white[ 10 ]<=0;
    white[ 11 ]<=0;
    white[ 12 ]<=0;
    white[ 13 ]<=0;
    white[ 14 ]<=0;
    white[ 15 ]<=0;
    white[ 16 ]<=0;
    white[ 17 ]<=0;
    white[ 18 ]<=0;
    white[ 19 ]<=0;
    white[ 20 ]<=0;
    white[ 21 ]<=0;
    white[ 22 ]<=0;
    white[ 23 ]<=1;
    white[ 24 ]<=1;
    white[ 25 ]<=1;
    white[ 26 ]<=1;
```

```
white[ 27 ]<=1;
white[ 28 ]<=1;
white[ 29 ]<=1;
white[ 30 ]<=1;
white[ 31 ]<=1;
white[ 32 ]<=1;
white[ 33 ]<=1;
white[ 34 ]<=1;
white[ 35 ]<=1;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=1;
white[ 80 ]<=1;
white[ 81 ]<=1;
white[ 82 ]<=1;
white[ 83 ]<=1;
white[ 84 ]<=1;
```

```
white[ 85 ]<=1;
white[ 86 ]<=1;
white[ 87 ]<=1;
white[ 88 ]<=1;
white[ 89 ]<=1;
white[ 90 ]<=1;
white[ 91 ]<=1;
white[ 92 ]<=1;
white[ 93 ]<=1;
white[ 94 ]<=1;
white[ 95 ]<=1;
white[ 96 ]<=1;
white[ 97 ]<=1;
white[ 98 ]<=1;
white[ 99 ]<=1;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=1;
white[ 138 ]<=1;
white[ 139 ]<=1;
white[ 140 ]<=1;
white[ 141 ]<=1;
white[ 142 ]<=1;
```

```
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=1;
white[ 155 ]<=1;
white[ 156 ]<=1;
white[ 157 ]<=1;
white[ 158 ]<=1;
white[ 159 ]<=1;
white[ 160 ]<=1;
white[ 161 ]<=1;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=0;
white[ 168 ]<=0;
white[ 169 ]<=0;
white[ 170 ]<=0;
white[ 171 ]<=0;
white[ 172 ]<=0;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=0;
white[ 180 ]<=0;
white[ 181 ]<=0;
white[ 182 ]<=0;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=1;
white[ 196 ]<=1;
white[ 197 ]<=1;
white[ 198 ]<=1;
white[ 199 ]<=1;
white[ 200 ]<=1;
```

```
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=1;
white[ 221 ]<=1;
white[ 222 ]<=1;
white[ 223 ]<=1;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=0;
white[ 230 ]<=0;
white[ 231 ]<=0;
white[ 232 ]<=0;
white[ 233 ]<=0;
white[ 234 ]<=0;
white[ 235 ]<=0;
white[ 236 ]<=0;
white[ 237 ]<=0;
white[ 238 ]<=0;
white[ 239 ]<=0;
white[ 240 ]<=0;
white[ 241 ]<=0;
white[ 242 ]<=0;
white[ 243 ]<=0;
white[ 244 ]<=0;
white[ 245 ]<=0;
white[ 246 ]<=0;
white[ 247 ]<=0;
white[ 248 ]<=0;
white[ 249 ]<=0;
white[ 250 ]<=0;
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=1;
white[ 254 ]<=1;
white[ 255 ]<=1;
white[ 256 ]<=1;
white[ 257 ]<=1;
white[ 258 ]<=1;
```

```
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=1;
white[ 283 ]<=1;
white[ 284 ]<=1;
white[ 285 ]<=1;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=0;
white[ 294 ]<=0;
white[ 295 ]<=0;
white[ 296 ]<=0;
white[ 297 ]<=0;
white[ 298 ]<=0;
white[ 299 ]<=0;
white[ 300 ]<=0;
white[ 301 ]<=0;
white[ 302 ]<=0;
white[ 303 ]<=0;
white[ 304 ]<=0;
white[ 305 ]<=0;
white[ 306 ]<=0;
white[ 307 ]<=0;
white[ 308 ]<=0;
white[ 309 ]<=0;
white[ 310 ]<=0;
white[ 311 ]<=0;
white[ 312 ]<=1;
white[ 313 ]<=1;
white[ 314 ]<=1;
white[ 315 ]<=1;
white[ 316 ]<=1;
```

```
white[ 317 ]<=1;
white[ 318 ]<=1;
white[ 319 ]<=1;
white[ 320 ]<=1;
white[ 321 ]<=1;
white[ 322 ]<=1;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=1;
white[ 344 ]<=1;
white[ 345 ]<=1;
white[ 346 ]<=1;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=0;
white[ 358 ]<=0;
white[ 359 ]<=0;
white[ 360 ]<=0;
white[ 361 ]<=0;
white[ 362 ]<=0;
white[ 363 ]<=0;
white[ 364 ]<=0;
white[ 365 ]<=0;
white[ 366 ]<=0;
white[ 367 ]<=0;
white[ 368 ]<=0;
white[ 369 ]<=0;
white[ 370 ]<=0;
white[ 371 ]<=1;
white[ 372 ]<=1;
white[ 373 ]<=1;
white[ 374 ]<=1;
```

```
white[ 375 ]<=1;
white[ 376 ]<=1;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=1;
white[ 383 ]<=1;
white[ 384 ]<=1;
white[ 385 ]<=1;
white[ 386 ]<=1;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=1;
white[ 405 ]<=1;
white[ 406 ]<=1;
white[ 407 ]<=1;
white[ 408 ]<=1;
white[ 409 ]<=0;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=0;
white[ 423 ]<=0;
white[ 424 ]<=0;
white[ 425 ]<=0;
white[ 426 ]<=0;
white[ 427 ]<=0;
white[ 428 ]<=0;
white[ 429 ]<=0;
white[ 430 ]<=1;
white[ 431 ]<=1;
white[ 432 ]<=1;
```

```
white[ 433 ]<=1;
white[ 434 ]<=1;
white[ 435 ]<=1;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=1;
white[ 447 ]<=1;
white[ 448 ]<=1;
white[ 449 ]<=1;
white[ 450 ]<=1;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=1;
white[ 466 ]<=1;
white[ 467 ]<=1;
white[ 468 ]<=1;
white[ 469 ]<=1;
white[ 470 ]<=0;
white[ 471 ]<=0;
white[ 472 ]<=0;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=0;
white[ 487 ]<=0;
white[ 488 ]<=1;
white[ 489 ]<=1;
white[ 490 ]<=1;
```

```
white[ 491 ]<=1;
white[ 492 ]<=1;
white[ 493 ]<=1;
white[ 494 ]<=1;
white[ 495 ]<=1;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=1;
white[ 511 ]<=1;
white[ 512 ]<=1;
white[ 513 ]<=1;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=1;
white[ 527 ]<=1;
white[ 528 ]<=1;
white[ 529 ]<=1;
white[ 530 ]<=1;
white[ 531 ]<=0;
white[ 532 ]<=0;
white[ 533 ]<=0;
white[ 534 ]<=0;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=1;
white[ 548 ]<=1;
```

```
white[ 549 ]<=1;
white[ 550 ]<=1;
white[ 551 ]<=1;
white[ 552 ]<=1;
white[ 553 ]<=1;
white[ 554 ]<=1;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=1;
white[ 576 ]<=1;
white[ 577 ]<=1;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=1;
white[ 587 ]<=1;
white[ 588 ]<=1;
white[ 589 ]<=1;
white[ 590 ]<=1;
white[ 591 ]<=1;
white[ 592 ]<=0;
white[ 593 ]<=0;
white[ 594 ]<=0;
white[ 595 ]<=0;
white[ 596 ]<=0;
white[ 597 ]<=0;
white[ 598 ]<=0;
white[ 599 ]<=0;
white[ 600 ]<=0;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
```

```
white[ 607 ]<=1;
white[ 608 ]<=1;
white[ 609 ]<=1;
white[ 610 ]<=1;
white[ 611 ]<=1;
white[ 612 ]<=1;
white[ 613 ]<=1;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=1;
white[ 640 ]<=1;
white[ 641 ]<=1;
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=1;
white[ 648 ]<=1;
white[ 649 ]<=1;
white[ 650 ]<=1;
white[ 651 ]<=1;
white[ 652 ]<=1;
white[ 653 ]<=0;
white[ 654 ]<=0;
white[ 655 ]<=0;
white[ 656 ]<=0;
white[ 657 ]<=0;
white[ 658 ]<=0;
white[ 659 ]<=0;
white[ 660 ]<=0;
white[ 661 ]<=0;
white[ 662 ]<=0;
white[ 663 ]<=0;
white[ 664 ]<=0;
```

```
white[ 665 ]<=0;
white[ 666 ]<=1;
white[ 667 ]<=1;
white[ 668 ]<=1;
white[ 669 ]<=1;
white[ 670 ]<=1;
white[ 671 ]<=1;
white[ 672 ]<=1;
white[ 673 ]<=1;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=1;
white[ 703 ]<=1;
white[ 704 ]<=1;
white[ 705 ]<=1;
white[ 706 ]<=1;
white[ 707 ]<=1;
white[ 708 ]<=1;
white[ 709 ]<=1;
white[ 710 ]<=1;
white[ 711 ]<=1;
white[ 712 ]<=1;
white[ 713 ]<=1;
white[ 714 ]<=0;
white[ 715 ]<=0;
white[ 716 ]<=0;
white[ 717 ]<=0;
white[ 718 ]<=0;
white[ 719 ]<=0;
white[ 720 ]<=0;
white[ 721 ]<=0;
white[ 722 ]<=0;
```

```
white[ 723 ]<=0;
white[ 724 ]<=0;
white[ 725 ]<=1;
white[ 726 ]<=1;
white[ 727 ]<=1;
white[ 728 ]<=1;
white[ 729 ]<=1;
white[ 730 ]<=1;
white[ 731 ]<=1;
white[ 732 ]<=1;
white[ 733 ]<=1;
white[ 734 ]<=1;
white[ 735 ]<=1;
white[ 736 ]<=1;
white[ 737 ]<=1;
white[ 738 ]<=1;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=1;
white[ 768 ]<=1;
white[ 769 ]<=1;
white[ 770 ]<=1;
white[ 771 ]<=1;
white[ 772 ]<=1;
white[ 773 ]<=1;
white[ 774 ]<=0;
white[ 775 ]<=0;
white[ 776 ]<=0;
white[ 777 ]<=0;
white[ 778 ]<=0;
white[ 779 ]<=0;
white[ 780 ]<=0;
```

```
white[ 781 ]<=0;
white[ 782 ]<=0;
white[ 783 ]<=0;
white[ 784 ]<=1;
white[ 785 ]<=1;
white[ 786 ]<=1;
white[ 787 ]<=1;
white[ 788 ]<=1;
white[ 789 ]<=1;
white[ 790 ]<=1;
white[ 791 ]<=1;
white[ 792 ]<=1;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=1;
white[ 800 ]<=1;
white[ 801 ]<=1;
white[ 802 ]<=1;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=1;
white[ 813 ]<=1;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=1;
white[ 829 ]<=1;
white[ 830 ]<=1;
white[ 831 ]<=1;
white[ 832 ]<=1;
white[ 833 ]<=1;
white[ 834 ]<=1;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=0;
```

```
white[ 839 ]<=0;
white[ 840 ]<=0;
white[ 841 ]<=0;
white[ 842 ]<=0;
white[ 843 ]<=0;
white[ 844 ]<=1;
white[ 845 ]<=1;
white[ 846 ]<=1;
white[ 847 ]<=1;
white[ 848 ]<=1;
white[ 849 ]<=1;
white[ 850 ]<=1;
white[ 851 ]<=1;
white[ 852 ]<=1;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=1;
white[ 863 ]<=1;
white[ 864 ]<=1;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=1;
white[ 872 ]<=1;
white[ 873 ]<=1;
white[ 874 ]<=1;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=1;
white[ 889 ]<=1;
white[ 890 ]<=1;
white[ 891 ]<=1;
white[ 892 ]<=1;
white[ 893 ]<=1;
white[ 894 ]<=1;
white[ 895 ]<=1;
white[ 896 ]<=0;
```

```
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=1;
white[ 904 ]<=1;
white[ 905 ]<=1;
white[ 906 ]<=1;
white[ 907 ]<=1;
white[ 908 ]<=1;
white[ 909 ]<=1;
white[ 910 ]<=1;
white[ 911 ]<=1;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=1;
white[ 927 ]<=1;
white[ 928 ]<=1;
white[ 929 ]<=1;
white[ 930 ]<=1;
white[ 931 ]<=1;
white[ 932 ]<=1;
white[ 933 ]<=1;
white[ 934 ]<=1;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=1;
white[ 950 ]<=1;
white[ 951 ]<=1;
white[ 952 ]<=1;
white[ 953 ]<=1;
white[ 954 ]<=1;
```

```
white[ 955 ]<=1;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=1;
white[ 964 ]<=1;
white[ 965 ]<=1;
white[ 966 ]<=1;
white[ 967 ]<=1;
white[ 968 ]<=1;
white[ 969 ]<=1;
white[ 970 ]<=1;
white[ 971 ]<=1;
white[ 972 ]<=1;
white[ 973 ]<=1;
white[ 974 ]<=1;
white[ 975 ]<=1;
white[ 976 ]<=1;
white[ 977 ]<=1;
white[ 978 ]<=1;
white[ 979 ]<=1;
white[ 980 ]<=1;
white[ 981 ]<=1;
white[ 982 ]<=1;
white[ 983 ]<=1;
white[ 984 ]<=1;
white[ 985 ]<=1;
white[ 986 ]<=1;
white[ 987 ]<=1;
white[ 988 ]<=1;
white[ 989 ]<=1;
white[ 990 ]<=1;
white[ 991 ]<=1;
white[ 992 ]<=1;
white[ 993 ]<=1;
white[ 994 ]<=1;
white[ 995 ]<=1;
white[ 996 ]<=1;
white[ 997 ]<=1;
white[ 998 ]<=1;
white[ 999 ]<=1;
white[ 1000 ]<=1;
white[ 1001 ]<=1;
white[ 1002 ]<=1;
white[ 1003 ]<=1;
white[ 1004 ]<=1;
white[ 1005 ]<=1;
white[ 1006 ]<=1;
white[ 1007 ]<=1;
white[ 1008 ]<=1;
white[ 1009 ]<=1;
white[ 1010 ]<=1;
white[ 1011 ]<=1;
white[ 1012 ]<=1;
```

```
white[ 1013 ]<=1;
white[ 1014 ]<=1;
white[ 1015 ]<=1;
white[ 1016 ]<=1;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=1;
white[ 1023 ]<=1;
white[ 1024 ]<=1;
white[ 1025 ]<=1;
white[ 1026 ]<=1;
white[ 1027 ]<=1;
white[ 1028 ]<=1;
white[ 1029 ]<=1;
white[ 1030 ]<=1;
white[ 1031 ]<=1;
white[ 1032 ]<=1;
white[ 1033 ]<=1;
white[ 1034 ]<=1;
white[ 1035 ]<=1;
white[ 1036 ]<=1;
white[ 1037 ]<=1;
white[ 1038 ]<=1;
white[ 1039 ]<=1;
white[ 1040 ]<=1;
white[ 1041 ]<=1;
white[ 1042 ]<=1;
white[ 1043 ]<=1;
white[ 1044 ]<=1;
white[ 1045 ]<=1;
white[ 1046 ]<=1;
white[ 1047 ]<=1;
white[ 1048 ]<=1;
white[ 1049 ]<=1;
white[ 1050 ]<=1;
white[ 1051 ]<=1;
white[ 1052 ]<=1;
white[ 1053 ]<=1;
white[ 1054 ]<=1;
white[ 1055 ]<=1;
white[ 1056 ]<=1;
white[ 1057 ]<=1;
white[ 1058 ]<=1;
white[ 1059 ]<=1;
white[ 1060 ]<=1;
white[ 1061 ]<=1;
white[ 1062 ]<=1;
white[ 1063 ]<=1;
white[ 1064 ]<=1;
white[ 1065 ]<=1;
white[ 1066 ]<=1;
white[ 1067 ]<=1;
white[ 1068 ]<=1;
white[ 1069 ]<=1;
white[ 1070 ]<=1;
```

```
white[ 1071 ]<=1;
white[ 1072 ]<=1;
white[ 1073 ]<=1;
white[ 1074 ]<=1;
white[ 1075 ]<=1;
white[ 1076 ]<=1;
white[ 1077 ]<=0;
white[ 1078 ]<=0;
white[ 1079 ]<=0;
white[ 1080 ]<=0;
white[ 1081 ]<=0;
white[ 1082 ]<=1;
white[ 1083 ]<=1;
white[ 1084 ]<=1;
white[ 1085 ]<=1;
white[ 1086 ]<=1;
white[ 1087 ]<=1;
white[ 1088 ]<=1;
white[ 1089 ]<=1;
white[ 1090 ]<=1;
white[ 1091 ]<=1;
white[ 1092 ]<=1;
white[ 1093 ]<=1;
white[ 1094 ]<=1;
white[ 1095 ]<=1;
white[ 1096 ]<=1;
white[ 1097 ]<=1;
white[ 1098 ]<=1;
white[ 1099 ]<=1;
white[ 1100 ]<=1;
white[ 1101 ]<=1;
white[ 1102 ]<=1;
white[ 1103 ]<=1;
white[ 1104 ]<=1;
white[ 1105 ]<=1;
white[ 1106 ]<=1;
white[ 1107 ]<=1;
white[ 1108 ]<=1;
white[ 1109 ]<=1;
white[ 1110 ]<=1;
white[ 1111 ]<=1;
white[ 1112 ]<=1;
white[ 1113 ]<=1;
white[ 1114 ]<=1;
white[ 1115 ]<=1;
white[ 1116 ]<=1;
white[ 1117 ]<=1;
white[ 1118 ]<=1;
white[ 1119 ]<=1;
white[ 1120 ]<=1;
white[ 1121 ]<=1;
white[ 1122 ]<=1;
white[ 1123 ]<=1;
white[ 1124 ]<=1;
white[ 1125 ]<=1;
white[ 1126 ]<=1;
white[ 1127 ]<=1;
white[ 1128 ]<=1;
```

```
white[ 1129 ]<=1;
white[ 1130 ]<=1;
white[ 1131 ]<=1;
white[ 1132 ]<=1;
white[ 1133 ]<=1;
white[ 1134 ]<=1;
white[ 1135 ]<=1;
white[ 1136 ]<=1;
white[ 1137 ]<=1;
white[ 1138 ]<=0;
white[ 1139 ]<=0;
white[ 1140 ]<=0;
white[ 1141 ]<=1;
white[ 1142 ]<=1;
white[ 1143 ]<=1;
white[ 1144 ]<=1;
white[ 1145 ]<=1;
white[ 1146 ]<=1;
white[ 1147 ]<=1;
white[ 1148 ]<=1;
white[ 1149 ]<=1;
white[ 1150 ]<=1;
white[ 1151 ]<=1;
white[ 1152 ]<=1;
white[ 1153 ]<=1;
white[ 1154 ]<=1;
white[ 1155 ]<=1;
white[ 1156 ]<=1;
white[ 1157 ]<=1;
white[ 1158 ]<=1;
white[ 1159 ]<=1;
white[ 1160 ]<=1;
white[ 1161 ]<=1;
white[ 1162 ]<=1;
white[ 1163 ]<=1;
white[ 1164 ]<=1;
white[ 1165 ]<=1;
white[ 1166 ]<=1;
white[ 1167 ]<=1;
white[ 1168 ]<=1;
white[ 1169 ]<=1;
white[ 1170 ]<=1;
white[ 1171 ]<=1;
white[ 1172 ]<=1;
white[ 1173 ]<=1;
white[ 1174 ]<=1;
white[ 1175 ]<=1;
white[ 1176 ]<=1;
white[ 1177 ]<=1;
white[ 1178 ]<=1;
white[ 1179 ]<=1;
white[ 1180 ]<=1;
white[ 1181 ]<=1;
white[ 1182 ]<=1;
white[ 1183 ]<=1;
white[ 1184 ]<=1;
white[ 1185 ]<=1;
white[ 1186 ]<=1;
```

```
white[ 1187 ]<=1;
white[ 1188 ]<=1;
white[ 1189 ]<=1;
white[ 1190 ]<=1;
white[ 1191 ]<=1;
white[ 1192 ]<=1;
white[ 1193 ]<=1;
white[ 1194 ]<=1;
white[ 1195 ]<=1;
white[ 1196 ]<=1;
white[ 1197 ]<=1;
white[ 1198 ]<=0;
white[ 1199 ]<=0;
white[ 1200 ]<=0;
white[ 1201 ]<=1;
white[ 1202 ]<=1;
white[ 1203 ]<=1;
white[ 1204 ]<=1;
white[ 1205 ]<=1;
white[ 1206 ]<=1;
white[ 1207 ]<=1;
white[ 1208 ]<=1;
white[ 1209 ]<=1;
white[ 1210 ]<=1;
white[ 1211 ]<=1;
white[ 1212 ]<=1;
white[ 1213 ]<=1;
white[ 1214 ]<=1;
white[ 1215 ]<=1;
white[ 1216 ]<=1;
white[ 1217 ]<=1;
white[ 1218 ]<=1;
white[ 1219 ]<=1;
white[ 1220 ]<=1;
white[ 1221 ]<=1;
white[ 1222 ]<=1;
white[ 1223 ]<=1;
white[ 1224 ]<=1;
white[ 1225 ]<=1;
white[ 1226 ]<=1;
white[ 1227 ]<=1;
white[ 1228 ]<=1;
white[ 1229 ]<=1;
white[ 1230 ]<=1;
white[ 1231 ]<=1;
white[ 1232 ]<=1;
white[ 1233 ]<=1;
white[ 1234 ]<=1;
white[ 1235 ]<=1;
white[ 1236 ]<=1;
white[ 1237 ]<=1;
white[ 1238 ]<=1;
white[ 1239 ]<=1;
white[ 1240 ]<=1;
white[ 1241 ]<=1;
white[ 1242 ]<=1;
white[ 1243 ]<=1;
white[ 1244 ]<=1;
```

```
white[ 1245 ]<=1;
white[ 1246 ]<=1;
white[ 1247 ]<=1;
white[ 1248 ]<=1;
white[ 1249 ]<=1;
white[ 1250 ]<=1;
white[ 1251 ]<=1;
white[ 1252 ]<=1;
white[ 1253 ]<=1;
white[ 1254 ]<=1;
white[ 1255 ]<=1;
white[ 1256 ]<=1;
white[ 1257 ]<=1;
white[ 1258 ]<=1;
white[ 1259 ]<=0;
white[ 1260 ]<=0;
white[ 1261 ]<=1;
white[ 1262 ]<=1;
white[ 1263 ]<=1;
white[ 1264 ]<=1;
white[ 1265 ]<=1;
white[ 1266 ]<=1;
white[ 1267 ]<=1;
white[ 1268 ]<=1;
white[ 1269 ]<=1;
white[ 1270 ]<=1;
white[ 1271 ]<=1;
white[ 1272 ]<=1;
white[ 1273 ]<=1;
white[ 1274 ]<=1;
white[ 1275 ]<=1;
white[ 1276 ]<=1;
white[ 1277 ]<=1;
white[ 1278 ]<=1;
white[ 1279 ]<=1;
white[ 1280 ]<=1;
white[ 1281 ]<=1;
white[ 1282 ]<=1;
white[ 1283 ]<=1;
white[ 1284 ]<=1;
white[ 1285 ]<=1;
white[ 1286 ]<=1;
white[ 1287 ]<=1;
white[ 1288 ]<=1;
white[ 1289 ]<=1;
white[ 1290 ]<=1;
white[ 1291 ]<=1;
white[ 1292 ]<=1;
white[ 1293 ]<=1;
white[ 1294 ]<=1;
white[ 1295 ]<=1;
white[ 1296 ]<=1;
white[ 1297 ]<=1;
white[ 1298 ]<=1;
white[ 1299 ]<=1;
white[ 1300 ]<=1;
white[ 1301 ]<=1;
white[ 1302 ]<=1;
```

```
white[ 1303 ]<=1;
white[ 1304 ]<=1;
white[ 1305 ]<=1;
white[ 1306 ]<=1;
white[ 1307 ]<=1;
white[ 1308 ]<=1;
white[ 1309 ]<=1;
white[ 1310 ]<=1;
white[ 1311 ]<=1;
white[ 1312 ]<=1;
white[ 1313 ]<=1;
white[ 1314 ]<=1;
white[ 1315 ]<=1;
white[ 1316 ]<=1;
white[ 1317 ]<=1;
white[ 1318 ]<=1;
white[ 1319 ]<=0;
white[ 1320 ]<=0;
white[ 1321 ]<=1;
white[ 1322 ]<=1;
white[ 1323 ]<=1;
white[ 1324 ]<=1;
white[ 1325 ]<=1;
white[ 1326 ]<=1;
white[ 1327 ]<=1;
white[ 1328 ]<=1;
white[ 1329 ]<=1;
white[ 1330 ]<=1;
white[ 1331 ]<=1;
white[ 1332 ]<=1;
white[ 1333 ]<=1;
white[ 1334 ]<=1;
white[ 1335 ]<=1;
white[ 1336 ]<=1;
white[ 1337 ]<=1;
white[ 1338 ]<=1;
white[ 1339 ]<=1;
white[ 1340 ]<=1;
white[ 1341 ]<=1;
white[ 1342 ]<=1;
white[ 1343 ]<=1;
white[ 1344 ]<=1;
white[ 1345 ]<=1;
white[ 1346 ]<=1;
white[ 1347 ]<=1;
white[ 1348 ]<=1;
white[ 1349 ]<=1;
white[ 1350 ]<=1;
white[ 1351 ]<=1;
white[ 1352 ]<=1;
white[ 1353 ]<=1;
white[ 1354 ]<=1;
white[ 1355 ]<=1;
white[ 1356 ]<=1;
white[ 1357 ]<=1;
white[ 1358 ]<=1;
white[ 1359 ]<=1;
white[ 1360 ]<=1;
```

```
white[ 1361 ]<=1;
white[ 1362 ]<=1;
white[ 1363 ]<=1;
white[ 1364 ]<=1;
white[ 1365 ]<=1;
white[ 1366 ]<=1;
white[ 1367 ]<=1;
white[ 1368 ]<=1;
white[ 1369 ]<=1;
white[ 1370 ]<=1;
white[ 1371 ]<=1;
white[ 1372 ]<=1;
white[ 1373 ]<=1;
white[ 1374 ]<=1;
white[ 1375 ]<=1;
white[ 1376 ]<=1;
white[ 1377 ]<=1;
white[ 1378 ]<=1;
white[ 1379 ]<=0;
white[ 1380 ]<=1;
white[ 1381 ]<=1;
white[ 1382 ]<=1;
white[ 1383 ]<=1;
white[ 1384 ]<=1;
white[ 1385 ]<=1;
white[ 1386 ]<=1;
white[ 1387 ]<=1;
white[ 1388 ]<=1;
white[ 1389 ]<=1;
white[ 1390 ]<=1;
white[ 1391 ]<=1;
white[ 1392 ]<=1;
white[ 1393 ]<=1;
white[ 1394 ]<=1;
white[ 1395 ]<=1;
white[ 1396 ]<=1;
white[ 1397 ]<=1;
white[ 1398 ]<=1;
white[ 1399 ]<=1;
white[ 1400 ]<=1;
white[ 1401 ]<=1;
white[ 1402 ]<=1;
white[ 1403 ]<=1;
white[ 1404 ]<=1;
white[ 1405 ]<=1;
white[ 1406 ]<=1;
white[ 1407 ]<=1;
white[ 1408 ]<=1;
white[ 1409 ]<=1;
white[ 1410 ]<=1;
white[ 1411 ]<=1;
white[ 1412 ]<=1;
white[ 1413 ]<=1;
white[ 1414 ]<=1;
white[ 1415 ]<=1;
white[ 1416 ]<=1;
white[ 1417 ]<=1;
white[ 1418 ]<=1;
```

```
white[ 1419 ]<=1;
white[ 1420 ]<=1;
white[ 1421 ]<=1;
white[ 1422 ]<=1;
white[ 1423 ]<=1;
white[ 1424 ]<=1;
white[ 1425 ]<=1;
white[ 1426 ]<=1;
white[ 1427 ]<=1;
white[ 1428 ]<=1;
white[ 1429 ]<=1;
white[ 1430 ]<=1;
white[ 1431 ]<=1;
white[ 1432 ]<=1;
white[ 1433 ]<=1;
white[ 1434 ]<=1;
white[ 1435 ]<=1;
white[ 1436 ]<=1;
white[ 1437 ]<=1;
white[ 1438 ]<=1;
white[ 1439 ]<=1;
white[ 1440 ]<=1;
white[ 1441 ]<=1;
white[ 1442 ]<=1;
white[ 1443 ]<=1;
white[ 1444 ]<=1;
white[ 1445 ]<=1;
white[ 1446 ]<=1;
white[ 1447 ]<=1;
white[ 1448 ]<=1;
white[ 1449 ]<=1;
white[ 1450 ]<=1;
white[ 1451 ]<=1;
white[ 1452 ]<=1;
white[ 1453 ]<=1;
white[ 1454 ]<=1;
white[ 1455 ]<=1;
white[ 1456 ]<=1;
white[ 1457 ]<=1;
white[ 1458 ]<=1;
white[ 1459 ]<=1;
white[ 1460 ]<=1;
white[ 1461 ]<=1;
white[ 1462 ]<=1;
white[ 1463 ]<=1;
white[ 1464 ]<=1;
white[ 1465 ]<=1;
white[ 1466 ]<=1;
white[ 1467 ]<=1;
white[ 1468 ]<=1;
white[ 1469 ]<=1;
white[ 1470 ]<=1;
white[ 1471 ]<=1;
white[ 1472 ]<=1;
white[ 1473 ]<=1;
white[ 1474 ]<=1;
white[ 1475 ]<=1;
white[ 1476 ]<=1;
```

```
white[ 1477 ]<=1;
white[ 1478 ]<=1;
white[ 1479 ]<=1;
white[ 1480 ]<=1;
white[ 1481 ]<=1;
white[ 1482 ]<=1;
white[ 1483 ]<=1;
white[ 1484 ]<=1;
white[ 1485 ]<=1;
white[ 1486 ]<=1;
white[ 1487 ]<=1;
white[ 1488 ]<=1;
white[ 1489 ]<=1;
white[ 1490 ]<=1;
white[ 1491 ]<=1;
white[ 1492 ]<=1;
white[ 1493 ]<=1;
white[ 1494 ]<=1;
white[ 1495 ]<=1;
white[ 1496 ]<=1;
white[ 1497 ]<=1;
white[ 1498 ]<=1;
white[ 1499 ]<=1;
white[ 1500 ]<=1;
white[ 1501 ]<=1;
white[ 1502 ]<=1;
white[ 1503 ]<=1;
white[ 1504 ]<=1;
white[ 1505 ]<=1;
white[ 1506 ]<=1;
white[ 1507 ]<=1;
white[ 1508 ]<=1;
white[ 1509 ]<=1;
white[ 1510 ]<=1;
white[ 1511 ]<=1;
white[ 1512 ]<=1;
white[ 1513 ]<=1;
white[ 1514 ]<=1;
white[ 1515 ]<=1;
white[ 1516 ]<=1;
white[ 1517 ]<=1;
white[ 1518 ]<=1;
white[ 1519 ]<=1;
white[ 1520 ]<=1;
white[ 1521 ]<=1;
white[ 1522 ]<=1;
white[ 1523 ]<=1;
white[ 1524 ]<=1;
white[ 1525 ]<=1;
white[ 1526 ]<=1;
white[ 1527 ]<=1;
white[ 1528 ]<=1;
white[ 1529 ]<=1;
white[ 1530 ]<=1;
white[ 1531 ]<=1;
white[ 1532 ]<=1;
white[ 1533 ]<=1;
white[ 1534 ]<=1;
```

```
white[ 1535 ]<=1;
white[ 1536 ]<=1;
white[ 1537 ]<=1;
white[ 1538 ]<=1;
white[ 1539 ]<=1;
white[ 1540 ]<=1;
white[ 1541 ]<=1;
white[ 1542 ]<=1;
white[ 1543 ]<=1;
white[ 1544 ]<=1;
white[ 1545 ]<=1;
white[ 1546 ]<=1;
white[ 1547 ]<=1;
white[ 1548 ]<=1;
white[ 1549 ]<=1;
white[ 1550 ]<=1;
white[ 1551 ]<=1;
white[ 1552 ]<=1;
white[ 1553 ]<=1;
white[ 1554 ]<=1;
white[ 1555 ]<=1;
white[ 1556 ]<=1;
white[ 1557 ]<=1;
white[ 1558 ]<=1;
white[ 1559 ]<=1;
white[ 1560 ]<=1;
white[ 1561 ]<=1;
white[ 1562 ]<=1;
white[ 1563 ]<=1;
white[ 1564 ]<=1;
white[ 1565 ]<=1;
white[ 1566 ]<=1;
white[ 1567 ]<=1;
white[ 1568 ]<=1;
white[ 1569 ]<=1;
white[ 1570 ]<=1;
white[ 1571 ]<=1;
white[ 1572 ]<=1;
white[ 1573 ]<=1;
white[ 1574 ]<=1;
white[ 1575 ]<=1;
white[ 1576 ]<=1;
white[ 1577 ]<=1;
white[ 1578 ]<=1;
white[ 1579 ]<=1;
white[ 1580 ]<=1;
white[ 1581 ]<=1;
white[ 1582 ]<=1;
white[ 1583 ]<=1;
white[ 1584 ]<=1;
white[ 1585 ]<=1;
white[ 1586 ]<=1;
white[ 1587 ]<=1;
white[ 1588 ]<=1;
white[ 1589 ]<=1;
white[ 1590 ]<=1;
white[ 1591 ]<=1;
white[ 1592 ]<=1;
```

```
white[ 1593 ]<=1;
white[ 1594 ]<=1;
white[ 1595 ]<=1;
white[ 1596 ]<=1;
white[ 1597 ]<=1;
white[ 1598 ]<=1;
white[ 1599 ]<=1;
white[ 1600 ]<=1;
white[ 1601 ]<=1;
white[ 1602 ]<=1;
white[ 1603 ]<=1;
white[ 1604 ]<=1;
white[ 1605 ]<=1;
white[ 1606 ]<=1;
white[ 1607 ]<=1;
white[ 1608 ]<=1;
white[ 1609 ]<=1;
white[ 1610 ]<=1;
white[ 1611 ]<=1;
white[ 1612 ]<=1;
white[ 1613 ]<=1;
white[ 1614 ]<=1;
white[ 1615 ]<=1;
white[ 1616 ]<=1;
white[ 1617 ]<=1;
white[ 1618 ]<=1;
white[ 1619 ]<=1;
white[ 1620 ]<=1;
white[ 1621 ]<=1;
white[ 1622 ]<=1;
white[ 1623 ]<=1;
white[ 1624 ]<=1;
white[ 1625 ]<=1;
white[ 1626 ]<=1;
white[ 1627 ]<=1;
white[ 1628 ]<=1;
white[ 1629 ]<=1;
white[ 1630 ]<=1;
white[ 1631 ]<=1;
white[ 1632 ]<=1;
white[ 1633 ]<=1;
white[ 1634 ]<=1;
white[ 1635 ]<=1;
white[ 1636 ]<=1;
white[ 1637 ]<=1;
white[ 1638 ]<=1;
white[ 1639 ]<=1;
white[ 1640 ]<=1;
white[ 1641 ]<=1;
white[ 1642 ]<=1;
white[ 1643 ]<=1;
white[ 1644 ]<=1;
white[ 1645 ]<=1;
white[ 1646 ]<=1;
white[ 1647 ]<=1;
white[ 1648 ]<=1;
white[ 1649 ]<=1;
white[ 1650 ]<=1;
```

```
white[ 1651 ]<=1;
white[ 1652 ]<=1;
white[ 1653 ]<=1;
white[ 1654 ]<=1;
white[ 1655 ]<=1;
white[ 1656 ]<=1;
white[ 1657 ]<=1;
white[ 1658 ]<=1;
white[ 1659 ]<=1;
white[ 1660 ]<=1;
white[ 1661 ]<=1;
white[ 1662 ]<=1;
white[ 1663 ]<=1;
white[ 1664 ]<=1;
white[ 1665 ]<=1;
white[ 1666 ]<=1;
white[ 1667 ]<=1;
white[ 1668 ]<=1;
white[ 1669 ]<=1;
white[ 1670 ]<=1;
white[ 1671 ]<=1;
white[ 1672 ]<=1;
white[ 1673 ]<=1;
white[ 1674 ]<=1;
white[ 1675 ]<=1;
white[ 1676 ]<=1;
white[ 1677 ]<=1;
white[ 1678 ]<=1;
white[ 1679 ]<=1;
white[ 1680 ]<=1;
white[ 1681 ]<=1;
white[ 1682 ]<=1;
white[ 1683 ]<=1;
white[ 1684 ]<=1;
white[ 1685 ]<=1;
white[ 1686 ]<=1;
white[ 1687 ]<=1;
white[ 1688 ]<=1;
white[ 1689 ]<=1;
white[ 1690 ]<=1;
white[ 1691 ]<=1;
white[ 1692 ]<=1;
white[ 1693 ]<=1;
white[ 1694 ]<=1;
white[ 1695 ]<=1;
white[ 1696 ]<=1;
white[ 1697 ]<=1;
white[ 1698 ]<=1;
white[ 1699 ]<=1;
white[ 1700 ]<=1;
white[ 1701 ]<=1;
white[ 1702 ]<=1;
white[ 1703 ]<=1;
white[ 1704 ]<=1;
white[ 1705 ]<=1;
white[ 1706 ]<=1;
white[ 1707 ]<=1;
white[ 1708 ]<=1;
```

```
white[ 1709 ]<=1;
white[ 1710 ]<=1;
white[ 1711 ]<=1;
white[ 1712 ]<=1;
white[ 1713 ]<=1;
white[ 1714 ]<=1;
white[ 1715 ]<=1;
white[ 1716 ]<=1;
white[ 1717 ]<=1;
white[ 1718 ]<=1;
white[ 1719 ]<=1;
white[ 1720 ]<=1;
white[ 1721 ]<=1;
white[ 1722 ]<=1;
white[ 1723 ]<=1;
white[ 1724 ]<=1;
white[ 1725 ]<=1;
white[ 1726 ]<=1;
white[ 1727 ]<=1;
white[ 1728 ]<=1;
white[ 1729 ]<=1;
white[ 1730 ]<=1;
white[ 1731 ]<=1;
white[ 1732 ]<=1;
white[ 1733 ]<=1;
white[ 1734 ]<=1;
white[ 1735 ]<=1;
white[ 1736 ]<=1;
white[ 1737 ]<=1;
white[ 1738 ]<=1;
white[ 1739 ]<=1;
white[ 1740 ]<=1;
white[ 1741 ]<=1;
white[ 1742 ]<=1;
white[ 1743 ]<=1;
white[ 1744 ]<=1;
white[ 1745 ]<=1;
white[ 1746 ]<=1;
white[ 1747 ]<=1;
white[ 1748 ]<=1;
white[ 1749 ]<=1;
white[ 1750 ]<=1;
white[ 1751 ]<=1;
white[ 1752 ]<=1;
white[ 1753 ]<=1;
white[ 1754 ]<=1;
white[ 1755 ]<=1;
white[ 1756 ]<=1;
white[ 1757 ]<=1;
white[ 1758 ]<=1;
white[ 1759 ]<=1;
white[ 1760 ]<=1;
white[ 1761 ]<=1;
white[ 1762 ]<=1;
white[ 1763 ]<=1;
white[ 1764 ]<=1;
white[ 1765 ]<=1;
white[ 1766 ]<=1;
```

```
white[ 1767 ]<=1;
white[ 1768 ]<=1;
white[ 1769 ]<=1;
white[ 1770 ]<=1;
white[ 1771 ]<=1;
white[ 1772 ]<=1;
white[ 1773 ]<=1;
white[ 1774 ]<=1;
white[ 1775 ]<=1;
white[ 1776 ]<=1;
white[ 1777 ]<=1;
white[ 1778 ]<=1;
white[ 1779 ]<=1;
white[ 1780 ]<=1;
white[ 1781 ]<=1;
white[ 1782 ]<=1;
white[ 1783 ]<=1;
white[ 1784 ]<=1;
white[ 1785 ]<=1;
white[ 1786 ]<=1;
white[ 1787 ]<=1;
white[ 1788 ]<=1;
white[ 1789 ]<=1;
white[ 1790 ]<=1;
white[ 1791 ]<=1;
white[ 1792 ]<=1;
white[ 1793 ]<=1;
white[ 1794 ]<=1;
white[ 1795 ]<=1;
white[ 1796 ]<=1;
white[ 1797 ]<=1;
white[ 1798 ]<=1;
white[ 1799 ]<=1;
white[ 1800 ]<=1;
white[ 1801 ]<=1;
white[ 1802 ]<=1;
white[ 1803 ]<=1;
white[ 1804 ]<=1;
white[ 1805 ]<=1;
white[ 1806 ]<=1;
white[ 1807 ]<=1;
white[ 1808 ]<=1;
white[ 1809 ]<=1;
white[ 1810 ]<=1;
white[ 1811 ]<=1;
white[ 1812 ]<=1;
white[ 1813 ]<=1;
white[ 1814 ]<=1;
white[ 1815 ]<=1;
white[ 1816 ]<=1;
white[ 1817 ]<=1;
white[ 1818 ]<=1;
white[ 1819 ]<=1;
white[ 1820 ]<=1;
white[ 1821 ]<=1;
white[ 1822 ]<=1;
white[ 1823 ]<=1;
white[ 1824 ]<=1;
```

```
white[ 1825 ]<=1;
white[ 1826 ]<=1;
white[ 1827 ]<=1;
white[ 1828 ]<=1;
white[ 1829 ]<=1;
white[ 1830 ]<=1;
white[ 1831 ]<=1;
white[ 1832 ]<=1;
white[ 1833 ]<=1;
white[ 1834 ]<=1;
white[ 1835 ]<=1;
white[ 1836 ]<=1;
white[ 1837 ]<=1;
white[ 1838 ]<=1;
white[ 1839 ]<=1;
white[ 1840 ]<=1;
white[ 1841 ]<=1;
white[ 1842 ]<=1;
white[ 1843 ]<=1;
white[ 1844 ]<=1;
white[ 1845 ]<=1;
white[ 1846 ]<=1;
white[ 1847 ]<=1;
white[ 1848 ]<=1;
white[ 1849 ]<=1;
white[ 1850 ]<=1;
white[ 1851 ]<=1;
white[ 1852 ]<=1;
white[ 1853 ]<=1;
white[ 1854 ]<=1;
white[ 1855 ]<=1;
white[ 1856 ]<=1;
white[ 1857 ]<=1;
white[ 1858 ]<=1;
white[ 1859 ]<=1;
white[ 1860 ]<=1;
white[ 1861 ]<=1;
white[ 1862 ]<=1;
white[ 1863 ]<=1;
white[ 1864 ]<=1;
white[ 1865 ]<=1;
white[ 1866 ]<=1;
white[ 1867 ]<=1;
white[ 1868 ]<=1;
white[ 1869 ]<=1;
white[ 1870 ]<=1;
white[ 1871 ]<=1;
white[ 1872 ]<=1;
white[ 1873 ]<=1;
white[ 1874 ]<=1;
white[ 1875 ]<=1;
white[ 1876 ]<=1;
white[ 1877 ]<=1;
white[ 1878 ]<=1;
white[ 1879 ]<=1;
white[ 1880 ]<=1;
white[ 1881 ]<=1;
white[ 1882 ]<=1;
```

```
white[ 1883 ]<=1;
white[ 1884 ]<=1;
white[ 1885 ]<=1;
white[ 1886 ]<=1;
white[ 1887 ]<=1;
white[ 1888 ]<=1;
white[ 1889 ]<=1;
white[ 1890 ]<=1;
white[ 1891 ]<=1;
white[ 1892 ]<=1;
white[ 1893 ]<=1;
white[ 1894 ]<=1;
white[ 1895 ]<=1;
white[ 1896 ]<=1;
white[ 1897 ]<=1;
white[ 1898 ]<=1;
white[ 1899 ]<=1;
white[ 1900 ]<=1;
white[ 1901 ]<=1;
white[ 1902 ]<=1;
white[ 1903 ]<=1;
white[ 1904 ]<=1;
white[ 1905 ]<=1;
white[ 1906 ]<=1;
white[ 1907 ]<=1;
white[ 1908 ]<=1;
white[ 1909 ]<=1;
white[ 1910 ]<=1;
white[ 1911 ]<=1;
white[ 1912 ]<=1;
white[ 1913 ]<=1;
white[ 1914 ]<=1;
white[ 1915 ]<=1;
white[ 1916 ]<=1;
white[ 1917 ]<=1;
white[ 1918 ]<=1;
white[ 1919 ]<=1;
white[ 1920 ]<=1;
white[ 1921 ]<=1;
white[ 1922 ]<=1;
white[ 1923 ]<=1;
white[ 1924 ]<=1;
white[ 1925 ]<=1;
white[ 1926 ]<=1;
white[ 1927 ]<=1;
white[ 1928 ]<=1;
white[ 1929 ]<=1;
white[ 1930 ]<=1;
white[ 1931 ]<=1;
white[ 1932 ]<=1;
white[ 1933 ]<=1;
white[ 1934 ]<=1;
white[ 1935 ]<=1;
white[ 1936 ]<=1;
white[ 1937 ]<=1;
white[ 1938 ]<=1;
white[ 1939 ]<=1;
white[ 1940 ]<=1;
```

```
white[ 1941 ]<=1;
white[ 1942 ]<=1;
white[ 1943 ]<=1;
white[ 1944 ]<=1;
white[ 1945 ]<=1;
white[ 1946 ]<=1;
white[ 1947 ]<=1;
white[ 1948 ]<=1;
white[ 1949 ]<=1;
white[ 1950 ]<=1;
white[ 1951 ]<=1;
white[ 1952 ]<=1;
white[ 1953 ]<=1;
white[ 1954 ]<=1;
white[ 1955 ]<=1;
white[ 1956 ]<=1;
white[ 1957 ]<=1;
white[ 1958 ]<=1;
white[ 1959 ]<=1;
white[ 1960 ]<=1;
white[ 1961 ]<=1;
white[ 1962 ]<=1;
white[ 1963 ]<=1;
white[ 1964 ]<=1;
white[ 1965 ]<=1;
white[ 1966 ]<=1;
white[ 1967 ]<=1;
white[ 1968 ]<=1;
white[ 1969 ]<=1;
white[ 1970 ]<=1;
white[ 1971 ]<=1;
white[ 1972 ]<=1;
white[ 1973 ]<=1;
white[ 1974 ]<=1;
white[ 1975 ]<=1;
white[ 1976 ]<=1;
white[ 1977 ]<=1;
white[ 1978 ]<=1;
white[ 1979 ]<=1;
white[ 1980 ]<=1;
white[ 1981 ]<=1;
white[ 1982 ]<=1;
white[ 1983 ]<=1;
white[ 1984 ]<=1;
white[ 1985 ]<=1;
white[ 1986 ]<=1;
white[ 1987 ]<=1;
white[ 1988 ]<=1;
white[ 1989 ]<=1;
white[ 1990 ]<=1;
white[ 1991 ]<=1;
white[ 1992 ]<=1;
white[ 1993 ]<=1;
white[ 1994 ]<=1;
white[ 1995 ]<=1;
white[ 1996 ]<=1;
white[ 1997 ]<=1;
white[ 1998 ]<=1;
```

```
white[ 1999 ]<=1;
white[ 2000 ]<=1;
white[ 2001 ]<=1;
white[ 2002 ]<=1;
white[ 2003 ]<=1;
white[ 2004 ]<=1;
white[ 2005 ]<=1;
white[ 2006 ]<=1;
white[ 2007 ]<=1;
white[ 2008 ]<=1;
white[ 2009 ]<=1;
white[ 2010 ]<=1;
white[ 2011 ]<=1;
white[ 2012 ]<=1;
white[ 2013 ]<=1;
white[ 2014 ]<=1;
white[ 2015 ]<=1;
white[ 2016 ]<=1;
white[ 2017 ]<=1;
white[ 2018 ]<=1;
white[ 2019 ]<=1;
white[ 2020 ]<=1;
white[ 2021 ]<=1;
white[ 2022 ]<=1;
white[ 2023 ]<=1;
white[ 2024 ]<=1;
white[ 2025 ]<=1;
white[ 2026 ]<=1;
white[ 2027 ]<=1;
white[ 2028 ]<=1;
white[ 2029 ]<=1;
white[ 2030 ]<=1;
white[ 2031 ]<=1;
white[ 2032 ]<=1;
white[ 2033 ]<=1;
white[ 2034 ]<=1;
white[ 2035 ]<=1;
white[ 2036 ]<=1;
white[ 2037 ]<=1;
white[ 2038 ]<=1;
white[ 2039 ]<=1;
white[ 2040 ]<=1;
white[ 2041 ]<=1;
white[ 2042 ]<=1;
white[ 2043 ]<=1;
white[ 2044 ]<=1;
white[ 2045 ]<=1;
white[ 2046 ]<=1;
white[ 2047 ]<=1;
white[ 2048 ]<=1;
white[ 2049 ]<=1;
white[ 2050 ]<=1;
white[ 2051 ]<=1;
white[ 2052 ]<=1;
white[ 2053 ]<=1;
white[ 2054 ]<=1;
white[ 2055 ]<=1;
white[ 2056 ]<=1;
```

```
white[ 2057 ]<=1;
white[ 2058 ]<=1;
white[ 2059 ]<=1;
white[ 2060 ]<=1;
white[ 2061 ]<=1;
white[ 2062 ]<=1;
white[ 2063 ]<=1;
white[ 2064 ]<=1;
white[ 2065 ]<=1;
white[ 2066 ]<=1;
white[ 2067 ]<=1;
white[ 2068 ]<=1;
white[ 2069 ]<=1;
white[ 2070 ]<=1;
white[ 2071 ]<=1;
white[ 2072 ]<=1;
white[ 2073 ]<=1;
white[ 2074 ]<=1;
white[ 2075 ]<=1;
white[ 2076 ]<=1;
white[ 2077 ]<=1;
white[ 2078 ]<=1;
white[ 2079 ]<=1;
white[ 2080 ]<=1;
white[ 2081 ]<=1;
white[ 2082 ]<=1;
white[ 2083 ]<=1;
white[ 2084 ]<=1;
white[ 2085 ]<=1;
white[ 2086 ]<=1;
white[ 2087 ]<=1;
white[ 2088 ]<=1;
white[ 2089 ]<=1;
white[ 2090 ]<=1;
white[ 2091 ]<=1;
white[ 2092 ]<=1;
white[ 2093 ]<=1;
white[ 2094 ]<=1;
white[ 2095 ]<=1;
white[ 2096 ]<=1;
white[ 2097 ]<=1;
white[ 2098 ]<=1;
white[ 2099 ]<=1;
white[ 2100 ]<=1;
white[ 2101 ]<=1;
white[ 2102 ]<=1;
white[ 2103 ]<=1;
white[ 2104 ]<=1;
white[ 2105 ]<=1;
white[ 2106 ]<=1;
white[ 2107 ]<=1;
white[ 2108 ]<=1;
white[ 2109 ]<=1;
white[ 2110 ]<=1;
white[ 2111 ]<=1;
white[ 2112 ]<=1;
white[ 2113 ]<=1;
white[ 2114 ]<=1;
```

```
white[ 2115 ]<=1;
white[ 2116 ]<=1;
white[ 2117 ]<=1;
white[ 2118 ]<=1;
white[ 2119 ]<=1;
white[ 2120 ]<=1;
white[ 2121 ]<=1;
white[ 2122 ]<=1;
white[ 2123 ]<=1;
white[ 2124 ]<=1;
white[ 2125 ]<=1;
white[ 2126 ]<=1;
white[ 2127 ]<=1;
white[ 2128 ]<=1;
white[ 2129 ]<=1;
white[ 2130 ]<=1;
white[ 2131 ]<=1;
white[ 2132 ]<=1;
white[ 2133 ]<=1;
white[ 2134 ]<=1;
white[ 2135 ]<=1;
white[ 2136 ]<=1;
white[ 2137 ]<=1;
white[ 2138 ]<=1;
white[ 2139 ]<=1;
white[ 2140 ]<=1;
white[ 2141 ]<=1;
white[ 2142 ]<=1;
white[ 2143 ]<=1;
white[ 2144 ]<=1;
white[ 2145 ]<=1;
white[ 2146 ]<=1;
white[ 2147 ]<=1;
white[ 2148 ]<=1;
white[ 2149 ]<=1;
white[ 2150 ]<=1;
white[ 2151 ]<=1;
white[ 2152 ]<=1;
white[ 2153 ]<=1;
white[ 2154 ]<=1;
white[ 2155 ]<=1;
white[ 2156 ]<=1;
white[ 2157 ]<=1;
white[ 2158 ]<=1;
white[ 2159 ]<=1;
white[ 2160 ]<=0;
white[ 2161 ]<=1;
white[ 2162 ]<=1;
white[ 2163 ]<=1;
white[ 2164 ]<=1;
white[ 2165 ]<=1;
white[ 2166 ]<=1;
white[ 2167 ]<=1;
white[ 2168 ]<=1;
white[ 2169 ]<=1;
white[ 2170 ]<=1;
white[ 2171 ]<=1;
white[ 2172 ]<=1;
```

```
white[ 2173 ]<=1;
white[ 2174 ]<=1;
white[ 2175 ]<=1;
white[ 2176 ]<=1;
white[ 2177 ]<=1;
white[ 2178 ]<=1;
white[ 2179 ]<=1;
white[ 2180 ]<=1;
white[ 2181 ]<=1;
white[ 2182 ]<=1;
white[ 2183 ]<=1;
white[ 2184 ]<=1;
white[ 2185 ]<=1;
white[ 2186 ]<=1;
white[ 2187 ]<=1;
white[ 2188 ]<=1;
white[ 2189 ]<=1;
white[ 2190 ]<=1;
white[ 2191 ]<=1;
white[ 2192 ]<=1;
white[ 2193 ]<=1;
white[ 2194 ]<=1;
white[ 2195 ]<=1;
white[ 2196 ]<=1;
white[ 2197 ]<=1;
white[ 2198 ]<=1;
white[ 2199 ]<=1;
white[ 2200 ]<=1;
white[ 2201 ]<=1;
white[ 2202 ]<=1;
white[ 2203 ]<=1;
white[ 2204 ]<=1;
white[ 2205 ]<=1;
white[ 2206 ]<=1;
white[ 2207 ]<=1;
white[ 2208 ]<=1;
white[ 2209 ]<=1;
white[ 2210 ]<=1;
white[ 2211 ]<=1;
white[ 2212 ]<=1;
white[ 2213 ]<=1;
white[ 2214 ]<=1;
white[ 2215 ]<=1;
white[ 2216 ]<=1;
white[ 2217 ]<=1;
white[ 2218 ]<=1;
white[ 2219 ]<=1;
white[ 2220 ]<=0;
white[ 2221 ]<=1;
white[ 2222 ]<=1;
white[ 2223 ]<=1;
white[ 2224 ]<=1;
white[ 2225 ]<=1;
white[ 2226 ]<=1;
white[ 2227 ]<=1;
white[ 2228 ]<=1;
white[ 2229 ]<=1;
white[ 2230 ]<=1;
```

```
white[ 2231 ]<=1;
white[ 2232 ]<=1;
white[ 2233 ]<=1;
white[ 2234 ]<=1;
white[ 2235 ]<=1;
white[ 2236 ]<=1;
white[ 2237 ]<=1;
white[ 2238 ]<=1;
white[ 2239 ]<=1;
white[ 2240 ]<=1;
white[ 2241 ]<=1;
white[ 2242 ]<=1;
white[ 2243 ]<=1;
white[ 2244 ]<=1;
white[ 2245 ]<=1;
white[ 2246 ]<=1;
white[ 2247 ]<=1;
white[ 2248 ]<=1;
white[ 2249 ]<=1;
white[ 2250 ]<=1;
white[ 2251 ]<=1;
white[ 2252 ]<=1;
white[ 2253 ]<=1;
white[ 2254 ]<=1;
white[ 2255 ]<=1;
white[ 2256 ]<=1;
white[ 2257 ]<=1;
white[ 2258 ]<=1;
white[ 2259 ]<=1;
white[ 2260 ]<=1;
white[ 2261 ]<=1;
white[ 2262 ]<=1;
white[ 2263 ]<=1;
white[ 2264 ]<=1;
white[ 2265 ]<=1;
white[ 2266 ]<=1;
white[ 2267 ]<=1;
white[ 2268 ]<=1;
white[ 2269 ]<=1;
white[ 2270 ]<=1;
white[ 2271 ]<=1;
white[ 2272 ]<=1;
white[ 2273 ]<=1;
white[ 2274 ]<=1;
white[ 2275 ]<=1;
white[ 2276 ]<=1;
white[ 2277 ]<=1;
white[ 2278 ]<=1;
white[ 2279 ]<=1;
white[ 2280 ]<=0;
white[ 2281 ]<=1;
white[ 2282 ]<=1;
white[ 2283 ]<=1;
white[ 2284 ]<=1;
white[ 2285 ]<=1;
white[ 2286 ]<=1;
white[ 2287 ]<=1;
white[ 2288 ]<=1;
```

```
white[ 2289 ]<=1;
white[ 2290 ]<=1;
white[ 2291 ]<=1;
white[ 2292 ]<=1;
white[ 2293 ]<=1;
white[ 2294 ]<=1;
white[ 2295 ]<=1;
white[ 2296 ]<=1;
white[ 2297 ]<=1;
white[ 2298 ]<=1;
white[ 2299 ]<=1;
white[ 2300 ]<=1;
white[ 2301 ]<=1;
white[ 2302 ]<=1;
white[ 2303 ]<=1;
white[ 2304 ]<=1;
white[ 2305 ]<=1;
white[ 2306 ]<=1;
white[ 2307 ]<=1;
white[ 2308 ]<=1;
white[ 2309 ]<=1;
white[ 2310 ]<=1;
white[ 2311 ]<=1;
white[ 2312 ]<=1;
white[ 2313 ]<=1;
white[ 2314 ]<=1;
white[ 2315 ]<=1;
white[ 2316 ]<=1;
white[ 2317 ]<=1;
white[ 2318 ]<=1;
white[ 2319 ]<=1;
white[ 2320 ]<=1;
white[ 2321 ]<=1;
white[ 2322 ]<=1;
white[ 2323 ]<=1;
white[ 2324 ]<=1;
white[ 2325 ]<=1;
white[ 2326 ]<=1;
white[ 2327 ]<=1;
white[ 2328 ]<=1;
white[ 2329 ]<=1;
white[ 2330 ]<=1;
white[ 2331 ]<=1;
white[ 2332 ]<=1;
white[ 2333 ]<=1;
white[ 2334 ]<=1;
white[ 2335 ]<=1;
white[ 2336 ]<=1;
white[ 2337 ]<=1;
white[ 2338 ]<=1;
white[ 2339 ]<=0;
white[ 2340 ]<=0;
white[ 2341 ]<=1;
white[ 2342 ]<=1;
white[ 2343 ]<=1;
white[ 2344 ]<=1;
white[ 2345 ]<=1;
white[ 2346 ]<=1;
```

```
white[ 2347 ]<=1;
white[ 2348 ]<=1;
white[ 2349 ]<=1;
white[ 2350 ]<=1;
white[ 2351 ]<=1;
white[ 2352 ]<=1;
white[ 2353 ]<=1;
white[ 2354 ]<=1;
white[ 2355 ]<=1;
white[ 2356 ]<=1;
white[ 2357 ]<=1;
white[ 2358 ]<=1;
white[ 2359 ]<=1;
white[ 2360 ]<=1;
white[ 2361 ]<=1;
white[ 2362 ]<=1;
white[ 2363 ]<=1;
white[ 2364 ]<=1;
white[ 2365 ]<=1;
white[ 2366 ]<=1;
white[ 2367 ]<=1;
white[ 2368 ]<=1;
white[ 2369 ]<=1;
white[ 2370 ]<=1;
white[ 2371 ]<=1;
white[ 2372 ]<=1;
white[ 2373 ]<=1;
white[ 2374 ]<=1;
white[ 2375 ]<=1;
white[ 2376 ]<=1;
white[ 2377 ]<=1;
white[ 2378 ]<=1;
white[ 2379 ]<=1;
white[ 2380 ]<=1;
white[ 2381 ]<=1;
white[ 2382 ]<=1;
white[ 2383 ]<=1;
white[ 2384 ]<=1;
white[ 2385 ]<=1;
white[ 2386 ]<=1;
white[ 2387 ]<=1;
white[ 2388 ]<=1;
white[ 2389 ]<=1;
white[ 2390 ]<=1;
white[ 2391 ]<=1;
white[ 2392 ]<=1;
white[ 2393 ]<=1;
white[ 2394 ]<=1;
white[ 2395 ]<=1;
white[ 2396 ]<=1;
white[ 2397 ]<=1;
white[ 2398 ]<=1;
white[ 2399 ]<=0;
white[ 2400 ]<=0;
white[ 2401 ]<=0;
white[ 2402 ]<=1;
white[ 2403 ]<=1;
white[ 2404 ]<=1;
```

```
white[ 2405 ]<=1;
white[ 2406 ]<=1;
white[ 2407 ]<=1;
white[ 2408 ]<=1;
white[ 2409 ]<=1;
white[ 2410 ]<=1;
white[ 2411 ]<=1;
white[ 2412 ]<=1;
white[ 2413 ]<=1;
white[ 2414 ]<=1;
white[ 2415 ]<=1;
white[ 2416 ]<=1;
white[ 2417 ]<=1;
white[ 2418 ]<=1;
white[ 2419 ]<=1;
white[ 2420 ]<=1;
white[ 2421 ]<=1;
white[ 2422 ]<=1;
white[ 2423 ]<=1;
white[ 2424 ]<=1;
white[ 2425 ]<=1;
white[ 2426 ]<=1;
white[ 2427 ]<=1;
white[ 2428 ]<=1;
white[ 2429 ]<=1;
white[ 2430 ]<=1;
white[ 2431 ]<=1;
white[ 2432 ]<=1;
white[ 2433 ]<=1;
white[ 2434 ]<=1;
white[ 2435 ]<=1;
white[ 2436 ]<=1;
white[ 2437 ]<=1;
white[ 2438 ]<=1;
white[ 2439 ]<=1;
white[ 2440 ]<=1;
white[ 2441 ]<=1;
white[ 2442 ]<=1;
white[ 2443 ]<=1;
white[ 2444 ]<=1;
white[ 2445 ]<=1;
white[ 2446 ]<=1;
white[ 2447 ]<=1;
white[ 2448 ]<=1;
white[ 2449 ]<=1;
white[ 2450 ]<=1;
white[ 2451 ]<=1;
white[ 2452 ]<=1;
white[ 2453 ]<=1;
white[ 2454 ]<=1;
white[ 2455 ]<=1;
white[ 2456 ]<=1;
white[ 2457 ]<=1;
white[ 2458 ]<=1;
white[ 2459 ]<=0;
white[ 2460 ]<=0;
white[ 2461 ]<=0;
white[ 2462 ]<=1;
```

```
white[ 2463 ]<=1;
white[ 2464 ]<=1;
white[ 2465 ]<=1;
white[ 2466 ]<=1;
white[ 2467 ]<=1;
white[ 2468 ]<=1;
white[ 2469 ]<=1;
white[ 2470 ]<=1;
white[ 2471 ]<=1;
white[ 2472 ]<=1;
white[ 2473 ]<=1;
white[ 2474 ]<=1;
white[ 2475 ]<=1;
white[ 2476 ]<=1;
white[ 2477 ]<=1;
white[ 2478 ]<=1;
white[ 2479 ]<=1;
white[ 2480 ]<=1;
white[ 2481 ]<=1;
white[ 2482 ]<=1;
white[ 2483 ]<=1;
white[ 2484 ]<=1;
white[ 2485 ]<=1;
white[ 2486 ]<=1;
white[ 2487 ]<=1;
white[ 2488 ]<=1;
white[ 2489 ]<=1;
white[ 2490 ]<=1;
white[ 2491 ]<=1;
white[ 2492 ]<=1;
white[ 2493 ]<=1;
white[ 2494 ]<=1;
white[ 2495 ]<=1;
white[ 2496 ]<=1;
white[ 2497 ]<=1;
white[ 2498 ]<=1;
white[ 2499 ]<=1;
white[ 2500 ]<=1;
white[ 2501 ]<=1;
white[ 2502 ]<=1;
white[ 2503 ]<=1;
white[ 2504 ]<=1;
white[ 2505 ]<=1;
white[ 2506 ]<=1;
white[ 2507 ]<=1;
white[ 2508 ]<=1;
white[ 2509 ]<=1;
white[ 2510 ]<=1;
white[ 2511 ]<=1;
white[ 2512 ]<=1;
white[ 2513 ]<=1;
white[ 2514 ]<=1;
white[ 2515 ]<=1;
white[ 2516 ]<=1;
white[ 2517 ]<=1;
white[ 2518 ]<=0;
white[ 2519 ]<=0;
white[ 2520 ]<=0;
```

```
white[ 2521 ]<=0;
white[ 2522 ]<=0;
white[ 2523 ]<=1;
white[ 2524 ]<=1;
white[ 2525 ]<=1;
white[ 2526 ]<=1;
white[ 2527 ]<=1;
white[ 2528 ]<=1;
white[ 2529 ]<=1;
white[ 2530 ]<=1;
white[ 2531 ]<=1;
white[ 2532 ]<=1;
white[ 2533 ]<=1;
white[ 2534 ]<=1;
white[ 2535 ]<=1;
white[ 2536 ]<=1;
white[ 2537 ]<=1;
white[ 2538 ]<=1;
white[ 2539 ]<=1;
white[ 2540 ]<=1;
white[ 2541 ]<=1;
white[ 2542 ]<=1;
white[ 2543 ]<=1;
white[ 2544 ]<=1;
white[ 2545 ]<=1;
white[ 2546 ]<=1;
white[ 2547 ]<=1;
white[ 2548 ]<=1;
white[ 2549 ]<=1;
white[ 2550 ]<=1;
white[ 2551 ]<=1;
white[ 2552 ]<=1;
white[ 2553 ]<=1;
white[ 2554 ]<=1;
white[ 2555 ]<=1;
white[ 2556 ]<=1;
white[ 2557 ]<=1;
white[ 2558 ]<=1;
white[ 2559 ]<=1;
white[ 2560 ]<=1;
white[ 2561 ]<=1;
white[ 2562 ]<=1;
white[ 2563 ]<=1;
white[ 2564 ]<=1;
white[ 2565 ]<=1;
white[ 2566 ]<=1;
white[ 2567 ]<=1;
white[ 2568 ]<=1;
white[ 2569 ]<=1;
white[ 2570 ]<=1;
white[ 2571 ]<=1;
white[ 2572 ]<=1;
white[ 2573 ]<=1;
white[ 2574 ]<=1;
white[ 2575 ]<=1;
white[ 2576 ]<=1;
white[ 2577 ]<=1;
white[ 2578 ]<=0;
```

```
white[ 2579 ]<=0;
white[ 2580 ]<=0;
white[ 2581 ]<=0;
white[ 2582 ]<=0;
white[ 2583 ]<=1;
white[ 2584 ]<=1;
white[ 2585 ]<=1;
white[ 2586 ]<=1;
white[ 2587 ]<=1;
white[ 2588 ]<=1;
white[ 2589 ]<=1;
white[ 2590 ]<=1;
white[ 2591 ]<=1;
white[ 2592 ]<=1;
white[ 2593 ]<=1;
white[ 2594 ]<=1;
white[ 2595 ]<=1;
white[ 2596 ]<=1;
white[ 2597 ]<=1;
white[ 2598 ]<=1;
white[ 2599 ]<=1;
white[ 2600 ]<=1;
white[ 2601 ]<=1;
white[ 2602 ]<=1;
white[ 2603 ]<=1;
white[ 2604 ]<=1;
white[ 2605 ]<=1;
white[ 2606 ]<=1;
white[ 2607 ]<=1;
white[ 2608 ]<=1;
white[ 2609 ]<=1;
white[ 2610 ]<=1;
white[ 2611 ]<=1;
white[ 2612 ]<=1;
white[ 2613 ]<=1;
white[ 2614 ]<=1;
white[ 2615 ]<=1;
white[ 2616 ]<=1;
white[ 2617 ]<=1;
white[ 2618 ]<=1;
white[ 2619 ]<=1;
white[ 2620 ]<=1;
white[ 2621 ]<=1;
white[ 2622 ]<=1;
white[ 2623 ]<=1;
white[ 2624 ]<=1;
white[ 2625 ]<=1;
white[ 2626 ]<=1;
white[ 2627 ]<=1;
white[ 2628 ]<=1;
white[ 2629 ]<=1;
white[ 2630 ]<=1;
white[ 2631 ]<=1;
white[ 2632 ]<=1;
white[ 2633 ]<=1;
white[ 2634 ]<=1;
white[ 2635 ]<=1;
white[ 2636 ]<=1;
```

```
white[ 2637 ]<=0;
white[ 2638 ]<=0;
white[ 2639 ]<=0;
white[ 2640 ]<=0;
white[ 2641 ]<=0;
white[ 2642 ]<=0;
white[ 2643 ]<=0;
white[ 2644 ]<=1;
white[ 2645 ]<=1;
white[ 2646 ]<=1;
white[ 2647 ]<=1;
white[ 2648 ]<=1;
white[ 2649 ]<=1;
white[ 2650 ]<=1;
white[ 2651 ]<=1;
white[ 2652 ]<=1;
white[ 2653 ]<=1;
white[ 2654 ]<=1;
white[ 2655 ]<=1;
white[ 2656 ]<=1;
white[ 2657 ]<=1;
white[ 2658 ]<=1;
white[ 2659 ]<=1;
white[ 2660 ]<=1;
white[ 2661 ]<=1;
white[ 2662 ]<=1;
white[ 2663 ]<=1;
white[ 2664 ]<=1;
white[ 2665 ]<=1;
white[ 2666 ]<=1;
white[ 2667 ]<=1;
white[ 2668 ]<=1;
white[ 2669 ]<=1;
white[ 2670 ]<=1;
white[ 2671 ]<=1;
white[ 2672 ]<=1;
white[ 2673 ]<=1;
white[ 2674 ]<=1;
white[ 2675 ]<=1;
white[ 2676 ]<=1;
white[ 2677 ]<=1;
white[ 2678 ]<=1;
white[ 2679 ]<=1;
white[ 2680 ]<=1;
white[ 2681 ]<=1;
white[ 2682 ]<=1;
white[ 2683 ]<=1;
white[ 2684 ]<=1;
white[ 2685 ]<=1;
white[ 2686 ]<=1;
white[ 2687 ]<=1;
white[ 2688 ]<=1;
white[ 2689 ]<=1;
white[ 2690 ]<=1;
white[ 2691 ]<=1;
white[ 2692 ]<=1;
white[ 2693 ]<=1;
white[ 2694 ]<=1;
```

```
white[ 2695 ]<=1;
white[ 2696 ]<=1;
white[ 2697 ]<=0;
white[ 2698 ]<=0;
white[ 2699 ]<=0;
white[ 2700 ]<=0;
white[ 2701 ]<=0;
white[ 2702 ]<=0;
white[ 2703 ]<=0;
white[ 2704 ]<=1;
white[ 2705 ]<=1;
white[ 2706 ]<=1;
white[ 2707 ]<=1;
white[ 2708 ]<=1;
white[ 2709 ]<=1;
white[ 2710 ]<=1;
white[ 2711 ]<=1;
white[ 2712 ]<=1;
white[ 2713 ]<=1;
white[ 2714 ]<=1;
white[ 2715 ]<=1;
white[ 2716 ]<=1;
white[ 2717 ]<=1;
white[ 2718 ]<=1;
white[ 2719 ]<=1;
white[ 2720 ]<=1;
white[ 2721 ]<=1;
white[ 2722 ]<=1;
white[ 2723 ]<=1;
white[ 2724 ]<=1;
white[ 2725 ]<=1;
white[ 2726 ]<=1;
white[ 2727 ]<=1;
white[ 2728 ]<=1;
white[ 2729 ]<=1;
white[ 2730 ]<=1;
white[ 2731 ]<=1;
white[ 2732 ]<=1;
white[ 2733 ]<=1;
white[ 2734 ]<=1;
white[ 2735 ]<=1;
white[ 2736 ]<=1;
white[ 2737 ]<=1;
white[ 2738 ]<=1;
white[ 2739 ]<=1;
white[ 2740 ]<=1;
white[ 2741 ]<=1;
white[ 2742 ]<=1;
white[ 2743 ]<=1;
white[ 2744 ]<=1;
white[ 2745 ]<=1;
white[ 2746 ]<=1;
white[ 2747 ]<=1;
white[ 2748 ]<=1;
white[ 2749 ]<=1;
white[ 2750 ]<=1;
white[ 2751 ]<=1;
white[ 2752 ]<=1;
```

```
white[ 2753 ]<=1;
white[ 2754 ]<=1;
white[ 2755 ]<=1;
white[ 2756 ]<=0;
white[ 2757 ]<=0;
white[ 2758 ]<=0;
white[ 2759 ]<=0;
white[ 2760 ]<=0;
white[ 2761 ]<=0;
white[ 2762 ]<=0;
white[ 2763 ]<=0;
white[ 2764 ]<=0;
white[ 2765 ]<=1;
white[ 2766 ]<=1;
white[ 2767 ]<=1;
white[ 2768 ]<=1;
white[ 2769 ]<=1;
white[ 2770 ]<=1;
white[ 2771 ]<=1;
white[ 2772 ]<=1;
white[ 2773 ]<=1;
white[ 2774 ]<=1;
white[ 2775 ]<=1;
white[ 2776 ]<=1;
white[ 2777 ]<=1;
white[ 2778 ]<=1;
white[ 2779 ]<=1;
white[ 2780 ]<=1;
white[ 2781 ]<=1;
white[ 2782 ]<=1;
white[ 2783 ]<=1;
white[ 2784 ]<=1;
white[ 2785 ]<=1;
white[ 2786 ]<=1;
white[ 2787 ]<=1;
white[ 2788 ]<=1;
white[ 2789 ]<=1;
white[ 2790 ]<=1;
white[ 2791 ]<=1;
white[ 2792 ]<=1;
white[ 2793 ]<=1;
white[ 2794 ]<=1;
white[ 2795 ]<=1;
white[ 2796 ]<=1;
white[ 2797 ]<=1;
white[ 2798 ]<=1;
white[ 2799 ]<=1;
white[ 2800 ]<=1;
white[ 2801 ]<=1;
white[ 2802 ]<=1;
white[ 2803 ]<=1;
white[ 2804 ]<=1;
white[ 2805 ]<=1;
white[ 2806 ]<=1;
white[ 2807 ]<=1;
white[ 2808 ]<=1;
white[ 2809 ]<=1;
white[ 2810 ]<=1;
```

```
white[ 2811 ]<=1;
white[ 2812 ]<=1;
white[ 2813 ]<=1;
white[ 2814 ]<=1;
white[ 2815 ]<=1;
white[ 2816 ]<=0;
white[ 2817 ]<=0;
white[ 2818 ]<=0;
white[ 2819 ]<=0;
white[ 2820 ]<=0;
white[ 2821 ]<=0;
white[ 2822 ]<=0;
white[ 2823 ]<=0;
white[ 2824 ]<=0;
white[ 2825 ]<=0;
white[ 2826 ]<=1;
white[ 2827 ]<=1;
white[ 2828 ]<=1;
white[ 2829 ]<=1;
white[ 2830 ]<=1;
white[ 2831 ]<=1;
white[ 2832 ]<=1;
white[ 2833 ]<=1;
white[ 2834 ]<=1;
white[ 2835 ]<=1;
white[ 2836 ]<=1;
white[ 2837 ]<=1;
white[ 2838 ]<=1;
white[ 2839 ]<=1;
white[ 2840 ]<=1;
white[ 2841 ]<=1;
white[ 2842 ]<=1;
white[ 2843 ]<=1;
white[ 2844 ]<=1;
white[ 2845 ]<=1;
white[ 2846 ]<=1;
white[ 2847 ]<=1;
white[ 2848 ]<=1;
white[ 2849 ]<=1;
white[ 2850 ]<=1;
white[ 2851 ]<=1;
white[ 2852 ]<=1;
white[ 2853 ]<=1;
white[ 2854 ]<=1;
white[ 2855 ]<=1;
white[ 2856 ]<=1;
white[ 2857 ]<=1;
white[ 2858 ]<=1;
white[ 2859 ]<=1;
white[ 2860 ]<=1;
white[ 2861 ]<=1;
white[ 2862 ]<=1;
white[ 2863 ]<=1;
white[ 2864 ]<=1;
white[ 2865 ]<=1;
white[ 2866 ]<=1;
white[ 2867 ]<=1;
white[ 2868 ]<=1;
```

```
white[ 2869 ]<=1;
white[ 2870 ]<=1;
white[ 2871 ]<=1;
white[ 2872 ]<=1;
white[ 2873 ]<=1;
white[ 2874 ]<=1;
white[ 2875 ]<=0;
white[ 2876 ]<=0;
white[ 2877 ]<=0;
white[ 2878 ]<=0;
white[ 2879 ]<=0;
white[ 2880 ]<=0;
white[ 2881 ]<=0;
white[ 2882 ]<=0;
white[ 2883 ]<=0;
white[ 2884 ]<=0;
white[ 2885 ]<=0;
white[ 2886 ]<=0;
white[ 2887 ]<=1;
white[ 2888 ]<=1;
white[ 2889 ]<=1;
white[ 2890 ]<=1;
white[ 2891 ]<=1;
white[ 2892 ]<=1;
white[ 2893 ]<=1;
white[ 2894 ]<=1;
white[ 2895 ]<=1;
white[ 2896 ]<=1;
white[ 2897 ]<=1;
white[ 2898 ]<=1;
white[ 2899 ]<=1;
white[ 2900 ]<=1;
white[ 2901 ]<=1;
white[ 2902 ]<=1;
white[ 2903 ]<=1;
white[ 2904 ]<=1;
white[ 2905 ]<=1;
white[ 2906 ]<=1;
white[ 2907 ]<=1;
white[ 2908 ]<=1;
white[ 2909 ]<=1;
white[ 2910 ]<=1;
white[ 2911 ]<=1;
white[ 2912 ]<=1;
white[ 2913 ]<=1;
white[ 2914 ]<=1;
white[ 2915 ]<=1;
white[ 2916 ]<=1;
white[ 2917 ]<=1;
white[ 2918 ]<=1;
white[ 2919 ]<=1;
white[ 2920 ]<=1;
white[ 2921 ]<=1;
white[ 2922 ]<=1;
white[ 2923 ]<=1;
white[ 2924 ]<=1;
white[ 2925 ]<=1;
white[ 2926 ]<=1;
```

```
white[ 2927 ]<=1;
white[ 2928 ]<=1;
white[ 2929 ]<=1;
white[ 2930 ]<=1;
white[ 2931 ]<=1;
white[ 2932 ]<=1;
white[ 2933 ]<=1;
white[ 2934 ]<=0;
white[ 2935 ]<=0;
white[ 2936 ]<=0;
white[ 2937 ]<=0;
white[ 2938 ]<=0;
white[ 2939 ]<=0;
white[ 2940 ]<=0;
white[ 2941 ]<=0;
white[ 2942 ]<=0;
white[ 2943 ]<=0;
white[ 2944 ]<=0;
white[ 2945 ]<=0;
white[ 2946 ]<=0;
white[ 2947 ]<=0;
white[ 2948 ]<=1;
white[ 2949 ]<=1;
white[ 2950 ]<=1;
white[ 2951 ]<=1;
white[ 2952 ]<=1;
white[ 2953 ]<=1;
white[ 2954 ]<=1;
white[ 2955 ]<=1;
white[ 2956 ]<=1;
white[ 2957 ]<=1;
white[ 2958 ]<=1;
white[ 2959 ]<=1;
white[ 2960 ]<=1;
white[ 2961 ]<=1;
white[ 2962 ]<=1;
white[ 2963 ]<=1;
white[ 2964 ]<=1;
white[ 2965 ]<=1;
white[ 2966 ]<=1;
white[ 2967 ]<=1;
white[ 2968 ]<=1;
white[ 2969 ]<=1;
white[ 2970 ]<=1;
white[ 2971 ]<=1;
white[ 2972 ]<=1;
white[ 2973 ]<=1;
white[ 2974 ]<=1;
white[ 2975 ]<=1;
white[ 2976 ]<=1;
white[ 2977 ]<=1;
white[ 2978 ]<=1;
white[ 2979 ]<=1;
white[ 2980 ]<=1;
white[ 2981 ]<=1;
white[ 2982 ]<=1;
white[ 2983 ]<=1;
white[ 2984 ]<=1;
```

```
white[ 2985 ]<=1;
white[ 2986 ]<=1;
white[ 2987 ]<=1;
white[ 2988 ]<=1;
white[ 2989 ]<=1;
white[ 2990 ]<=1;
white[ 2991 ]<=1;
white[ 2992 ]<=1;
white[ 2993 ]<=1;
white[ 2994 ]<=0;
white[ 2995 ]<=0;
white[ 2996 ]<=0;
white[ 2997 ]<=0;
white[ 2998 ]<=0;
white[ 2999 ]<=0;
white[ 3000 ]<=0;
white[ 3001 ]<=0;
white[ 3002 ]<=0;
white[ 3003 ]<=0;
white[ 3004 ]<=0;
white[ 3005 ]<=0;
white[ 3006 ]<=0;
white[ 3007 ]<=0;
white[ 3008 ]<=0;
white[ 3009 ]<=1;
white[ 3010 ]<=1;
white[ 3011 ]<=1;
white[ 3012 ]<=1;
white[ 3013 ]<=1;
white[ 3014 ]<=1;
white[ 3015 ]<=1;
white[ 3016 ]<=1;
white[ 3017 ]<=1;
white[ 3018 ]<=1;
white[ 3019 ]<=1;
white[ 3020 ]<=1;
white[ 3021 ]<=1;
white[ 3022 ]<=1;
white[ 3023 ]<=1;
white[ 3024 ]<=1;
white[ 3025 ]<=1;
white[ 3026 ]<=1;
white[ 3027 ]<=1;
white[ 3028 ]<=1;
white[ 3029 ]<=1;
white[ 3030 ]<=1;
white[ 3031 ]<=1;
white[ 3032 ]<=1;
white[ 3033 ]<=1;
white[ 3034 ]<=1;
white[ 3035 ]<=1;
white[ 3036 ]<=1;
white[ 3037 ]<=1;
white[ 3038 ]<=1;
white[ 3039 ]<=1;
white[ 3040 ]<=1;
white[ 3041 ]<=1;
white[ 3042 ]<=1;
```

```
white[ 3043 ]<=1;
white[ 3044 ]<=1;
white[ 3045 ]<=1;
white[ 3046 ]<=1;
white[ 3047 ]<=1;
white[ 3048 ]<=1;
white[ 3049 ]<=1;
white[ 3050 ]<=1;
white[ 3051 ]<=1;
white[ 3052 ]<=1;
white[ 3053 ]<=0;
white[ 3054 ]<=0;
white[ 3055 ]<=0;
white[ 3056 ]<=0;
white[ 3057 ]<=0;
white[ 3058 ]<=0;
white[ 3059 ]<=0;
white[ 3060 ]<=0;
white[ 3061 ]<=0;
white[ 3062 ]<=0;
white[ 3063 ]<=0;
white[ 3064 ]<=0;
white[ 3065 ]<=0;
white[ 3066 ]<=0;
white[ 3067 ]<=0;
white[ 3068 ]<=0;
white[ 3069 ]<=0;
white[ 3070 ]<=1;
white[ 3071 ]<=1;
white[ 3072 ]<=1;
white[ 3073 ]<=1;
white[ 3074 ]<=1;
white[ 3075 ]<=1;
white[ 3076 ]<=1;
white[ 3077 ]<=1;
white[ 3078 ]<=1;
white[ 3079 ]<=1;
white[ 3080 ]<=1;
white[ 3081 ]<=1;
white[ 3082 ]<=1;
white[ 3083 ]<=1;
white[ 3084 ]<=1;
white[ 3085 ]<=1;
white[ 3086 ]<=1;
white[ 3087 ]<=1;
white[ 3088 ]<=1;
white[ 3089 ]<=1;
white[ 3090 ]<=1;
white[ 3091 ]<=1;
white[ 3092 ]<=1;
white[ 3093 ]<=1;
white[ 3094 ]<=1;
white[ 3095 ]<=1;
white[ 3096 ]<=1;
white[ 3097 ]<=1;
white[ 3098 ]<=1;
white[ 3099 ]<=1;
white[ 3100 ]<=1;
```

```
white[ 3101 ]<=1;
white[ 3102 ]<=1;
white[ 3103 ]<=1;
white[ 3104 ]<=1;
white[ 3105 ]<=1;
white[ 3106 ]<=1;
white[ 3107 ]<=1;
white[ 3108 ]<=1;
white[ 3109 ]<=1;
white[ 3110 ]<=1;
white[ 3111 ]<=1;
white[ 3112 ]<=0;
white[ 3113 ]<=0;
white[ 3114 ]<=0;
white[ 3115 ]<=0;
white[ 3116 ]<=0;
white[ 3117 ]<=0;
white[ 3118 ]<=0;
white[ 3119 ]<=0;
white[ 3120 ]<=0;
white[ 3121 ]<=0;
white[ 3122 ]<=0;
white[ 3123 ]<=0;
white[ 3124 ]<=0;
white[ 3125 ]<=0;
white[ 3126 ]<=0;
white[ 3127 ]<=0;
white[ 3128 ]<=0;
white[ 3129 ]<=0;
white[ 3130 ]<=1;
white[ 3131 ]<=1;
white[ 3132 ]<=1;
white[ 3133 ]<=1;
white[ 3134 ]<=1;
white[ 3135 ]<=1;
white[ 3136 ]<=1;
white[ 3137 ]<=1;
white[ 3138 ]<=1;
white[ 3139 ]<=1;
white[ 3140 ]<=1;
white[ 3141 ]<=1;
white[ 3142 ]<=1;
white[ 3143 ]<=1;
white[ 3144 ]<=1;
white[ 3145 ]<=1;
white[ 3146 ]<=1;
white[ 3147 ]<=1;
white[ 3148 ]<=1;
white[ 3149 ]<=1;
white[ 3150 ]<=1;
white[ 3151 ]<=1;
white[ 3152 ]<=1;
white[ 3153 ]<=1;
white[ 3154 ]<=1;
white[ 3155 ]<=1;
white[ 3156 ]<=1;
white[ 3157 ]<=1;
white[ 3158 ]<=1;
```

```
white[ 3159 ]<=1;
white[ 3160 ]<=1;
white[ 3161 ]<=1;
white[ 3162 ]<=1;
white[ 3163 ]<=1;
white[ 3164 ]<=1;
white[ 3165 ]<=1;
white[ 3166 ]<=1;
white[ 3167 ]<=1;
white[ 3168 ]<=1;
white[ 3169 ]<=1;
white[ 3170 ]<=1;
white[ 3171 ]<=0;
white[ 3172 ]<=0;
white[ 3173 ]<=0;
white[ 3174 ]<=0;
white[ 3175 ]<=0;
white[ 3176 ]<=0;
white[ 3177 ]<=0;
white[ 3178 ]<=0;
white[ 3179 ]<=0;
white[ 3180 ]<=0;
white[ 3181 ]<=0;
white[ 3182 ]<=0;
white[ 3183 ]<=0;
white[ 3184 ]<=0;
white[ 3185 ]<=0;
white[ 3186 ]<=0;
white[ 3187 ]<=0;
white[ 3188 ]<=0;
white[ 3189 ]<=0;
white[ 3190 ]<=0;
white[ 3191 ]<=0;
white[ 3192 ]<=1;
white[ 3193 ]<=1;
white[ 3194 ]<=1;
white[ 3195 ]<=1;
white[ 3196 ]<=1;
white[ 3197 ]<=1;
white[ 3198 ]<=1;
white[ 3199 ]<=1;
white[ 3200 ]<=1;
white[ 3201 ]<=1;
white[ 3202 ]<=1;
white[ 3203 ]<=1;
white[ 3204 ]<=1;
white[ 3205 ]<=1;
white[ 3206 ]<=1;
white[ 3207 ]<=1;
white[ 3208 ]<=1;
white[ 3209 ]<=1;
white[ 3210 ]<=1;
white[ 3211 ]<=1;
white[ 3212 ]<=1;
white[ 3213 ]<=1;
white[ 3214 ]<=1;
white[ 3215 ]<=1;
white[ 3216 ]<=1;
```

```
white[ 3217 ]<=1;
white[ 3218 ]<=1;
white[ 3219 ]<=1;
white[ 3220 ]<=1;
white[ 3221 ]<=1;
white[ 3222 ]<=1;
white[ 3223 ]<=1;
white[ 3224 ]<=1;
white[ 3225 ]<=1;
white[ 3226 ]<=1;
white[ 3227 ]<=1;
white[ 3228 ]<=1;
white[ 3229 ]<=1;
white[ 3230 ]<=0;
white[ 3231 ]<=0;
white[ 3232 ]<=0;
white[ 3233 ]<=0;
white[ 3234 ]<=0;
white[ 3235 ]<=0;
white[ 3236 ]<=0;
white[ 3237 ]<=0;
white[ 3238 ]<=0;
white[ 3239 ]<=0;
white[ 3240 ]<=0;
white[ 3241 ]<=0;
white[ 3242 ]<=0;
white[ 3243 ]<=0;
white[ 3244 ]<=0;
white[ 3245 ]<=0;
white[ 3246 ]<=0;
white[ 3247 ]<=0;
white[ 3248 ]<=0;
white[ 3249 ]<=0;
white[ 3250 ]<=0;
white[ 3251 ]<=0;
white[ 3252 ]<=0;
white[ 3253 ]<=1;
white[ 3254 ]<=1;
white[ 3255 ]<=1;
white[ 3256 ]<=1;
white[ 3257 ]<=1;
white[ 3258 ]<=1;
white[ 3259 ]<=1;
white[ 3260 ]<=1;
white[ 3261 ]<=1;
white[ 3262 ]<=1;
white[ 3263 ]<=1;
white[ 3264 ]<=1;
white[ 3265 ]<=1;
white[ 3266 ]<=1;
white[ 3267 ]<=1;
white[ 3268 ]<=1;
white[ 3269 ]<=1;
white[ 3270 ]<=1;
white[ 3271 ]<=1;
white[ 3272 ]<=1;
white[ 3273 ]<=1;
white[ 3274 ]<=1;
```

```
white[ 3275 ]<=1;
white[ 3276 ]<=1;
white[ 3277 ]<=1;
white[ 3278 ]<=1;
white[ 3279 ]<=1;
white[ 3280 ]<=1;
white[ 3281 ]<=1;
white[ 3282 ]<=1;
white[ 3283 ]<=1;
white[ 3284 ]<=1;
white[ 3285 ]<=1;
white[ 3286 ]<=1;
white[ 3287 ]<=1;
white[ 3288 ]<=0;
white[ 3289 ]<=0;
white[ 3290 ]<=0;
white[ 3291 ]<=0;
white[ 3292 ]<=0;
white[ 3293 ]<=0;
white[ 3294 ]<=0;
white[ 3295 ]<=0;
white[ 3296 ]<=0;
white[ 3297 ]<=0;
white[ 3298 ]<=0;
white[ 3299 ]<=0;
white[ 3300 ]<=0;
white[ 3301 ]<=0;
white[ 3302 ]<=0;
white[ 3303 ]<=0;
white[ 3304 ]<=0;
white[ 3305 ]<=0;
white[ 3306 ]<=0;
white[ 3307 ]<=0;
white[ 3308 ]<=0;
white[ 3309 ]<=0;
white[ 3310 ]<=0;
white[ 3311 ]<=0;
white[ 3312 ]<=0;
white[ 3313 ]<=0;
white[ 3314 ]<=1;
white[ 3315 ]<=1;
white[ 3316 ]<=1;
white[ 3317 ]<=1;
white[ 3318 ]<=1;
white[ 3319 ]<=1;
white[ 3320 ]<=1;
white[ 3321 ]<=1;
white[ 3322 ]<=1;
white[ 3323 ]<=1;
white[ 3324 ]<=1;
white[ 3325 ]<=1;
white[ 3326 ]<=1;
white[ 3327 ]<=1;
white[ 3328 ]<=1;
white[ 3329 ]<=1;
white[ 3330 ]<=1;
white[ 3331 ]<=1;
white[ 3332 ]<=1;
```

```
white[ 3333 ]<=1;
white[ 3334 ]<=1;
white[ 3335 ]<=1;
white[ 3336 ]<=1;
white[ 3337 ]<=1;
white[ 3338 ]<=1;
white[ 3339 ]<=1;
white[ 3340 ]<=1;
white[ 3341 ]<=1;
white[ 3342 ]<=1;
white[ 3343 ]<=1;
white[ 3344 ]<=1;
white[ 3345 ]<=1;
white[ 3346 ]<=1;
white[ 3347 ]<=0;
white[ 3348 ]<=0;
white[ 3349 ]<=0;
white[ 3350 ]<=0;
white[ 3351 ]<=0;
white[ 3352 ]<=0;
white[ 3353 ]<=0;
white[ 3354 ]<=0;
white[ 3355 ]<=0;
white[ 3356 ]<=0;
white[ 3357 ]<=0;
white[ 3358 ]<=0;
white[ 3359 ]<=0;
white[ 3360 ]<=0;
white[ 3361 ]<=0;
white[ 3362 ]<=0;
white[ 3363 ]<=0;
white[ 3364 ]<=0;
white[ 3365 ]<=0;
white[ 3366 ]<=0;
white[ 3367 ]<=0;
white[ 3368 ]<=0;
white[ 3369 ]<=0;
white[ 3370 ]<=0;
white[ 3371 ]<=0;
white[ 3372 ]<=0;
white[ 3373 ]<=0;
white[ 3374 ]<=0;
white[ 3375 ]<=0;
white[ 3376 ]<=1;
white[ 3377 ]<=1;
white[ 3378 ]<=1;
white[ 3379 ]<=1;
white[ 3380 ]<=1;
white[ 3381 ]<=1;
white[ 3382 ]<=1;
white[ 3383 ]<=1;
white[ 3384 ]<=1;
white[ 3385 ]<=1;
white[ 3386 ]<=1;
white[ 3387 ]<=1;
white[ 3388 ]<=1;
white[ 3389 ]<=1;
white[ 3390 ]<=1;
```

```
white[ 3391 ]<=1;
white[ 3392 ]<=1;
white[ 3393 ]<=1;
white[ 3394 ]<=1;
white[ 3395 ]<=1;
white[ 3396 ]<=1;
white[ 3397 ]<=1;
white[ 3398 ]<=1;
white[ 3399 ]<=1;
white[ 3400 ]<=1;
white[ 3401 ]<=1;
white[ 3402 ]<=1;
white[ 3403 ]<=1;
white[ 3404 ]<=1;
white[ 3405 ]<=0;
white[ 3406 ]<=0;
white[ 3407 ]<=0;
white[ 3408 ]<=0;
white[ 3409 ]<=0;
white[ 3410 ]<=0;
white[ 3411 ]<=0;
white[ 3412 ]<=0;
white[ 3413 ]<=0;
white[ 3414 ]<=0;
white[ 3415 ]<=0;
white[ 3416 ]<=0;
white[ 3417 ]<=0;
white[ 3418 ]<=0;
white[ 3419 ]<=0;
white[ 3420 ]<=0;
white[ 3421 ]<=0;
white[ 3422 ]<=0;
white[ 3423 ]<=0;
white[ 3424 ]<=0;
white[ 3425 ]<=0;
white[ 3426 ]<=0;
white[ 3427 ]<=0;
white[ 3428 ]<=0;
white[ 3429 ]<=0;
white[ 3430 ]<=0;
white[ 3431 ]<=0;
white[ 3432 ]<=0;
white[ 3433 ]<=0;
white[ 3434 ]<=0;
white[ 3435 ]<=0;
white[ 3436 ]<=0;
white[ 3437 ]<=0;
white[ 3438 ]<=1;
white[ 3439 ]<=1;
white[ 3440 ]<=1;
white[ 3441 ]<=1;
white[ 3442 ]<=1;
white[ 3443 ]<=1;
white[ 3444 ]<=1;
white[ 3445 ]<=1;
white[ 3446 ]<=1;
white[ 3447 ]<=1;
white[ 3448 ]<=1;
```

```
white[ 3449 ]<=1;
white[ 3450 ]<=1;
white[ 3451 ]<=1;
white[ 3452 ]<=1;
white[ 3453 ]<=1;
white[ 3454 ]<=1;
white[ 3455 ]<=1;
white[ 3456 ]<=1;
white[ 3457 ]<=1;
white[ 3458 ]<=1;
white[ 3459 ]<=1;
white[ 3460 ]<=1;
white[ 3461 ]<=1;
white[ 3462 ]<=1;
white[ 3463 ]<=0;
white[ 3464 ]<=0;
white[ 3465 ]<=0;
white[ 3466 ]<=0;
white[ 3467 ]<=0;
white[ 3468 ]<=0;
white[ 3469 ]<=0;
white[ 3470 ]<=0;
white[ 3471 ]<=0;
white[ 3472 ]<=0;
white[ 3473 ]<=0;
white[ 3474 ]<=0;
white[ 3475 ]<=0;
white[ 3476 ]<=0;
white[ 3477 ]<=0;
white[ 3478 ]<=0;
white[ 3479 ]<=0;
white[ 3480 ]<=0;
white[ 3481 ]<=0;
white[ 3482 ]<=0;
white[ 3483 ]<=0;
white[ 3484 ]<=0;
white[ 3485 ]<=0;
white[ 3486 ]<=0;
white[ 3487 ]<=0;
white[ 3488 ]<=0;
white[ 3489 ]<=0;
white[ 3490 ]<=0;
white[ 3491 ]<=0;
white[ 3492 ]<=0;
white[ 3493 ]<=0;
white[ 3494 ]<=0;
white[ 3495 ]<=0;
white[ 3496 ]<=0;
white[ 3497 ]<=0;
white[ 3498 ]<=0;
white[ 3499 ]<=0;
white[ 3500 ]<=1;
white[ 3501 ]<=1;
white[ 3502 ]<=1;
white[ 3503 ]<=1;
white[ 3504 ]<=1;
white[ 3505 ]<=1;
white[ 3506 ]<=1;
```

```
white[ 3507 ]<=1;
white[ 3508 ]<=1;
white[ 3509 ]<=1;
white[ 3510 ]<=1;
white[ 3511 ]<=1;
white[ 3512 ]<=1;
white[ 3513 ]<=1;
white[ 3514 ]<=1;
white[ 3515 ]<=1;
white[ 3516 ]<=1;
white[ 3517 ]<=1;
white[ 3518 ]<=1;
white[ 3519 ]<=1;
white[ 3520 ]<=1;
white[ 3521 ]<=0;
white[ 3522 ]<=0;
white[ 3523 ]<=0;
white[ 3524 ]<=0;
white[ 3525 ]<=0;
white[ 3526 ]<=0;
white[ 3527 ]<=0;
white[ 3528 ]<=0;
white[ 3529 ]<=0;
white[ 3530 ]<=0;
white[ 3531 ]<=0;
white[ 3532 ]<=0;
white[ 3533 ]<=0;
white[ 3534 ]<=0;
white[ 3535 ]<=0;
white[ 3536 ]<=0;
white[ 3537 ]<=0;
white[ 3538 ]<=0;
white[ 3539 ]<=0;
white[ 3540 ]<=0;
white[ 3541 ]<=0;
white[ 3542 ]<=0;
white[ 3543 ]<=0;
white[ 3544 ]<=0;
white[ 3545 ]<=0;
white[ 3546 ]<=0;
white[ 3547 ]<=0;
white[ 3548 ]<=0;
white[ 3549 ]<=0;
white[ 3550 ]<=0;
white[ 3551 ]<=0;
white[ 3552 ]<=0;
white[ 3553 ]<=0;
white[ 3554 ]<=0;
white[ 3555 ]<=0;
white[ 3556 ]<=0;
white[ 3557 ]<=0;
white[ 3558 ]<=0;
white[ 3559 ]<=0;
white[ 3560 ]<=0;
white[ 3561 ]<=0;
white[ 3562 ]<=0;
white[ 3563 ]<=1;
white[ 3564 ]<=1;
```

```
white[ 3565 ]<=1;
white[ 3566 ]<=1;
white[ 3567 ]<=1;
white[ 3568 ]<=1;
white[ 3569 ]<=1;
white[ 3570 ]<=1;
white[ 3571 ]<=1;
white[ 3572 ]<=1;
white[ 3573 ]<=1;
white[ 3574 ]<=1;
white[ 3575 ]<=1;
white[ 3576 ]<=1;
white[ 3577 ]<=1;
white[ 3578 ]<=0;
white[ 3579 ]<=0;
white[ 3580 ]<=0;
white[ 3581 ]<=0;
white[ 3582 ]<=0;
white[ 3583 ]<=0;
white[ 3584 ]<=0;
white[ 3585 ]<=0;
white[ 3586 ]<=0;
white[ 3587 ]<=0;
white[ 3588 ]<=0;
white[ 3589 ]<=0;
white[ 3590 ]<=0;
white[ 3591 ]<=0;
white[ 3592 ]<=0;
white[ 3593 ]<=0;
white[ 3594 ]<=0;
white[ 3595 ]<=0;
white[ 3596 ]<=0;
white[ 3597 ]<=0;
white[ 3598 ]<=0;
white[ 3599 ]<=0;
end

//indicator
18'd99839:begin
white[ 0 ]<=0;
white[ 1 ]<=0;
white[ 2 ]<=0;
white[ 3 ]<=0;
white[ 4 ]<=0;
white[ 5 ]<=0;
white[ 6 ]<=0;
white[ 7 ]<=0;
white[ 8 ]<=0;
white[ 9 ]<=0;
white[ 10 ]<=0;
white[ 11 ]<=0;
white[ 12 ]<=0;
white[ 13 ]<=0;
white[ 14 ]<=0;
white[ 15 ]<=0;
white[ 16 ]<=0;
white[ 17 ]<=0;
white[ 18 ]<=0;
```

```
white[ 19 ]<=0;
white[ 20 ]<=0;
white[ 21 ]<=0;
white[ 22 ]<=0;
white[ 23 ]<=0;
white[ 24 ]<=0;
white[ 25 ]<=0;
white[ 26 ]<=0;
white[ 27 ]<=0;
white[ 28 ]<=0;
white[ 29 ]<=0;
white[ 30 ]<=0;
white[ 31 ]<=0;
white[ 32 ]<=0;
white[ 33 ]<=0;
white[ 34 ]<=0;
white[ 35 ]<=0;
white[ 36 ]<=0;
white[ 37 ]<=0;
white[ 38 ]<=0;
white[ 39 ]<=0;
white[ 40 ]<=0;
white[ 41 ]<=0;
white[ 42 ]<=0;
white[ 43 ]<=0;
white[ 44 ]<=0;
white[ 45 ]<=0;
white[ 46 ]<=0;
white[ 47 ]<=0;
white[ 48 ]<=0;
white[ 49 ]<=0;
white[ 50 ]<=0;
white[ 51 ]<=0;
white[ 52 ]<=0;
white[ 53 ]<=0;
white[ 54 ]<=0;
white[ 55 ]<=0;
white[ 56 ]<=0;
white[ 57 ]<=0;
white[ 58 ]<=0;
white[ 59 ]<=0;
white[ 60 ]<=0;
white[ 61 ]<=0;
white[ 62 ]<=0;
white[ 63 ]<=0;
white[ 64 ]<=0;
white[ 65 ]<=0;
white[ 66 ]<=0;
white[ 67 ]<=0;
white[ 68 ]<=0;
white[ 69 ]<=0;
white[ 70 ]<=0;
white[ 71 ]<=0;
white[ 72 ]<=0;
white[ 73 ]<=0;
white[ 74 ]<=0;
white[ 75 ]<=0;
white[ 76 ]<=0;
```

```
white[ 77 ]<=0;
white[ 78 ]<=0;
white[ 79 ]<=0;
white[ 80 ]<=0;
white[ 81 ]<=0;
white[ 82 ]<=0;
white[ 83 ]<=0;
white[ 84 ]<=0;
white[ 85 ]<=1;
white[ 86 ]<=1;
white[ 87 ]<=1;
white[ 88 ]<=1;
white[ 89 ]<=1;
white[ 90 ]<=1;
white[ 91 ]<=1;
white[ 92 ]<=1;
white[ 93 ]<=1;
white[ 94 ]<=1;
white[ 95 ]<=1;
white[ 96 ]<=0;
white[ 97 ]<=0;
white[ 98 ]<=0;
white[ 99 ]<=0;
white[ 100 ]<=0;
white[ 101 ]<=0;
white[ 102 ]<=0;
white[ 103 ]<=0;
white[ 104 ]<=0;
white[ 105 ]<=0;
white[ 106 ]<=0;
white[ 107 ]<=0;
white[ 108 ]<=0;
white[ 109 ]<=0;
white[ 110 ]<=0;
white[ 111 ]<=0;
white[ 112 ]<=0;
white[ 113 ]<=0;
white[ 114 ]<=0;
white[ 115 ]<=0;
white[ 116 ]<=0;
white[ 117 ]<=0;
white[ 118 ]<=0;
white[ 119 ]<=0;
white[ 120 ]<=0;
white[ 121 ]<=0;
white[ 122 ]<=0;
white[ 123 ]<=0;
white[ 124 ]<=0;
white[ 125 ]<=0;
white[ 126 ]<=0;
white[ 127 ]<=0;
white[ 128 ]<=0;
white[ 129 ]<=0;
white[ 130 ]<=0;
white[ 131 ]<=0;
white[ 132 ]<=0;
white[ 133 ]<=0;
white[ 134 ]<=0;
```

```
white[ 135 ]<=0;
white[ 136 ]<=0;
white[ 137 ]<=0;
white[ 138 ]<=0;
white[ 139 ]<=0;
white[ 140 ]<=0;
white[ 141 ]<=0;
white[ 142 ]<=0;
white[ 143 ]<=1;
white[ 144 ]<=1;
white[ 145 ]<=1;
white[ 146 ]<=1;
white[ 147 ]<=1;
white[ 148 ]<=1;
white[ 149 ]<=1;
white[ 150 ]<=1;
white[ 151 ]<=1;
white[ 152 ]<=1;
white[ 153 ]<=1;
white[ 154 ]<=1;
white[ 155 ]<=1;
white[ 156 ]<=1;
white[ 157 ]<=1;
white[ 158 ]<=0;
white[ 159 ]<=0;
white[ 160 ]<=0;
white[ 161 ]<=0;
white[ 162 ]<=0;
white[ 163 ]<=0;
white[ 164 ]<=0;
white[ 165 ]<=0;
white[ 166 ]<=0;
white[ 167 ]<=0;
white[ 168 ]<=0;
white[ 169 ]<=0;
white[ 170 ]<=0;
white[ 171 ]<=0;
white[ 172 ]<=0;
white[ 173 ]<=0;
white[ 174 ]<=0;
white[ 175 ]<=0;
white[ 176 ]<=0;
white[ 177 ]<=0;
white[ 178 ]<=0;
white[ 179 ]<=0;
white[ 180 ]<=0;
white[ 181 ]<=0;
white[ 182 ]<=0;
white[ 183 ]<=0;
white[ 184 ]<=0;
white[ 185 ]<=0;
white[ 186 ]<=0;
white[ 187 ]<=0;
white[ 188 ]<=0;
white[ 189 ]<=0;
white[ 190 ]<=0;
white[ 191 ]<=0;
white[ 192 ]<=0;
```

```
white[ 193 ]<=0;
white[ 194 ]<=0;
white[ 195 ]<=0;
white[ 196 ]<=0;
white[ 197 ]<=0;
white[ 198 ]<=0;
white[ 199 ]<=0;
white[ 200 ]<=0;
white[ 201 ]<=1;
white[ 202 ]<=1;
white[ 203 ]<=1;
white[ 204 ]<=1;
white[ 205 ]<=1;
white[ 206 ]<=1;
white[ 207 ]<=1;
white[ 208 ]<=1;
white[ 209 ]<=1;
white[ 210 ]<=1;
white[ 211 ]<=1;
white[ 212 ]<=1;
white[ 213 ]<=1;
white[ 214 ]<=1;
white[ 215 ]<=1;
white[ 216 ]<=1;
white[ 217 ]<=1;
white[ 218 ]<=1;
white[ 219 ]<=1;
white[ 220 ]<=0;
white[ 221 ]<=0;
white[ 222 ]<=0;
white[ 223 ]<=0;
white[ 224 ]<=0;
white[ 225 ]<=0;
white[ 226 ]<=0;
white[ 227 ]<=0;
white[ 228 ]<=0;
white[ 229 ]<=0;
white[ 230 ]<=0;
white[ 231 ]<=0;
white[ 232 ]<=0;
white[ 233 ]<=0;
white[ 234 ]<=0;
white[ 235 ]<=0;
white[ 236 ]<=0;
white[ 237 ]<=0;
white[ 238 ]<=0;
white[ 239 ]<=0;
white[ 240 ]<=0;
white[ 241 ]<=0;
white[ 242 ]<=0;
white[ 243 ]<=0;
white[ 244 ]<=0;
white[ 245 ]<=0;
white[ 246 ]<=0;
white[ 247 ]<=0;
white[ 248 ]<=0;
white[ 249 ]<=0;
white[ 250 ]<=0;
```

```
white[ 251 ]<=0;
white[ 252 ]<=0;
white[ 253 ]<=0;
white[ 254 ]<=0;
white[ 255 ]<=0;
white[ 256 ]<=0;
white[ 257 ]<=0;
white[ 258 ]<=0;
white[ 259 ]<=1;
white[ 260 ]<=1;
white[ 261 ]<=1;
white[ 262 ]<=1;
white[ 263 ]<=1;
white[ 264 ]<=1;
white[ 265 ]<=1;
white[ 266 ]<=1;
white[ 267 ]<=1;
white[ 268 ]<=1;
white[ 269 ]<=1;
white[ 270 ]<=1;
white[ 271 ]<=1;
white[ 272 ]<=1;
white[ 273 ]<=1;
white[ 274 ]<=1;
white[ 275 ]<=1;
white[ 276 ]<=1;
white[ 277 ]<=1;
white[ 278 ]<=1;
white[ 279 ]<=1;
white[ 280 ]<=1;
white[ 281 ]<=1;
white[ 282 ]<=0;
white[ 283 ]<=0;
white[ 284 ]<=0;
white[ 285 ]<=0;
white[ 286 ]<=0;
white[ 287 ]<=0;
white[ 288 ]<=0;
white[ 289 ]<=0;
white[ 290 ]<=0;
white[ 291 ]<=0;
white[ 292 ]<=0;
white[ 293 ]<=0;
white[ 294 ]<=0;
white[ 295 ]<=0;
white[ 296 ]<=0;
white[ 297 ]<=0;
white[ 298 ]<=0;
white[ 299 ]<=0;
white[ 300 ]<=0;
white[ 301 ]<=0;
white[ 302 ]<=0;
white[ 303 ]<=0;
white[ 304 ]<=0;
white[ 305 ]<=0;
white[ 306 ]<=0;
white[ 307 ]<=0;
white[ 308 ]<=0;
```

```
white[ 309 ]<=0;
white[ 310 ]<=0;
white[ 311 ]<=0;
white[ 312 ]<=0;
white[ 313 ]<=0;
white[ 314 ]<=0;
white[ 315 ]<=0;
white[ 316 ]<=0;
white[ 317 ]<=0;
white[ 318 ]<=1;
white[ 319 ]<=1;
white[ 320 ]<=1;
white[ 321 ]<=1;
white[ 322 ]<=1;
white[ 323 ]<=1;
white[ 324 ]<=1;
white[ 325 ]<=1;
white[ 326 ]<=1;
white[ 327 ]<=1;
white[ 328 ]<=1;
white[ 329 ]<=1;
white[ 330 ]<=1;
white[ 331 ]<=1;
white[ 332 ]<=1;
white[ 333 ]<=1;
white[ 334 ]<=1;
white[ 335 ]<=1;
white[ 336 ]<=1;
white[ 337 ]<=1;
white[ 338 ]<=1;
white[ 339 ]<=1;
white[ 340 ]<=1;
white[ 341 ]<=1;
white[ 342 ]<=1;
white[ 343 ]<=0;
white[ 344 ]<=0;
white[ 345 ]<=0;
white[ 346 ]<=0;
white[ 347 ]<=0;
white[ 348 ]<=0;
white[ 349 ]<=0;
white[ 350 ]<=0;
white[ 351 ]<=0;
white[ 352 ]<=0;
white[ 353 ]<=0;
white[ 354 ]<=0;
white[ 355 ]<=0;
white[ 356 ]<=0;
white[ 357 ]<=0;
white[ 358 ]<=0;
white[ 359 ]<=0;
white[ 360 ]<=0;
white[ 361 ]<=0;
white[ 362 ]<=0;
white[ 363 ]<=0;
white[ 364 ]<=0;
white[ 365 ]<=0;
white[ 366 ]<=0;
```

```
white[ 367 ]<=0;
white[ 368 ]<=0;
white[ 369 ]<=0;
white[ 370 ]<=0;
white[ 371 ]<=0;
white[ 372 ]<=0;
white[ 373 ]<=0;
white[ 374 ]<=0;
white[ 375 ]<=0;
white[ 376 ]<=0;
white[ 377 ]<=1;
white[ 378 ]<=1;
white[ 379 ]<=1;
white[ 380 ]<=1;
white[ 381 ]<=1;
white[ 382 ]<=1;
white[ 383 ]<=1;
white[ 384 ]<=1;
white[ 385 ]<=1;
white[ 386 ]<=1;
white[ 387 ]<=1;
white[ 388 ]<=1;
white[ 389 ]<=1;
white[ 390 ]<=1;
white[ 391 ]<=1;
white[ 392 ]<=1;
white[ 393 ]<=1;
white[ 394 ]<=1;
white[ 395 ]<=1;
white[ 396 ]<=1;
white[ 397 ]<=1;
white[ 398 ]<=1;
white[ 399 ]<=1;
white[ 400 ]<=1;
white[ 401 ]<=1;
white[ 402 ]<=1;
white[ 403 ]<=1;
white[ 404 ]<=0;
white[ 405 ]<=0;
white[ 406 ]<=0;
white[ 407 ]<=0;
white[ 408 ]<=0;
white[ 409 ]<=0;
white[ 410 ]<=0;
white[ 411 ]<=0;
white[ 412 ]<=0;
white[ 413 ]<=0;
white[ 414 ]<=0;
white[ 415 ]<=0;
white[ 416 ]<=0;
white[ 417 ]<=0;
white[ 418 ]<=0;
white[ 419 ]<=0;
white[ 420 ]<=0;
white[ 421 ]<=0;
white[ 422 ]<=0;
white[ 423 ]<=0;
white[ 424 ]<=0;
```

```
white[ 425 ]<=0;
white[ 426 ]<=0;
white[ 427 ]<=0;
white[ 428 ]<=0;
white[ 429 ]<=0;
white[ 430 ]<=0;
white[ 431 ]<=0;
white[ 432 ]<=0;
white[ 433 ]<=0;
white[ 434 ]<=0;
white[ 435 ]<=0;
white[ 436 ]<=1;
white[ 437 ]<=1;
white[ 438 ]<=1;
white[ 439 ]<=1;
white[ 440 ]<=1;
white[ 441 ]<=1;
white[ 442 ]<=1;
white[ 443 ]<=1;
white[ 444 ]<=1;
white[ 445 ]<=1;
white[ 446 ]<=1;
white[ 447 ]<=1;
white[ 448 ]<=1;
white[ 449 ]<=1;
white[ 450 ]<=1;
white[ 451 ]<=1;
white[ 452 ]<=1;
white[ 453 ]<=1;
white[ 454 ]<=1;
white[ 455 ]<=1;
white[ 456 ]<=1;
white[ 457 ]<=1;
white[ 458 ]<=1;
white[ 459 ]<=1;
white[ 460 ]<=1;
white[ 461 ]<=1;
white[ 462 ]<=1;
white[ 463 ]<=1;
white[ 464 ]<=1;
white[ 465 ]<=0;
white[ 466 ]<=0;
white[ 467 ]<=0;
white[ 468 ]<=0;
white[ 469 ]<=0;
white[ 470 ]<=0;
white[ 471 ]<=0;
white[ 472 ]<=0;
white[ 473 ]<=0;
white[ 474 ]<=0;
white[ 475 ]<=0;
white[ 476 ]<=0;
white[ 477 ]<=0;
white[ 478 ]<=0;
white[ 479 ]<=0;
white[ 480 ]<=0;
white[ 481 ]<=0;
white[ 482 ]<=0;
```

```
white[ 483 ]<=0;
white[ 484 ]<=0;
white[ 485 ]<=0;
white[ 486 ]<=0;
white[ 487 ]<=0;
white[ 488 ]<=0;
white[ 489 ]<=0;
white[ 490 ]<=0;
white[ 491 ]<=0;
white[ 492 ]<=0;
white[ 493 ]<=0;
white[ 494 ]<=0;
white[ 495 ]<=0;
white[ 496 ]<=1;
white[ 497 ]<=1;
white[ 498 ]<=1;
white[ 499 ]<=1;
white[ 500 ]<=1;
white[ 501 ]<=1;
white[ 502 ]<=1;
white[ 503 ]<=1;
white[ 504 ]<=1;
white[ 505 ]<=1;
white[ 506 ]<=1;
white[ 507 ]<=1;
white[ 508 ]<=1;
white[ 509 ]<=1;
white[ 510 ]<=1;
white[ 511 ]<=1;
white[ 512 ]<=1;
white[ 513 ]<=1;
white[ 514 ]<=1;
white[ 515 ]<=1;
white[ 516 ]<=1;
white[ 517 ]<=1;
white[ 518 ]<=1;
white[ 519 ]<=1;
white[ 520 ]<=1;
white[ 521 ]<=1;
white[ 522 ]<=1;
white[ 523 ]<=1;
white[ 524 ]<=1;
white[ 525 ]<=1;
white[ 526 ]<=0;
white[ 527 ]<=0;
white[ 528 ]<=0;
white[ 529 ]<=0;
white[ 530 ]<=0;
white[ 531 ]<=0;
white[ 532 ]<=0;
white[ 533 ]<=0;
white[ 534 ]<=0;
white[ 535 ]<=0;
white[ 536 ]<=0;
white[ 537 ]<=0;
white[ 538 ]<=0;
white[ 539 ]<=0;
white[ 540 ]<=0;
```

```
white[ 541 ]<=0;
white[ 542 ]<=0;
white[ 543 ]<=0;
white[ 544 ]<=0;
white[ 545 ]<=0;
white[ 546 ]<=0;
white[ 547 ]<=0;
white[ 548 ]<=0;
white[ 549 ]<=0;
white[ 550 ]<=0;
white[ 551 ]<=0;
white[ 552 ]<=0;
white[ 553 ]<=0;
white[ 554 ]<=0;
white[ 555 ]<=1;
white[ 556 ]<=1;
white[ 557 ]<=1;
white[ 558 ]<=1;
white[ 559 ]<=1;
white[ 560 ]<=1;
white[ 561 ]<=1;
white[ 562 ]<=1;
white[ 563 ]<=1;
white[ 564 ]<=1;
white[ 565 ]<=1;
white[ 566 ]<=1;
white[ 567 ]<=1;
white[ 568 ]<=1;
white[ 569 ]<=1;
white[ 570 ]<=1;
white[ 571 ]<=1;
white[ 572 ]<=1;
white[ 573 ]<=1;
white[ 574 ]<=1;
white[ 575 ]<=1;
white[ 576 ]<=1;
white[ 577 ]<=1;
white[ 578 ]<=1;
white[ 579 ]<=1;
white[ 580 ]<=1;
white[ 581 ]<=1;
white[ 582 ]<=1;
white[ 583 ]<=1;
white[ 584 ]<=1;
white[ 585 ]<=1;
white[ 586 ]<=0;
white[ 587 ]<=0;
white[ 588 ]<=0;
white[ 589 ]<=0;
white[ 590 ]<=0;
white[ 591 ]<=0;
white[ 592 ]<=0;
white[ 593 ]<=0;
white[ 594 ]<=0;
white[ 595 ]<=0;
white[ 596 ]<=0;
white[ 597 ]<=0;
white[ 598 ]<=0;
```

```
white[ 599 ]<=0;
white[ 600 ]<=0;
white[ 601 ]<=0;
white[ 602 ]<=0;
white[ 603 ]<=0;
white[ 604 ]<=0;
white[ 605 ]<=0;
white[ 606 ]<=0;
white[ 607 ]<=0;
white[ 608 ]<=0;
white[ 609 ]<=0;
white[ 610 ]<=0;
white[ 611 ]<=0;
white[ 612 ]<=0;
white[ 613 ]<=0;
white[ 614 ]<=1;
white[ 615 ]<=1;
white[ 616 ]<=1;
white[ 617 ]<=1;
white[ 618 ]<=1;
white[ 619 ]<=1;
white[ 620 ]<=1;
white[ 621 ]<=1;
white[ 622 ]<=1;
white[ 623 ]<=1;
white[ 624 ]<=1;
white[ 625 ]<=1;
white[ 626 ]<=1;
white[ 627 ]<=1;
white[ 628 ]<=1;
white[ 629 ]<=1;
white[ 630 ]<=1;
white[ 631 ]<=1;
white[ 632 ]<=1;
white[ 633 ]<=1;
white[ 634 ]<=1;
white[ 635 ]<=1;
white[ 636 ]<=1;
white[ 637 ]<=1;
white[ 638 ]<=1;
white[ 639 ]<=1;
white[ 640 ]<=1;
white[ 641 ]<=1;
white[ 642 ]<=1;
white[ 643 ]<=1;
white[ 644 ]<=1;
white[ 645 ]<=1;
white[ 646 ]<=1;
white[ 647 ]<=0;
white[ 648 ]<=0;
white[ 649 ]<=0;
white[ 650 ]<=0;
white[ 651 ]<=0;
white[ 652 ]<=0;
white[ 653 ]<=0;
white[ 654 ]<=0;
white[ 655 ]<=0;
white[ 656 ]<=0;
```

```
white[ 657 ]<=0;
white[ 658 ]<=0;
white[ 659 ]<=0;
white[ 660 ]<=0;
white[ 661 ]<=0;
white[ 662 ]<=0;
white[ 663 ]<=0;
white[ 664 ]<=0;
white[ 665 ]<=0;
white[ 666 ]<=0;
white[ 667 ]<=0;
white[ 668 ]<=0;
white[ 669 ]<=0;
white[ 670 ]<=0;
white[ 671 ]<=0;
white[ 672 ]<=0;
white[ 673 ]<=0;
white[ 674 ]<=1;
white[ 675 ]<=1;
white[ 676 ]<=1;
white[ 677 ]<=1;
white[ 678 ]<=1;
white[ 679 ]<=1;
white[ 680 ]<=1;
white[ 681 ]<=1;
white[ 682 ]<=1;
white[ 683 ]<=1;
white[ 684 ]<=1;
white[ 685 ]<=1;
white[ 686 ]<=1;
white[ 687 ]<=1;
white[ 688 ]<=1;
white[ 689 ]<=1;
white[ 690 ]<=1;
white[ 691 ]<=1;
white[ 692 ]<=1;
white[ 693 ]<=1;
white[ 694 ]<=1;
white[ 695 ]<=1;
white[ 696 ]<=1;
white[ 697 ]<=1;
white[ 698 ]<=1;
white[ 699 ]<=1;
white[ 700 ]<=1;
white[ 701 ]<=1;
white[ 702 ]<=1;
white[ 703 ]<=1;
white[ 704 ]<=1;
white[ 705 ]<=1;
white[ 706 ]<=1;
white[ 707 ]<=0;
white[ 708 ]<=0;
white[ 709 ]<=0;
white[ 710 ]<=0;
white[ 711 ]<=0;
white[ 712 ]<=0;
white[ 713 ]<=0;
white[ 714 ]<=0;
```

```
white[ 715 ]<=0;
white[ 716 ]<=0;
white[ 717 ]<=0;
white[ 718 ]<=0;
white[ 719 ]<=0;
white[ 720 ]<=0;
white[ 721 ]<=0;
white[ 722 ]<=0;
white[ 723 ]<=0;
white[ 724 ]<=0;
white[ 725 ]<=0;
white[ 726 ]<=0;
white[ 727 ]<=0;
white[ 728 ]<=0;
white[ 729 ]<=0;
white[ 730 ]<=0;
white[ 731 ]<=0;
white[ 732 ]<=0;
white[ 733 ]<=1;
white[ 734 ]<=1;
white[ 735 ]<=1;
white[ 736 ]<=1;
white[ 737 ]<=1;
white[ 738 ]<=1;
white[ 739 ]<=1;
white[ 740 ]<=1;
white[ 741 ]<=1;
white[ 742 ]<=1;
white[ 743 ]<=1;
white[ 744 ]<=1;
white[ 745 ]<=1;
white[ 746 ]<=1;
white[ 747 ]<=1;
white[ 748 ]<=1;
white[ 749 ]<=1;
white[ 750 ]<=1;
white[ 751 ]<=1;
white[ 752 ]<=1;
white[ 753 ]<=1;
white[ 754 ]<=1;
white[ 755 ]<=1;
white[ 756 ]<=1;
white[ 757 ]<=1;
white[ 758 ]<=1;
white[ 759 ]<=1;
white[ 760 ]<=1;
white[ 761 ]<=1;
white[ 762 ]<=1;
white[ 763 ]<=1;
white[ 764 ]<=1;
white[ 765 ]<=1;
white[ 766 ]<=1;
white[ 767 ]<=1;
white[ 768 ]<=0;
white[ 769 ]<=0;
white[ 770 ]<=0;
white[ 771 ]<=0;
white[ 772 ]<=0;
```

```
white[ 773 ]<=0;
white[ 774 ]<=0;
white[ 775 ]<=0;
white[ 776 ]<=0;
white[ 777 ]<=0;
white[ 778 ]<=0;
white[ 779 ]<=0;
white[ 780 ]<=0;
white[ 781 ]<=0;
white[ 782 ]<=0;
white[ 783 ]<=0;
white[ 784 ]<=0;
white[ 785 ]<=0;
white[ 786 ]<=0;
white[ 787 ]<=0;
white[ 788 ]<=0;
white[ 789 ]<=0;
white[ 790 ]<=0;
white[ 791 ]<=0;
white[ 792 ]<=0;
white[ 793 ]<=1;
white[ 794 ]<=1;
white[ 795 ]<=1;
white[ 796 ]<=1;
white[ 797 ]<=1;
white[ 798 ]<=1;
white[ 799 ]<=1;
white[ 800 ]<=1;
white[ 801 ]<=1;
white[ 802 ]<=1;
white[ 803 ]<=1;
white[ 804 ]<=1;
white[ 805 ]<=1;
white[ 806 ]<=1;
white[ 807 ]<=1;
white[ 808 ]<=1;
white[ 809 ]<=1;
white[ 810 ]<=1;
white[ 811 ]<=1;
white[ 812 ]<=0;
white[ 813 ]<=0;
white[ 814 ]<=1;
white[ 815 ]<=1;
white[ 816 ]<=1;
white[ 817 ]<=1;
white[ 818 ]<=1;
white[ 819 ]<=1;
white[ 820 ]<=1;
white[ 821 ]<=1;
white[ 822 ]<=1;
white[ 823 ]<=1;
white[ 824 ]<=1;
white[ 825 ]<=1;
white[ 826 ]<=1;
white[ 827 ]<=1;
white[ 828 ]<=0;
white[ 829 ]<=0;
white[ 830 ]<=0;
```

```
white[ 831 ]<=0;
white[ 832 ]<=0;
white[ 833 ]<=0;
white[ 834 ]<=0;
white[ 835 ]<=0;
white[ 836 ]<=0;
white[ 837 ]<=0;
white[ 838 ]<=0;
white[ 839 ]<=0;
white[ 840 ]<=0;
white[ 841 ]<=0;
white[ 842 ]<=0;
white[ 843 ]<=0;
white[ 844 ]<=0;
white[ 845 ]<=0;
white[ 846 ]<=0;
white[ 847 ]<=0;
white[ 848 ]<=0;
white[ 849 ]<=0;
white[ 850 ]<=0;
white[ 851 ]<=0;
white[ 852 ]<=0;
white[ 853 ]<=1;
white[ 854 ]<=1;
white[ 855 ]<=1;
white[ 856 ]<=1;
white[ 857 ]<=1;
white[ 858 ]<=1;
white[ 859 ]<=1;
white[ 860 ]<=1;
white[ 861 ]<=1;
white[ 862 ]<=1;
white[ 863 ]<=1;
white[ 864 ]<=1;
white[ 865 ]<=1;
white[ 866 ]<=1;
white[ 867 ]<=1;
white[ 868 ]<=1;
white[ 869 ]<=1;
white[ 870 ]<=1;
white[ 871 ]<=0;
white[ 872 ]<=0;
white[ 873 ]<=0;
white[ 874 ]<=0;
white[ 875 ]<=1;
white[ 876 ]<=1;
white[ 877 ]<=1;
white[ 878 ]<=1;
white[ 879 ]<=1;
white[ 880 ]<=1;
white[ 881 ]<=1;
white[ 882 ]<=1;
white[ 883 ]<=1;
white[ 884 ]<=1;
white[ 885 ]<=1;
white[ 886 ]<=1;
white[ 887 ]<=1;
white[ 888 ]<=0;
```

```
white[ 889 ]<=0;
white[ 890 ]<=0;
white[ 891 ]<=0;
white[ 892 ]<=0;
white[ 893 ]<=0;
white[ 894 ]<=0;
white[ 895 ]<=0;
white[ 896 ]<=0;
white[ 897 ]<=0;
white[ 898 ]<=0;
white[ 899 ]<=0;
white[ 900 ]<=0;
white[ 901 ]<=0;
white[ 902 ]<=0;
white[ 903 ]<=0;
white[ 904 ]<=0;
white[ 905 ]<=0;
white[ 906 ]<=0;
white[ 907 ]<=0;
white[ 908 ]<=0;
white[ 909 ]<=0;
white[ 910 ]<=0;
white[ 911 ]<=0;
white[ 912 ]<=1;
white[ 913 ]<=1;
white[ 914 ]<=1;
white[ 915 ]<=1;
white[ 916 ]<=1;
white[ 917 ]<=1;
white[ 918 ]<=1;
white[ 919 ]<=1;
white[ 920 ]<=1;
white[ 921 ]<=1;
white[ 922 ]<=1;
white[ 923 ]<=1;
white[ 924 ]<=1;
white[ 925 ]<=1;
white[ 926 ]<=1;
white[ 927 ]<=1;
white[ 928 ]<=1;
white[ 929 ]<=1;
white[ 930 ]<=1;
white[ 931 ]<=0;
white[ 932 ]<=0;
white[ 933 ]<=0;
white[ 934 ]<=0;
white[ 935 ]<=1;
white[ 936 ]<=1;
white[ 937 ]<=1;
white[ 938 ]<=1;
white[ 939 ]<=1;
white[ 940 ]<=1;
white[ 941 ]<=1;
white[ 942 ]<=1;
white[ 943 ]<=1;
white[ 944 ]<=1;
white[ 945 ]<=1;
white[ 946 ]<=1;
```

```
white[ 947 ]<=1;
white[ 948 ]<=1;
white[ 949 ]<=0;
white[ 950 ]<=0;
white[ 951 ]<=0;
white[ 952 ]<=0;
white[ 953 ]<=0;
white[ 954 ]<=0;
white[ 955 ]<=0;
white[ 956 ]<=0;
white[ 957 ]<=0;
white[ 958 ]<=0;
white[ 959 ]<=0;
white[ 960 ]<=0;
white[ 961 ]<=0;
white[ 962 ]<=0;
white[ 963 ]<=0;
white[ 964 ]<=0;
white[ 965 ]<=0;
white[ 966 ]<=0;
white[ 967 ]<=0;
white[ 968 ]<=0;
white[ 969 ]<=0;
white[ 970 ]<=0;
white[ 971 ]<=0;
white[ 972 ]<=1;
white[ 973 ]<=1;
white[ 974 ]<=1;
white[ 975 ]<=1;
white[ 976 ]<=1;
white[ 977 ]<=1;
white[ 978 ]<=1;
white[ 979 ]<=1;
white[ 980 ]<=1;
white[ 981 ]<=1;
white[ 982 ]<=1;
white[ 983 ]<=1;
white[ 984 ]<=1;
white[ 985 ]<=1;
white[ 986 ]<=1;
white[ 987 ]<=1;
white[ 988 ]<=1;
white[ 989 ]<=1;
white[ 990 ]<=1;
white[ 991 ]<=0;
white[ 992 ]<=0;
white[ 993 ]<=0;
white[ 994 ]<=0;
white[ 995 ]<=1;
white[ 996 ]<=1;
white[ 997 ]<=1;
white[ 998 ]<=1;
white[ 999 ]<=1;
white[ 1000 ]<=1;
white[ 1001 ]<=1;
white[ 1002 ]<=1;
white[ 1003 ]<=1;
white[ 1004 ]<=1;
```

```
white[ 1005 ]<=1;
white[ 1006 ]<=1;
white[ 1007 ]<=1;
white[ 1008 ]<=1;
white[ 1009 ]<=0;
white[ 1010 ]<=0;
white[ 1011 ]<=0;
white[ 1012 ]<=0;
white[ 1013 ]<=0;
white[ 1014 ]<=0;
white[ 1015 ]<=0;
white[ 1016 ]<=0;
white[ 1017 ]<=0;
white[ 1018 ]<=0;
white[ 1019 ]<=0;
white[ 1020 ]<=0;
white[ 1021 ]<=0;
white[ 1022 ]<=0;
white[ 1023 ]<=0;
white[ 1024 ]<=0;
white[ 1025 ]<=0;
white[ 1026 ]<=0;
white[ 1027 ]<=0;
white[ 1028 ]<=0;
white[ 1029 ]<=0;
white[ 1030 ]<=0;
white[ 1031 ]<=1;
white[ 1032 ]<=1;
white[ 1033 ]<=1;
white[ 1034 ]<=1;
white[ 1035 ]<=1;
white[ 1036 ]<=1;
white[ 1037 ]<=1;
white[ 1038 ]<=1;
white[ 1039 ]<=1;
white[ 1040 ]<=1;
white[ 1041 ]<=1;
white[ 1042 ]<=1;
white[ 1043 ]<=1;
white[ 1044 ]<=1;
white[ 1045 ]<=1;
white[ 1046 ]<=1;
white[ 1047 ]<=1;
white[ 1048 ]<=1;
white[ 1049 ]<=1;
white[ 1050 ]<=1;
white[ 1051 ]<=1;
white[ 1052 ]<=1;
white[ 1053 ]<=1;
white[ 1054 ]<=1;
white[ 1055 ]<=1;
white[ 1056 ]<=1;
white[ 1057 ]<=1;
white[ 1058 ]<=1;
white[ 1059 ]<=1;
white[ 1060 ]<=1;
white[ 1061 ]<=1;
white[ 1062 ]<=1;
```

```
white[ 1063 ]<=1;
white[ 1064 ]<=1;
white[ 1065 ]<=1;
white[ 1066 ]<=1;
white[ 1067 ]<=1;
white[ 1068 ]<=1;
white[ 1069 ]<=1;
white[ 1070 ]<=0;
white[ 1071 ]<=0;
white[ 1072 ]<=0;
white[ 1073 ]<=0;
white[ 1074 ]<=0;
white[ 1075 ]<=0;
white[ 1076 ]<=0;
white[ 1077 ]<=0;
white[ 1078 ]<=0;
white[ 1079 ]<=0;
white[ 1080 ]<=0;
white[ 1081 ]<=0;
white[ 1082 ]<=0;
white[ 1083 ]<=0;
white[ 1084 ]<=0;
white[ 1085 ]<=0;
white[ 1086 ]<=0;
white[ 1087 ]<=0;
white[ 1088 ]<=0;
white[ 1089 ]<=0;
white[ 1090 ]<=1;
white[ 1091 ]<=1;
white[ 1092 ]<=1;
white[ 1093 ]<=1;
white[ 1094 ]<=1;
white[ 1095 ]<=1;
white[ 1096 ]<=1;
white[ 1097 ]<=1;
white[ 1098 ]<=1;
white[ 1099 ]<=1;
white[ 1100 ]<=1;
white[ 1101 ]<=1;
white[ 1102 ]<=1;
white[ 1103 ]<=1;
white[ 1104 ]<=1;
white[ 1105 ]<=1;
white[ 1106 ]<=1;
white[ 1107 ]<=1;
white[ 1108 ]<=1;
white[ 1109 ]<=1;
white[ 1110 ]<=1;
white[ 1111 ]<=1;
white[ 1112 ]<=1;
white[ 1113 ]<=1;
white[ 1114 ]<=1;
white[ 1115 ]<=1;
white[ 1116 ]<=1;
white[ 1117 ]<=1;
white[ 1118 ]<=1;
white[ 1119 ]<=1;
white[ 1120 ]<=1;
```

```
white[ 1121 ]<=1;
white[ 1122 ]<=1;
white[ 1123 ]<=1;
white[ 1124 ]<=1;
white[ 1125 ]<=1;
white[ 1126 ]<=1;
white[ 1127 ]<=1;
white[ 1128 ]<=1;
white[ 1129 ]<=1;
white[ 1130 ]<=1;
white[ 1131 ]<=0;
white[ 1132 ]<=0;
white[ 1133 ]<=0;
white[ 1134 ]<=0;
white[ 1135 ]<=0;
white[ 1136 ]<=0;
white[ 1137 ]<=0;
white[ 1138 ]<=0;
white[ 1139 ]<=0;
white[ 1140 ]<=0;
white[ 1141 ]<=0;
white[ 1142 ]<=0;
white[ 1143 ]<=0;
white[ 1144 ]<=0;
white[ 1145 ]<=0;
white[ 1146 ]<=0;
white[ 1147 ]<=0;
white[ 1148 ]<=0;
white[ 1149 ]<=0;
white[ 1150 ]<=1;
white[ 1151 ]<=1;
white[ 1152 ]<=1;
white[ 1153 ]<=1;
white[ 1154 ]<=1;
white[ 1155 ]<=1;
white[ 1156 ]<=1;
white[ 1157 ]<=1;
white[ 1158 ]<=1;
white[ 1159 ]<=1;
white[ 1160 ]<=1;
white[ 1161 ]<=1;
white[ 1162 ]<=1;
white[ 1163 ]<=1;
white[ 1164 ]<=1;
white[ 1165 ]<=1;
white[ 1166 ]<=1;
white[ 1167 ]<=1;
white[ 1168 ]<=1;
white[ 1169 ]<=1;
white[ 1170 ]<=1;
white[ 1171 ]<=1;
white[ 1172 ]<=1;
white[ 1173 ]<=1;
white[ 1174 ]<=1;
white[ 1175 ]<=1;
white[ 1176 ]<=1;
white[ 1177 ]<=1;
white[ 1178 ]<=1;
```

```
white[ 1179 ]<=1;
white[ 1180 ]<=1;
white[ 1181 ]<=1;
white[ 1182 ]<=1;
white[ 1183 ]<=1;
white[ 1184 ]<=1;
white[ 1185 ]<=1;
white[ 1186 ]<=1;
white[ 1187 ]<=1;
white[ 1188 ]<=1;
white[ 1189 ]<=1;
white[ 1190 ]<=1;
white[ 1191 ]<=0;
white[ 1192 ]<=0;
white[ 1193 ]<=0;
white[ 1194 ]<=0;
white[ 1195 ]<=0;
white[ 1196 ]<=0;
white[ 1197 ]<=0;
white[ 1198 ]<=0;
white[ 1199 ]<=0;
white[ 1200 ]<=0;
white[ 1201 ]<=0;
white[ 1202 ]<=0;
white[ 1203 ]<=0;
white[ 1204 ]<=0;
white[ 1205 ]<=0;
white[ 1206 ]<=0;
white[ 1207 ]<=0;
white[ 1208 ]<=1;
white[ 1209 ]<=1;
white[ 1210 ]<=1;
white[ 1211 ]<=1;
white[ 1212 ]<=1;
white[ 1213 ]<=1;
white[ 1214 ]<=1;
white[ 1215 ]<=1;
white[ 1216 ]<=1;
white[ 1217 ]<=1;
white[ 1218 ]<=1;
white[ 1219 ]<=1;
white[ 1220 ]<=1;
white[ 1221 ]<=1;
white[ 1222 ]<=1;
white[ 1223 ]<=1;
white[ 1224 ]<=1;
white[ 1225 ]<=1;
white[ 1226 ]<=1;
white[ 1227 ]<=1;
white[ 1228 ]<=1;
white[ 1229 ]<=1;
white[ 1230 ]<=1;
white[ 1231 ]<=1;
white[ 1232 ]<=1;
white[ 1233 ]<=1;
white[ 1234 ]<=1;
white[ 1235 ]<=1;
white[ 1236 ]<=1;
```

```
white[ 1237 ]<=1;
white[ 1238 ]<=1;
white[ 1239 ]<=1;
white[ 1240 ]<=1;
white[ 1241 ]<=1;
white[ 1242 ]<=1;
white[ 1243 ]<=1;
white[ 1244 ]<=1;
white[ 1245 ]<=1;
white[ 1246 ]<=1;
white[ 1247 ]<=1;
white[ 1248 ]<=1;
white[ 1249 ]<=1;
white[ 1250 ]<=1;
white[ 1251 ]<=1;
white[ 1252 ]<=1;
white[ 1253 ]<=0;
white[ 1254 ]<=0;
white[ 1255 ]<=0;
white[ 1256 ]<=0;
white[ 1257 ]<=0;
white[ 1258 ]<=0;
white[ 1259 ]<=0;
white[ 1260 ]<=0;
white[ 1261 ]<=0;
white[ 1262 ]<=0;
white[ 1263 ]<=0;
white[ 1264 ]<=0;
white[ 1265 ]<=0;
white[ 1266 ]<=1;
white[ 1267 ]<=1;
white[ 1268 ]<=1;
white[ 1269 ]<=1;
white[ 1270 ]<=1;
white[ 1271 ]<=1;
white[ 1272 ]<=1;
white[ 1273 ]<=1;
white[ 1274 ]<=1;
white[ 1275 ]<=1;
white[ 1276 ]<=1;
white[ 1277 ]<=1;
white[ 1278 ]<=1;
white[ 1279 ]<=1;
white[ 1280 ]<=1;
white[ 1281 ]<=1;
white[ 1282 ]<=1;
white[ 1283 ]<=1;
white[ 1284 ]<=1;
white[ 1285 ]<=1;
white[ 1286 ]<=1;
white[ 1287 ]<=1;
white[ 1288 ]<=1;
white[ 1289 ]<=1;
white[ 1290 ]<=1;
white[ 1291 ]<=1;
white[ 1292 ]<=1;
white[ 1293 ]<=1;
white[ 1294 ]<=1;
```

```
white[ 1295 ]<=1;
white[ 1296 ]<=1;
white[ 1297 ]<=1;
white[ 1298 ]<=1;
white[ 1299 ]<=1;
white[ 1300 ]<=1;
white[ 1301 ]<=1;
white[ 1302 ]<=1;
white[ 1303 ]<=1;
white[ 1304 ]<=1;
white[ 1305 ]<=1;
white[ 1306 ]<=1;
white[ 1307 ]<=1;
white[ 1308 ]<=1;
white[ 1309 ]<=1;
white[ 1310 ]<=1;
white[ 1311 ]<=1;
white[ 1312 ]<=1;
white[ 1313 ]<=1;
white[ 1314 ]<=1;
white[ 1315 ]<=0;
white[ 1316 ]<=0;
white[ 1317 ]<=0;
white[ 1318 ]<=0;
white[ 1319 ]<=0;
white[ 1320 ]<=0;
white[ 1321 ]<=0;
white[ 1322 ]<=0;
white[ 1323 ]<=0;
white[ 1324 ]<=0;
white[ 1325 ]<=0;
white[ 1326 ]<=1;
white[ 1327 ]<=1;
white[ 1328 ]<=1;
white[ 1329 ]<=1;
white[ 1330 ]<=1;
white[ 1331 ]<=1;
white[ 1332 ]<=1;
white[ 1333 ]<=1;
white[ 1334 ]<=1;
white[ 1335 ]<=1;
white[ 1336 ]<=1;
white[ 1337 ]<=1;
white[ 1338 ]<=1;
white[ 1339 ]<=1;
white[ 1340 ]<=1;
white[ 1341 ]<=1;
white[ 1342 ]<=1;
white[ 1343 ]<=1;
white[ 1344 ]<=1;
white[ 1345 ]<=1;
white[ 1346 ]<=1;
white[ 1347 ]<=1;
white[ 1348 ]<=1;
white[ 1349 ]<=1;
white[ 1350 ]<=1;
white[ 1351 ]<=1;
white[ 1352 ]<=1;
```

```
white[ 1353 ]<=1;
white[ 1354 ]<=1;
white[ 1355 ]<=1;
white[ 1356 ]<=1;
white[ 1357 ]<=1;
white[ 1358 ]<=1;
white[ 1359 ]<=1;
white[ 1360 ]<=1;
white[ 1361 ]<=1;
white[ 1362 ]<=1;
white[ 1363 ]<=1;
white[ 1364 ]<=1;
white[ 1365 ]<=1;
white[ 1366 ]<=1;
white[ 1367 ]<=1;
white[ 1368 ]<=1;
white[ 1369 ]<=1;
white[ 1370 ]<=1;
white[ 1371 ]<=1;
white[ 1372 ]<=1;
white[ 1373 ]<=1;
white[ 1374 ]<=1;
white[ 1375 ]<=0;
white[ 1376 ]<=0;
white[ 1377 ]<=0;
white[ 1378 ]<=0;
white[ 1379 ]<=0;
white[ 1380 ]<=0;
white[ 1381 ]<=0;
white[ 1382 ]<=0;
white[ 1383 ]<=0;
white[ 1384 ]<=0;
white[ 1385 ]<=1;
white[ 1386 ]<=1;
white[ 1387 ]<=1;
white[ 1388 ]<=1;
white[ 1389 ]<=1;
white[ 1390 ]<=1;
white[ 1391 ]<=1;
white[ 1392 ]<=1;
white[ 1393 ]<=1;
white[ 1394 ]<=1;
white[ 1395 ]<=1;
white[ 1396 ]<=1;
white[ 1397 ]<=1;
white[ 1398 ]<=1;
white[ 1399 ]<=1;
white[ 1400 ]<=1;
white[ 1401 ]<=1;
white[ 1402 ]<=1;
white[ 1403 ]<=1;
white[ 1404 ]<=1;
white[ 1405 ]<=1;
white[ 1406 ]<=1;
white[ 1407 ]<=1;
white[ 1408 ]<=1;
white[ 1409 ]<=1;
white[ 1410 ]<=1;
```

```
white[ 1411 ]<=1;
white[ 1412 ]<=1;
white[ 1413 ]<=1;
white[ 1414 ]<=1;
white[ 1415 ]<=1;
white[ 1416 ]<=1;
white[ 1417 ]<=1;
white[ 1418 ]<=1;
white[ 1419 ]<=1;
white[ 1420 ]<=1;
white[ 1421 ]<=1;
white[ 1422 ]<=1;
white[ 1423 ]<=1;
white[ 1424 ]<=1;
white[ 1425 ]<=1;
white[ 1426 ]<=1;
white[ 1427 ]<=1;
white[ 1428 ]<=1;
white[ 1429 ]<=1;
white[ 1430 ]<=1;
white[ 1431 ]<=1;
white[ 1432 ]<=1;
white[ 1433 ]<=1;
white[ 1434 ]<=1;
white[ 1435 ]<=1;
white[ 1436 ]<=0;
white[ 1437 ]<=0;
white[ 1438 ]<=0;
white[ 1439 ]<=0;
white[ 1440 ]<=0;
white[ 1441 ]<=0;
white[ 1442 ]<=0;
white[ 1443 ]<=0;
white[ 1444 ]<=0;
white[ 1445 ]<=1;
white[ 1446 ]<=1;
white[ 1447 ]<=1;
white[ 1448 ]<=1;
white[ 1449 ]<=1;
white[ 1450 ]<=1;
white[ 1451 ]<=1;
white[ 1452 ]<=1;
white[ 1453 ]<=1;
white[ 1454 ]<=1;
white[ 1455 ]<=1;
white[ 1456 ]<=1;
white[ 1457 ]<=1;
white[ 1458 ]<=1;
white[ 1459 ]<=1;
white[ 1460 ]<=1;
white[ 1461 ]<=1;
white[ 1462 ]<=1;
white[ 1463 ]<=1;
white[ 1464 ]<=1;
white[ 1465 ]<=1;
white[ 1466 ]<=1;
white[ 1467 ]<=1;
white[ 1468 ]<=1;
```

```
white[ 1469 ]<=1;
white[ 1470 ]<=1;
white[ 1471 ]<=1;
white[ 1472 ]<=1;
white[ 1473 ]<=1;
white[ 1474 ]<=1;
white[ 1475 ]<=1;
white[ 1476 ]<=1;
white[ 1477 ]<=1;
white[ 1478 ]<=1;
white[ 1479 ]<=1;
white[ 1480 ]<=1;
white[ 1481 ]<=1;
white[ 1482 ]<=1;
white[ 1483 ]<=1;
white[ 1484 ]<=1;
white[ 1485 ]<=1;
white[ 1486 ]<=1;
white[ 1487 ]<=1;
white[ 1488 ]<=1;
white[ 1489 ]<=1;
white[ 1490 ]<=1;
white[ 1491 ]<=1;
white[ 1492 ]<=1;
white[ 1493 ]<=1;
white[ 1494 ]<=1;
white[ 1495 ]<=1;
white[ 1496 ]<=0;
white[ 1497 ]<=0;
white[ 1498 ]<=0;
white[ 1499 ]<=0;
white[ 1500 ]<=0;
white[ 1501 ]<=0;
white[ 1502 ]<=0;
white[ 1503 ]<=0;
white[ 1504 ]<=0;
white[ 1505 ]<=1;
white[ 1506 ]<=1;
white[ 1507 ]<=1;
white[ 1508 ]<=1;
white[ 1509 ]<=1;
white[ 1510 ]<=1;
white[ 1511 ]<=1;
white[ 1512 ]<=1;
white[ 1513 ]<=1;
white[ 1514 ]<=1;
white[ 1515 ]<=1;
white[ 1516 ]<=1;
white[ 1517 ]<=1;
white[ 1518 ]<=1;
white[ 1519 ]<=1;
white[ 1520 ]<=1;
white[ 1521 ]<=1;
white[ 1522 ]<=1;
white[ 1523 ]<=1;
white[ 1524 ]<=1;
white[ 1525 ]<=1;
white[ 1526 ]<=1;
```

```
white[ 1527 ]<=1;
white[ 1528 ]<=1;
white[ 1529 ]<=1;
white[ 1530 ]<=1;
white[ 1531 ]<=1;
white[ 1532 ]<=1;
white[ 1533 ]<=1;
white[ 1534 ]<=1;
white[ 1535 ]<=1;
white[ 1536 ]<=1;
white[ 1537 ]<=1;
white[ 1538 ]<=1;
white[ 1539 ]<=1;
white[ 1540 ]<=1;
white[ 1541 ]<=1;
white[ 1542 ]<=1;
white[ 1543 ]<=1;
white[ 1544 ]<=1;
white[ 1545 ]<=1;
white[ 1546 ]<=1;
white[ 1547 ]<=1;
white[ 1548 ]<=1;
white[ 1549 ]<=1;
white[ 1550 ]<=1;
white[ 1551 ]<=1;
white[ 1552 ]<=1;
white[ 1553 ]<=1;
white[ 1554 ]<=1;
white[ 1555 ]<=1;
white[ 1556 ]<=0;
white[ 1557 ]<=0;
white[ 1558 ]<=0;
white[ 1559 ]<=0;
white[ 1560 ]<=0;
white[ 1561 ]<=0;
white[ 1562 ]<=0;
white[ 1563 ]<=0;
white[ 1564 ]<=0;
white[ 1565 ]<=1;
white[ 1566 ]<=1;
white[ 1567 ]<=1;
white[ 1568 ]<=1;
white[ 1569 ]<=1;
white[ 1570 ]<=1;
white[ 1571 ]<=1;
white[ 1572 ]<=1;
white[ 1573 ]<=1;
white[ 1574 ]<=1;
white[ 1575 ]<=1;
white[ 1576 ]<=1;
white[ 1577 ]<=1;
white[ 1578 ]<=1;
white[ 1579 ]<=1;
white[ 1580 ]<=1;
white[ 1581 ]<=1;
white[ 1582 ]<=1;
white[ 1583 ]<=1;
white[ 1584 ]<=1;
```

```
white[ 1585 ]<=1;
white[ 1586 ]<=1;
white[ 1587 ]<=1;
white[ 1588 ]<=1;
white[ 1589 ]<=1;
white[ 1590 ]<=1;
white[ 1591 ]<=1;
white[ 1592 ]<=1;
white[ 1593 ]<=1;
white[ 1594 ]<=1;
white[ 1595 ]<=1;
white[ 1596 ]<=1;
white[ 1597 ]<=1;
white[ 1598 ]<=1;
white[ 1599 ]<=1;
white[ 1600 ]<=1;
white[ 1601 ]<=1;
white[ 1602 ]<=1;
white[ 1603 ]<=1;
white[ 1604 ]<=1;
white[ 1605 ]<=1;
white[ 1606 ]<=1;
white[ 1607 ]<=1;
white[ 1608 ]<=1;
white[ 1609 ]<=1;
white[ 1610 ]<=1;
white[ 1611 ]<=1;
white[ 1612 ]<=1;
white[ 1613 ]<=1;
white[ 1614 ]<=1;
white[ 1615 ]<=1;
white[ 1616 ]<=0;
white[ 1617 ]<=0;
white[ 1618 ]<=0;
white[ 1619 ]<=0;
white[ 1620 ]<=0;
white[ 1621 ]<=0;
white[ 1622 ]<=0;
white[ 1623 ]<=0;
white[ 1624 ]<=0;
white[ 1625 ]<=1;
white[ 1626 ]<=1;
white[ 1627 ]<=1;
white[ 1628 ]<=1;
white[ 1629 ]<=1;
white[ 1630 ]<=1;
white[ 1631 ]<=1;
white[ 1632 ]<=1;
white[ 1633 ]<=1;
white[ 1634 ]<=1;
white[ 1635 ]<=1;
white[ 1636 ]<=1;
white[ 1637 ]<=1;
white[ 1638 ]<=1;
white[ 1639 ]<=1;
white[ 1640 ]<=1;
white[ 1641 ]<=1;
white[ 1642 ]<=1;
```

```
white[ 1643 ]<=1;
white[ 1644 ]<=1;
white[ 1645 ]<=1;
white[ 1646 ]<=1;
white[ 1647 ]<=1;
white[ 1648 ]<=1;
white[ 1649 ]<=1;
white[ 1650 ]<=1;
white[ 1651 ]<=1;
white[ 1652 ]<=1;
white[ 1653 ]<=1;
white[ 1654 ]<=1;
white[ 1655 ]<=1;
white[ 1656 ]<=1;
white[ 1657 ]<=1;
white[ 1658 ]<=1;
white[ 1659 ]<=1;
white[ 1660 ]<=1;
white[ 1661 ]<=1;
white[ 1662 ]<=1;
white[ 1663 ]<=1;
white[ 1664 ]<=1;
white[ 1665 ]<=1;
white[ 1666 ]<=1;
white[ 1667 ]<=1;
white[ 1668 ]<=1;
white[ 1669 ]<=1;
white[ 1670 ]<=1;
white[ 1671 ]<=1;
white[ 1672 ]<=1;
white[ 1673 ]<=1;
white[ 1674 ]<=1;
white[ 1675 ]<=1;
white[ 1676 ]<=0;
white[ 1677 ]<=0;
white[ 1678 ]<=0;
white[ 1679 ]<=0;
white[ 1680 ]<=0;
white[ 1681 ]<=0;
white[ 1682 ]<=0;
white[ 1683 ]<=0;
white[ 1684 ]<=0;
white[ 1685 ]<=0;
white[ 1686 ]<=1;
white[ 1687 ]<=1;
white[ 1688 ]<=1;
white[ 1689 ]<=1;
white[ 1690 ]<=1;
white[ 1691 ]<=1;
white[ 1692 ]<=1;
white[ 1693 ]<=1;
white[ 1694 ]<=1;
white[ 1695 ]<=1;
white[ 1696 ]<=1;
white[ 1697 ]<=1;
white[ 1698 ]<=1;
white[ 1699 ]<=1;
white[ 1700 ]<=1;
```

```
white[ 1701 ]<=1;
white[ 1702 ]<=1;
white[ 1703 ]<=1;
white[ 1704 ]<=1;
white[ 1705 ]<=1;
white[ 1706 ]<=1;
white[ 1707 ]<=1;
white[ 1708 ]<=1;
white[ 1709 ]<=1;
white[ 1710 ]<=1;
white[ 1711 ]<=1;
white[ 1712 ]<=1;
white[ 1713 ]<=1;
white[ 1714 ]<=1;
white[ 1715 ]<=1;
white[ 1716 ]<=1;
white[ 1717 ]<=1;
white[ 1718 ]<=1;
white[ 1719 ]<=1;
white[ 1720 ]<=1;
white[ 1721 ]<=1;
white[ 1722 ]<=1;
white[ 1723 ]<=1;
white[ 1724 ]<=1;
white[ 1725 ]<=1;
white[ 1726 ]<=1;
white[ 1727 ]<=1;
white[ 1728 ]<=1;
white[ 1729 ]<=1;
white[ 1730 ]<=1;
white[ 1731 ]<=1;
white[ 1732 ]<=1;
white[ 1733 ]<=1;
white[ 1734 ]<=1;
white[ 1735 ]<=0;
white[ 1736 ]<=0;
white[ 1737 ]<=0;
white[ 1738 ]<=0;
white[ 1739 ]<=0;
white[ 1740 ]<=0;
white[ 1741 ]<=0;
white[ 1742 ]<=0;
white[ 1743 ]<=0;
white[ 1744 ]<=0;
white[ 1745 ]<=0;
white[ 1746 ]<=0;
white[ 1747 ]<=0;
white[ 1748 ]<=0;
white[ 1749 ]<=1;
white[ 1750 ]<=1;
white[ 1751 ]<=1;
white[ 1752 ]<=1;
white[ 1753 ]<=1;
white[ 1754 ]<=1;
white[ 1755 ]<=1;
white[ 1756 ]<=1;
white[ 1757 ]<=1;
white[ 1758 ]<=1;
```

```
white[ 1759 ]<=1;
white[ 1760 ]<=1;
white[ 1761 ]<=1;
white[ 1762 ]<=1;
white[ 1763 ]<=1;
white[ 1764 ]<=1;
white[ 1765 ]<=1;
white[ 1766 ]<=1;
white[ 1767 ]<=1;
white[ 1768 ]<=1;
white[ 1769 ]<=1;
white[ 1770 ]<=1;
white[ 1771 ]<=1;
white[ 1772 ]<=1;
white[ 1773 ]<=1;
white[ 1774 ]<=1;
white[ 1775 ]<=1;
white[ 1776 ]<=1;
white[ 1777 ]<=1;
white[ 1778 ]<=1;
white[ 1779 ]<=1;
white[ 1780 ]<=1;
white[ 1781 ]<=1;
white[ 1782 ]<=1;
white[ 1783 ]<=1;
white[ 1784 ]<=1;
white[ 1785 ]<=1;
white[ 1786 ]<=1;
white[ 1787 ]<=1;
white[ 1788 ]<=1;
white[ 1789 ]<=1;
white[ 1790 ]<=1;
white[ 1791 ]<=1;
white[ 1792 ]<=1;
white[ 1793 ]<=0;
white[ 1794 ]<=0;
white[ 1795 ]<=0;
white[ 1796 ]<=0;
white[ 1797 ]<=0;
white[ 1798 ]<=0;
white[ 1799 ]<=0;
white[ 1800 ]<=0;
white[ 1801 ]<=0;
white[ 1802 ]<=0;
white[ 1803 ]<=0;
white[ 1804 ]<=0;
white[ 1805 ]<=0;
white[ 1806 ]<=0;
white[ 1807 ]<=0;
white[ 1808 ]<=0;
white[ 1809 ]<=1;
white[ 1810 ]<=1;
white[ 1811 ]<=1;
white[ 1812 ]<=1;
white[ 1813 ]<=1;
white[ 1814 ]<=1;
white[ 1815 ]<=1;
white[ 1816 ]<=1;
```

```
white[ 1817 ]<=1;
white[ 1818 ]<=1;
white[ 1819 ]<=1;
white[ 1820 ]<=1;
white[ 1821 ]<=1;
white[ 1822 ]<=1;
white[ 1823 ]<=1;
white[ 1824 ]<=1;
white[ 1825 ]<=1;
white[ 1826 ]<=1;
white[ 1827 ]<=1;
white[ 1828 ]<=1;
white[ 1829 ]<=1;
white[ 1830 ]<=1;
white[ 1831 ]<=1;
white[ 1832 ]<=1;
white[ 1833 ]<=1;
white[ 1834 ]<=1;
white[ 1835 ]<=1;
white[ 1836 ]<=1;
white[ 1837 ]<=1;
white[ 1838 ]<=1;
white[ 1839 ]<=1;
white[ 1840 ]<=1;
white[ 1841 ]<=1;
white[ 1842 ]<=1;
white[ 1843 ]<=1;
white[ 1844 ]<=1;
white[ 1845 ]<=1;
white[ 1846 ]<=1;
white[ 1847 ]<=1;
white[ 1848 ]<=1;
white[ 1849 ]<=1;
white[ 1850 ]<=1;
white[ 1851 ]<=1;
white[ 1852 ]<=1;
white[ 1853 ]<=0;
white[ 1854 ]<=0;
white[ 1855 ]<=0;
white[ 1856 ]<=0;
white[ 1857 ]<=0;
white[ 1858 ]<=0;
white[ 1859 ]<=0;
white[ 1860 ]<=0;
white[ 1861 ]<=0;
white[ 1862 ]<=0;
white[ 1863 ]<=0;
white[ 1864 ]<=0;
white[ 1865 ]<=0;
white[ 1866 ]<=0;
white[ 1867 ]<=0;
white[ 1868 ]<=0;
white[ 1869 ]<=1;
white[ 1870 ]<=1;
white[ 1871 ]<=1;
white[ 1872 ]<=1;
white[ 1873 ]<=1;
white[ 1874 ]<=1;
```

```
white[ 1875 ]<=1;
white[ 1876 ]<=1;
white[ 1877 ]<=1;
white[ 1878 ]<=1;
white[ 1879 ]<=1;
white[ 1880 ]<=1;
white[ 1881 ]<=1;
white[ 1882 ]<=1;
white[ 1883 ]<=1;
white[ 1884 ]<=1;
white[ 1885 ]<=1;
white[ 1886 ]<=1;
white[ 1887 ]<=1;
white[ 1888 ]<=1;
white[ 1889 ]<=1;
white[ 1890 ]<=1;
white[ 1891 ]<=1;
white[ 1892 ]<=1;
white[ 1893 ]<=1;
white[ 1894 ]<=1;
white[ 1895 ]<=1;
white[ 1896 ]<=1;
white[ 1897 ]<=1;
white[ 1898 ]<=1;
white[ 1899 ]<=1;
white[ 1900 ]<=1;
white[ 1901 ]<=1;
white[ 1902 ]<=1;
white[ 1903 ]<=1;
white[ 1904 ]<=1;
white[ 1905 ]<=1;
white[ 1906 ]<=1;
white[ 1907 ]<=1;
white[ 1908 ]<=1;
white[ 1909 ]<=1;
white[ 1910 ]<=1;
white[ 1911 ]<=1;
white[ 1912 ]<=1;
white[ 1913 ]<=0;
white[ 1914 ]<=0;
white[ 1915 ]<=0;
white[ 1916 ]<=0;
white[ 1917 ]<=0;
white[ 1918 ]<=0;
white[ 1919 ]<=0;
white[ 1920 ]<=0;
white[ 1921 ]<=0;
white[ 1922 ]<=0;
white[ 1923 ]<=0;
white[ 1924 ]<=0;
white[ 1925 ]<=0;
white[ 1926 ]<=0;
white[ 1927 ]<=0;
white[ 1928 ]<=1;
white[ 1929 ]<=1;
white[ 1930 ]<=1;
white[ 1931 ]<=1;
white[ 1932 ]<=1;
```

```
white[ 1933 ]<=1;
white[ 1934 ]<=1;
white[ 1935 ]<=1;
white[ 1936 ]<=1;
white[ 1937 ]<=1;
white[ 1938 ]<=1;
white[ 1939 ]<=1;
white[ 1940 ]<=1;
white[ 1941 ]<=1;
white[ 1942 ]<=1;
white[ 1943 ]<=1;
white[ 1944 ]<=1;
white[ 1945 ]<=1;
white[ 1946 ]<=1;
white[ 1947 ]<=1;
white[ 1948 ]<=1;
white[ 1949 ]<=1;
white[ 1950 ]<=1;
white[ 1951 ]<=1;
white[ 1952 ]<=1;
white[ 1953 ]<=1;
white[ 1954 ]<=1;
white[ 1955 ]<=1;
white[ 1956 ]<=1;
white[ 1957 ]<=1;
white[ 1958 ]<=1;
white[ 1959 ]<=1;
white[ 1960 ]<=1;
white[ 1961 ]<=1;
white[ 1962 ]<=1;
white[ 1963 ]<=1;
white[ 1964 ]<=1;
white[ 1965 ]<=1;
white[ 1966 ]<=1;
white[ 1967 ]<=1;
white[ 1968 ]<=1;
white[ 1969 ]<=1;
white[ 1970 ]<=1;
white[ 1971 ]<=1;
white[ 1972 ]<=1;
white[ 1973 ]<=0;
white[ 1974 ]<=0;
white[ 1975 ]<=0;
white[ 1976 ]<=0;
white[ 1977 ]<=0;
white[ 1978 ]<=0;
white[ 1979 ]<=0;
white[ 1980 ]<=0;
white[ 1981 ]<=0;
white[ 1982 ]<=0;
white[ 1983 ]<=0;
white[ 1984 ]<=0;
white[ 1985 ]<=0;
white[ 1986 ]<=0;
white[ 1987 ]<=0;
white[ 1988 ]<=1;
white[ 1989 ]<=1;
white[ 1990 ]<=1;
```

```
white[ 1991 ]<=1;
white[ 1992 ]<=1;
white[ 1993 ]<=1;
white[ 1994 ]<=1;
white[ 1995 ]<=1;
white[ 1996 ]<=1;
white[ 1997 ]<=1;
white[ 1998 ]<=1;
white[ 1999 ]<=1;
white[ 2000 ]<=1;
white[ 2001 ]<=1;
white[ 2002 ]<=1;
white[ 2003 ]<=1;
white[ 2004 ]<=1;
white[ 2005 ]<=1;
white[ 2006 ]<=1;
white[ 2007 ]<=1;
white[ 2008 ]<=1;
white[ 2009 ]<=1;
white[ 2010 ]<=1;
white[ 2011 ]<=1;
white[ 2012 ]<=1;
white[ 2013 ]<=1;
white[ 2014 ]<=1;
white[ 2015 ]<=1;
white[ 2016 ]<=1;
white[ 2017 ]<=1;
white[ 2018 ]<=1;
white[ 2019 ]<=1;
white[ 2020 ]<=1;
white[ 2021 ]<=1;
white[ 2022 ]<=1;
white[ 2023 ]<=1;
white[ 2024 ]<=1;
white[ 2025 ]<=1;
white[ 2026 ]<=1;
white[ 2027 ]<=1;
white[ 2028 ]<=1;
white[ 2029 ]<=1;
white[ 2030 ]<=1;
white[ 2031 ]<=1;
white[ 2032 ]<=1;
white[ 2033 ]<=0;
white[ 2034 ]<=0;
white[ 2035 ]<=0;
white[ 2036 ]<=0;
white[ 2037 ]<=0;
white[ 2038 ]<=0;
white[ 2039 ]<=0;
white[ 2040 ]<=0;
white[ 2041 ]<=0;
white[ 2042 ]<=0;
white[ 2043 ]<=0;
white[ 2044 ]<=0;
white[ 2045 ]<=0;
white[ 2046 ]<=0;
white[ 2047 ]<=0;
white[ 2048 ]<=1;
```

```
white[ 2049 ]<=1;
white[ 2050 ]<=1;
white[ 2051 ]<=1;
white[ 2052 ]<=1;
white[ 2053 ]<=1;
white[ 2054 ]<=1;
white[ 2055 ]<=1;
white[ 2056 ]<=1;
white[ 2057 ]<=1;
white[ 2058 ]<=1;
white[ 2059 ]<=1;
white[ 2060 ]<=1;
white[ 2061 ]<=1;
white[ 2062 ]<=1;
white[ 2063 ]<=1;
white[ 2064 ]<=1;
white[ 2065 ]<=1;
white[ 2066 ]<=1;
white[ 2067 ]<=1;
white[ 2068 ]<=1;
white[ 2069 ]<=1;
white[ 2070 ]<=1;
white[ 2071 ]<=1;
white[ 2072 ]<=1;
white[ 2073 ]<=1;
white[ 2074 ]<=1;
white[ 2075 ]<=1;
white[ 2076 ]<=1;
white[ 2077 ]<=1;
white[ 2078 ]<=1;
white[ 2079 ]<=1;
white[ 2080 ]<=1;
white[ 2081 ]<=1;
white[ 2082 ]<=1;
white[ 2083 ]<=1;
white[ 2084 ]<=1;
white[ 2085 ]<=1;
white[ 2086 ]<=1;
white[ 2087 ]<=1;
white[ 2088 ]<=1;
white[ 2089 ]<=1;
white[ 2090 ]<=1;
white[ 2091 ]<=1;
white[ 2092 ]<=1;
white[ 2093 ]<=0;
white[ 2094 ]<=0;
white[ 2095 ]<=0;
white[ 2096 ]<=0;
white[ 2097 ]<=0;
white[ 2098 ]<=0;
white[ 2099 ]<=0;
white[ 2100 ]<=0;
white[ 2101 ]<=0;
white[ 2102 ]<=0;
white[ 2103 ]<=0;
white[ 2104 ]<=0;
white[ 2105 ]<=0;
white[ 2106 ]<=0;
```

```
white[ 2107 ]<=0;
white[ 2108 ]<=1;
white[ 2109 ]<=1;
white[ 2110 ]<=1;
white[ 2111 ]<=1;
white[ 2112 ]<=1;
white[ 2113 ]<=1;
white[ 2114 ]<=1;
white[ 2115 ]<=1;
white[ 2116 ]<=1;
white[ 2117 ]<=1;
white[ 2118 ]<=1;
white[ 2119 ]<=1;
white[ 2120 ]<=1;
white[ 2121 ]<=1;
white[ 2122 ]<=1;
white[ 2123 ]<=1;
white[ 2124 ]<=1;
white[ 2125 ]<=1;
white[ 2126 ]<=1;
white[ 2127 ]<=1;
white[ 2128 ]<=1;
white[ 2129 ]<=1;
white[ 2130 ]<=1;
white[ 2131 ]<=1;
white[ 2132 ]<=1;
white[ 2133 ]<=1;
white[ 2134 ]<=1;
white[ 2135 ]<=1;
white[ 2136 ]<=1;
white[ 2137 ]<=1;
white[ 2138 ]<=1;
white[ 2139 ]<=1;
white[ 2140 ]<=1;
white[ 2141 ]<=1;
white[ 2142 ]<=1;
white[ 2143 ]<=1;
white[ 2144 ]<=1;
white[ 2145 ]<=1;
white[ 2146 ]<=1;
white[ 2147 ]<=1;
white[ 2148 ]<=1;
white[ 2149 ]<=1;
white[ 2150 ]<=1;
white[ 2151 ]<=1;
white[ 2152 ]<=1;
white[ 2153 ]<=1;
white[ 2154 ]<=0;
white[ 2155 ]<=0;
white[ 2156 ]<=0;
white[ 2157 ]<=0;
white[ 2158 ]<=0;
white[ 2159 ]<=0;
white[ 2160 ]<=0;
white[ 2161 ]<=0;
white[ 2162 ]<=0;
white[ 2163 ]<=0;
white[ 2164 ]<=0;
```

```
white[ 2165 ]<=0;
white[ 2166 ]<=0;
white[ 2167 ]<=0;
white[ 2168 ]<=1;
white[ 2169 ]<=1;
white[ 2170 ]<=1;
white[ 2171 ]<=1;
white[ 2172 ]<=1;
white[ 2173 ]<=1;
white[ 2174 ]<=1;
white[ 2175 ]<=1;
white[ 2176 ]<=1;
white[ 2177 ]<=1;
white[ 2178 ]<=1;
white[ 2179 ]<=1;
white[ 2180 ]<=1;
white[ 2181 ]<=1;
white[ 2182 ]<=1;
white[ 2183 ]<=1;
white[ 2184 ]<=1;
white[ 2185 ]<=1;
white[ 2186 ]<=1;
white[ 2187 ]<=1;
white[ 2188 ]<=1;
white[ 2189 ]<=1;
white[ 2190 ]<=1;
white[ 2191 ]<=1;
white[ 2192 ]<=1;
white[ 2193 ]<=1;
white[ 2194 ]<=1;
white[ 2195 ]<=1;
white[ 2196 ]<=1;
white[ 2197 ]<=1;
white[ 2198 ]<=1;
white[ 2199 ]<=1;
white[ 2200 ]<=1;
white[ 2201 ]<=1;
white[ 2202 ]<=1;
white[ 2203 ]<=1;
white[ 2204 ]<=1;
white[ 2205 ]<=1;
white[ 2206 ]<=1;
white[ 2207 ]<=1;
white[ 2208 ]<=1;
white[ 2209 ]<=1;
white[ 2210 ]<=1;
white[ 2211 ]<=1;
white[ 2212 ]<=1;
white[ 2213 ]<=1;
white[ 2214 ]<=0;
white[ 2215 ]<=0;
white[ 2216 ]<=0;
white[ 2217 ]<=0;
white[ 2218 ]<=0;
white[ 2219 ]<=0;
white[ 2220 ]<=0;
white[ 2221 ]<=0;
white[ 2222 ]<=0;
```

```
white[ 2223 ]<=0;
white[ 2224 ]<=0;
white[ 2225 ]<=0;
white[ 2226 ]<=0;
white[ 2227 ]<=0;
white[ 2228 ]<=1;
white[ 2229 ]<=1;
white[ 2230 ]<=1;
white[ 2231 ]<=1;
white[ 2232 ]<=1;
white[ 2233 ]<=1;
white[ 2234 ]<=1;
white[ 2235 ]<=1;
white[ 2236 ]<=1;
white[ 2237 ]<=1;
white[ 2238 ]<=1;
white[ 2239 ]<=1;
white[ 2240 ]<=1;
white[ 2241 ]<=1;
white[ 2242 ]<=1;
white[ 2243 ]<=1;
white[ 2244 ]<=1;
white[ 2245 ]<=1;
white[ 2246 ]<=1;
white[ 2247 ]<=1;
white[ 2248 ]<=1;
white[ 2249 ]<=1;
white[ 2250 ]<=1;
white[ 2251 ]<=1;
white[ 2252 ]<=1;
white[ 2253 ]<=1;
white[ 2254 ]<=1;
white[ 2255 ]<=1;
white[ 2256 ]<=1;
white[ 2257 ]<=1;
white[ 2258 ]<=1;
white[ 2259 ]<=1;
white[ 2260 ]<=1;
white[ 2261 ]<=1;
white[ 2262 ]<=1;
white[ 2263 ]<=1;
white[ 2264 ]<=1;
white[ 2265 ]<=1;
white[ 2266 ]<=1;
white[ 2267 ]<=1;
white[ 2268 ]<=1;
white[ 2269 ]<=1;
white[ 2270 ]<=1;
white[ 2271 ]<=1;
white[ 2272 ]<=1;
white[ 2273 ]<=1;
white[ 2274 ]<=0;
white[ 2275 ]<=0;
white[ 2276 ]<=0;
white[ 2277 ]<=0;
white[ 2278 ]<=0;
white[ 2279 ]<=0;
white[ 2280 ]<=0;
```

```
white[ 2281 ]<=0;
white[ 2282 ]<=0;
white[ 2283 ]<=0;
white[ 2284 ]<=0;
white[ 2285 ]<=0;
white[ 2286 ]<=0;
white[ 2287 ]<=0;
white[ 2288 ]<=1;
white[ 2289 ]<=1;
white[ 2290 ]<=1;
white[ 2291 ]<=1;
white[ 2292 ]<=1;
white[ 2293 ]<=1;
white[ 2294 ]<=1;
white[ 2295 ]<=1;
white[ 2296 ]<=1;
white[ 2297 ]<=1;
white[ 2298 ]<=1;
white[ 2299 ]<=1;
white[ 2300 ]<=1;
white[ 2301 ]<=1;
white[ 2302 ]<=1;
white[ 2303 ]<=1;
white[ 2304 ]<=1;
white[ 2305 ]<=1;
white[ 2306 ]<=1;
white[ 2307 ]<=1;
white[ 2308 ]<=1;
white[ 2309 ]<=1;
white[ 2310 ]<=0;
white[ 2311 ]<=1;
white[ 2312 ]<=1;
white[ 2313 ]<=1;
white[ 2314 ]<=1;
white[ 2315 ]<=1;
white[ 2316 ]<=1;
white[ 2317 ]<=1;
white[ 2318 ]<=1;
white[ 2319 ]<=1;
white[ 2320 ]<=1;
white[ 2321 ]<=1;
white[ 2322 ]<=1;
white[ 2323 ]<=1;
white[ 2324 ]<=1;
white[ 2325 ]<=1;
white[ 2326 ]<=1;
white[ 2327 ]<=1;
white[ 2328 ]<=1;
white[ 2329 ]<=1;
white[ 2330 ]<=1;
white[ 2331 ]<=1;
white[ 2332 ]<=1;
white[ 2333 ]<=1;
white[ 2334 ]<=0;
white[ 2335 ]<=0;
white[ 2336 ]<=0;
white[ 2337 ]<=0;
white[ 2338 ]<=0;
```

```
white[ 2339 ]<=0;
white[ 2340 ]<=0;
white[ 2341 ]<=0;
white[ 2342 ]<=0;
white[ 2343 ]<=0;
white[ 2344 ]<=0;
white[ 2345 ]<=0;
white[ 2346 ]<=0;
white[ 2347 ]<=1;
white[ 2348 ]<=1;
white[ 2349 ]<=1;
white[ 2350 ]<=1;
white[ 2351 ]<=1;
white[ 2352 ]<=1;
white[ 2353 ]<=1;
white[ 2354 ]<=1;
white[ 2355 ]<=1;
white[ 2356 ]<=1;
white[ 2357 ]<=1;
white[ 2358 ]<=1;
white[ 2359 ]<=1;
white[ 2360 ]<=1;
white[ 2361 ]<=1;
white[ 2362 ]<=1;
white[ 2363 ]<=1;
white[ 2364 ]<=1;
white[ 2365 ]<=1;
white[ 2366 ]<=1;
white[ 2367 ]<=1;
white[ 2368 ]<=1;
white[ 2369 ]<=1;
white[ 2370 ]<=1;
white[ 2371 ]<=1;
white[ 2372 ]<=1;
white[ 2373 ]<=1;
white[ 2374 ]<=1;
white[ 2375 ]<=1;
white[ 2376 ]<=1;
white[ 2377 ]<=1;
white[ 2378 ]<=1;
white[ 2379 ]<=1;
white[ 2380 ]<=1;
white[ 2381 ]<=1;
white[ 2382 ]<=1;
white[ 2383 ]<=1;
white[ 2384 ]<=1;
white[ 2385 ]<=1;
white[ 2386 ]<=1;
white[ 2387 ]<=1;
white[ 2388 ]<=1;
white[ 2389 ]<=1;
white[ 2390 ]<=1;
white[ 2391 ]<=1;
white[ 2392 ]<=1;
white[ 2393 ]<=1;
white[ 2394 ]<=0;
white[ 2395 ]<=0;
white[ 2396 ]<=0;
```

```
white[ 2397 ]<=0;
white[ 2398 ]<=0;
white[ 2399 ]<=0;
white[ 2400 ]<=0;
white[ 2401 ]<=0;
white[ 2402 ]<=0;
white[ 2403 ]<=0;
white[ 2404 ]<=0;
white[ 2405 ]<=0;
white[ 2406 ]<=0;
white[ 2407 ]<=1;
white[ 2408 ]<=1;
white[ 2409 ]<=1;
white[ 2410 ]<=1;
white[ 2411 ]<=1;
white[ 2412 ]<=1;
white[ 2413 ]<=1;
white[ 2414 ]<=1;
white[ 2415 ]<=1;
white[ 2416 ]<=1;
white[ 2417 ]<=1;
white[ 2418 ]<=1;
white[ 2419 ]<=1;
white[ 2420 ]<=1;
white[ 2421 ]<=1;
white[ 2422 ]<=1;
white[ 2423 ]<=1;
white[ 2424 ]<=0;
white[ 2425 ]<=0;
white[ 2426 ]<=1;
white[ 2427 ]<=1;
white[ 2428 ]<=1;
white[ 2429 ]<=1;
white[ 2430 ]<=1;
white[ 2431 ]<=1;
white[ 2432 ]<=1;
white[ 2433 ]<=1;
white[ 2434 ]<=1;
white[ 2435 ]<=0;
white[ 2436 ]<=0;
white[ 2437 ]<=1;
white[ 2438 ]<=1;
white[ 2439 ]<=1;
white[ 2440 ]<=1;
white[ 2441 ]<=1;
white[ 2442 ]<=1;
white[ 2443 ]<=1;
white[ 2444 ]<=1;
white[ 2445 ]<=1;
white[ 2446 ]<=1;
white[ 2447 ]<=1;
white[ 2448 ]<=1;
white[ 2449 ]<=1;
white[ 2450 ]<=1;
white[ 2451 ]<=1;
white[ 2452 ]<=1;
white[ 2453 ]<=1;
white[ 2454 ]<=0;
```

```
white[ 2455 ]<=0;
white[ 2456 ]<=0;
white[ 2457 ]<=0;
white[ 2458 ]<=0;
white[ 2459 ]<=0;
white[ 2460 ]<=0;
white[ 2461 ]<=0;
white[ 2462 ]<=0;
white[ 2463 ]<=0;
white[ 2464 ]<=0;
white[ 2465 ]<=0;
white[ 2466 ]<=0;
white[ 2467 ]<=1;
white[ 2468 ]<=1;
white[ 2469 ]<=1;
white[ 2470 ]<=1;
white[ 2471 ]<=1;
white[ 2472 ]<=1;
white[ 2473 ]<=1;
white[ 2474 ]<=1;
white[ 2475 ]<=1;
white[ 2476 ]<=1;
white[ 2477 ]<=1;
white[ 2478 ]<=1;
white[ 2479 ]<=1;
white[ 2480 ]<=1;
white[ 2481 ]<=1;
white[ 2482 ]<=1;
white[ 2483 ]<=0;
white[ 2484 ]<=0;
white[ 2485 ]<=0;
white[ 2486 ]<=1;
white[ 2487 ]<=1;
white[ 2488 ]<=1;
white[ 2489 ]<=1;
white[ 2490 ]<=1;
white[ 2491 ]<=1;
white[ 2492 ]<=1;
white[ 2493 ]<=1;
white[ 2494 ]<=1;
white[ 2495 ]<=0;
white[ 2496 ]<=0;
white[ 2497 ]<=0;
white[ 2498 ]<=1;
white[ 2499 ]<=1;
white[ 2500 ]<=1;
white[ 2501 ]<=1;
white[ 2502 ]<=1;
white[ 2503 ]<=1;
white[ 2504 ]<=1;
white[ 2505 ]<=1;
white[ 2506 ]<=1;
white[ 2507 ]<=1;
white[ 2508 ]<=1;
white[ 2509 ]<=1;
white[ 2510 ]<=1;
white[ 2511 ]<=1;
white[ 2512 ]<=1;
```

```
white[ 2513 ]<=1;
white[ 2514 ]<=0;
white[ 2515 ]<=0;
white[ 2516 ]<=0;
white[ 2517 ]<=0;
white[ 2518 ]<=0;
white[ 2519 ]<=0;
white[ 2520 ]<=0;
white[ 2521 ]<=0;
white[ 2522 ]<=0;
white[ 2523 ]<=0;
white[ 2524 ]<=0;
white[ 2525 ]<=0;
white[ 2526 ]<=0;
white[ 2527 ]<=1;
white[ 2528 ]<=1;
white[ 2529 ]<=1;
white[ 2530 ]<=1;
white[ 2531 ]<=1;
white[ 2532 ]<=1;
white[ 2533 ]<=1;
white[ 2534 ]<=1;
white[ 2535 ]<=1;
white[ 2536 ]<=1;
white[ 2537 ]<=1;
white[ 2538 ]<=1;
white[ 2539 ]<=0;
white[ 2540 ]<=0;
white[ 2541 ]<=1;
white[ 2542 ]<=1;
white[ 2543 ]<=1;
white[ 2544 ]<=1;
white[ 2545 ]<=1;
white[ 2546 ]<=0;
white[ 2547 ]<=0;
white[ 2548 ]<=1;
white[ 2549 ]<=1;
white[ 2550 ]<=1;
white[ 2551 ]<=1;
white[ 2552 ]<=1;
white[ 2553 ]<=1;
white[ 2554 ]<=0;
white[ 2555 ]<=1;
white[ 2556 ]<=1;
white[ 2557 ]<=1;
white[ 2558 ]<=1;
white[ 2559 ]<=1;
white[ 2560 ]<=0;
white[ 2561 ]<=0;
white[ 2562 ]<=1;
white[ 2563 ]<=1;
white[ 2564 ]<=1;
white[ 2565 ]<=1;
white[ 2566 ]<=1;
white[ 2567 ]<=1;
white[ 2568 ]<=1;
white[ 2569 ]<=1;
white[ 2570 ]<=1;
```

```
white[ 2571 ]<=1;
white[ 2572 ]<=1;
white[ 2573 ]<=1;
white[ 2574 ]<=0;
white[ 2575 ]<=0;
white[ 2576 ]<=0;
white[ 2577 ]<=0;
white[ 2578 ]<=0;
white[ 2579 ]<=0;
white[ 2580 ]<=0;
white[ 2581 ]<=0;
white[ 2582 ]<=0;
white[ 2583 ]<=0;
white[ 2584 ]<=0;
white[ 2585 ]<=0;
white[ 2586 ]<=0;
white[ 2587 ]<=1;
white[ 2588 ]<=1;
white[ 2589 ]<=1;
white[ 2590 ]<=1;
white[ 2591 ]<=1;
white[ 2592 ]<=1;
white[ 2593 ]<=1;
white[ 2594 ]<=1;
white[ 2595 ]<=1;
white[ 2596 ]<=1;
white[ 2597 ]<=1;
white[ 2598 ]<=1;
white[ 2599 ]<=1;
white[ 2600 ]<=1;
white[ 2601 ]<=1;
white[ 2602 ]<=1;
white[ 2603 ]<=1;
white[ 2604 ]<=1;
white[ 2605 ]<=1;
white[ 2606 ]<=1;
white[ 2607 ]<=1;
white[ 2608 ]<=0;
white[ 2609 ]<=1;
white[ 2610 ]<=1;
white[ 2611 ]<=1;
white[ 2612 ]<=0;
white[ 2613 ]<=1;
white[ 2614 ]<=1;
white[ 2615 ]<=1;
white[ 2616 ]<=1;
white[ 2617 ]<=1;
white[ 2618 ]<=1;
white[ 2619 ]<=1;
white[ 2620 ]<=1;
white[ 2621 ]<=1;
white[ 2622 ]<=1;
white[ 2623 ]<=1;
white[ 2624 ]<=1;
white[ 2625 ]<=1;
white[ 2626 ]<=1;
white[ 2627 ]<=1;
white[ 2628 ]<=1;
```

```
white[ 2629 ]<=1;
white[ 2630 ]<=1;
white[ 2631 ]<=1;
white[ 2632 ]<=1;
white[ 2633 ]<=1;
white[ 2634 ]<=1;
white[ 2635 ]<=0;
white[ 2636 ]<=0;
white[ 2637 ]<=0;
white[ 2638 ]<=0;
white[ 2639 ]<=0;
white[ 2640 ]<=0;
white[ 2641 ]<=0;
white[ 2642 ]<=0;
white[ 2643 ]<=0;
white[ 2644 ]<=0;
white[ 2645 ]<=0;
white[ 2646 ]<=0;
white[ 2647 ]<=1;
white[ 2648 ]<=1;
white[ 2649 ]<=1;
white[ 2650 ]<=1;
white[ 2651 ]<=1;
white[ 2652 ]<=1;
white[ 2653 ]<=1;
white[ 2654 ]<=1;
white[ 2655 ]<=1;
white[ 2656 ]<=1;
white[ 2657 ]<=1;
white[ 2658 ]<=1;
white[ 2659 ]<=1;
white[ 2660 ]<=1;
white[ 2661 ]<=1;
white[ 2662 ]<=1;
white[ 2663 ]<=1;
white[ 2664 ]<=1;
white[ 2665 ]<=1;
white[ 2666 ]<=1;
white[ 2667 ]<=1;
white[ 2668 ]<=0;
white[ 2669 ]<=1;
white[ 2670 ]<=1;
white[ 2671 ]<=1;
white[ 2672 ]<=0;
white[ 2673 ]<=1;
white[ 2674 ]<=1;
white[ 2675 ]<=1;
white[ 2676 ]<=1;
white[ 2677 ]<=1;
white[ 2678 ]<=1;
white[ 2679 ]<=1;
white[ 2680 ]<=1;
white[ 2681 ]<=1;
white[ 2682 ]<=1;
white[ 2683 ]<=1;
white[ 2684 ]<=1;
white[ 2685 ]<=1;
white[ 2686 ]<=1;
```

```
white[ 2687 ]<=1;
white[ 2688 ]<=1;
white[ 2689 ]<=1;
white[ 2690 ]<=1;
white[ 2691 ]<=1;
white[ 2692 ]<=1;
white[ 2693 ]<=1;
white[ 2694 ]<=1;
white[ 2695 ]<=0;
white[ 2696 ]<=0;
white[ 2697 ]<=0;
white[ 2698 ]<=0;
white[ 2699 ]<=0;
white[ 2700 ]<=0;
white[ 2701 ]<=0;
white[ 2702 ]<=0;
white[ 2703 ]<=0;
white[ 2704 ]<=0;
white[ 2705 ]<=0;
white[ 2706 ]<=0;
white[ 2707 ]<=1;
white[ 2708 ]<=1;
white[ 2709 ]<=1;
white[ 2710 ]<=1;
white[ 2711 ]<=1;
white[ 2712 ]<=1;
white[ 2713 ]<=1;
white[ 2714 ]<=1;
white[ 2715 ]<=1;
white[ 2716 ]<=0;
white[ 2717 ]<=1;
white[ 2718 ]<=1;
white[ 2719 ]<=1;
white[ 2720 ]<=1;
white[ 2721 ]<=1;
white[ 2722 ]<=1;
white[ 2723 ]<=1;
white[ 2724 ]<=1;
white[ 2725 ]<=0;
white[ 2726 ]<=0;
white[ 2727 ]<=1;
white[ 2728 ]<=1;
white[ 2729 ]<=1;
white[ 2730 ]<=1;
white[ 2731 ]<=1;
white[ 2732 ]<=1;
white[ 2733 ]<=1;
white[ 2734 ]<=1;
white[ 2735 ]<=0;
white[ 2736 ]<=1;
white[ 2737 ]<=1;
white[ 2738 ]<=1;
white[ 2739 ]<=1;
white[ 2740 ]<=1;
white[ 2741 ]<=1;
white[ 2742 ]<=1;
white[ 2743 ]<=1;
white[ 2744 ]<=1;
```

```
white[ 2745 ]<=1;
white[ 2746 ]<=1;
white[ 2747 ]<=1;
white[ 2748 ]<=1;
white[ 2749 ]<=1;
white[ 2750 ]<=1;
white[ 2751 ]<=1;
white[ 2752 ]<=1;
white[ 2753 ]<=1;
white[ 2754 ]<=1;
white[ 2755 ]<=0;
white[ 2756 ]<=0;
white[ 2757 ]<=0;
white[ 2758 ]<=0;
white[ 2759 ]<=0;
white[ 2760 ]<=0;
white[ 2761 ]<=0;
white[ 2762 ]<=0;
white[ 2763 ]<=0;
white[ 2764 ]<=0;
white[ 2765 ]<=0;
white[ 2766 ]<=0;
white[ 2767 ]<=1;
white[ 2768 ]<=1;
white[ 2769 ]<=1;
white[ 2770 ]<=1;
white[ 2771 ]<=1;
white[ 2772 ]<=1;
white[ 2773 ]<=1;
white[ 2774 ]<=1;
white[ 2775 ]<=1;
white[ 2776 ]<=1;
white[ 2777 ]<=1;
white[ 2778 ]<=1;
white[ 2779 ]<=1;
white[ 2780 ]<=1;
white[ 2781 ]<=1;
white[ 2782 ]<=1;
white[ 2783 ]<=1;
white[ 2784 ]<=1;
white[ 2785 ]<=1;
white[ 2786 ]<=1;
white[ 2787 ]<=1;
white[ 2788 ]<=1;
white[ 2789 ]<=0;
white[ 2790 ]<=0;
white[ 2791 ]<=0;
white[ 2792 ]<=1;
white[ 2793 ]<=1;
white[ 2794 ]<=1;
white[ 2795 ]<=1;
white[ 2796 ]<=1;
white[ 2797 ]<=1;
white[ 2798 ]<=1;
white[ 2799 ]<=1;
white[ 2800 ]<=1;
white[ 2801 ]<=1;
white[ 2802 ]<=1;
```

```
white[ 2803 ]<=1;
white[ 2804 ]<=1;
white[ 2805 ]<=1;
white[ 2806 ]<=1;
white[ 2807 ]<=1;
white[ 2808 ]<=1;
white[ 2809 ]<=1;
white[ 2810 ]<=1;
white[ 2811 ]<=1;
white[ 2812 ]<=1;
white[ 2813 ]<=1;
white[ 2814 ]<=1;
white[ 2815 ]<=0;
white[ 2816 ]<=0;
white[ 2817 ]<=0;
white[ 2818 ]<=0;
white[ 2819 ]<=0;
white[ 2820 ]<=0;
white[ 2821 ]<=0;
white[ 2822 ]<=0;
white[ 2823 ]<=0;
white[ 2824 ]<=0;
white[ 2825 ]<=0;
white[ 2826 ]<=0;
white[ 2827 ]<=1;
white[ 2828 ]<=1;
white[ 2829 ]<=1;
white[ 2830 ]<=1;
white[ 2831 ]<=1;
white[ 2832 ]<=1;
white[ 2833 ]<=1;
white[ 2834 ]<=1;
white[ 2835 ]<=1;
white[ 2836 ]<=1;
white[ 2837 ]<=1;
white[ 2838 ]<=1;
white[ 2839 ]<=1;
white[ 2840 ]<=1;
white[ 2841 ]<=1;
white[ 2842 ]<=1;
white[ 2843 ]<=1;
white[ 2844 ]<=1;
white[ 2845 ]<=1;
white[ 2846 ]<=1;
white[ 2847 ]<=1;
white[ 2848 ]<=1;
white[ 2849 ]<=0;
white[ 2850 ]<=0;
white[ 2851 ]<=0;
white[ 2852 ]<=1;
white[ 2853 ]<=1;
white[ 2854 ]<=1;
white[ 2855 ]<=1;
white[ 2856 ]<=1;
white[ 2857 ]<=1;
white[ 2858 ]<=1;
white[ 2859 ]<=1;
white[ 2860 ]<=1;
```

```
white[ 2861 ]<=1;
white[ 2862 ]<=1;
white[ 2863 ]<=1;
white[ 2864 ]<=1;
white[ 2865 ]<=1;
white[ 2866 ]<=1;
white[ 2867 ]<=1;
white[ 2868 ]<=1;
white[ 2869 ]<=1;
white[ 2870 ]<=1;
white[ 2871 ]<=1;
white[ 2872 ]<=1;
white[ 2873 ]<=1;
white[ 2874 ]<=1;
white[ 2875 ]<=0;
white[ 2876 ]<=0;
white[ 2877 ]<=0;
white[ 2878 ]<=0;
white[ 2879 ]<=0;
white[ 2880 ]<=0;
white[ 2881 ]<=0;
white[ 2882 ]<=0;
white[ 2883 ]<=0;
white[ 2884 ]<=0;
white[ 2885 ]<=0;
white[ 2886 ]<=0;
white[ 2887 ]<=1;
white[ 2888 ]<=1;
white[ 2889 ]<=1;
white[ 2890 ]<=1;
white[ 2891 ]<=1;
white[ 2892 ]<=1;
white[ 2893 ]<=1;
white[ 2894 ]<=1;
white[ 2895 ]<=1;
white[ 2896 ]<=1;
white[ 2897 ]<=1;
white[ 2898 ]<=1;
white[ 2899 ]<=1;
white[ 2900 ]<=1;
white[ 2901 ]<=1;
white[ 2902 ]<=1;
white[ 2903 ]<=1;
white[ 2904 ]<=1;
white[ 2905 ]<=0;
white[ 2906 ]<=1;
white[ 2907 ]<=1;
white[ 2908 ]<=1;
white[ 2909 ]<=1;
white[ 2910 ]<=1;
white[ 2911 ]<=1;
white[ 2912 ]<=1;
white[ 2913 ]<=1;
white[ 2914 ]<=1;
white[ 2915 ]<=0;
white[ 2916 ]<=1;
white[ 2917 ]<=1;
white[ 2918 ]<=1;
```

```
white[ 2919 ]<=1;
white[ 2920 ]<=1;
white[ 2921 ]<=1;
white[ 2922 ]<=1;
white[ 2923 ]<=1;
white[ 2924 ]<=1;
white[ 2925 ]<=1;
white[ 2926 ]<=1;
white[ 2927 ]<=1;
white[ 2928 ]<=1;
white[ 2929 ]<=1;
white[ 2930 ]<=1;
white[ 2931 ]<=1;
white[ 2932 ]<=1;
white[ 2933 ]<=1;
white[ 2934 ]<=1;
white[ 2935 ]<=0;
white[ 2936 ]<=0;
white[ 2937 ]<=0;
white[ 2938 ]<=0;
white[ 2939 ]<=0;
white[ 2940 ]<=0;
white[ 2941 ]<=0;
white[ 2942 ]<=0;
white[ 2943 ]<=0;
white[ 2944 ]<=0;
white[ 2945 ]<=0;
white[ 2946 ]<=0;
white[ 2947 ]<=1;
white[ 2948 ]<=1;
white[ 2949 ]<=1;
white[ 2950 ]<=1;
white[ 2951 ]<=1;
white[ 2952 ]<=1;
white[ 2953 ]<=1;
white[ 2954 ]<=1;
white[ 2955 ]<=1;
white[ 2956 ]<=1;
white[ 2957 ]<=1;
white[ 2958 ]<=1;
white[ 2959 ]<=1;
white[ 2960 ]<=1;
white[ 2961 ]<=1;
white[ 2962 ]<=1;
white[ 2963 ]<=1;
white[ 2964 ]<=1;
white[ 2965 ]<=0;
white[ 2966 ]<=1;
white[ 2967 ]<=1;
white[ 2968 ]<=1;
white[ 2969 ]<=1;
white[ 2970 ]<=1;
white[ 2971 ]<=1;
white[ 2972 ]<=1;
white[ 2973 ]<=1;
white[ 2974 ]<=1;
white[ 2975 ]<=0;
white[ 2976 ]<=1;
```

```
white[ 2977 ]<=1;
white[ 2978 ]<=1;
white[ 2979 ]<=1;
white[ 2980 ]<=1;
white[ 2981 ]<=1;
white[ 2982 ]<=1;
white[ 2983 ]<=1;
white[ 2984 ]<=1;
white[ 2985 ]<=1;
white[ 2986 ]<=1;
white[ 2987 ]<=1;
white[ 2988 ]<=1;
white[ 2989 ]<=1;
white[ 2990 ]<=1;
white[ 2991 ]<=1;
white[ 2992 ]<=1;
white[ 2993 ]<=1;
white[ 2994 ]<=1;
white[ 2995 ]<=0;
white[ 2996 ]<=0;
white[ 2997 ]<=0;
white[ 2998 ]<=0;
white[ 2999 ]<=0;
white[ 3000 ]<=0;
white[ 3001 ]<=0;
white[ 3002 ]<=0;
white[ 3003 ]<=0;
white[ 3004 ]<=0;
white[ 3005 ]<=0;
white[ 3006 ]<=0;
white[ 3007 ]<=1;
white[ 3008 ]<=1;
white[ 3009 ]<=1;
white[ 3010 ]<=1;
white[ 3011 ]<=1;
white[ 3012 ]<=1;
white[ 3013 ]<=1;
white[ 3014 ]<=1;
white[ 3015 ]<=1;
white[ 3016 ]<=1;
white[ 3017 ]<=1;
white[ 3018 ]<=1;
white[ 3019 ]<=1;
white[ 3020 ]<=1;
white[ 3021 ]<=1;
white[ 3022 ]<=1;
white[ 3023 ]<=1;
white[ 3024 ]<=1;
white[ 3025 ]<=1;
white[ 3026 ]<=1;
white[ 3027 ]<=1;
white[ 3028 ]<=1;
white[ 3029 ]<=1;
white[ 3030 ]<=1;
white[ 3031 ]<=1;
white[ 3032 ]<=1;
white[ 3033 ]<=1;
white[ 3034 ]<=1;
```

```
white[ 3035 ]<=0;
white[ 3036 ]<=1;
white[ 3037 ]<=1;
white[ 3038 ]<=1;
white[ 3039 ]<=1;
white[ 3040 ]<=1;
white[ 3041 ]<=1;
white[ 3042 ]<=1;
white[ 3043 ]<=1;
white[ 3044 ]<=1;
white[ 3045 ]<=1;
white[ 3046 ]<=1;
white[ 3047 ]<=1;
white[ 3048 ]<=1;
white[ 3049 ]<=1;
white[ 3050 ]<=1;
white[ 3051 ]<=1;
white[ 3052 ]<=1;
white[ 3053 ]<=1;
white[ 3054 ]<=1;
white[ 3055 ]<=0;
white[ 3056 ]<=0;
white[ 3057 ]<=0;
white[ 3058 ]<=0;
white[ 3059 ]<=0;
white[ 3060 ]<=0;
white[ 3061 ]<=0;
white[ 3062 ]<=0;
white[ 3063 ]<=0;
white[ 3064 ]<=0;
white[ 3065 ]<=0;
white[ 3066 ]<=0;
white[ 3067 ]<=1;
white[ 3068 ]<=1;
white[ 3069 ]<=1;
white[ 3070 ]<=1;
white[ 3071 ]<=1;
white[ 3072 ]<=1;
white[ 3073 ]<=1;
white[ 3074 ]<=1;
white[ 3075 ]<=1;
white[ 3076 ]<=1;
white[ 3077 ]<=1;
white[ 3078 ]<=1;
white[ 3079 ]<=1;
white[ 3080 ]<=0;
white[ 3081 ]<=1;
white[ 3082 ]<=1;
white[ 3083 ]<=1;
white[ 3084 ]<=1;
white[ 3085 ]<=1;
white[ 3086 ]<=0;
white[ 3087 ]<=1;
white[ 3088 ]<=1;
white[ 3089 ]<=1;
white[ 3090 ]<=1;
white[ 3091 ]<=1;
white[ 3092 ]<=1;
```

```
white[ 3093 ]<=1;
white[ 3094 ]<=0;
white[ 3095 ]<=1;
white[ 3096 ]<=1;
white[ 3097 ]<=1;
white[ 3098 ]<=1;
white[ 3099 ]<=1;
white[ 3100 ]<=1;
white[ 3101 ]<=1;
white[ 3102 ]<=1;
white[ 3103 ]<=1;
white[ 3104 ]<=1;
white[ 3105 ]<=1;
white[ 3106 ]<=1;
white[ 3107 ]<=1;
white[ 3108 ]<=1;
white[ 3109 ]<=1;
white[ 3110 ]<=1;
white[ 3111 ]<=1;
white[ 3112 ]<=1;
white[ 3113 ]<=1;
white[ 3114 ]<=1;
white[ 3115 ]<=0;
white[ 3116 ]<=0;
white[ 3117 ]<=0;
white[ 3118 ]<=0;
white[ 3119 ]<=0;
white[ 3120 ]<=0;
white[ 3121 ]<=0;
white[ 3122 ]<=0;
white[ 3123 ]<=0;
white[ 3124 ]<=0;
white[ 3125 ]<=0;
white[ 3126 ]<=0;
white[ 3127 ]<=1;
white[ 3128 ]<=1;
white[ 3129 ]<=1;
white[ 3130 ]<=1;
white[ 3131 ]<=1;
white[ 3132 ]<=1;
white[ 3133 ]<=1;
white[ 3134 ]<=1;
white[ 3135 ]<=1;
white[ 3136 ]<=1;
white[ 3137 ]<=1;
white[ 3138 ]<=1;
white[ 3139 ]<=1;
white[ 3140 ]<=0;
white[ 3141 ]<=1;
white[ 3142 ]<=1;
white[ 3143 ]<=1;
white[ 3144 ]<=1;
white[ 3145 ]<=1;
white[ 3146 ]<=0;
white[ 3147 ]<=0;
white[ 3148 ]<=0;
white[ 3149 ]<=0;
white[ 3150 ]<=0;
```

```
white[ 3151 ]<=0;
white[ 3152 ]<=0;
white[ 3153 ]<=0;
white[ 3154 ]<=1;
white[ 3155 ]<=1;
white[ 3156 ]<=1;
white[ 3157 ]<=1;
white[ 3158 ]<=1;
white[ 3159 ]<=1;
white[ 3160 ]<=0;
white[ 3161 ]<=1;
white[ 3162 ]<=1;
white[ 3163 ]<=1;
white[ 3164 ]<=1;
white[ 3165 ]<=1;
white[ 3166 ]<=1;
white[ 3167 ]<=1;
white[ 3168 ]<=1;
white[ 3169 ]<=1;
white[ 3170 ]<=1;
white[ 3171 ]<=1;
white[ 3172 ]<=1;
white[ 3173 ]<=1;
white[ 3174 ]<=1;
white[ 3175 ]<=0;
white[ 3176 ]<=0;
white[ 3177 ]<=0;
white[ 3178 ]<=0;
white[ 3179 ]<=0;
white[ 3180 ]<=0;
white[ 3181 ]<=0;
white[ 3182 ]<=0;
white[ 3183 ]<=0;
white[ 3184 ]<=0;
white[ 3185 ]<=0;
white[ 3186 ]<=0;
white[ 3187 ]<=1;
white[ 3188 ]<=1;
white[ 3189 ]<=1;
white[ 3190 ]<=1;
white[ 3191 ]<=1;
white[ 3192 ]<=1;
white[ 3193 ]<=1;
white[ 3194 ]<=1;
white[ 3195 ]<=1;
white[ 3196 ]<=1;
white[ 3197 ]<=1;
white[ 3198 ]<=0;
white[ 3199 ]<=1;
white[ 3200 ]<=1;
white[ 3201 ]<=1;
white[ 3202 ]<=1;
white[ 3203 ]<=1;
white[ 3204 ]<=1;
white[ 3205 ]<=0;
white[ 3206 ]<=0;
white[ 3207 ]<=0;
white[ 3208 ]<=0;
```

```
white[ 3209 ]<=0;
white[ 3210 ]<=0;
white[ 3211 ]<=0;
white[ 3212 ]<=0;
white[ 3213 ]<=0;
white[ 3214 ]<=1;
white[ 3215 ]<=1;
white[ 3216 ]<=1;
white[ 3217 ]<=1;
white[ 3218 ]<=1;
white[ 3219 ]<=1;
white[ 3220 ]<=0;
white[ 3221 ]<=1;
white[ 3222 ]<=0;
white[ 3223 ]<=1;
white[ 3224 ]<=1;
white[ 3225 ]<=1;
white[ 3226 ]<=1;
white[ 3227 ]<=1;
white[ 3228 ]<=1;
white[ 3229 ]<=1;
white[ 3230 ]<=1;
white[ 3231 ]<=1;
white[ 3232 ]<=1;
white[ 3233 ]<=1;
white[ 3234 ]<=1;
white[ 3235 ]<=0;
white[ 3236 ]<=0;
white[ 3237 ]<=0;
white[ 3238 ]<=0;
white[ 3239 ]<=0;
white[ 3240 ]<=0;
white[ 3241 ]<=0;
white[ 3242 ]<=0;
white[ 3243 ]<=0;
white[ 3244 ]<=0;
white[ 3245 ]<=0;
white[ 3246 ]<=0;
white[ 3247 ]<=1;
white[ 3248 ]<=1;
white[ 3249 ]<=1;
white[ 3250 ]<=1;
white[ 3251 ]<=1;
white[ 3252 ]<=1;
white[ 3253 ]<=1;
white[ 3254 ]<=1;
white[ 3255 ]<=1;
white[ 3256 ]<=1;
white[ 3257 ]<=1;
white[ 3258 ]<=1;
white[ 3259 ]<=0;
white[ 3260 ]<=1;
white[ 3261 ]<=0;
white[ 3262 ]<=1;
white[ 3263 ]<=1;
white[ 3264 ]<=1;
white[ 3265 ]<=1;
white[ 3266 ]<=0;
```

```
white[ 3267 ]<=0;
white[ 3268 ]<=0;
white[ 3269 ]<=0;
white[ 3270 ]<=0;
white[ 3271 ]<=0;
white[ 3272 ]<=0;
white[ 3273 ]<=0;
white[ 3274 ]<=0;
white[ 3275 ]<=0;
white[ 3276 ]<=1;
white[ 3277 ]<=1;
white[ 3278 ]<=1;
white[ 3279 ]<=0;
white[ 3280 ]<=1;
white[ 3281 ]<=1;
white[ 3282 ]<=1;
white[ 3283 ]<=1;
white[ 3284 ]<=1;
white[ 3285 ]<=1;
white[ 3286 ]<=1;
white[ 3287 ]<=1;
white[ 3288 ]<=1;
white[ 3289 ]<=1;
white[ 3290 ]<=1;
white[ 3291 ]<=1;
white[ 3292 ]<=1;
white[ 3293 ]<=1;
white[ 3294 ]<=1;
white[ 3295 ]<=0;
white[ 3296 ]<=0;
white[ 3297 ]<=0;
white[ 3298 ]<=0;
white[ 3299 ]<=0;
white[ 3300 ]<=0;
white[ 3301 ]<=0;
white[ 3302 ]<=0;
white[ 3303 ]<=0;
white[ 3304 ]<=0;
white[ 3305 ]<=0;
white[ 3306 ]<=0;
white[ 3307 ]<=1;
white[ 3308 ]<=1;
white[ 3309 ]<=1;
white[ 3310 ]<=1;
white[ 3311 ]<=1;
white[ 3312 ]<=1;
white[ 3313 ]<=1;
white[ 3314 ]<=1;
white[ 3315 ]<=1;
white[ 3316 ]<=1;
white[ 3317 ]<=1;
white[ 3318 ]<=1;
white[ 3319 ]<=1;
white[ 3320 ]<=1;
white[ 3321 ]<=0;
white[ 3322 ]<=1;
white[ 3323 ]<=1;
white[ 3324 ]<=1;
```

```
white[ 3325 ]<=1;
white[ 3326 ]<=0;
white[ 3327 ]<=0;
white[ 3328 ]<=0;
white[ 3329 ]<=0;
white[ 3330 ]<=0;
white[ 3331 ]<=0;
white[ 3332 ]<=0;
white[ 3333 ]<=0;
white[ 3334 ]<=0;
white[ 3335 ]<=1;
white[ 3336 ]<=1;
white[ 3337 ]<=1;
white[ 3338 ]<=1;
white[ 3339 ]<=0;
white[ 3340 ]<=1;
white[ 3341 ]<=1;
white[ 3342 ]<=1;
white[ 3343 ]<=1;
white[ 3344 ]<=1;
white[ 3345 ]<=1;
white[ 3346 ]<=1;
white[ 3347 ]<=1;
white[ 3348 ]<=1;
white[ 3349 ]<=1;
white[ 3350 ]<=1;
white[ 3351 ]<=1;
white[ 3352 ]<=1;
white[ 3353 ]<=1;
white[ 3354 ]<=1;
white[ 3355 ]<=0;
white[ 3356 ]<=0;
white[ 3357 ]<=0;
white[ 3358 ]<=0;
white[ 3359 ]<=0;
white[ 3360 ]<=0;
white[ 3361 ]<=0;
white[ 3362 ]<=0;
white[ 3363 ]<=0;
white[ 3364 ]<=0;
white[ 3365 ]<=0;
white[ 3366 ]<=0;
white[ 3367 ]<=1;
white[ 3368 ]<=1;
white[ 3369 ]<=1;
white[ 3370 ]<=1;
white[ 3371 ]<=1;
white[ 3372 ]<=1;
white[ 3373 ]<=1;
white[ 3374 ]<=1;
white[ 3375 ]<=1;
white[ 3376 ]<=1;
white[ 3377 ]<=1;
white[ 3378 ]<=1;
white[ 3379 ]<=1;
white[ 3380 ]<=1;
white[ 3381 ]<=1;
white[ 3382 ]<=1;
```

```
white[ 3383 ]<=1;
white[ 3384 ]<=1;
white[ 3385 ]<=1;
white[ 3386 ]<=0;
white[ 3387 ]<=0;
white[ 3388 ]<=0;
white[ 3389 ]<=0;
white[ 3390 ]<=0;
white[ 3391 ]<=0;
white[ 3392 ]<=0;
white[ 3393 ]<=0;
white[ 3394 ]<=0;
white[ 3395 ]<=1;
white[ 3396 ]<=1;
white[ 3397 ]<=1;
white[ 3398 ]<=1;
white[ 3399 ]<=1;
white[ 3400 ]<=1;
white[ 3401 ]<=1;
white[ 3402 ]<=1;
white[ 3403 ]<=1;
white[ 3404 ]<=1;
white[ 3405 ]<=1;
white[ 3406 ]<=1;
white[ 3407 ]<=1;
white[ 3408 ]<=1;
white[ 3409 ]<=1;
white[ 3410 ]<=1;
white[ 3411 ]<=1;
white[ 3412 ]<=1;
white[ 3413 ]<=1;
white[ 3414 ]<=1;
white[ 3415 ]<=0;
white[ 3416 ]<=0;
white[ 3417 ]<=0;
white[ 3418 ]<=0;
white[ 3419 ]<=0;
white[ 3420 ]<=0;
white[ 3421 ]<=0;
white[ 3422 ]<=0;
white[ 3423 ]<=0;
white[ 3424 ]<=0;
white[ 3425 ]<=0;
white[ 3426 ]<=0;
white[ 3427 ]<=1;
white[ 3428 ]<=1;
white[ 3429 ]<=1;
white[ 3430 ]<=1;
white[ 3431 ]<=1;
white[ 3432 ]<=1;
white[ 3433 ]<=1;
white[ 3434 ]<=1;
white[ 3435 ]<=1;
white[ 3436 ]<=1;
white[ 3437 ]<=1;
white[ 3438 ]<=1;
white[ 3439 ]<=1;
white[ 3440 ]<=1;
```

```
white[ 3441 ]<=1;
white[ 3442 ]<=1;
white[ 3443 ]<=1;
white[ 3444 ]<=1;
white[ 3445 ]<=0;
white[ 3446 ]<=0;
white[ 3447 ]<=0;
white[ 3448 ]<=0;
white[ 3449 ]<=0;
white[ 3450 ]<=0;
white[ 3451 ]<=0;
white[ 3452 ]<=0;
white[ 3453 ]<=0;
white[ 3454 ]<=0;
white[ 3455 ]<=0;
white[ 3456 ]<=1;
white[ 3457 ]<=1;
white[ 3458 ]<=1;
white[ 3459 ]<=1;
white[ 3460 ]<=0;
white[ 3461 ]<=1;
white[ 3462 ]<=1;
white[ 3463 ]<=1;
white[ 3464 ]<=1;
white[ 3465 ]<=1;
white[ 3466 ]<=1;
white[ 3467 ]<=1;
white[ 3468 ]<=1;
white[ 3469 ]<=1;
white[ 3470 ]<=1;
white[ 3471 ]<=1;
white[ 3472 ]<=1;
white[ 3473 ]<=1;
white[ 3474 ]<=1;
white[ 3475 ]<=0;
white[ 3476 ]<=0;
white[ 3477 ]<=0;
white[ 3478 ]<=0;
white[ 3479 ]<=0;
white[ 3480 ]<=0;
white[ 3481 ]<=0;
white[ 3482 ]<=0;
white[ 3483 ]<=0;
white[ 3484 ]<=0;
white[ 3485 ]<=0;
white[ 3486 ]<=0;
white[ 3487 ]<=1;
white[ 3488 ]<=1;
white[ 3489 ]<=1;
white[ 3490 ]<=1;
white[ 3491 ]<=1;
white[ 3492 ]<=1;
white[ 3493 ]<=1;
white[ 3494 ]<=1;
white[ 3495 ]<=1;
white[ 3496 ]<=1;
white[ 3497 ]<=1;
white[ 3498 ]<=1;
```

```
white[ 3499 ]<=1;
white[ 3500 ]<=1;
white[ 3501 ]<=1;
white[ 3502 ]<=1;
white[ 3503 ]<=1;
white[ 3504 ]<=1;
white[ 3505 ]<=1;
white[ 3506 ]<=0;
white[ 3507 ]<=0;
white[ 3508 ]<=0;
white[ 3509 ]<=0;
white[ 3510 ]<=0;
white[ 3511 ]<=0;
white[ 3512 ]<=0;
white[ 3513 ]<=0;
white[ 3514 ]<=0;
white[ 3515 ]<=1;
white[ 3516 ]<=1;
white[ 3517 ]<=1;
white[ 3518 ]<=1;
white[ 3519 ]<=1;
white[ 3520 ]<=1;
white[ 3521 ]<=1;
white[ 3522 ]<=1;
white[ 3523 ]<=1;
white[ 3524 ]<=1;
white[ 3525 ]<=1;
white[ 3526 ]<=1;
white[ 3527 ]<=1;
white[ 3528 ]<=1;
white[ 3529 ]<=1;
white[ 3530 ]<=1;
white[ 3531 ]<=1;
white[ 3532 ]<=1;
white[ 3533 ]<=1;
white[ 3534 ]<=0;
white[ 3535 ]<=0;
white[ 3536 ]<=0;
white[ 3537 ]<=0;
white[ 3538 ]<=0;
white[ 3539 ]<=0;
white[ 3540 ]<=0;
white[ 3541 ]<=0;
white[ 3542 ]<=0;
white[ 3543 ]<=0;
white[ 3544 ]<=0;
white[ 3545 ]<=0;
white[ 3546 ]<=0;
white[ 3547 ]<=1;
white[ 3548 ]<=1;
white[ 3549 ]<=1;
white[ 3550 ]<=1;
white[ 3551 ]<=1;
white[ 3552 ]<=1;
white[ 3553 ]<=1;
white[ 3554 ]<=1;
white[ 3555 ]<=1;
white[ 3556 ]<=1;
```

```
                white[ 3557 ]<=1;
                white[ 3558 ]<=1;
                white[ 3559 ]<=1;
                white[ 3560 ]<=1;
                white[ 3561 ]<=1;
                white[ 3562 ]<=1;
                white[ 3563 ]<=1;
                white[ 3564 ]<=1;
                white[ 3565 ]<=1;
                white[ 3566 ]<=1;
                white[ 3567 ]<=1;
                white[ 3568 ]<=1;
                white[ 3569 ]<=0;
                white[ 3570 ]<=0;
                white[ 3571 ]<=0;
                white[ 3572 ]<=1;
                white[ 3573 ]<=1;
                white[ 3574 ]<=1;
                white[ 3575 ]<=1;
                white[ 3576 ]<=1;
                white[ 3577 ]<=1;
                white[ 3578 ]<=1;
                white[ 3579 ]<=1;
                white[ 3580 ]<=1;
                white[ 3581 ]<=1;
                white[ 3582 ]<=1;
                white[ 3583 ]<=1;
                white[ 3584 ]<=1;
                white[ 3585 ]<=1;
                white[ 3586 ]<=1;
                white[ 3587 ]<=1;
                white[ 3588 ]<=1;
                white[ 3589 ]<=1;
                white[ 3590 ]<=1;
                white[ 3591 ]<=1;
                white[ 3592 ]<=1;
                white[ 3593 ]<=1;
                white[ 3594 ]<=0;
                white[ 3595 ]<=0;
                white[ 3596 ]<=0;
                white[ 3597 ]<=0;
                white[ 3598 ]<=0;
                white[ 3599 ]<=0;
                end
endcase
end
end

always@(posedge iclk or negedge ireset_n)
begin
      if(!ireset_n)
      begin
            oOVERRIDE<= 0;
            oADDR <= 18'bZ;
            counter <= 6'd59;
      end
      else if(display_area)
      begin
```

```
            if(counter == 0)
                counter <= 6'd59;
            else
                counter <= counter-1;
            oOVERRIDE <=white[counter + 60*(iy_cnt-isymbol_pos_y)];
            oADDR <= iBASE_ADDR + counter + 60*(iy_cnt-isymbol_pos_y);
        end
        else
        begin
            oOVERRIDE <= 0;
            oADDR <= 18'bZ;
        end
end

endmodule
```

three_wire_controller.v

```
// --------------------------------------------------------------------

// Copyright (c) 2005 by Terasic Technologies Inc.

// --------------------------------------------------------------------

//

// Permission:

//

//     Terasic grants permission to use and modify this code for use

//     in synthesis for all Terasic Development Boards and Altera Development

//     Kits made by Terasic.    Other use of this code, including the selling

//     ,duplication, or modification of any portion is strictly prohibited.

//

// Disclaimer:

//

//     This VHDL/Verilog or C/C++ source code is intended as a design reference

//     which illustrates how these types of functions can be implemented.

//     It is the user's responsibility to verify their design for

//     consistency and functionality through the use of formal

//     verification methods.    Terasic provides no warranty regarding the use

//     or functionality of this code.
```

```
//
// --------------------------------------------------------------------
//
//                          Terasic Technologies Inc
//                          356 Fu-Shin E. Rd Sec. 1. JhuBei City,
//                          HsinChu County, Taiwan
//                          302
//
//                          web: http://www.terasic.com/
//                          email: support@terasic.com
//
// --------------------------------------------------------------------
//
// Major Functions: 3-Wire Serial Bus Controller
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
//     Ver  :| Author              :| Mod. Date :| Changes Made:
//     V1.0 :| Johnny Chen         :| 06/03/23  :|          Initial Revision
// --------------------------------------------------------------------

module three_wire_controller(     //     Host Side
                                  iCLK,
                                  iRST,
                                  iDATA,
                                  iSTR,
                                  oACK,
```

```
                                oRDY,
                                oCLK,
                                //  Serial Side
                                oSCEN,
                                SDA,
                                oSCLK    );
//  Host Side
input           iCLK;
input           iRST;
input           iSTR;
input   [15:0]  iDATA;
output          oACK;
output          oRDY;
output          oCLK;
//  Serial Side
output          oSCEN;
inout           SDA;
output          oSCLK;
//  Internal Register and Wire
reg             mSPI_CLK;
reg     [15:0]  mSPI_CLK_DIV;
reg             mSEN;
reg             mSDATA;
reg             mSCLK;
reg             mACK;
reg     [4:0] mST;

parameter   CLK_Freq =  50000000;   //  50  MHz
parameter   SPI_Freq  =  20000;     //  20  KHz
```

```
//  Serial Clock Generator
always@(posedge iCLK or negedge iRST)
begin
    if(!iRST)
    begin
        mSPI_CLK<=  0;
        mSPI_CLK_DIV<=  0;
    end
    else
    begin
        if( mSPI_CLK_DIV  < (CLK_Freq/SPI_Freq) )
        mSPI_CLK_DIV<=  mSPI_CLK_DIV+1;
        else
        begin
            mSPI_CLK_DIV<=  0;
            mSPI_CLK    <=  ~mSPI_CLK;
        end
    end
end
//  Parallel to Serial
always@(negedge mSPI_CLK or negedge iRST)
begin
    if(!iRST)
    begin
        mSEN    <=  1'b1;
        mSCLK   <=  1'b0;
        mSDATA  <=  1'bz;
        mACK    <=  1'b0;
```

```
        mST       <=   4'h00;
    end
    else
    begin
        if(iSTR)
        begin
            if(mST<17)
            mST <=   mST+1'b1;
            if(mST==0)
            begin
                mSEN     <=   1'b0;
                mSCLK    <=   1'b1;
            end
            else if(mST==8)
            mACK     <=   SDA;
            else if(mST==16 && mSCLK)
            begin
                mSEN     <=   1'b1;
                mSCLK    <=   1'b0;
            end
            if(mST<16)
            mSDATA  <=   iDATA[15-mST];
        end
        else
        begin
            mSEN     <=   1'b1;
            mSCLK    <=   1'b0;
            mSDATA  <=   1'bz;
            mACK     <=   1'b0;
```

```
                mST     <=  4'h00;
            end
        end
end

assign  oACK        =   mACK;
assign  oRDY        =   (mST==17)   ?   1'b1 :   1'b0;
assign  oSCEN       =   mSEN;
assign  oSCLK       =   mSCLK   &   mSPI_CLK;
assign  SDA =   (mST==8) ?  1'bz :
                            (mST==17)   ?   1'bz :
                                            mSDATA ;
assign  oCLK        =   mSPI_CLK;

endmodule
```

Timmer.v

```
module Timmer(
                clk_ltm,
                reset_n,
                x_cnt,
                y_cnt,
                state,
                oOVERRIDE,
                oADDR
                );

input clk_ltm;
input reset_n;
input [3:0] state;
input [10:0] x_cnt;
input [9:0] y_cnt;
output oOVERRIDE;
output [17:0] oADDR;
//PARAMETER
parameter BASE_ADDR = 89025;
parameter INIT = 4'b0100;
parameter START = 4'b0101;
parameter PAUSE = 4'b0110;
parameter H_LINE = 1056;
parameter V_LINE = 525;
```

```
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
parameter clock_base = 18'd89039;
//WIRE & REG
reg [3:0] inumber_s;
reg [3:0] inumber_ts;
reg [3:0] inumber_min;
reg [26:0] counter;

wire [3:0] num_input_s;
wire [3:0] num_input_ts;
wire [3:0] num_input_min;
wire [26:0] oCNT;
wire [3:0] OVERRIDE;
wire flag1;
wire flag2;
wire [17:0] address;

symbol clock(
                        .iclk(clk_ltm),
                        .ireset_n(reset_n),
                        .ix_cnt(x_cnt),
                        .iy_cnt(y_cnt),
                        .isymbol_pos_x(Hsync_Blank-1+730),
                        .isymbol_pos_y(Vertical_Back_Porch+10),
                        .iBASE_ADDR(clock_base),
                        .oOVERRIDE(OVERRIDE[3]),
                        .oADDR(address)
);
//////////////////////Timmer//////////////////////////
number LED0(
                        .iclk(clk_ltm),
                        .iReset_n(reset_n),
                        .ixcount(x_cnt),
                        .iycount(y_cnt),
                        .iled_pos_x(Hsync_Blank-1+600),
                        .iled_pos_y(Vertical_Back_Porch),
                        .inumber(num_input_s),
                        .oOVERRIDE(OVERRIDE[0])
);

number LED1(
                        .iclk(clk_ltm),
                        .iReset_n(reset_n),
                        .ixcount(x_cnt),
                        .iycount(y_cnt),
                        .iled_pos_x(Hsync_Blank-1+640),
                        .iled_pos_y(Vertical_Back_Porch),
                        .inumber(num_input_ts),
                        .oOVERRIDE(OVERRIDE[1])
);

number LED2(
                        .iclk(clk_ltm),
                        .iReset_n(reset_n),
                        .ixcount(x_cnt),
```

```
                        .iycount(y_cnt),
                        .iled_pos_x(Hsync_Blank-1+680),
                        .iled_pos_y(Vertical_Back_Porch),
                        .inumber(num_input_min),
                        .oOVERRIDE(OVERRIDE[2])
);

assign flag1 = (oCNT == 26'd24999999) ? 1'b1 : 1'b0;
assign flag2 = (inumber_s == 0) ? 1'b1 : 1'b0;
assign num_input_s = inumber_s;
assign num_input_ts = inumber_ts;
assign num_input_min = inumber_min;
assign oCNT = counter;


always@(posedge clk_ltm or negedge reset_n)
begin
      if(!reset_n)
      begin
      inumber_min <= 4'd2;
      inumber_ts <= 4'd0;
      inumber_s <= 4'd0;
      counter <= 26'b0;
      end
      else if(state == INIT)
      begin
      inumber_min <= 4'd2;
      inumber_ts <= 4'd0;
      inumber_s <= 4'd0;
      counter <= 26'b0;
      end
      else if(state == PAUSE)
      begin
      inumber_min <= inumber_min;
      inumber_ts <= inumber_ts;
      inumber_s <= inumber_s;
      end
      else if(state == START)
            begin
            if(flag1)
            begin
                  if(flag2)
                        begin
                              if(inumber_ts != 0)
                                    begin
                                          inumber_ts <= inumber_ts - 1;
                                          inumber_s <= 4'd9;
                                          counter <= 26'b0;
                                    end
                              else
                                    begin
                                          if(inumber_min != 0)
                                                begin
                                                      inumber_min <= inumber_min -1;
                                                      inumber_ts    <= 4'd5;
                                                      inumber_s     <= 4'd9;
                                                      counter <= 26'b0;
                                                end
```

```
                                    else
                                        begin
                                            inumber_min <= 4'd2;
                                            inumber_ts <= 4'd0;
                                            inumber_s <= 4'd0;
                                            counter <= 26'b0;
                                        end
                                end
                            end
                        else
                        begin
                        inumber_s <= inumber_s-1;
                        counter <= 26'b0;
                        end
                end
        else
                counter <= counter +1;
        end
end

////OUTPUT
assign oOVERRIDE = OVERRIDE[0] || OVERRIDE[1] || OVERRIDE[2] || OVERRIDE[3];
assign oADDR = (OVERRIDE[0] || OVERRIDE[1] || OVERRIDE[2]) ? BASE_ADDR:
                            OVERRIDE[3] ? address : 18'bZ;

endmodule
```

touch_irq_detector.v

```
// --------------------------------------------------------------------

// Copyright (c) 2005 by Terasic Technologies Inc.

// --------------------------------------------------------------------

//

// Permission:

//

//     Terasic grants permission to use and modify this code for use

//     in synthesis for all Terasic Development Boards and Altera Development

//     Kits made by Terasic.    Other use of this code, including the selling

//     ,duplication, or modification of any portion is strictly prohibited.

//

// Disclaimer:

//

//     This VHDL/Verilog or C/C++ source code is intended as a design reference
```

```
//      which illustrates how these types of functions can be implemented.
//      It is the user's responsibility to verify their design for
//      consistency and functionality through the use of formal
//      verification methods.    Terasic provides no warranty regarding the use
//      or functionality of this code.
//
// --------------------------------------------------------------------
//
//                                 Terasic Technologies Inc
//                                 356 Fu-Shin E. Rd Sec. 1. JhuBei City,
//                                 HsinChu County, Taiwan
//                                 302
//
//                                 web: http://www.terasic.com/
//                                 email: support@terasic.com
//
// --------------------------------------------------------------------
//
// Major Functions:
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
//      Ver    :| Author                    :| Mod. Date :| Changes Made:
//      V1.0 :| Johnny Fan                 :| 07/06/30    :| Initial Revision
// --------------------------------------------------------------------
module touch_irq_detector   (
                                clk,
```

```
                        reset_n,
                        iTOUCH_IRQ,
                        iX_COORD,
                        iY_COORD,
                        iNEW_COORD,
                        oDISPLAY_MODE
                        );

//=============================================================================
// PARAMETER declarations
//=============================================================================

parameter      TOUCH_CNT_CLEAR = 24'h2fffff;    // total photo numbers

//=============================================================================
// PORT declarations
//=============================================================================
input      wire       clk;     // system clock 50Mhz
input      wire       reset_n;// system reset
input                 iTOUCH_IRQ;
input      [11:0]     iX_COORD;              // X coordinate form touch panel
input      [11:0]     iY_COORD;              // Y coordinate form touch panel
input                 iNEW_COORD;                 // new coordinates indicate
output     oDISPLAY_MODE;          // displaed photo number
//=============================================================================
// REG/WIRE declarations
//=============================================================================
reg                   touch_en;
reg                   touch_en_clr;
```

```
reg   [24:0]      touch_delay_cnt;
reg   oDISPLAY_MODE;
//=============================================================================
// Structural coding
//=============================================================================

always@(posedge clk or negedge reset_n)
      begin
            if (!reset_n)
                  touch_en <= 0;
            else if (touch_en_clr)
                  touch_en <= 0;
            else if (iTOUCH_IRQ)
                  touch_en <= 1;
      end



always@(posedge clk or negedge reset_n)
      begin
            if (!reset_n)
                  begin
                        touch_delay_cnt <= 0;
                        touch_en_clr <= 0;
                  end
            else if (touch_delay_cnt == TOUCH_CNT_CLEAR)
                  begin
                        touch_delay_cnt <= 0;
                        touch_en_clr <= 1;
                  end
```

```
        else if (touch_en)
            touch_delay_cnt <= touch_delay_cnt + 1;
        else
            begin
                touch_delay_cnt <= 0;
                touch_en_clr <= 0;
            end
    end


always@(posedge clk or negedge reset_n)
    begin
        if (!reset_n)
            oDISPLAY_MODE <= 0;
        else if(iTOUCH_IRQ ==0)
            oDISPLAY_MODE <= 0;
        else if (iTOUCH_IRQ && !touch_en)
            oDISPLAY_MODE <= oDISPLAY_MODE + 1;
    end

endmodule
```

welcome.v

```
module welcome(
                    iclk,
                    ireset_n,
                    ix_cnt,
                    iy_cnt,
                    istate,
                    oADDR);

//=============================================
//Parameter=============================
//=============================================
parameter A = 5'd1;
parameter B = 5'd2;
parameter C = 5'd3;
parameter D = 5'd4;
```

```
parameter E = 5'd5;
parameter F = 5'd6;
parameter G = 5'd7;
parameter H = 5'd8;
parameter I = 5'd9;
parameter J = 5'd10;
parameter K = 5'd11;
parameter L = 5'd12;
parameter M = 5'd13;
parameter N = 5'd14;
parameter O = 5'd15;
parameter P = 5'd16;
parameter Q = 5'd17;
parameter R = 5'd18;
parameter S = 5'd19;
parameter T = 5'd20;
parameter U = 5'd21;
parameter V = 5'd22;
parameter W = 5'd23;
parameter X = 5'd24;
parameter Y = 5'd25;
parameter Z = 5'd26;

parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
parameter indicator_base = 18'd99839;
//=============================================
//I/O DECALARATION=============================
//=============================================
input iclk;
input ireset_n;
input [10:0] ix_cnt;
input [9:0] iy_cnt;
input [3:0] istate;
output [17:0] oADDR;

//=============================================
//WIRE & REG=============================
//=============================================
wire [4:0] letter[0:24];
wire [10:0] letter_pos_x[0:24];
wire [9:0] letter_pos_y[0:24];
wire [25:0] override;
wire [17:0] address[0:25];
reg [17:0] oADDR;
reg [10:0] symbol_pos_x;
reg [9:0] symbol_pos_y;
wire [3:0] state;

//=============================================
//Structured logic=============================
//=============================================
assign state = istate;
```

```
//Assign Letters
assign letter[0] = W;
assign letter[1] = H;
assign letter[2] = A;
assign letter[3] = C;
assign letter[4] = A;
assign letter[5] = M;
assign letter[6] = O;
assign letter[7] = L;
assign letter[8] = E;
assign letter[9] = E;
assign letter[10] = A;
assign letter[11] = S;
assign letter[12] = Y;
assign letter[13] = N;
assign letter[14] = O;
assign letter[15] = R;
assign letter[16] = M;
assign letter[17] = A;
assign letter[18] = L;
assign letter[19] = H;
assign letter[20] = A;
assign letter[21] = R;
assign letter[22] = D;
assign letter[23] = G;
assign letter[24] = O;

//assign letter position
//whac a mole
assign letter_pos_x[0] = Hsync_Blank-2+544;
assign letter_pos_x[1] = Hsync_Blank-2+512;
assign letter_pos_x[2] = Hsync_Blank-2+480;
assign letter_pos_x[3] = Hsync_Blank-2+448;
assign letter_pos_x[4] = Hsync_Blank-2+384;
assign letter_pos_x[5] = Hsync_Blank-2+320;
assign letter_pos_x[6] = Hsync_Blank-2+288;
assign letter_pos_x[7] = Hsync_Blank-2+256;
assign letter_pos_x[8] = Hsync_Blank-2+224;
//easy
assign letter_pos_x[9] = Hsync_Blank-2+432;
assign letter_pos_x[10] = Hsync_Blank-2+400;
assign letter_pos_x[11] = Hsync_Blank-2+368;
assign letter_pos_x[12] = Hsync_Blank-2+336;
//normal
assign letter_pos_x[13] = Hsync_Blank-2+464;
assign letter_pos_x[14] = Hsync_Blank-2+432;
assign letter_pos_x[15] = Hsync_Blank-2+400;
assign letter_pos_x[16] = Hsync_Blank-2+368;
assign letter_pos_x[17] = Hsync_Blank-2+336;
assign letter_pos_x[18] = Hsync_Blank-2+304;
//hard
assign letter_pos_x[19] = Hsync_Blank-2+432;
assign letter_pos_x[20] = Hsync_Blank-2+400;
assign letter_pos_x[21] = Hsync_Blank-2+368;
assign letter_pos_x[22] = Hsync_Blank-2+336;
//go
assign letter_pos_x[23] = Hsync_Blank-2+74;
assign letter_pos_x[24] = Hsync_Blank-2+42;
```

```
assign letter_pos_y[0] = Vertical_Back_Porch-1+80;
assign letter_pos_y[1] = Vertical_Back_Porch-1+75;
assign letter_pos_y[2] = Vertical_Back_Porch-1+70;
assign letter_pos_y[3] = Vertical_Back_Porch-1+65;
assign letter_pos_y[4] = Vertical_Back_Porch-1+60;
assign letter_pos_y[5] = Vertical_Back_Porch-1+65;
assign letter_pos_y[6] = Vertical_Back_Porch-1+70;
assign letter_pos_y[7] = Vertical_Back_Porch-1+75;
assign letter_pos_y[8] = Vertical_Back_Porch-1+80;

assign letter_pos_y[9] = Vertical_Back_Porch-1+160;
assign letter_pos_y[10] = Vertical_Back_Porch-1+160;
assign letter_pos_y[11] = Vertical_Back_Porch-1+160;
assign letter_pos_y[12] = Vertical_Back_Porch-1+160;

assign letter_pos_y[13] = Vertical_Back_Porch-1+240;
assign letter_pos_y[14] = Vertical_Back_Porch-1+240;
assign letter_pos_y[15] = Vertical_Back_Porch-1+240;
assign letter_pos_y[16] = Vertical_Back_Porch-1+240;
assign letter_pos_y[17] = Vertical_Back_Porch-1+240;
assign letter_pos_y[18] = Vertical_Back_Porch-1+240;

assign letter_pos_y[19] = Vertical_Back_Porch-1+320;
assign letter_pos_y[20] = Vertical_Back_Porch-1+320;
assign letter_pos_y[21] = Vertical_Back_Porch-1+320;
assign letter_pos_y[22] = Vertical_Back_Porch-1+320;

assign letter_pos_y[23] = Vertical_Back_Porch-1+408;
assign letter_pos_y[24] = Vertical_Back_Porch-1+408;

letter letter0(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[0]),
                        .iletter_pos_y(letter_pos_y[0]),
                        .iletter(letter[0]),
                        .oOVERRIDE(override[0]),
                        .oADDR(address[0])
                        );

letter letter1(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[1]),
                        .iletter_pos_y(letter_pos_y[1]),
                        .iletter(letter[1]),
                        .oOVERRIDE(override[1]),
                        .oADDR(address[1])
                        );

letter letter2(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
```

```
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[2]),
                .iletter_pos_y(letter_pos_y[2]),
                .iletter(letter[2]),
                .oOVERRIDE(override[2]),
                .oADDR(address[2])
                );

letter letter3(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[3]),
                .iletter_pos_y(letter_pos_y[3]),
                .iletter(letter[3]),
                .oOVERRIDE(override[3]),
                .oADDR(address[3])
                );

letter letter4(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[4]),
                .iletter_pos_y(letter_pos_y[4]),
                .iletter(letter[4]),
                .oOVERRIDE(override[4]),
                .oADDR(address[4])
                );

letter letter5(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[5]),
                .iletter_pos_y(letter_pos_y[5]),
                .iletter(letter[5]),
                .oOVERRIDE(override[5]),
                .oADDR(address[5])
                );

letter letter6(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[6]),
                .iletter_pos_y(letter_pos_y[6]),
                .iletter(letter[6]),
                .oOVERRIDE(override[6]),
                .oADDR(address[6])
                );

letter letter7(
```

```
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[7]),
                .iletter_pos_y(letter_pos_y[7]),
                .iletter(letter[7]),
                .oOVERRIDE(override[7]),
                .oADDR(address[7])
                );

letter letter8(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[8]),
                .iletter_pos_y(letter_pos_y[8]),
                .iletter(letter[8]),
                .oOVERRIDE(override[8]),
                .oADDR(address[8])
                );

letter letter9(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[9]),
                .iletter_pos_y(letter_pos_y[9]),
                .iletter(letter[9]),
                .oOVERRIDE(override[9]),
                .oADDR(address[9])
                );

letter letter10(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[10]),
                .iletter_pos_y(letter_pos_y[10]),
                .iletter(letter[10]),
                .oOVERRIDE(override[10]),
                .oADDR(address[10])
                );

letter letter11(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[11]),
                .iletter_pos_y(letter_pos_y[11]),
                .iletter(letter[11]),
                .oOVERRIDE(override[11]),
                .oADDR(address[11])
                );
```

```
letter letter12(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[12]),
                .iletter_pos_y(letter_pos_y[12]),
                .iletter(letter[12]),
                .oOVERRIDE(override[12]),
                .oADDR(address[12])
                );

letter letter13(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[13]),
                .iletter_pos_y(letter_pos_y[13]),
                .iletter(letter[13]),
                .oOVERRIDE(override[13]),
                .oADDR(address[13])
                );

letter letter14(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[14]),
                .iletter_pos_y(letter_pos_y[14]),
                .iletter(letter[14]),
                .oOVERRIDE(override[14]),
                .oADDR(address[14])
                );

letter letter15(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[15]),
                .iletter_pos_y(letter_pos_y[15]),
                .iletter(letter[15]),
                .oOVERRIDE(override[15]),
                .oADDR(address[15])
                );

letter letter16(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[16]),
                .iletter_pos_y(letter_pos_y[16]),
                .iletter(letter[16]),
                .oOVERRIDE(override[16]),
```

```
                .oADDR(address[16])
                );

letter letter17(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[17]),
                .iletter_pos_y(letter_pos_y[17]),
                .iletter(letter[17]),
                .oOVERRIDE(override[17]),
                .oADDR(address[17])
                );

letter letter18(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[18]),
                .iletter_pos_y(letter_pos_y[18]),
                .iletter(letter[18]),
                .oOVERRIDE(override[18]),
                .oADDR(address[18])
                );

letter letter19(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[19]),
                .iletter_pos_y(letter_pos_y[19]),
                .iletter(letter[19]),
                .oOVERRIDE(override[19]),
                .oADDR(address[19])
                );

letter letter20(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[20]),
                .iletter_pos_y(letter_pos_y[20]),
                .iletter(letter[20]),
                .oOVERRIDE(override[20]),
                .oADDR(address[20])
                );

letter letter21(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[21]),
                .iletter_pos_y(letter_pos_y[21]),
```

```
                        .iletter(letter[21]),
                        .oOVERRIDE(override[21]),
                        .oADDR(address[21])
                        );

letter letter22(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[22]),
                        .iletter_pos_y(letter_pos_y[22]),
                        .iletter(letter[22]),
                        .oOVERRIDE(override[22]),
                        .oADDR(address[22])
                        );


letter letter23(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[23]),
                        .iletter_pos_y(letter_pos_y[23]),
                        .iletter(letter[23]),
                        .oOVERRIDE(override[23]),
                        .oADDR(address[23])
                        );

letter letter24(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[24]),
                        .iletter_pos_y(letter_pos_y[24]),
                        .iletter(letter[24]),
                        .oOVERRIDE(override[24]),
                        .oADDR(address[24])
                        );

symbol indicator(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .isymbol_pos_x(symbol_pos_x),
                        .isymbol_pos_y(symbol_pos_y),
                        .iBASE_ADDR(indicator_base),
                        .oOVERRIDE(override[25]),
                        .oADDR(address[25])
                        );

always@(posedge iclk or negedge ireset_n)
begin
    if(!ireset_n)
    begin
```

```
            symbol_pos_x <= Hsync_Blank-2+528;
            symbol_pos_y <= Vertical_Back_Porch-1+144;
        end
        else if(state == 4'b0000)
        begin
            symbol_pos_x <= Hsync_Blank-2+528;
            symbol_pos_y <= Vertical_Back_Porch-1+144;
        end
        else if(state == 4'b0001)
        begin
            symbol_pos_x <= Hsync_Blank-2+528;
            symbol_pos_y <= Vertical_Back_Porch-1+224;
        end
        else if(state == 4'b0010)
        begin
            symbol_pos_x <= Hsync_Blank-2+528;
            symbol_pos_y <= Vertical_Back_Porch-1+304;
        end
end

always@(posedge iclk or negedge ireset_n)
begin
        if(!ireset_n)
            oADDR <= 18'bZ;
        else
            case(override)
            26'b00000000000000000000000001 : oADDR <= address[0];
            26'b00000000000000000000000010 : oADDR <= address[1];
            26'b00000000000000000000000100 : oADDR <= address[2];
            26'b00000000000000000000001000 : oADDR <= address[3];
            26'b00000000000000000000010000 : oADDR <= address[4];
            26'b00000000000000000000100000 : oADDR <= address[5];
            26'b00000000000000000001000000 : oADDR <= address[6];
            26'b00000000000000000010000000 : oADDR <= address[7];
            26'b00000000000000000100000000 : oADDR <= address[8];
            26'b00000000000000001000000000 : oADDR <= address[9];
            26'b00000000000000010000000000 : oADDR <= address[10];
            26'b00000000000000100000000000 : oADDR <= address[11];
            26'b00000000000001000000000000 : oADDR <= address[12];
            26'b00000000000010000000000000 : oADDR <= address[13];
            26'b00000000000100000000000000 : oADDR <= address[14];
            26'b00000000001000000000000000 : oADDR <= address[15];
            26'b00000000010000000000000000 : oADDR <= address[16];
            26'b00000000100000000000000000 : oADDR <= address[17];
            26'b00000001000000000000000000 : oADDR <= address[18];
            26'b00000010000000000000000000 : oADDR <= address[19];
            26'b00000100000000000000000000 : oADDR <= address[20];
            26'b00001000000000000000000000 : oADDR <= address[21];
            26'b00010000000000000000000000 : oADDR <= address[22];
            26'b00100000000000000000000000 : oADDR <= address[23];
            26'b01000000000000000000000000 : oADDR <= address[24];
            26'b10000000000000000000000000 : oADDR <= address[25];
            default: oADDR <= 18'd89025;
            endcase;
end

endmodule
```

win.v

```
module win(
                        iclk,
                        ireset_n,
                        ix_cnt,
                        iy_cnt,
                        oADDR);

//================================================
//Parameter=============================
//================================================
parameter A = 5'd1;
parameter B = 5'd2;
parameter C = 5'd3;
parameter D = 5'd4;
parameter E = 5'd5;
parameter F = 5'd6;
parameter G = 5'd7;
parameter H = 5'd8;
parameter I = 5'd9;
parameter J = 5'd10;
parameter K = 5'd11;
parameter L = 5'd12;
parameter M = 5'd13;
parameter N = 5'd14;
parameter O = 5'd15;
parameter P = 5'd16;
parameter Q = 5'd17;
parameter R = 5'd18;
parameter S = 5'd19;
parameter T = 5'd20;
parameter U = 5'd21;
parameter V = 5'd22;
parameter W = 5'd23;
parameter X = 5'd24;
parameter Y = 5'd25;
parameter Z = 5'd26;

parameter H_LINE = 1056;
parameter V_LINE = 525;
parameter Hsync_Blank = 216;
parameter Hsync_Front_Porch = 40;
parameter Vertical_Back_Porch = 35;
parameter Vertical_Front_Porch = 10;
//================================================
//I/O DECALARATION=============================
//================================================
input iclk;
input ireset_n;
input [10:0] ix_cnt;
input [9:0] iy_cnt;
output [17:0] oADDR;

//================================================
//WIRE & REG=============================
//================================================
wire [4:0] letter[0:20];
```

```
wire [10:0] letter_pos_x[0:20];
wire [9:0] letter_pos_y[0:20];
wire [20:0] override;
wire [17:0] address[0:20];
reg [17:0] oADDR;
//============================================
//Structured logic==============================
//============================================
//Assign Letters
assign letter[0] = Y;
assign letter[1] = O;
assign letter[2] = U;
assign letter[3] = W;
assign letter[4] = I;
assign letter[5] = N;
assign letter[6] = C;
assign letter[7] = O;
assign letter[8] = N;
assign letter[9] = G;
assign letter[10] = R;
assign letter[11] = A;
assign letter[12] = D;
assign letter[13] = U;
assign letter[14] = L;
assign letter[15] = A;
assign letter[16] = T;
assign letter[17] = I;
assign letter[18] = O;
assign letter[19] = N;
assign letter[20] = S;


//assign letter position
assign letter_pos_x[0] = Hsync_Blank-2+480;
assign letter_pos_x[1] = Hsync_Blank-2+448;
assign letter_pos_x[2] = Hsync_Blank-2+416;
assign letter_pos_x[3] = Hsync_Blank-2+352;
assign letter_pos_x[4] = Hsync_Blank-2+320;
assign letter_pos_x[5] = Hsync_Blank-2+288;

assign letter_pos_x[6] = Hsync_Blank-2+608;
assign letter_pos_x[7] = Hsync_Blank-2+576;
assign letter_pos_x[8] = Hsync_Blank-2+544;
assign letter_pos_x[9] = Hsync_Blank-2+512;
assign letter_pos_x[10] = Hsync_Blank-2+480;
assign letter_pos_x[11] = Hsync_Blank-2+448;
assign letter_pos_x[12] = Hsync_Blank-2+416;
assign letter_pos_x[13] = Hsync_Blank-2+384;
assign letter_pos_x[14] = Hsync_Blank-2+352;
assign letter_pos_x[15] = Hsync_Blank-2+320;
assign letter_pos_x[16] = Hsync_Blank-2+288;
assign letter_pos_x[17] = Hsync_Blank-2+256;
assign letter_pos_x[18] = Hsync_Blank-2+224;
assign letter_pos_x[19] = Hsync_Blank-2+192;
assign letter_pos_x[20] = Hsync_Blank-2+160;


assign letter_pos_y[0] = Vertical_Back_Porch-1+160;
```

```
assign letter_pos_y[1] = Vertical_Back_Porch-1+160;
assign letter_pos_y[2] = Vertical_Back_Porch-1+160;
assign letter_pos_y[3] = Vertical_Back_Porch-1+160;
assign letter_pos_y[4] = Vertical_Back_Porch-1+160;
assign letter_pos_y[5] = Vertical_Back_Porch-1+160;

assign letter_pos_y[6] = Vertical_Back_Porch-1+240;
assign letter_pos_y[7] = Vertical_Back_Porch-1+240;
assign letter_pos_y[8] = Vertical_Back_Porch-1+240;
assign letter_pos_y[9] = Vertical_Back_Porch-1+240;
assign letter_pos_y[10] = Vertical_Back_Porch-1+240;
assign letter_pos_y[11] = Vertical_Back_Porch-1+240;
assign letter_pos_y[12] = Vertical_Back_Porch-1+240;
assign letter_pos_y[13] = Vertical_Back_Porch-1+240;
assign letter_pos_y[14] = Vertical_Back_Porch-1+240;
assign letter_pos_y[15] = Vertical_Back_Porch-1+240;
assign letter_pos_y[16] = Vertical_Back_Porch-1+240;
assign letter_pos_y[17] = Vertical_Back_Porch-1+240;
assign letter_pos_y[18] = Vertical_Back_Porch-1+240;
assign letter_pos_y[19] = Vertical_Back_Porch-1+240;
assign letter_pos_y[20] = Vertical_Back_Porch-1+240;


letter letter0(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[0]),
                        .iletter_pos_y(letter_pos_y[0]),
                        .iletter(letter[0]),
                        .oOVERRIDE(override[0]),
                        .oADDR(address[0])
                        );

letter letter1(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[1]),
                        .iletter_pos_y(letter_pos_y[1]),
                        .iletter(letter[1]),
                        .oOVERRIDE(override[1]),
                        .oADDR(address[1])
                        );

letter letter2(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[2]),
                        .iletter_pos_y(letter_pos_y[2]),
                        .iletter(letter[2]),
                        .oOVERRIDE(override[2]),
                        .oADDR(address[2])
                        );
```

```
letter letter3(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[3]),
                .iletter_pos_y(letter_pos_y[3]),
                .iletter(letter[3]),
                .oOVERRIDE(override[3]),
                .oADDR(address[3])
                );

letter letter4(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[4]),
                .iletter_pos_y(letter_pos_y[4]),
                .iletter(letter[4]),
                .oOVERRIDE(override[4]),
                .oADDR(address[4])
                );

letter letter5(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[5]),
                .iletter_pos_y(letter_pos_y[5]),
                .iletter(letter[5]),
                .oOVERRIDE(override[5]),
                .oADDR(address[5])
                );

letter letter6(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[6]),
                .iletter_pos_y(letter_pos_y[6]),
                .iletter(letter[6]),
                .oOVERRIDE(override[6]),
                .oADDR(address[6])
                );

letter letter7(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[7]),
                .iletter_pos_y(letter_pos_y[7]),
                .iletter(letter[7]),
                .oOVERRIDE(override[7]),
```

```
                .oADDR(address[7])
                );

letter letter8(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[8]),
                .iletter_pos_y(letter_pos_y[8]),
                .iletter(letter[8]),
                .oOVERRIDE(override[8]),
                .oADDR(address[8])
                );

letter letter9(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[9]),
                .iletter_pos_y(letter_pos_y[9]),
                .iletter(letter[9]),
                .oOVERRIDE(override[9]),
                .oADDR(address[9])
                );

letter letter10(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[10]),
                .iletter_pos_y(letter_pos_y[10]),
                .iletter(letter[10]),
                .oOVERRIDE(override[10]),
                .oADDR(address[10])
                );

letter letter11(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[11]),
                .iletter_pos_y(letter_pos_y[11]),
                .iletter(letter[11]),
                .oOVERRIDE(override[11]),
                .oADDR(address[11])
                );

letter letter12(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[12]),
                .iletter_pos_y(letter_pos_y[12]),
```

```
                .iletter(letter[12]),
                .oOVERRIDE(override[12]),
                .oADDR(address[12])
                );

letter letter13(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[13]),
                .iletter_pos_y(letter_pos_y[13]),
                .iletter(letter[13]),
                .oOVERRIDE(override[13]),
                .oADDR(address[13])
                );

letter letter14(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[14]),
                .iletter_pos_y(letter_pos_y[14]),
                .iletter(letter[14]),
                .oOVERRIDE(override[14]),
                .oADDR(address[14])
                );

letter letter15(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[15]),
                .iletter_pos_y(letter_pos_y[15]),
                .iletter(letter[15]),
                .oOVERRIDE(override[15]),
                .oADDR(address[15])
                );

letter letter16(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
                .iletter_pos_x(letter_pos_x[16]),
                .iletter_pos_y(letter_pos_y[16]),
                .iletter(letter[16]),
                .oOVERRIDE(override[16]),
                .oADDR(address[16])
                );

letter letter17(
                .iclk(iclk),
                .ireset_n(ireset_n),
                .ix_cnt(ix_cnt),
                .iy_cnt(iy_cnt),
```

```
                        .iletter_pos_x(letter_pos_x[17]),
                        .iletter_pos_y(letter_pos_y[17]),
                        .iletter(letter[17]),
                        .oOVERRIDE(override[17]),
                        .oADDR(address[17])
                        );

letter letter18(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[18]),
                        .iletter_pos_y(letter_pos_y[18]),
                        .iletter(letter[18]),
                        .oOVERRIDE(override[18]),
                        .oADDR(address[18])
                        );

letter letter19(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[19]),
                        .iletter_pos_y(letter_pos_y[19]),
                        .iletter(letter[19]),
                        .oOVERRIDE(override[19]),
                        .oADDR(address[19])
                        );

letter letter20(
                        .iclk(iclk),
                        .ireset_n(ireset_n),
                        .ix_cnt(ix_cnt),
                        .iy_cnt(iy_cnt),
                        .iletter_pos_x(letter_pos_x[20]),
                        .iletter_pos_y(letter_pos_y[20]),
                        .iletter(letter[20]),
                        .oOVERRIDE(override[20]),
                        .oADDR(address[20])
                        );


always@(posedge iclk or negedge ireset_n)
begin
      if(!ireset_n)
            oADDR <= 18'bZ;
      else
            case(override)
            21'b000000000000000000001 : oADDR <= address[0];
            21'b000000000000000000010 : oADDR <= address[1];
            21'b000000000000000000100 : oADDR <= address[2];
            21'b000000000000000001000 : oADDR <= address[3];
            21'b000000000000000010000 : oADDR <= address[4];
            21'b000000000000000100000 : oADDR <= address[5];
            21'b000000000000001000000 : oADDR <= address[6];
            21'b000000000000010000000 : oADDR <= address[7];
```

```
        21'b000000000000100000000 : oADDR <= address[8];
        21'b000000000001000000000 : oADDR <= address[9];
        21'b000000000010000000000 : oADDR <= address[10];
        21'b000000000100000000000 : oADDR <= address[11];
        21'b000000001000000000000 : oADDR <= address[12];
        21'b000000010000000000000 : oADDR <= address[13];
        21'b000000100000000000000 : oADDR <= address[14];
        21'b000001000000000000000 : oADDR <= address[15];
        21'b000010000000000000000 : oADDR <= address[16];
        21'b000100000000000000000 : oADDR <= address[17];
        21'b001000000000000000000 : oADDR <= address[18];
        21'b010000000000000000000 : oADDR <= address[19];
        21'b100000000000000000000 : oADDR <= address[20];
        default: oADDR <= 18'd89025;
        endcase;
end

endmodule
```

xycoord_interface.v

```
module xycoord_interface    (
                            clk,
                            reset_n,
                            read,
                            write,
                            chipselect,
                            address,
                            readdata,
                            writedata,
                        iX_COORD,
                        iY_COORD
                        ////////////////////
                        );


//=======================================================================
// PORT declarations
//=======================================================================
input       clk;     // system clock 50Mhz
input reset_n;// system reset
input       read;
input write;
input chipselect;
input [4:0]address;
output [15:0]readdata;
input [15:0]writedata;
input       [11:0]      iX_COORD;
input       [11:0]      iY_COORD;

//=======================================================================
// REG/WIRE declarations
//=======================================================================
reg         [11:0]      oX_COORD;
reg         [11:0]      oY_COORD;
reg [15:0] readdata_buff;
```

```
//=============================================================================
// Structural coding
//=============================================================================

assign readdata = readdata_buff;
//write the x y coordinates to the avalon bus
always@(posedge clk or negedge reset_n)
	begin
		if(!reset_n)
			readdata_buff <= 12'b0;
		else if (chipselect && read)
			begin
					if(address == 5'b00000)
						readdata_buff <= iX_COORD;
					else if (address == 5'b00001)
						readdata_buff <= iY_COORD;
			end
	end


endmodule
```

functions.h

```
#include <stdio.h>
#include <system.h>
#include <io.h>
#include "background.h"
#include "letters.h"
#include "music.h"

//read x y coordinates, offset = 0 for x and offset = 1 for y
#define IORD_XY_DATA(base,offset) \
	IORD_16DIRECT(base,(offset)*2)
#define IOWR_SRAM_DATA(base,offset,data)\
	IOWR_16DIRECT(base,(offset)*2,data)
#define IORD_SRAM_DATA(base,offset)\
	IORD_16DIRECT(base,(offset)*2)

	// offset = 0 control panel 32bit
	// offset = 1 control mole 32bit
#define IOWR_CONTROL(base,offset,data) \
	IOWR_32DIRECT(base,(offset)*4,data)

//Initialization
void init()
{
	int base = 0;
	int i;
	int data = 0;

   for(i=0;i<32000;i++)
   {
	   data = background[i];
	   IOWR_SRAM_DATA(SRAM_MUX_0_BASE,i,data);
```

```
   } //shifted by 64000

  // printf("parameter control_panel_base = %d\n",base);
   for(i=0; i< 16000;i++)
   {
         data = control_panel[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+64000,i,data); //shift 32000 from the base

   }

  // printf("parameter mole_shape_base = %d\n",base);
   for(i=0; i< 14400;i++)
   {
         data = mole_shape[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+96000,i,data);  //shift  48000  16bit  from  the
base
   }

//writing 26 letters into SRAM
   for(i=0; i<1024;i++)
   {
         data = A[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+124800,i,data);  //shift  48000  16bit  from  the
base
   }

   for(i=0; i<1024;i++)
   {
         data = B[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+126848,i,data);  //shift  48000  16bit  from  the
base
   }

   for(i=0; i<1024;i++)
   {
         data = C[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+128896,i,data);  //shift  48000  16bit  from  the
base
         base = base +1;
   }
   for(i=0; i<1024;i++)
   {
         data = D[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+130944,i,data);  //shift  48000  16bit  from  the
base
   }
   for(i=0; i<1024;i++)
   {
         data = E[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+132992,i,data);  //shift  48000  16bit  from  the
base
   }
   for(i=0; i<1024;i++)
   {
         data = F[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+135040,i,data);  //shift  48000  16bit  from  the
base
   }
```

```
  for(i=0; i<1024;i++)
  {
        data = G[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+137088,i,data); //shift 48000 16bit from the
base
  }
  for(i=0; i<1024;i++)
  {
        data = H[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+139136,i,data); //shift 48000 16bit from the
base
  }
  for(i=0; i<1024;i++)
  {
        data = I[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+141184,i,data); //shift 48000 16bit from the
base
  }
  for(i=0; i<1024;i++)
  {
        data = J[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+143232,i,data); //shift 48000 16bit from the
base
  }
  for(i=0; i<1024;i++)
  {
        data = K[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+145280,i,data); //shift 48000 16bit from the
base
  }
  for(i=0; i<1024;i++)
  {
        data = L[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+147328,i,data); //shift 48000 16bit from the
base
  }
  for(i=0; i<1024;i++)
  {
        data = M[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+149376,i,data); //shift 48000 16bit from the
base
  }
  for(i=0; i<1024;i++)
  {
        data = N[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+151424,i,data); //shift 48000 16bit from the
base
  }
  for(i=0; i<1024;i++)
  {
        data = O[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+153472,i,data); //shift 48000 16bit from the
base
  }
  for(i=0; i<1024;i++)
  {
        data = P[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+155520,i,data); //shift 48000 16bit from the
```

```
base
   }
   for(i=0; i<1024;i++)
   {
         data = Q[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+157568,i,data); //shift 48000 16bit from the
base
   }
   for(i=0; i<1024;i++)
   {
         data = R[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+159616,i,data); //shift 48000 16bit from the
base
   }
   for(i=0; i<1024;i++)
   {
         data = S[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+161664,i,data); //shift 48000 16bit from the
base
   }
   for(i=0; i<1024;i++)
   {
         data = T[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+163712,i,data); //shift 48000 16bit from the
base
   }
   for(i=0; i<1024;i++)
   {
         data = U[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+165760,i,data); //shift 48000 16bit from the
base
   }
   for(i=0; i<1024;i++)
   {
         data = V[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+167808,i,data); //shift 48000 16bit from the
base
   }
   for(i=0; i<1024;i++)
   {
         data = W[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+169856,i,data); //shift 48000 16bit from the
base
   }
   for(i=0; i<1024;i++)
   {
         data = X[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+171904,i,data); //shift 48000 16bit from the
base
   }
   for(i=0; i<1024;i++)
   {
         data = Y[i];
         IOWR_SRAM_DATA(SRAM_MUX_0_BASE+173952,i,data); //shift 48000 16bit from the
base
   }
   for(i=0; i<1024;i++)
   {
```

```
        data = Z[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+176000,i,data); //shift 48000 16bit from the
base
    }
    //write black hole
    for(i=0; i<15;i++)
    {
        data = xcolor[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+178048,i,data); //shift 48000 16bit from the
base
    }

    for(i=0;i<3600;i++)
    {
        data = clock1[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+178078,i,data); //shift 48000 16bit from the
base
    }

    for(i=0;i<3600;i++)
    {
        data = heart[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+185278,i,data); //shift 48000 16bit from the
base
    }

    for(i=0;i<3600;i++)
    {
        data = coin[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+192478,i,data); //shift 48000 16bit from the
base
    }

    for(i=0;i<3600;i++)
    {
        data = indicator[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+199678,i,data);
    }

    for(i=0;i<14400;i++)
    {
        data = bottle1[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+206878,i,data);
    }

    for(i=0;i<14400;i++)
    {
        data = evil[i];
        IOWR_SRAM_DATA(SRAM_MUX_0_BASE+235678,i,data);
    }
}
```

main.c

```
#include <stdio.h>
#include <system.h>
#include <io.h>
#include "functions.h"
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <sys/alt_irq.h> // the irq functions
#include <sys/types.h>

#define random(x) (rand()%x)

int i=0;
int j;
int m;
int n;
int time_interval=0;
unsigned data_temp;
int state = 0;
int score0 = 0;
int score1 = 0;
int score2 = 0;
int game_time=0;
int life = 3;
int moleP=0;
int showtime=0;
int counter = 0;
int counter_previous = 0;
int random_number=0;
long int panel_command=0;
long int mole_command=0;
int mole_status=0;
int level_mode = 0;
int mole_command1=0;
int mole_command2=0;
int mole_command3=0;
int item=0;
int animal2=0;
int whichmole;
clock_t start_t;
clock_t end_t;
clock_t total_t;




int touch(int state_in){
	int x;
	int y;
	x = IORD_XY_DATA(XYCOORD_INTERFACE_0_BASE,0);
	y = IORD_XY_DATA(XYCOORD_INTERFACE_0_BASE,1);
	if(state_in == 0x4 || state_in == 0x5 || state_in == 0x6 || state_in == 0x7 )
	{
		if(x>0xA10 && x< 0xED0 && y> 0x430 && y< 0x7F0)
			return 0;//mole0
		else if(x>0xA10 && x< 0xED0 && y>0x7F1 && y< 0xBB0)
```

```
                return 1;
        else if(x>0xA10 && x< 0xED0 && y>0xBB0 && y< 0xF40)
                return 2;
        else if(x>0x530 && x< 0xA10 && y> 0x430 && y< 0x7F0)
                return 3;
        else if(x>0x530 && x< 0xA10 && y>0x7F1 && y< 0xBB0)
                return 4;
        else if(x>0x530 && x< 0xA10 && y>0xBB0 && y< 0xF40)
                return 5;
        else if(x>0x0C0 && x< 0x530 && y> 0x430 && y< 0x7F0)
                return 6;
        else if(x>0x0C0 && x< 0x530 && y>0x7F1 && y< 0xBB0)
                return 7;
        else if(x>0x0C0 && x< 0x530 && y>0xBB0 && y< 0xF40)
                return 8;//mole8
        else if(x>0x530 && x<0x790 && y>0x0 && y<0x430)
                return 9; //start key
        else if(x>0x2F0 && x< 0x530 && y>0x0 && y<0x430)
                return 0xA;//pause key
        else if(x>0x0 && x< 0x2F0 && y>0x0 && y<0x430)
                return 0xB; //quit key
        else
                return 0xC;
    }

    else if(state_in == 0x0 || state_in == 0x1 || state_in == 0x2 || state_in == 0x3)
    {
        if(x>0x810 && x< 0xA90 && y> 0x580 && y< 0xA10)
                return 0;// easy
        else if(x>0x610 && x< 0x810 && y> 0x580 && y< 0xA10)
                return 1; // normal
        else if(x>0x370 && x< 0x610 && y> 0x580 && y< 0xA10)
                return 2; // hard
        else if(x>0x0 && x< 0x350 && y> 0xCD0 && y< 0xF98){
                music = 1;
                return 3;} //go
        else
                return 4;
    }
}



int mole_position(int num,int status){

    switch(num%3){
        case 0: item=0;    // mole
                break;
        case 1: item=1;    //bomb
                break;
        case 2: item=2; //bottle
                break;
    }

    while (num%3!=0){
        num--;
    }
    int i=num/3;
```

```
      whichmole = i;
    mole_command = item*pow(8,i)+status*pow(8,i)*4;
    return mole_command;
}



//Music interrupt function
short audio_to_play;
short *music_to_play;
short music_to_play_temp;
short sound_to_play_temp;
short *remember_music_to_play;
short *sound_to_play;
int music = 0;
int remember_music;
int remember_musiclength;
int music_length;
int sound = 0;
int sound_length;
int a,b,c,d,e;
int aa,bb,cc,dd,ee,ff;
int pause = 0;
int pre_pause = 0;

static void standbymusic(){
//    switch(music){
//    case 0: music_length = 368139; break;
//    case 1:    music_length = 78912;    break;
//    case 2:    music_length = 368140; break;
//    case 3:    music_length = 80640;   break;
//    case 4:    music_length = 87552;   break;
//    default:   break;
//    }
      if (b != 1 && d != 1 && e != 1){
      switch(music){
                  case 0: music_length = 368139; if(a==0){i=0;} a=1; b=0 ;c=0;d=0;e = 0;
                                                              music_to_play       =       standby;
break;
                  case 1:    music_length = 78912;    if(b==0){i=0;} b=1; a=0;c=0;d=0;e = 0;
                                                              music_to_play       =       lady_go;
break;
                  case 2:    music_length = 368140; if(c==0){i=0;} c=1; a=0;b=0;d=0;e = 0;
                                                             music_to_play = background_music;
break;
                  case 3:    music_length = 67026;    if(d==0){i=0;} d=1; a=0;b=0;c=0;e = 0;
                                                              music_to_play = victory;       break;
                  case 4:    music_length = 63295;    if(e==0){i=0;} e=1; a=0;b=0;c=0;d = 0;
                                                              music_to_play       =       defeat;
break;
//                          case 5: music_length = 5790;    if(aa==0) m=1; aa=1; bb=0; dd=0;
ee=0; ff = 0;
//
music_to_play = hit;        break;
//                          case 6:    music_length    =    14008;        if(bb==0){m=1;}    bb=1;
aa=0;cc=0;dd=0;ee=0;ff = 0;
//
music_to_play = gold;       break;
```

```
//                              case 7:   music_length    =    13540;    if(cc==0){m=1;}    cc=1;
aa=0;bb=0;dd=0;ee=0;ff = 0;
//
music_to_play = explo;     break;
//                              case 8:   music_length    =    6571;     if(dd==0){m=1;}    dd=1;
aa=0;bb=0;cc=0;ee=0;ff = 0;
//
music_to_play = miss;      break;
//                              case 9:   music_length    =    1961;     if(ee==0){m=1;}    ee=1;
aa=0;bb=0;cc=0;dd=0;ff = 0;
//
music_to_play = button_chose; break;
//                              case 10:  music_length    =    1128;      if(ff==0){m=1;}    ff=1;
aa=0;bb=0;cc=0;dd=0;ee = 0;
//
music_to_play = button_wood;   break;
                        default:   break;
                        }
      }
                              switch(sound){
                                          case 0: sound_length = 5790;
                                                      if(aa==0) m=0;
                                                      aa=1;
                                                      bb=0;
                                                      cc=0;
                                                      dd=0;
                                                      ee=0;
                                                      ff = 0;
                                                      sound_to_play = hit;
                                                      break;
                                          case 1:    sound_length = 14008;    if(bb==0){m=0;} bb=1;
aa=0;cc=0;dd=0;ee=0;ff = 0;
                                                                                  sound_to_play
= gold;       break;
                                          case 2:    sound_length  =  13540;  if(cc==0){m=0;}  cc=1;
aa=0;bb=0;dd=0;ee=0;ff = 0;
                                                                                  sound_to_play
= explo;      break;
                                          case 3:    sound_length  =  6571;   if(dd==0){m=0;}  dd=1;
aa=0;bb=0;cc=0;ee=0;ff = 0;
                                                                                  sound_to_play
= miss;       break;
                                          case 4:    sound_length  =  1961;   if(ee==0){m=0;}  ee=1;
aa=0;bb=0;cc=0;dd=0;ff = 0;
                                                                                  sound_to_play
= button_chose; break;
                                          case 5:    sound_length  =  1128;    if(ff==0){m=0;}  ff=1;
aa=0;bb=0;cc=0;dd=0;ee = 0;
                                                                                  sound_to_play
= button_wood;    break;
                                          default: aa=0;bb=0;cc=0;dd=0;ee=0;ff = 0; m = 0;    break;
                                          }



      for(j=0;j<2048;j++)
```

```
            {

              if (pause == 1){audio_to_play=0;}
//              else if (m!=0){audio_to_play = music_to_play[m];}
//              else audio_to_play = music_to_play[i];
              else if (sound !=6) {
                    music_to_play_temp = music_to_play[i];
                    sound_to_play_temp = sound_to_play[m];
                    audio_to_play = (music_to_play_temp >> 2) + ((sound_to_play_temp>>2)*3);
              }
              else audio_to_play = music_to_play[i];

              IOWR_16DIRECT(MUSICCONTROLLER_0_BASE,j*2,audio_to_play);
                              i++;
                              m++;

              if(i == music_length){ i = 0;
                 if (music == 1) {music = 2; b = 0;}
                 else if (music == 3 || music == 4) {state = 0;d = 0; e = 0;music=0;}

              }
                 //       if(i == 368139){ i = 0;}
              if(m == sound_length){
                    if (pre_pause == 1){pause = 1;}
                    m = 0; sound = 6;}
            }

}


void check_mole(){


      if (random_number==0 || random_number==1 || random_number==2){
            moleP=0;
      }else if (random_number==3 || random_number==4 || random_number==5){
            moleP=1;
      }else if (random_number==6 || random_number==7 || random_number==8){
            moleP=2;
      }else if (random_number==9 || random_number==10 || random_number==11){
            moleP=3;
      }else if (random_number==12 || random_number==13 || random_number==14){
            moleP=4;
      }else if (random_number==15 || random_number==16 || random_number==17){
            moleP=5;
      }else if (random_number==18 || random_number==19 || random_number==20){
            moleP=6;
      }else if (random_number==21 || random_number==22 || random_number==23){
            moleP=7;
      }else if (random_number==24 || random_number==25 || random_number==26){
            moleP=8;
      }
}

void state_sel(int state_in)
{
     check_mole();
     switch(state_in)
```

```
{
        //Welcome-Easy
    case 0x0:
        if(touch(state_in) != 4)
        {
            if(touch(state_in)== 0)
            {
                state = 0x0; //easy
                sound = 4;
            }
            else if(touch(state_in) == 1)
            {
                state = 0x1; // normal
               music = 0;
                sound = 4;
            }
            else if(touch(state_in) == 2)
            {
                state = 0x2; // hard
              music = 0;
                sound = 4;
            }
            else if(touch(state_in)==3){
                state = 0x4;
                level_mode=0;
                music = 2;
                sound = 4;
            }
        }

        break;
        //WELCOME-NORMAL
    case 0x1:
        if(touch(state_in) != 4)
        {
            if(touch(state_in)== 0)
            {
                state = 0;
                music = 0;
                sound = 4;
            }
            else if(touch(state_in) == 1)
            {
                state = 0x1;
                music =0;
                sound = 4;
            }
            else if(touch(state_in) == 2)
            {
                state = 0x2;
                music = 0;
                sound = 4;
            }
            else if(touch(state_in)==3){
                state = 0x4;
                level_mode=1;
                music = 2;
                sound = 4;
```

```
            }
        }
        break;
        //WELCOME-HARD
    case 0x2:
        if(touch(state_in) != 4)
        {
            if(touch(state_in)== 0)
            {
                state = 0;
                music = 0;
                sound = 4;
            }
            else if(touch(state_in) == 1)
            {
                state = 0x1;
                music = 0;
                sound = 4;
            }
            else if(touch(state_in) == 2)
            {
                state = 0x2;
                music = 0;
                sound = 4;
            }
            else if(touch(state_in)==3){
                state = 0x4;
                level_mode=2;
                music = 2;
                sound = 4;
            }
        }
        break;
        //unused state
    case 0x3:
        break;

        //PLAY-INIT
    case 0x4:
        if(touch(state_in) == 0xB)
        {
            state = 0x0;
            music = 0;
            sound = 5;
        }
        else if(touch(state_in) == 0x9)
        {
            state = 0x5;
            music = 2;
            sound =5;

        }
        break;

        //PLAY- ONGOING
    case 0x5:
        if(touch(state_in) == 0xB)
        {
```

```
            state = 0x0;
            sound =5;
            music =0;
        }
        else if(touch(state_in) == 0xA)
        {
            state = 0x6;
            sound =5;
            music = 5;
            pause = 1;
        }
        else if(touch(state_in) == moleP){
            mole_status = 0;
            if(item == 0)
                sound = 0;
            else if(item == 1)
                sound = 2;
            else
                sound = 1;
        }
        else
            sound = 3;
        break;

        //PLAY- PAUSE
    case 0x6:
        if(touch(state_in) == 0xB)
        {
            state = 0x0;
            sound =5;
            pause = 0;
            music = 2;
        }
        else if(touch(state_in) == 0x9)
        {
            state = 0x5;
            sound =5;
            music = 0;
        }
        break;

    case 0x7:
        break;
    case 0x8:
        music = 4;
        break;
    case 0x9:
        break;
    case 0xA:
        break;
    case 0xB:
        break;
    case 0xC:
        music = 3;
        break;
    case 0xD:
        break;
    case 0xE:
```

```
                break;
        case 0xF:
                break;
    }
}

typedef enum {True, False} bool;

static void statedetect()
{
    state_sel(state);
    panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
    IOWR_CONTROL(INTERRUPT_0_BASE,0,0);
    IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
}


int main (int argc, const char * argv[])
{



    //Music interrupt
    alt_irq_register(MUSICCONTROLLER_0_IRQ, NULL,(void*)standbymusic);
    //touch interrupt
    alt_irq_register(INTERRUPT_0_IRQ, NULL,(void*)statedetect);

    init();

    // initialization
    IOWR_CONTROL(COMMAND_0_BASE,0,0x30);
    IOWR_CONTROL(COMMAND_0_BASE,1,0);

    while (1){
    LOOP: if (state==0x0 || state==0x1 || state==0x2){
        life=3;
        score0=0;
        score1=0;
        score2=0;
        panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
        IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
        IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
    }

    if (state==0x4){
        if (level_mode==0)
        {
            while(state == 0x4 || state == 0x5 || state ==0x6)
            {
            if (state==0x5)
            {
//              start_t=clock();
//              printf("start time: %d \n", start_t/CLOCKS_PER_SEC);
                while(1 && game_time <= 10000000 ){
                    random_number = rand()%27;
```

```
                mole_status=1;
                mole_command = mole_position( random_number,mole_status);
                panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;



                IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);


                while (mole_status==1 && counter<100000 ){
                    counter++;
                    game_time++;

                    //pause
                    if (state==6){
                    while (state==6){
                        panel_command = state+ life*16 + score0*16*16 +
score1*16*16*16 + score2*16*16*16*16;
                        IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                        IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                    }

                    }
                    else if (state==0x0){

                        IOWR_CONTROL(COMMAND_0_BASE,1,0);
                        IOWR_CONTROL(COMMAND_0_BASE,0,0);
                        goto LOOP;

                    }
                }

                //end_t=clock();
                //printf("end time: %d \n", end_t/CLOCKS_PER_SEC);

            //  total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
                //printf("time interval: %f\n",total_t);
                //12000000

                if (counter==100000 ){
                    if (item==0){
                    life--;
                    }
                     if (life<=0){
                        state=0xC;
                        music = 4;
                        panel_command = state+ life*16 + score0*16*16 +
score1*16*16*16 + score2*16*16*16*16;
                        IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                        IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                        while (showtime<500000){
                            showtime++;
                        }
                        showtime=0;
                        state=0;
                        IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
```

```
                    IOWR_CONTROL(COMMAND_0_BASE,0,0x0);
                    goto LOOP;

                }else{
                    mole_status=0;
                    panel_command = state+ life*16 + score0*16*16 +
score1*16*16*16 + score2*16*16*16*16;
                    mole_command =
mole_position(random_number,mole_status);
                    IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                    IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                }
            }else{
                if(item == 0){
                    if(score0 == 0){
                        score0= score0+5;
                    }else if(score1 < 9 ){
                                score0 =0;
                                score1++;
                            }else{
                                    score2++;
                                    score1 = 0;
                                    score0 = 0;
                                }
                    panel_command = state+ life*16 + score0*16*16 +
score1*16*16*16 + score2*16*16*16*16;
                    mole_command = mole_position(random_number, mole_status);
                    IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                    IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                    item=0;

                }else if(item == 2) {
                                    if (life!=3){
                                        life = life +1;//life limit
                                    }

                }else if(item == 1){

                    if(life>0){
                        life = life -1;
                    }else{
                        state=0xC;
                        music = 4;
                        panel_command = state+ life*16 + score0*16*16 +
score1*16*16*16 + score2*16*16*16*16;
                        IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                        IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                        while (showtime<500000){
                            showtime++;
                        }
                        showtime=0;
                        state=0;
                        IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                        IOWR_CONTROL(COMMAND_0_BASE,0,0x0);
                        goto LOOP;
                    }
                }
                 panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
```

```
score2*16*16*16*16;
                        mole_command = mole_position(random_number, mole_status);
                        IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                        IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                        item=0;

                    }

                counter = 0;

                }
                 state=8;
                 music = 3;
                 game_time=0;
                 panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
                 IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                 IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                 while (showtime<500000){
                       showtime++;
                 }
                 showtime=0;
                 state=0;
                 IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                 IOWR_CONTROL(COMMAND_0_BASE,0,0x0);
                 goto LOOP;
          }
     }

     }
     else if (level_mode==1)        {
          while(state == 0x4 || state == 0x5 || state ==0x6)
          {
          if (state==0x5)
          {
//              start_t=clock();
//              printf("start time: %d \n", start_t/CLOCKS_PER_SEC);
          while(1 && game_time <= 10000000 ){
               random_number = rand()%27;

               mole_status=1;
               mole_command = mole_position( random_number,mole_status);
               panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;



               IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
               IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);


               while (mole_status==1 && counter<70000 ){
                    counter++;
                    game_time++;

                    //pause
                    if (state==6){
                    while (state==6){
```

```
                panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
                IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
            }

            }
            else if (state==0x0){

                IOWR_CONTROL(COMMAND_0_BASE,1,0);
                IOWR_CONTROL(COMMAND_0_BASE,0,0);
                goto LOOP;

            }
        }


        if (counter==70000 ){
            if (item==0){
            life--;
            }
              if (life<=0){
                  state=0xC;
                  music = 4;
                  panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
                  IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                  IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                  while (showtime<500000){
                      showtime++;
                  }
                  showtime=0;
                  state=0;
                  IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                  IOWR_CONTROL(COMMAND_0_BASE,0,0x0);
                  goto LOOP;

                    }else{
                      mole_status=0;
                      panel_command = state+ life*16 + score0*16*16 +
score1*16*16*16 + score2*16*16*16*16;
                      mole_command = mole_position(random_number,mole_status);
                      IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                      IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                  }
        }else{
            if(item == 0){
                if(score0 == 0){
                    score0= score0+5;
                }else if(score1 < 9 ){
                                score0 =0;
                                score1++;
                            }else{
                                    score2++;
                                    score1 = 0;
                                    score0 = 0;
                                }
                panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
```

```
score2*16*16*16*16;
                        mole_command = mole_position(random_number, mole_status);
                        IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                        IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                        item=0;

                    }else if(item == 2) {
                                        if (life!=3){
                                            life = life +1;//life limit
                                        }

                    }else if(item == 1){

                        if(life>0){
                            life = life -1;
                        }else{
                            state=0xC;
                            music = 4;
                            panel_command = state+ life*16 + score0*16*16 +
score1*16*16*16 + score2*16*16*16*16;
                            IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                            IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                             while (showtime<500000){
                                  showtime++;
                             }
                             showtime=0;
                             state=0;
                             IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                             IOWR_CONTROL(COMMAND_0_BASE,0,0x0);
                             goto LOOP;
                        }
                    }
                     panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
                     mole_command = mole_position(random_number, mole_status);
                     IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                     IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                     item=0;

                }

            counter = 0;

            }
             state=8;
             music = 3;
             game_time=0;
             panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
             IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
             IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
             while (showtime<500000){
                  showtime++;
             }
             showtime=0;
             state=0;
             IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
             IOWR_CONTROL(COMMAND_0_BASE,0,0x0);
```

```
            goto LOOP;
      }
}


      }else if (level_mode==2)        {
            while(state == 0x4 || state == 0x5 || state ==0x6)
            {
            if (state==0x5)
            {

            while(1 && game_time <= 10000000 ){
                  random_number = rand()%27;

                  mole_status=1;
                  mole_command = mole_position( random_number,mole_status);
                  panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;



                  IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                  IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);


                  while (mole_status==1 && counter<40000 ){
                        counter++;
                        game_time++;

                        //pause
                        if (state==6){
                        while (state==6){
                              panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
                              IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                              IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                        }

                        }
                        else if (state==0x0){

                              IOWR_CONTROL(COMMAND_0_BASE,1,0);
                              IOWR_CONTROL(COMMAND_0_BASE,0,0);
                              goto LOOP;

                        }
                  }


                  if (counter==40000 ){
                        if (item==0){
                        life--;
                        }
                         if (life<=0){
                               state=0xC;
                               music = 4;
                               panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
```

```
                        IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                        IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                        while (showtime<500000){
                              showtime++;
                        }
                        showtime=0;
                        state=0;
                        IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                        IOWR_CONTROL(COMMAND_0_BASE,0,0x0);
                        goto LOOP;

                            }else{
                              mole_status=0;
                              panel_command = state+ life*16 + score0*16*16 +
score1*16*16*16 + score2*16*16*16*16;
                              mole_command = mole_position(random_number,mole_status);
                              IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                              IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                        }
                  }else{
                        if(item == 0){
                              if(score0 == 0){
                                    score0= score0+5;
                              }else if(score1 < 9 ){
                                          score0 =0;
                                          score1++;
                                    }else{
                                                score2++;
                                                score1 = 0;
                                                score0 = 0;
                                          }
                              panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
                              mole_command = mole_position(random_number, mole_status);
                              IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                              IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                              item=0;

                        }else if(item == 2) {
                                          if (life!=3){
                                                life = life +1;//life limit
                                          }

                        }else if(item == 1){

                              if(life>0){
                                    life = life -1;
                              }else{
                                    state=0xC;
                                    music = 4;
                                    panel_command = state+ life*16 + score0*16*16 +
score1*16*16*16 + score2*16*16*16*16;
                                    IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                                    IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                                     while (showtime<500000){
                                          showtime++;
                                     }
                                     showtime=0;
```

```
                                state=0;
                                IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                                IOWR_CONTROL(COMMAND_0_BASE,0,0x0);
                                goto LOOP;
                            }
                        }
                         panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
                         mole_command = mole_position(random_number, mole_status);
                         IOWR_CONTROL(COMMAND_0_BASE,1,mole_command);
                         IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                         item=0;

                    }

                counter = 0;

                }
                  state=8;
                  music = 3;
                  game_time=0;
                  panel_command = state+ life*16 + score0*16*16 + score1*16*16*16 +
score2*16*16*16*16;
                  IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                  IOWR_CONTROL(COMMAND_0_BASE,0,panel_command);
                  while (showtime<500000){
                        showtime++;
                  }
                  showtime=0;
                  state=0;
                  IOWR_CONTROL(COMMAND_0_BASE,1,0x0);
                  IOWR_CONTROL(COMMAND_0_BASE,0,0x0);
                  goto LOOP;
          }
}

          }
}
          }
          return 0;
}
```

background.h

```
const int background[]= {

      25832,38510,38510,30058,36397,25832,25800,34285,38543,27978,23752,23687,15299,19
493,27945,27978,36461,36461,27977,36462,30122,23751,30090,32235,32203,36429,21638,195
26,34349,32236,25897,19526,21606,25865,19525,25864,21638,23751,15299,15331,25864,2167
1,15332,23784,11106,19493,27945,19493,40623,23751,36396,38542,40622,34348,30122,34380
,34348,19557,36461,34380,17444,21702,25864,15331,36461,32235,21638,38542,13250,23815,
28009,32235,38541,25896,36493,30090,32235,25896,21670,38542,38542,23783,23751,23751,2
3816,32235,21670,13219,19557,23783,30090,25896,27977,25864,15299,13186,21670,23783,34
283,27977,40623,36429,21670,34315,44913,32235,38574,27976,40655,40654,40655,21637,216
70,21606,27977,25896,25864,23719,42768,32203,38574,25897,27977,21670,23783,30090,3220
3,27977,17412,40687,30121,25863,34347,28040,25927,25894,32266,25927,30186,38605,28040
```

,38605,36460,40686,25863,34316,34316,13218,30058,17445,34316,38606,32202,32235,30122,25896,25863,36461,34315,25863,36461,30122,30090,11073,30090,21670,25896,34315,36460,32235,32267,30122,30122,30122,25928,34348,32267,36493,23783,34348,30089,36428,27977,34316,15363,30121,25896,34315,25928,25895,28040,28040,36492,34347,25960,28008,34379,36460,40654,28008,

25865,30090,30091,32204,36430,23720,15300,30026,44817,30026,25833,32139,30091,36397,32204,30091,34316,40687,23751,32203,25864,34348,34348,30090,17412,27977,38542,38542,34316,30058,15300,34317,34316,23752,19558,27978,19526,25864,17412,30090,19525,30090,32236,36430,19558,13219,30090,40623,40655,21638,38510,25831,32202,25896,25928,30154,32235,28009,23783,34315,23783,38574,27944,27976,34315,28009,17444,34348,19525,30090,23815,32235,30154,19557,40687,28009,19589,25864,15331,30090,28009,19525,25864,19557,32235,30122,17444,25864,32203,36429,25864,32236,32236,30122,19525,11106,23751,38542,36396,44881,30090,42768,36461,17444,38574,19557,32235,30122,38542,36428,38542,15266,27944,42735,36429,28009,34316,30090,40688,34317,25897,32203,42768,28009,34316,38574,34348,27976,36461,34347,30154,23750,34347,32234,23782,19620,34346,38605,30185,38573,30121,36492,40686,25928,27976,28009,30122,30058,30058,30090,30090,40687,42768,32203,21637,27976,28009,38574,32202,27976,42800,30089,21638,25831,25896,36429,25896,25896,40654,36460,30089,36428,34348,23815,30155,32235,30155,28041,21702,32267,30122,32267,34316,32235,28009,38541,25863,27976,30121,30120,30121,25927,36492,34347,28040,23814,32266,23815,32266,23814,

36430,23751,32204,34284,32171,21639,38478,36333,32139,38446,32171,25832,32139,30059,19525,23752,34316,38575,25864,21671,21638,25864,25896,32235,15299,23750,36429,25897,17445,13219,21671,23719,25897,25832,30123,28010,27945,19558,30123,25897,36429,34349,34284,38542,25864,21639,25865,40623,21606,19525,17412,25864,38509,30122,25863,30154,32202,25896,25928,25863,25863,32235,40622,23751,23783,28009,17412,21638,21670,21670,23751,32235,23783,28009,34316,25864,19557,23783,21670,21702,23751,36429,32235,19557,30090,27977,32236,21638,32236,38542,25864,34316,34348,25896,13219,34348,32235,27977,34348,30090,21638,36396,42768,17412,42800,17412,23783,42800,40687,44880,40654,19525,34316,42768,32170,36396,25864,25864,34317,30090,21639,32236,42768,34348,38574,36493,36428,23783,44912,28009,30121,19556,36428,32234,32266,36492,40685,40685,36492,38637,28008,34379,36493,28008,30089,34316,30090,30058,25832,30090,23783,30154,34348,25864,30154,32235,25896,38541,32234,36460,36461,21670,34348,27976,30089,34283,32234,40687,34348,25863,23815,32202,36493,25928,34380,38607,28009,23783,28041,25928,36493,32202,38574,42735,28009,32235,32202,30121,28041,28008,32234,28008,25927,34347,38605,28008,32266,21669,34347,30121,

25865,23719,42736,32172,38543,19526,25833,29994,30026,36398,34284,42704,25865,34284,6848,13154,17380,36429,28010,19525,38542,21670,21670,32202,32202,30089,19525,23783,11105,11073,32204,25832,25897,23752,25865,34317,25865,30123,32268,34316,30123,28010,32236,34317,34317,19526,19493,30058,25864,15299,15299,23718,38541,21637,17476,30154,30122,28009,21670,30089,30090,25864,32203,44848,17379,30090,21638,27977,32236,21638,19589,38574,23815,34380,23783,25896,19557,17476,28009,25864,21670,38542,23815,23815,28041,32267,32203,19557,28009,38542,21638,23751,25864,30123,30123,28009,32171,30090,34284,27977,30058,40655,40655,11073,36461,23751,32203,38574,36428,38541,40687,36428,34316,30090,23751,42735,27977,32171,34349,38543,27945,30091,36429,32235,38574,32202,36460,34347,42831,23815,38573,15331,34380,38573,17475,38572,42831,23814,32266,23782,21701,40718,28008,23815,30122,28009,25832,23751,17412,30058,11073,23816,25896,25928,36461,40687,25896,36461,38574,23815,32170,27976,23751,23751,36428,30057,34283,32202,32234,25863,21637,28009,36494,23815,30122,30155,25928,25896,30154,30122,38574,32235,38510,44881,23751,34315,34347,23782,25927,30121,21701,32233,25927,23814,30153,21669,30154,21702,38572,40653,

27945,32203,32204,23752,30091,15235,19494,27881,34252,25800,19493,32204,36430,38543,17412,21639,21638,34284,25897,21671,32203,25896,30089,25928,32235,38574,8992,21670,13218,23783,30090,32236,8993,13187,23752,38510,28010,38543,23784,34316,21639,34349,28010,34349,36462,21671,19525,34316,27977,8960,23718,34315,36428,25863,23783,15363,21

702,21669,25928,19589,32235,40655,30090,30089,30057,34316,15331,27977,19557,19525,280
09,36428,36428,28041,19557,32235,34380,21702,32267,30122,23783,32235,32267,32235,3223
5,28009,32203,25864,32236,42768,30090,32203,40655,40623,38542,30090,32235,25832,27945
,36429,27977,36429,40687,11105,21670,30122,38542,38541,32202,27976,36461,28009,42832,
17412,30089,36461,30058,30122,30091,40656,34349,25864,36461,36461,25928,34380,25895,3
4379,32267,30154,38606,23783,32203,34380,21669,40718,38573,28040,23782,25927,17443,34
379,15362,17476,36461,25863,32203,36429,21606,30090,23751,25896,28009,28041,36461,322
35,30089,40654,40687,30089,40654,34283,19525,23783,36428,42735,34283,32235,30121,2586
3,25896,21637,19589,17476,21670,19589,28041,34348,25928,32235,34348,28009,32235,30090
,27977,34283,23783,25895,25895,23814,36492,36460,30121,25895,34347,23782,34379,28008,
36460,40718,

30091,30090,23687,25832,21606,6848,27913,32139,19462,23655,27978,23752,30058,385
75,23751,19525,32236,30090,17412,21638,17412,32203,30122,32202,21670,32267,21670,3012
2,23815,40687,28010,23784,13186,25864,23784,38510,23752,21606,25833,21639,21639,32171
,25865,40623,25832,17412,17412,42768,25896,15331,32203,32235,30122,34348,21670,23750,
19524,36461,28041,23783,25864,34348,32203,21670,27977,28009,38574,32235,17444,25896,3
4315,42767,38606,23750,32235,36525,28073,30186,30155,32235,38574,30154,23784,28009,23
784,38542,30090,32236,28009,38542,44848,32171,40622,44849,30090,36397,23751,25865,195
26,34349,27945,38542,42800,17412,34315,23815,32202,40687,25863,32267,32202,25928,2592
8,19557,28009,30090,34315,23751,27977,36397,32171,17412,30090,32202,17444,32267,23783
,28009,36461,34380,28009,30090,17444,34348,30154,32234,32202,34380,21637,36460,17475,
23814,15297,32202,32234,34315,34380,38574,27977,23783,30122,34316,36461,32235,38541,2
5896,34316,36461,28009,34315,36461,30122,32202,25896,28008,38541,36461,34380,25895,25
895,32267,15363,15330,21702,23783,25928,38574,32267,28009,32267,34381,27977,27976,321
70,30089,25863,21637,23750,32202,17411,34380,36460,28041,32266,32234,30121,28073,1962
1,34379,40751,

36397,30090,34317,25832,27946,36398,36398,19494,21542,17348,25865,21639,23751,13
187,17412,21671,34316,34348,11041,23783,30090,23751,30090,21637,21670,23815,28009,237
83,32235,36493,21638,34316,34284,32203,19526,23752,17381,23752,30059,30058,25865,2372
0,25865,23719,13186,8960,27945,34284,23751,23751,36461,17412,30089,27977,32235,34380,
28009,30154,38541,25928,25928,28041,23783,13186,23783,32235,32203,38542,17444,25896,2
7976,30089,25896,25863,34380,25961,21702,23815,40719,34380,28042,25896,32235,27977,34
317,25897,25897,30123,27945,38574,38510,30057,46962,38510,34284,30090,21574,17413,343
16,28010,21638,34316,30090,34315,40687,30089,30122,36493,34347,34347,28041,21702,2589
5,32235,30122,21702,27976,30154,21638,36429,34316,15299,23783,19557,21670,32203,30090
,30122,27976,36429,36429,42800,27977,32235,21670,32235,34315,28008,13217,36428,19524,
21669,25862,42766,36460,42799,36460,36461,25863,21670,25864,38542,36429,34348,36461,3
0122,36461,25864,17444,34348,34315,34315,40654,30089,32235,30154,25928,32267,23782,30
154,32267,30121,17476,30121,34347,30121,32267,25928,30154,25928,32235,17476,27976,406
22,34315,23718,32202,30057,36460,13218,36428,36460,23782,25928,28008,36460,23814,2589
5,28073,40718,

27978,34284,36430,23720,17381,36398,38511,32139,30059,15268,30058,13187,4736,153
00,25897,25897,30090,40687,21606,30058,23719,19492,38574,17444,21702,23783,30122,3009
0,32235,25896,23783,34348,27977,40688,30091,32203,21606,38543,42769,36365,21606,23719
,25865,25832,17413,19526,27977,36429,21671,17444,32203,25864,23751,30090,36461,30089,
30122,34380,34348,23815,32235,30090,34348,21670,30122,23815,27977,32235,15331,25896,2
3783,21638,17444,21702,23783,19621,21734,32267,40719,30187,21703,23783,38607,34316,36
430,25864,25864,30090,40688,27977,19557,36429,42735,32203,36429,32203,25832,25865,448
82,28010,17412,23783,30090,34316,32235,27977,36461,32267,38574,32235,30154,28041,3226
7,36460,21702,28041,25928,36428,21638,36461,32203,17412,38574,21638,30122,36461,30090
,32203,32202,28009,28009,38574,25864,28009,25864,32203,25896,19556,21702,34380,6912,3
6460,23750,40654,42799,40718,34348,30089,32267,34315,34316,38542,36461,34348,32203,19
525,34316,32235,32235,34380,32235,34315,30122,25863,34348,23783,25863,21702,30089,301
54,30089,32267,30154,28040,36461,36461,38573,32234,30122,28009,28009,19525,21670,4065
5,30089,21637,42800,38574,40654,17444,36460,34347,21701,23815,38573,30121,23814,30186

,36492,34379,

21639,23719,30091,27913,19526,27946,30027,34285,42737,13187,8929,19493,27978,321
71,36429,38542,23783,21671,36461,19525,11041,17412,17444,19557,23751,32235,28009,2804
1,36493,25928,30155,23816,30123,42800,32203,30123,19493,44849,44817,21639,34317,30091
,25865,32171,25864,27978,28009,34316,30090,30090,36429,15331,15299,28009,34348,32202,
30154,17476,30122,25864,28009,27977,34380,25896,32203,40687,19557,30090,25896,34348,3
4348,21637,30122,27976,19524,32267,28009,23815,28042,32235,32235,32267,44913,34348,38
510,32171,25832,23752,38575,21670,25863,27977,25831,30057,30090,44849,27977,34284,343
17,21638,32203,25864,36396,34316,23750,38574,42832,23750,32235,21702,25928,25928,3438
0,34380,25863,28008,23815,32202,25896,30090,25896,27977,32203,27977,38574,32203,28009
,38542,23783,42768,30122,40687,23783,27977,32235,23783,27977,40687,30121,34380,17411,
34347,30121,36460,38573,42767,36460,34315,34380,36428,32203,36429,30090,25896,23783,2
1670,34348,27977,30089,32235,27976,28009,30122,25895,32202,23750,30154,25927,32234,32
234,30121,36493,32299,25927,40687,40686,32234,32234,34348,25928,25896,11105,34315,343
16,34316,34348,42767,36396,36493,15298,32267,30121,23750,25895,40718,32234,32234,3437
9,38605,23814,

30091,32236,38543,30091,30091,15300,17413,25865,36398,19526,23720,36430,30123,40
688,34284,38510,17412,30090,42736,23751,30091,36429,21670,28009,23783,23783,30090,322
35,36461,34380,36429,40687,32236,28009,30090,23784,17413,30058,38511,27978,25833,2583
3,30091,30091,34316,42736,23752,15299,28010,36462,28009,13186,27977,23751,38541,38542
,32235,44881,23751,40687,21637,30090,32235,19557,34316,34348,11073,40655,30090,23783,
11105,36429,28009,25928,30154,34348,23815,21669,38574,17444,25896,28009,34349,19557,1
7380,32235,17413,17380,38542,34283,17412,30089,30122,28009,25896,38542,34349,21638,23
719,32171,27945,21638,34316,32203,19557,44880,25896,28009,25864,23783,30154,30154,322
34,25895,23782,11137,25895,32234,32202,30090,27977,28009,32235,23783,34316,42768,1955
7,34316,30090,36461,25896,27977,25864,25896,30122,32235,28009,36428,44913,21702,30089
,40719,38573,34347,25863,36460,32234,34347,36460,36461,28009,25864,23783,23783,30090,
21638,32235,25896,23783,34315,23783,25895,38541,25895,32234,25863,36492,25895,28040,3
2267,34379,36460,32299,25895,38573,34347,28041,32234,28009,23847,17509,21670,36429,25
896,34316,32235,36429,25896,34315,28008,38574,30121,30121,30121,34379,32234,30153,301
53,32298,34411,

30091,34284,23752,23719,27978,17381,25801,27979,36365,11042,23720,38543,32171,36
429,38542,36397,30090,21638,32171,38542,32203,34348,23718,32235,19557,27977,36493,238
15,30122,36461,25864,28009,25897,23784,36462,23751,19493,38510,19494,27978,19526,3005
9,23752,38543,42736,44849,36397,15267,21638,36461,21638,21638,32235,30122,28009,27977
,34348,34348,15331,30090,15331,30122,36429,19557,30090,36429,17444,34316,40720,30090,
17444,38542,27944,36428,23783,36429,28009,19557,17476,23815,34348,15331,30123,17444,2
3751,38542,32171,13154,40655,30057,32235,40687,30122,21638,13218,38542,30123,17445,23
751,32235,34316,30090,40655,36429,27976,30122,15331,21638,30154,25928,23783,25895,280
41,28008,28041,25895,34315,38606,38542,27977,25896,36429,25896,21638,27977,42767,2589
6,34316,30122,25864,15331,32235,30122,30090,17444,38542,28009,32203,40687,30121,38573
,34380,25863,30153,27976,36428,27976,25895,32234,28041,27977,19557,23751,21638,30090,
36461,32235,28009,30122,34348,27976,25896,40654,23750,28008,25895,36428,34379,34347,4
0718,32266,25895,23814,38606,32299,34347,19588,32267,25961,30154,28041,30154,28009,27
944,25864,23751,36428,27976,28009,38541,34315,30122,34315,40654,25928,25895,32267,322
34,32298,30184,

30123,25865,13219,19494,21607,19461,30091,21640,21639,34285,19494,30091,23752,32
204,36430,34317,32139,25832,11073,23687,30090,30122,32170,32235,27977,34316,30122,258
64,28009,34380,32267,23783,32203,23751,27977,25832,40623,36430,11074,21639,11074,3428
4,36430,36397,32204,32204,36430,27978,25832,36397,27977,30058,36397,40622,36429,25864
,30122,34316,23783,32235,17444,36429,32203,25896,32235,30089,34316,28009,38542,28009,
21638,34316,25896,38574,19525,28009,40687,25896,32203,25896,32235,34315,23751,30090,2
5864,44849,46962,25832,40623,28009,25896,38541,32267,25929,19557,38607,27977,30122,32
203,30090,40623,23783,34316,30090,25896,25896,15299,23783,23783,30090,21670,15331,237

82,25928,23782,34379,27976,34347,28009,23815,30090,32267,25928,30090,38542,36429,3009
0,25896,23783,27976,17412,30122,32203,36429,23751,21638,23783,32235,28009,25896,36428
,19556,19556,28008,30153,32234,21637,25895,32234,34315,28009,30090,27977,30090,19525,
34316,17444,25864,25896,32267,28008,30122,38541,27976,25863,23814,40654,30186,34347,3
2266,30153,15362,19588,32298,32299,30186,21734,28008,32267,32300,34380,30155,23815,27
977,25864,28009,36461,32235,36429,32235,28041,38573,34347,42767,19588,27976,36460,238
14,28039,25926,

30091,21671,15268,11042,27946,21574,36430,27946,13219,27946,19558,23720,34284,42
769,34316,25832,21606,17380,17348,30026,36397,17412,30058,30122,34316,34348,15331,258
96,38541,28009,28041,19557,36461,28009,19558,32203,34317,32171,4736,21606,27978,34317
,42736,42736,23719,19493,38510,25832,30090,34284,27977,38542,27945,36397,34349,19525,
25896,15331,21638,30122,23783,28009,19557,27977,32203,42768,25896,27977,28009,32203,2
1605,36429,25831,21605,38509,30122,46994,38542,32203,30090,34316,30122,28009,36461,25
896,36429,34316,27977,44881,38542,25928,42799,38606,42799,36494,30155,34349,32203,258
64,25831,40655,25831,38509,30090,27977,28009,34316,30090,25896,25928,32203,21702,3008
9,23783,19589,38573,32266,32234,21670,36429,32235,32235,25928,30154,32203,30089,36461
,21670,36428,30090,23783,32203,30122,42800,32203,25896,23783,34316,32235,30089,36461,
21670,32267,25928,42767,28008,23814,21637,27976,34347,28009,32236,23783,30090,23783,2
1670,21637,25864,32202,38574,28008,36460,21670,30089,40686,21637,28008,30121,25959,23
782,32266,21669,15362,36460,34379,32266,17508,28041,34380,32267,36494,30154,32203,364
61,30122,25896,34348,25864,38542,27977,36429,32235,30121,30121,28008,32266,38605,1750
7,28039,19587,

23752,23752,15300,2656,38543,34317,38576,34285,13187,25865,21639,15300,30091,385
10,25864,21638,15267,11074,27945,27945,40655,17412,25864,30058,21606,27977,17444,3223
5,34348,28009,4736,32202,38541,40687,19558,32203,34284,17380,21639,32204,34317,32204,
36397,34284,23719,23719,30058,27978,21606,40623,23719,38542,11074,25864,32203,25831,2
5864,21638,25864,21670,19525,30122,28009,36461,32203,30122,27977,27977,30122,32235,36
397,38477,23751,17412,44848,40655,36429,38542,40654,40687,38541,32202,32235,32203,280
09,32203,27945,25832,44881,27977,30122,38606,28008,38638,32267,28009,15331,30122,2375
1,25864,34316,30057,32203,28009,23783,25896,40655,30090,23751,28009,40655,38541,32235
,32234,34380,34347,30153,34315,23782,38574,40687,34315,25928,30122,19589,21670,30090,
36429,38574,28009,32235,32235,21670,23783,40687,23751,25864,40654,21670,25896,30122,3
0122,30089,36428,28008,34347,30121,25895,25927,32234,21670,11105,23751,28009,32203,34
348,25864,34316,32202,30089,25863,34315,21669,36460,30154,17411,21701,30121,30186,238
14,34411,28040,23814,38572,30153,25959,30186,25927,34380,30186,32300,36461,32235,4065
5,28009,27977,30090,27977,40687,21702,21670,23783,28041,21669,34347,34315,30153,21669
,25894,32265,

6848,32172,25865,15300,32139,34317,40624,21607,19494,17446,36397,19493,36397,322
04,17380,19493,19493,15267,40623,32171,40655,19493,17444,30090,19493,21605,36429,3857
4,32235,17444,28009,36428,38542,28009,25864,25864,32203,30058,23752,30058,25865,30058
,30058,25865,30058,38510,19526,25832,34284,38542,27945,38542,11041,32171,21606,23751,
34284,19525,30090,32170,27977,30122,30090,30090,28009,19557,25896,30122,19525,44913,3
2204,30090,23687,32138,40655,42736,25864,36461,38574,34348,34315,32202,34316,23751,32
203,36461,25864,25896,28009,21702,28009,30122,28041,38606,34380,34381,34348,32203,258
96,38574,34348,32170,34348,32203,21670,38574,23751,30122,28009,27977,30090,34348,4276
7,34315,36460,36493,38573,30121,23815,34380,40654,32267,21702,30122,17476,13218,27977
,25896,32235,25864,23783,30090,38542,21670,21670,30122,25896,34348,25864,25896,25863,
34315,30089,32202,30089,38541,40686,21637,30121,30154,17444,17444,32203,34348,38541,3
2235,25896,38574,32202,28009,30122,30089,21637,38574,19556,30121,23782,28008,34380,21
702,34379,21701,32266,30153,32234,28040,34379,30153,32299,30186,30154,38607,30122,364
61,30090,25864,30122,32203,28041,27977,23815,30122,21670,25863,30154,30154,25895,2166
9,30185,32265,

15332,30091,38543,17381,19493,27946,25865,19559,27978,15333,30059,25832,38510,25
832,15267,32171,23687,27945,36429,21639,19526,21606,36397,25832,25864,21605,34348,322

03,30122,36460,34315,30122,30089,21670,17412,17444,38510,25864,30090,28010,23752,1955
8,17413,21671,32204,34284,17380,11041,34317,32204,23719,27978,21606,27945,25800,21573
,30058,27977,25864,32203,30122,30057,32235,21670,17444,23751,25864,28009,19525,32203,
40656,30090,21606,36429,27977,25831,6848,32235,36461,25863,25928,30122,40687,32235,34
380,38574,30090,21638,25864,32235,38606,34347,36493,38573,36460,23783,34380,25896,322
35,36461,23783,25863,28009,32235,23751,38574,17379,23719,36429,32170,15267,21670,4068
7,19557,32267,30154,32266,28041,25896,28041,32234,28008,32202,25863,32235,32267,28041
,23783,34348,38574,11105,36461,34348,32235,40687,27976,27977,25896,28009,15299,25896,
23751,36429,36493,28008,38573,32234,38541,32234,28041,23750,38574,34316,38541,36461,3
4348,34348,32235,30122,30089,30122,30089,30089,28009,23783,32202,25928,30121,30154,25
863,34346,23814,34379,36492,23814,38573,25960,25895,25928,28008,23815,38573,30122,343
48,28009,36429,34348,32268,30123,27977,25896,30090,25928,23783,28009,25928,25928,2378
2,30153,28040,

17413,25897,40623,8961,21639,30059,36398,17446,27979,32205,27946,32172,34252,215
74,17380,30091,27945,30090,21606,27945,23719,32171,27945,27977,21606,11105,27944,2797
6,38541,23815,42767,32267,32235,15331,19557,27977,40623,17380,32203,23752,23752,25832
,17413,23784,30091,34284,30091,19526,19493,25865,27978,36397,30091,21606,30091,25832,
38542,36397,27977,46962,25863,27976,32202,32267,28009,21670,21670,36461,30123,32203,3
6397,25865,25832,27945,27945,28009,19557,38542,27976,32235,36461,34380,38541,34380,34
315,32235,21670,34348,34348,25896,36493,30121,30122,34380,36461,21670,38574,25896,237
83,36428,28041,38606,21670,27977,15331,17412,11073,36429,30090,36429,21606,21605,2375
1,30122,30154,34380,21669,19556,28008,25928,25895,23815,32235,25896,36493,32267,28009
,21638,21638,23751,17444,23751,38574,30122,40654,21670,23783,30089,32203,23751,30090,
30089,28009,25896,23750,38574,25895,36460,25895,40686,36461,38574,36461,34315,34315,2
3783,38574,30122,38574,23815,30089,36428,28009,21637,25896,30122,23783,32234,30121,32
234,34347,36492,30153,32233,25927,23781,25895,21669,34347,34347,19557,30154,27976,301
22,25864,36429,36493,34380,30090,27977,23783,32203,28009,30122,28009,23750,28009,2166
9,32266,30153,

30091,30058,40591,23752,34317,32204,38511,25833,13187,21640,42769,23720,34317,44
849,27945,40656,13187,17412,17413,36429,28010,38543,19494,11009,13121,21638,21606,280
09,23783,23750,36461,36461,36461,15298,32235,19557,27945,17412,34284,34316,32204,3431
6,19525,36397,34284,44882,32171,23719,13187,30059,40624,38543,25865,21606,25832,34284
,40623,36429,25832,19525,25864,27976,28009,30122,30090,23783,28041,32235,30090,25864,
21606,17380,23719,21639,30122,30090,21670,30089,25896,28009,32267,40654,40686,28041,3
8573,30122,21670,38574,19589,19589,34380,40718,32267,36493,32202,21637,36461,32267,32
234,34380,32234,19589,11105,34315,32235,25896,21605,46961,21670,36429,34316,38509,195
57,38574,30121,38541,23814,25928,34380,19588,28041,28041,28041,23783,34316,30122,2586
4,19557,25864,21670,34316,27977,32203,25864,32235,23783,32202,30122,34348,40654,32235
,27977,19557,30090,25863,30089,30121,38541,28008,42799,34348,30154,40686,21702,34380,
36461,32235,32202,40655,21638,32202,25864,27977,27977,36429,27977,34316,30122,25863,2
5895,30153,30121,32201,30121,25894,25895,30121,25895,36460,36428,17476,25896,25863,25
863,23783,32235,32235,30155,32235,25897,34349,30154,30122,28009,30122,28041,32234,364
28,36493,25895,

30058,30058,36397,21574,32139,36398,36366,17414,15268,17381,36398,25833,23719,36
397,27913,25865,23719,19493,19526,25865,40688,42769,19493,13186,25864,21606,17379,300
57,28009,23750,32235,30121,30122,30122,23783,21638,34348,15267,30090,42801,40688,3009
0,30091,38543,38543,34317,36430,30058,30059,27945,27978,34284,36365,38543,23720,30026
,34284,32203,17444,32203,25863,30057,23750,17476,38574,30122,30122,25896,27977,17412,
21639,13154,30058,23719,27945,19493,13186,23783,25928,13218,30121,34380,32267,15363,3
6460,36460,30155,28041,34380,34380,34380,34347,23750,19556,23815,8992,42768,34348,343
80,36493,28008,19556,30121,34348,30090,36429,30090,25831,13154,30090,32171,23751,3431
6,34316,19557,21702,28008,32234,28073,25895,34347,36460,34347,25896,34348,30090,30090
,30122,32235,30122,32203,25864,28009,36461,28008,25895,38574,30122,32203,38574,36461,
30122,25864,34348,27977,23750,34348,36460,17443,34379,30121,23782,42767,27976,21702,3
6428,30122,32235,30090,34316,32203,30090,25896,36429,36461,25864,38542,38542,21670,34

347,28041,25863,25895,34346,25927,19556,34379,32266,32234,19589,23815,28009,21702,237
82,28009,32203,27977,25929,34316,25864,30090,30123,25896,25896,34348,30089,34348,3642
8,36461,25895,

36365,32171,27945,19526,19494,36397,40624,15268,27946,21607,36398,19494,30058,34
285,25865,13187,32171,30091,23752,38510,38575,30123,32204,21606,30090,17412,38542,321
70,34380,28009,25863,21669,21637,36428,17412,17412,23751,27945,27977,25864,36462,3009
1,30090,42801,23752,21639,36397,44849,36397,23752,17413,34252,36398,38511,40623,30058
,32171,19493,30090,36429,21638,36396,36461,28009,34316,30122,32235,17444,23751,13219,
25865,25832,8928,30058,23719,27977,25896,21670,36461,21670,38573,28041,23815,21702,34
380,34380,28042,25961,36494,28041,19556,34347,17443,25863,25863,25863,38574,38541,301
22,36525,30153,19621,40687,40687,32235,34348,30090,25831,19525,27945,27977,25864,4068
7,15298,13218,30121,32267,30122,34412,36492,19556,36493,30122,25896,25928,32235,34348
,32267,32203,32203,27977,30090,28009,44913,15298,21637,42767,21637,30090,19525,32235,
32203,23783,34316,36429,28009,36428,32234,19524,32202,34347,38573,40654,21702,23783,3
6428,40654,25896,32203,36461,36429,36429,30090,32203,30122,23783,34348,40687,30057,25
863,27976,19524,25895,25895,25895,40653,30121,32234,28008,19589,30122,21670,17476,259
28,34347,21670,25896,25929,21671,28042,32268,34316,25896,30122,30090,32267,32202,3008
9,28041,23815,

21606,36397,40591,17380,17348,21574,36430,17413,15268,19494,38543,23752,27978,36
365,30091,27945,21639,32171,34317,34317,23816,27945,36397,19493,15267,23719,30122,280
09,36428,36493,17443,30121,28041,34347,28041,23783,36429,30122,34284,19526,30058,2800
9,32203,23751,34284,23751,30058,36397,23752,27978,23687,19494,30026,32172,36397,32171
,19461,19493,40623,38574,36461,32170,27977,27977,36429,30122,30090,25864,25864,17445,
36397,27945,27978,34317,25832,21638,23751,23783,40654,30090,32267,27976,23782,32234,3
6460,38573,21702,19622,32299,13218,23782,25928,17443,25896,32170,42767,28009,36493,30
186,32267,36493,23814,28008,36461,23783,28009,34348,23719,27977,25864,30090,21638,300
58,15331,28009,34315,32234,28008,34412,38606,17476,32267,23783,30122,28041,34380,3438
0,28009,36429,21670,32202,34348,30057,38574,19492,27976,32235,21637,28009,30122,40655
,28009,30122,32235,25863,30154,32267,38606,34347,28008,30121,40686,30089,28041,34315,
34347,34380,17411,38574,36428,23783,30122,27977,38542,40687,19493,38542,36461,30090,2
5864,28008,30121,36460,30088,25862,40718,27975,21701,23782,32234,36460,32267,19557,32
267,32234,17477,23783,21702,23783,30090,38542,30090,23784,34348,34316,30090,34348,174
76,30122,25929,

38510,34220,49011,19461,30091,19494,34317,27978,6816,19494,32204,21607,13155,406
56,34285,17413,17381,25897,21671,30090,19526,28010,38543,17413,15267,40655,30122,3220
3,30089,38573,32234,32266,32266,40686,19524,34315,25832,25864,40623,27945,25832,32171
,42768,19493,32203,30025,34316,38542,23752,13154,36398,17349,30027,25865,32139,34220,
17381,36398,27946,36430,25865,28010,19525,36461,36461,27977,15299,34316,30058,21606,2
7946,25833,27913,38511,36397,17380,36397,27977,21638,28009,32235,19557,13217,17443,32
266,38605,28008,34347,42799,34347,28041,25928,27976,36428,32202,40719,36461,34380,322
67,30186,28073,38573,36460,36429,19589,30122,17444,17412,30122,27977,40622,25864,1533
1,25896,36461,40687,42800,28040,32234,30121,28040,32267,36460,30153,38573,28009,34348
,19557,38606,19557,30122,34316,30090,36396,30058,40655,30057,21605,25831,34348,36461,
23750,30122,36428,30089,34315,34315,44880,32202,27976,25895,32234,28009,32234,28009,2
3815,23750,23815,32235,38574,36429,34283,30090,40655,40655,30058,40623,40623,21605,32
235,25896,40622,36493,19556,21669,25927,17475,34347,25927,38573,32266,30186,25895,364
93,30122,19589,23815,30090,17444,32235,28042,25896,23816,32235,34380,32235,30155,1959
0,30155,28010,

25800,36365,34285,25833,25833,15268,32204,36398,25865,25865,23752,23687,30059,30
026,27978,17381,15300,30123,27978,19558,8960,25897,30058,15300,15300,27945,21638,3015
4,28009,32266,32267,36492,40686,34347,21605,21637,23718,23718,25832,32171,15299,19492
,40622,34283,27944,38509,36397,36429,30058,30026,40624,25833,30027,32140,38479,25800,
32139,32139,30091,21639,21671,32171,38575,34348,36428,30089,36429,40655,36397,32203,3
2172,36430,21607,36365,40624,17381,34285,19493,25897,21671,27977,21670,13185,25928,28

007,36459,40686,25895,40686,36492,30122,30154,40654,38541,30154,36460,34315,38573,322
67,30153,30154,40751,34380,36429,28009,17476,25864,19525,25864,36396,40655,21606,1949
2,36428,34348,27976,36461,30089,23782,28040,36460,17443,38573,34347,32299,23847,21670
,21669,36461,17444,30155,38542,23751,21670,36429,42768,32138,30057,30090,30057,28008,
30089,36493,44880,32202,44912,27976,30154,36428,40654,36461,36525,28008,32267,25895,2
3782,21670,30122,23783,38606,42735,32170,19460,17412,23751,36397,27945,27977,21605,23
783,28009,36429,34315,27976,30153,32201,25894,30120,36459,28007,23814,23782,32235,322
35,36461,25864,27977,38574,25896,32235,30154,21670,21670,36461,36461,30154,34316,2586
4,36429,25897,

17380,25832,30059,40624,25832,30091,27946,38511,34317,19526,8929,15268,30059,279
78,23752,25865,21639,32204,42769,17445,4736,6848,27913,19526,21639,27977,21670,25928,
32203,25928,32266,25895,32266,25863,30122,36461,40655,34316,17412,25799,32170,19493,2
3718,36429,17379,27944,38542,36429,30058,36365,34317,32204,23688,34253,38478,21574,36
398,30091,32172,21607,38511,23784,40655,23751,36461,25863,30090,30122,34348,21639,384
78,34285,21574,21639,44882,13155,27945,19526,23784,30123,32203,38574,28008,28040,2592
7,40685,36460,23815,25863,28008,38606,30154,27976,32235,32202,23815,30122,23782,28041
,42864,25960,38638,30121,32235,23815,25896,32235,25896,40622,27977,36396,38509,19557,
42800,21637,23750,28009,21702,21702,30089,38573,21701,23814,32234,28008,17443,28009,2
8041,25928,23783,30090,32203,19589,17444,40655,40623,32171,32235,46993,23783,21637,34
348,28009,36493,32234,42799,30154,30122,32235,34315,28041,23847,36460,28041,34380,259
28,23783,30121,32235,36428,36429,32203,21638,21638,27912,30090,23751,17412,13186,2586
4,32235,32170,36460,21669,34379,25927,30153,30121,38605,30153,30185,23782,38606,28009
,32267,19557,28009,34316,32267,28009,32268,30155,32235,36429,36461,27977,28009,27977,
27977,32203,

13122,19461,23720,30058,25897,15332,27946,38543,36398,21639,6816,480,27978,19494,
19526,36430,17445,32204,23784,17413,30090,23751,38510,21607,23752,27945,15299,11105,2
3783,25896,25895,40686,40654,32202,32235,28009,30090,34284,23686,17412,40655,36364,30
090,34316,17412,13153,25864,30090,30058,38510,34317,19461,19494,21639,44785,34252,153
00,38478,23752,15300,34317,25865,30090,32203,25896,25896,30090,38542,32170,15299,3220
4,36365,32204,25833,32172,17348,19526,25865,21638,40687,34348,40654,38573,34347,40686
,34380,40686,15330,30089,21702,30122,36461,19524,25896,32202,34348,32235,23750,21702,
38606,19589,36461,19557,27977,21670,34316,32203,34348,38542,32203,40622,30057,21605,3
6428,23718,32170,25896,36428,19589,25928,25863,27976,38573,32234,34380,19589,30122,30
122,34348,34348,36461,34380,25864,34348,25831,32203,36429,30090,34348,25863,38574,385
73,19589,38606,28041,21669,30089,36428,21670,19589,25928,25928,40686,25895,36493,3226
7,30089,28041,40654,30122,32203,32203,42800,28009,32170,30090,27977,27945,19492,38542
,23718,32202,25863,17443,38573,28008,32234,32234,38573,36492,28040,19621,38606,25896,
25896,28041,25928,32235,32235,23815,34348,34348,38607,36461,27977,30155,30155,30122,3
0123,42768,

23719,36397,23719,21639,27977,25865,40623,38543,32204,19494,19526,4704,25865,279
46,30059,44817,19526,15300,11041,28010,36397,30123,32171,19493,21638,17380,19557,3220
3,25864,25896,21702,34347,34347,19556,15299,19525,25832,38510,13187,21606,36429,27945
,30057,34283,17380,15299,23719,21606,27945,38510,34252,36365,27913,13122,34285,15268,
17349,36366,27946,17413,25865,38543,19557,34348,36428,32202,34316,36429,23751,23783,4
2704,32203,32204,32204,21639,23751,19526,36397,25864,27977,25896,40687,42800,30121,40
686,32234,25928,25896,32235,27976,32235,32267,32203,30122,25896,25863,23751,17476,322
03,21702,21670,30122,25864,32235,25864,34316,32203,36462,32171,42736,38542,38542,2794
4,36429,28009,23718,32235,17444,27976,34348,32202,30121,28009,36428,28041,30121,30154
,30121,38573,32267,36493,34380,34348,28009,23718,19524,27977,42800,25896,25896,44880,
36461,23782,32202,19556,28008,23782,30154,21670,27976,38573,36460,32267,21702,40718,3
4379,40686,30121,32234,34315,32202,36461,38542,38542,34316,30090,34316,40655,23718,38
542,27944,25831,15331,21637,34347,23750,32234,40686,34347,28008,28008,23815,36461,195
57,23783,34316,30122,34380,30122,28041,30122,28041,36461,28041,27977,28009,34348,2800
9,27977,38574,

25864,36429,17380,13154,30123,34284,44849,34317,34317,27913,34285,27946,21607,21639,19526,36430,15300,21639,13154,27945,30123,17445,27945,25832,36429,15267,30090,28010,21702,17444,15331,27976,21669,11105,23751,19525,25864,27945,13154,30058,42768,13186,30058,38509,27945,23718,38542,32138,27945,32139,40623,40624,30026,15235,17381,23752,13155,25833,40624,17413,27978,30058,17412,42768,32235,38574,32170,28009,32235,19492,44849,25832,38510,15267,8960,40623,34317,23751,25864,34316,34348,40687,40686,30121,40654,30089,30121,42767,32234,30121,34380,32235,21670,25864,25864,25864,11073,25864,25929,13218,34316,38607,32268,38574,25864,28009,36429,30122,27945,27945,38510,36429,21638,34316,32235,25896,36429,23750,34283,38509,19557,17411,30089,32267,28008,25928,28041,28008,25896,25895,30154,30153,36460,28009,25863,25896,21670,40655,32202,30122,28041,23815,25928,28008,30154,42767,25928,40686,36428,28041,34348,38573,32234,32267,40719,38605,40719,40719,17443,34347,38541,34348,23751,38541,36429,34283,27944,34348,25864,44881,23718,19524,25863,28009,27976,17443,28009,40654,32267,23783,38606,34348,30122,19589,19557,21702,38542,32203,28041,32235,30122,34348,40655,32235,34348,28042,34381,25896,36397,38542,

27977,34316,19525,28010,23751,34316,42769,44850,32172,23720,32172,34252,21607,19494,21606,27945,21607,25897,27978,27945,25865,21671,30058,36397,23719,17380,34316,36429,19557,30154,21702,21670,25896,23750,36461,36461,38574,25832,21639,40656,36429,19525,19557,32235,30090,36429,38542,21638,34284,42736,30091,36430,32171,30091,25865,19461,13155,11010,34252,25865,32204,23751,17412,40655,32170,30089,32202,27977,32203,15331,36429,21638,34284,15299,25832,38510,19493,25832,25864,42735,32235,30122,32235,25895,34315,23750,23783,38541,32235,25896,25864,21670,13218,28009,21670,19525,17444,32203,32235,30090,34348,38542,38574,40655,19525,13186,23783,23751,19525,30090,27977,25896,27977,34316,27977,21638,30090,30122,36429,21638,17412,21670,34315,30089,28009,21670,23782,23783,30121,17508,28040,28041,21669,40686,23783,34348,25896,25863,30154,25896,19589,30089,34347,32267,34347,36460,34379,32234,28041,28008,32234,28073,30121,23815,38606,36460,32267,32234,25895,34380,32202,36428,36396,44881,23751,38542,17412,32203,36429,30058,28009,30090,32202,32235,25896,25896,25928,30122,36461,30089,34380,32235,19524,19589,19557,17444,32267,25896,21670,25896,28009,30122,34381,36461,21703,32235,34348,25864,34316,25831,

28010,40655,36462,17413,32203,40688,25832,19493,27946,34285,40591,30059,21639,15300,15267,21639,23784,25897,17412,32204,23751,17380,38477,25864,25865,23751,27977,42769,28009,13250,19557,25863,30122,30089,30122,30090,38543,30090,25832,36429,36397,17413,30122,34316,15299,21670,32171,30090,38543,34284,32171,42768,40623,27945,21639,19494,480,19462,30059,38510,42769,23751,25831,36429,30057,30089,34315,34315,21638,19525,30057,21605,34284,25832,32203,36429,28009,23719,28009,34316,32202,32203,34348,25863,28009,21670,25928,28009,36428,30089,30122,28009,36429,27977,32203,28009,25864,34349,36429,36461,27977,23751,36461,34380,15331,25864,30122,32203,21670,36462,32171,23719,25864,36429,27977,30090,40654,32235,32235,28009,32170,27977,30122,30122,32235,23783,21637,11072,32267,11136,19556,23782,23782,34347,25896,34315,34380,25928,25863,23783,27976,36493,40686,36461,27976,32234,34347,19556,34380,36492,28008,34412,34347,25928,30154,30121,25928,25895,34347,25895,32202,36428,44880,36429,11105,34283,21605,34316,40622,34284,30090,28009,32235,36461,19525,27977,32235,32202,30122,27976,23815,28009,21702,21637,40719,25863,30122,25896,28009,28041,28009,19557,28009,21638,17476,30122,32203,30122,36429,21606,

25896,32236,21671,23784,13218,30090,15267,6816,15300,25865,21607,15300,15268,23720,19494,32204,23752,23752,25897,38542,32203,27977,32203,44849,36429,30091,21606,30123,21671,19525,25863,34316,21670,21670,23751,25864,36429,27977,15267,34284,21638,32204,30091,36397,15267,11073,30058,27945,38543,32203,27978,42768,17412,25865,27978,21607,11074,30026,15300,32139,17413,38542,32171,30089,30057,27976,25896,30122,21638,34348,15266,32203,32170,28009,30090,32235,27977,40655,32170,21606,25864,30122,25863,23783,15298,38574,21670,30122,23783,23815,32235,44880,25864,25896,34316,42800,25864,34348,38542,19525,21606,25864,36429,30122,17444,25896,21638,30122,21638,36429,32203,17412,30090,38574,30057,25896,40655,25896,30090,32203,32235,30089,32203,28009,30089,30154,23782,19556,25960,15395,17475,32267,34412,34379,28009,30154,30122,25896,38574,30089,19524,

32202,36461,34380,30121,34347,38541,21701,25895,32234,30122,34380,28008,25895,32234,34347,25927,34379,21702,30121,28008,28008,42735,34315,13185,28009,30122,34316,34348,32203,23783,28009,32235,34348,21670,34316,42768,25896,23783,28009,30090,28041,32235,23782,36460,34347,25863,21670,40654,27976,32203,23815,34348,25896,25896,28009,34380,32267,38542,27977,

23751,38574,21671,15299,30090,19526,13187,13154,15268,19494,21607,30059,17381,27978,17381,21639,17412,32203,32203,25897,27977,23784,17380,40623,25864,40623,27978,30058,19558,34316,23783,38574,23751,25864,28009,38542,36461,21671,19525,23751,23719,40623,38543,38543,32171,11073,23719,32171,27978,23751,21606,30058,25864,30090,34284,27946,36398,21639,15268,15268,21639,27977,27977,30122,34316,27976,13218,28009,28009,28009,23751,34283,30057,34316,30090,34348,25864,32203,30122,21638,27977,30090,30154,30122,17444,40687,25896,30122,21670,23750,38542,32235,36429,21637,25864,40655,32203,32203,21638,32203,15299,19525,36461,27976,36429,32235,32235,32235,23751,36429,34316,23783,40655,23783,36397,27977,38574,19525,36396,32203,38574,30089,23750,27977,30122,21637,25863,30089,30121,30186,32234,32299,32266,25927,21702,28009,30122,17476,34348,25928,30122,32202,38574,25928,32267,30121,32235,28008,25895,28008,34348,34347,30154,30121,40686,34379,25928,36460,27976,30154,19589,34315,32202,27976,11072,27944,21638,36461,23783,32202,19525,17444,32203,27977,32203,40687,30090,27977,23751,28009,25928,38574,32267,23782,28041,32234,32234,23783,36461,30122,34348,32203,40687,28009,30122,32203,34348,25896,36429,32171,

23783,38542,28010,28010,32171,25832,32204,17413,15267,32204,38511,32171,30058,21606,25865,21606,30091,27978,15299,23751,27977,27977,13186,23751,27978,38542,40656,34284,25864,44913,13186,32235,23783,25896,38574,32235,40687,34348,25864,17412,36429,25864,32171,40623,30091,15300,30058,34317,27945,34284,32203,32171,25832,38510,36430,34284,40623,23752,13122,25800,32139,30090,38542,23718,34315,34315,36429,15331,27977,28009,30058,27945,30058,32171,32203,23751,21605,30090,21670,25896,42800,30057,42767,19525,23751,40655,21702,27976,30089,32235,36461,32235,25896,32235,32235,30090,40687,21638,23751,27977,23718,42768,30122,17411,36461,28041,34316,17444,34316,34316,25864,30090,32203,17412,27977,32235,21605,27977,32203,36461,32203,19557,30057,25896,34315,32267,17444,30121,32234,30121,32234,32267,32234,28040,23783,28009,32235,23783,23783,21670,30090,40687,40719,42800,28009,25863,32202,23783,25895,32234,40719,23815,25895,30121,30089,34347,27976,28008,30154,13250,23750,36460,21637,21637,13217,34348,21670,28009,19557,34316,28009,19557,40655,23751,40687,32203,25896,28009,23750,27977,23783,34348,25895,25895,28041,28008,32234,36461,38573,34380,32267,21637,40654,30122,27977,30154,21638,21702,28042,28009,

36429,32236,40656,34317,27945,25832,36430,19526,21607,30091,32172,21639,23719,36397,23752,34317,32204,23784,11041,21639,38542,25896,19493,36396,32204,34316,23752,27945,19526,34349,13218,34348,19557,34348,34348,27977,32203,23783,30057,27944,30058,32203,27978,36430,44882,6848,25865,25865,36397,23752,40623,36397,38542,34316,27977,34317,34284,36430,17348,17413,25865,32203,42735,17412,32203,40622,40654,25896,36429,32203,32235,27945,25832,36397,34284,17380,4736,23751,21605,40622,17411,23718,42768,15299,36429,36396,32235,38574,25896,23783,13218,15331,15331,32234,34348,34315,34315,23751,17412,27977,38542,40687,30122,6880,27976,32235,34348,17444,13186,34316,36461,36461,30090,28009,27977,30057,19492,25864,34316,32235,30090,21637,30089,23750,34283,17444,25895,32267,21702,19556,30154,38605,32266,34380,28041,34380,34348,21670,30090,25864,30090,34348,40687,32235,42800,32267,32202,27976,30121,42799,30089,25895,32202,34315,15363,28041,34347,17443,30121,13217,36428,30121,17411,30122,25863,30089,36428,32202,27977,38542,36429,32235,36461,23783,40655,25896,21670,21638,30122,32202,27976,36493,17411,36460,28041,25927,30154,36493,30154,32267,32267,28009,34380,28041,25864,32203,19589,21702,28041,25896,

21638,30091,25865,19493,19494,40623,38543,23752,21606,13155,23752,38543,21607,30058,38543,30091,23751,34284,6880,25897,34348,21670,40623,44816,21638,4704,8961,25865,27978,34284,32235,32235,27976,32235,34380,21670,32267,19557,23751,40655,19492,34284,25832,27978,32171,11074,27978,19526,30091,32204,32204,30090,30058,25864,13186,23752,25

832,34284,44817,21606,36365,21638,40622,23718,38541,32235,30089,36429,38574,34349,342
84,19493,13187,21606,27945,8992,6848,23751,34284,30057,30057,25864,36429,21670,38509,
32203,36493,23783,23783,17476,19556,25895,19556,30121,32234,36493,30154,28041,25863,2
5864,44848,40655,34348,34347,28009,32235,30122,32235,32235,25928,38542,34316,32203,19
557,25896,23718,23750,30057,25928,32267,36461,19524,21670,28008,36396,25863,38573,343
47,21637,21669,17476,32267,38606,32202,28041,28041,30090,40687,38574,23751,32235,2375
1,36429,38509,44848,19524,40654,36428,40654,32235,23815,32202,25895,27976,15330,17443
,40654,30089,23750,25928,38541,30089,23782,36461,38541,38541,32202,32235,30122,30090,
36429,32235,32235,38542,42768,28009,23783,34348,34316,30122,32267,23782,17443,36492,3
0121,25895,32266,30121,17443,34347,36460,34380,30121,23782,28009,36461,25928,30122,36
461,30122,

19526,21638,17413,30058,32172,46963,25833,21607,15300,19526,19526,30091,23752,32
171,30091,17380,25897,25897,13219,36397,30090,15332,34251,30058,19493,15267,15300,279
46,19526,21638,32203,28009,34348,32267,38541,19589,23815,21670,30122,23718,17412,2797
7,30058,44849,30058,19493,32172,21607,21639,27978,38510,38510,25864,19557,13154,27977
,36397,36397,15268,25832,40655,17412,38477,34316,36429,32235,27977,30122,32203,32203,
30058,11074,23719,17412,21638,19557,13154,30090,34316,25831,36429,19557,32170,44881,2
7977,27977,30122,23815,25896,21670,23783,28040,25928,38606,38573,34380,32234,38541,32
202,15331,40687,40687,30121,44880,17411,40654,32202,32235,21702,21638,34348,40655,279
77,17412,27977,17444,25864,32170,30121,25863,21637,19589,27976,25895,25863,38509,3431
5,34315,17476,30121,23782,38606,28041,32202,27977,15363,19589,30155,32203,17444,32235
,15267,40655,36461,40687,6848,25896,32235,30089,19524,23750,25928,17444,28041,25928,2
3750,34347,36428,17411,25895,27976,23750,28009,36428,36461,25928,34315,36461,32235,32
203,36429,21638,30122,42800,38574,36429,27977,38574,36429,30154,32235,25863,21669,322
66,30153,25927,30121,25895,21701,32234,32267,25928,32202,28008,32267,38541,32235,3012
2,28041,25928,

25865,32237,25865,21639,25833,38511,21607,32204,34317,19494,38510,30058,34285,21
639,23752,13154,19525,23784,23752,13186,32203,19558,38542,30025,27913,32204,32171,300
59,27945,32203,28010,30090,32235,28008,34315,28041,30154,32267,30089,19524,23783,2586
4,21671,23752,28010,21639,30091,17413,32236,15332,27977,40655,17444,32203,25832,36429
,32204,25832,34316,27977,15267,36396,25864,30057,34315,27944,23783,25864,34316,17445,
17445,21638,25832,19493,27978,19493,32171,32203,25864,38574,30090,23718,30122,40622,3
2170,11073,23783,30122,28041,28041,34347,36493,21734,25927,23814,34347,25928,28041,32
234,19556,36428,34315,19556,38508,25863,38541,30089,32234,34380,25928,30090,23751,195
89,19557,36428,34315,38509,40622,25895,15363,28008,34379,23815,27976,32202,27976,2586
3,19524,17444,27976,38574,36428,21702,25928,19557,21703,23783,27977,34284,19525,21606
,23719,40622,44816,30057,25864,34315,25863,28008,15298,23750,32267,23815,32267,30122,
34347,36460,36428,38573,34348,34315,28008,32235,25896,36428,38574,32235,32267,25896,3
8542,28009,19557,25864,34348,21638,34316,25896,38574,36428,34348,36461,34347,21669,32
266,25927,28007,23814,23781,32234,25895,30121,30089,34347,36460,38573,34348,38573,280
08,30153,34380,

40656,38543,6848,19461,15268,25800,19526,40656,21606,15300,27945,25865,34284,195
26,11041,17380,15299,32236,25864,19557,23751,34348,36461,23719,38477,27913,34317,3217
1,19526,30091,36461,28009,23815,32202,28009,28009,36493,38606,40654,28009,32203,30090
,38510,30091,23719,21607,23752,23752,25832,19526,21606,32204,30058,30123,23751,30090,
32203,21606,30058,23719,23751,44849,32170,34316,32235,38542,25864,40655,30122,32203,1
9525,36430,27978,32171,36397,32171,30058,25832,19525,36429,30122,15266,15266,23783,25
863,19557,34348,19557,25895,23815,34379,28040,15330,19621,23847,28073,32234,32299,238
14,34347,19524,40654,19556,36428,34315,32202,19556,19524,32203,28041,36461,27977,1955
7,21638,38509,36428,36428,40687,28008,21637,38573,30154,23783,32234,32170,30089,15331
,36428,27976,30057,21637,25863,25863,36493,19557,32235,32203,15299,30090,19557,32138,
23686,23751,40655,36429,34316,36428,28009,23750,13250,28009,40719,23783,30154,32267,2
8009,38574,34348,32202,36460,32235,28041,36460,28009,36428,40687,32202,32202,28009,34
316,27977,25896,23751,27977,23751,38574,25864,42767,17411,21702,38573,25863,23750,259
27,28040,23814,23782,32266,23782,34346,32266,28008,28040,32267,40719,34347,32202,3860

6,30154,36492,

32204,25865,23720,36430,6784,19494,23719,38575,23751,30090,25832,30090,23784,279
77,17380,17412,19525,32236,36429,25897,21638,36462,23718,34316,38510,19525,32171,3213
9,19493,25832,30058,32202,34283,28009,25896,25928,30122,34316,27944,36397,32171,30058
,27945,27945,15267,23752,27946,27945,23687,30058,21606,30058,36397,32203,19525,15331,
40688,30091,30091,23719,25832,40623,23687,34284,27977,42768,25799,36461,32203,32203,1
9493,34316,42736,21638,23752,34284,28010,32236,32204,23784,28009,25864,17444,23751,32
170,25831,28009,8992,30089,21637,25928,30153,17476,21669,19621,34379,25960,25960,2800
8,28041,32235,25863,32234,21669,34347,28008,23783,21670,25897,21671,30058,21606,30058
,23783,34316,28009,23783,42800,30122,21638,28009,36461,30089,38574,25895,25830,27975,
34314,40654,34347,19524,34283,30057,30122,25863,36428,34316,21670,28009,34284,42736,3
0058,34251,32139,34284,34251,23751,30057,21605,30089,30089,36493,19589,30121,32234,30
089,32234,30154,34315,36493,32202,28009,40687,23783,32203,40687,27976,32235,34316,300
90,34348,27977,19525,23783,32203,36461,32203,30090,19524,23783,30121,34347,30121,3226
7,28008,32234,34347,30121,30121,40686,34347,28008,23814,36460,38638,30154,30154,36460
,34347,34347,

19526,23752,21574,32204,19462,15300,27945,32203,21638,21671,23783,21671,28010,23
752,19493,25897,38575,32236,34316,21671,21671,21671,30090,34283,30058,23719,34284,153
00,34284,34284,30090,42703,34315,23718,32203,32235,30090,21670,15267,27977,25832,3428
4,13154,8929,13122,27913,21574,27913,23719,30058,30058,27978,23687,25865,23751,6880,3
0058,28010,23752,13187,19493,21574,23719,42704,27978,34284,19525,40622,34283,19557,27
945,21638,46962,13154,13187,36430,30091,19525,15332,15332,32171,36397,25864,27944,343
15,23750,27976,15299,21670,34315,15363,40719,15395,28040,32299,30186,34412,28073,3012
1,32267,36493,36493,21701,19589,38573,44913,25864,17444,40623,21639,19493,27978,25864
,32171,21671,23783,32235,28009,32203,23751,17444,42768,25863,34347,30089,19524,32169,
30056,44880,44880,30121,42799,21670,21637,32202,32203,23751,17444,30090,34317,36365,3
0091,25833,25800,27913,23719,19493,15266,28009,32202,23815,38606,19589,32266,25895,36
460,25895,30089,40686,25895,34347,30122,36461,32202,30122,32235,32235,38542,32235,237
51,32235,23751,21671,19557,30122,44913,38542,23783,30089,25896,21670,30154,38573,2807
3,28073,28040,38541,30153,34346,34314,32233,34314,19556,34347,34379,28041,23815,32234
,32234,30089,

32171,40656,15268,34252,15268,36398,19525,34316,25864,19558,32204,32171,27945,27
977,21671,27977,36462,17412,32204,17412,21606,21638,34316,23719,25832,23752,27978,131
55,46963,27945,42736,34316,30090,32203,36429,25864,17412,25896,21638,25864,32139,3217
1,17348,21606,34317,32171,30026,34317,23687,34317,32139,21639,13155,25865,32236,27977
,17445,25864,19493,17413,25800,23719,32139,38478,25832,30090,19525,30057,30090,34316,
25831,32203,40623,13154,34317,34349,23752,28010,21638,15299,23751,36429,34316,42800,3
0089,27944,30057,34283,40622,23815,23782,34380,30186,21734,23815,25928,30186,28041,25
928,23783,27976,34380,17443,30121,32234,34348,13218,28009,32203,32203,17412,27978,300
90,30058,27977,25896,30090,25864,28009,32203,30090,40687,32202,21637,40621,30089,3854
0,25863,38573,42767,21637,40686,13185,25896,32202,32267,19524,23783,34316,36397,21638
,30058,27946,23687,23751,25832,17412,21638,30090,32203,23750,40654,32202,30121,21637,
27976,21669,32202,36461,30154,34380,34380,25864,25928,38574,30154,38606,32202,28009,1
9557,32203,25864,25864,30090,32236,30090,32235,34348,30122,25896,19557,25896,30154,25
928,28041,25960,36460,28007,21669,30088,30121,36492,19588,30153,40719,19589,21670,343
80,25896,36428,

30090,25832,19494,13155,23720,21607,38543,36462,23784,15299,23784,28010,25865,34
317,32204,32204,28010,28010,25832,25897,17412,28010,38510,30091,25832,30058,34317,216
39,19461,34252,34317,44849,32203,34316,32203,21670,8993,34316,40623,27913,21606,38510
,23687,38478,32171,25800,34317,27978,21639,32171,32139,19493,13187,27946,40623,25865,
19558,23751,32204,27978,32172,36430,34252,25864,36429,32203,23719,38542,32202,30089,3
0090,34283,38542,32236,28010,36430,19558,21638,25864,30090,17380,30058,34316,23718,32
170,17379,25863,34347,40686,21669,23815,34347,34379,17508,32266,32234,23847,23815,216
70,25896,25896,17444,13250,32234,32234,32202,32203,28009,25864,40655,27977,38510,2794

5,21638,34316,40655,32203,23784,30058,27945,23751,34316,42735,30089,34348,30121,38541
,21669,36460,34315,32202,15330,11105,28009,25896,27976,30089,32202,19492,32171,36429,
27945,32171,42736,19493,15267,25864,17444,34316,42735,28009,36428,25896,34315,25895,2
8041,25896,23750,38574,34347,34348,38574,21702,25896,34380,32235,32235,36461,30089,21
670,30122,21638,21638,30090,34316,30123,30122,34316,34348,25896,17444,32235,32235,301
22,23783,25895,30154,23814,17410,25927,28040,28040,13217,34347,36492,19556,32234,3223
4,32267,36460,

25832,32204,23687,4672,6784,25865,38542,25864,19526,2656,15299,17445,28010,30058,
27946,30091,25865,19558,19526,23719,19493,34316,44849,19493,15300,21607,36398,30059,3
0091,27945,32171,34284,34284,19557,44881,11105,32171,34349,34284,23719,21639,42769,40
624,34317,34317,27946,23720,27945,30058,40656,25864,32204,23719,36430,42736,30091,385
10,30058,30058,25800,40624,38510,25832,32203,40655,36429,27977,30122,25896,32267,3646
1,30122,27977,30122,21670,40655,30122,25864,30123,28010,21671,36462,30090,17380,27944
,23718,17411,25895,27976,23750,17443,30121,38573,19588,40686,38573,30154,32235,25896,
25896,28041,25928,30154,30154,34315,32235,30122,21638,23783,36461,42768,44849,32171,3
0058,38542,38542,32236,27977,23751,27945,17412,40687,44913,25831,27976,30089,32234,17
443,32202,32202,30121,28008,34347,25895,36460,28009,28009,30122,25864,34316,46962,321
71,38510,42736,23751,13186,32203,25864,23751,27977,34348,32202,28009,19557,32202,3642
8,23815,34348,34348,36428,36461,36428,23783,30122,32235,21702,30121,40686,32267,21702
,32203,21638,19558,38574,34316,32203,36429,27977,34284,19525,23751,38574,38542,25896,
23815,25863,21701,23814,28007,28008,21701,23781,32233,36460,36493,34347,32299,32299,3
0153,32234,

23719,19525,27978,15235,21607,40624,25864,17412,23752,36462,21639,25897,34349,32
204,25833,25897,27946,27978,19558,27978,30091,30090,36397,11074,30091,13154,30059,342
52,15300,23687,23720,30058,19557,23751,25897,17412,32203,36430,34252,23687,40623,4488
2,30058,27946,32172,17380,32139,15300,15300,40624,23751,30026,30090,23751,27945,25832
,34284,30058,40624,25832,38511,19493,23719,25864,34348,23783,25863,34348,15331,38541,
34315,25896,30089,23783,8992,25929,25897,36429,19525,17477,25864,36429,21670,13186,23
718,32203,27944,34347,40654,27976,11136,30121,32266,32234,36492,25928,21702,42799,237
83,23815,36396,27976,32202,34348,30122,36461,32203,25864,32203,36429,38574,36429,2160
5,23751,34284,38542,36397,23784,34284,32204,27945,32235,38542,30122,21605,34315,30122
,28008,30121,30121,30154,32202,36460,32267,32267,25895,30089,34380,19525,30090,42800,
25864,30090,21638,36397,23719,36461,32203,30090,28009,36396,28009,30090,15331,30122,2
1670,34348,34380,36428,25896,38574,32235,27977,34316,30122,30154,36460,32234,34347,23
750,32202,28009,28009,42768,27977,34316,42768,27977,34316,23718,27977,40622,36428,152
99,40654,23815,28008,30089,34347,28007,21668,34379,34379,36492,30153,30121,30154,3226
7,25895,32234,

27945,21606,23719,30058,34252,34317,19493,21606,34316,34284,25832,30058,30091,23
752,27978,34285,21607,27978,23720,21639,25865,27945,27978,19493,38510,6816,17381,2160
7,21639,19494,34284,25800,17380,34316,25864,27977,32203,30122,40591,40656,36397,38478
,32172,19494,23720,17380,25833,32204,25865,40624,23751,21606,27977,34252,23719,30091,
32204,25800,19526,27946,32171,21574,21639,17413,36429,30090,25896,36428,21702,34380,2
5928,32234,25895,25896,15331,28009,30155,28009,21671,25864,25864,30122,21605,27977,32
203,27977,27976,32202,30154,25895,21669,25927,34347,42799,32267,36460,27976,42800,237
18,34283,42735,32202,25896,36461,28009,36429,27976,36461,19557,40654,38574,38574,1529
9,30090,25864,32204,32203,27978,40624,40656,25865,21606,25864,36461,25896,32235,25896
,36460,23815,28041,34379,36493,38573,21701,30153,19556,21702,38541,25863,28009,36428,
30090,32203,34284,30090,36429,40623,30058,27945,19525,46961,36461,25832,19525,27944,2
7976,36428,38574,32267,32202,36461,28041,28041,34348,34348,34380,36460,32234,32266,23
782,25863,30154,32203,23783,30058,34316,40656,25864,23751,32171,30090,27944,40655,216
38,23783,21605,19524,34315,28008,30088,32266,40718,28008,23814,30153,25927,25928,3229
9,21702,30153,

19525,27978,32171,34317,25833,32139,25864,19525,32171,27945,23719,42736,30059,17
380,21607,34285,21607,32172,30091,15300,27946,34285,34284,32171,27945,34285,30026,215

74,40624,27978,15299,23751,19492,34348,32203,23751,27977,30123,38542,25832,23719,2375
2,23752,19526,15235,23720,36365,19493,25800,40656,40655,27977,30058,36397,17413,21606
,36398,38543,15267,19461,32204,21606,17380,23751,40622,21605,17411,23750,19589,30121,
30121,38574,32234,17443,32234,28041,28042,30122,30090,19558,27977,25896,27977,21606,4
0655,21605,34316,30089,15363,32202,30121,38541,32267,34315,30154,34380,38574,42832,25
863,36461,38542,38541,27977,36493,28009,17412,25896,36461,28041,34348,30122,28009,279
77,27977,21638,38575,30123,27945,40591,38543,30026,21606,21606,38542,38574,34315,3220
2,36493,34380,25895,28040,38606,28040,21734,19589,21702,23782,30121,40654,32235,27976
,30122,30057,19525,30122,30058,32171,32171,21639,30058,36397,36429,21638,27945,21605,
25864,23783,36461,30089,32235,36493,28009,25896,28041,17477,32235,38541,34347,34347,2
1702,21702,34380,28009,28010,44881,30090,32171,23751,19525,36429,32203,27944,38574,30
090,15298,28009,34380,32234,38573,28040,32233,32266,19588,34347,25927,32266,30121,280
08,19588,25895,

25896,30122,40623,21606,38543,23719,34284,27977,27945,30058,40656,32171,34317,21
639,34285,19526,23720,19494,25865,27946,32172,21607,23752,34284,15267,46962,30091,216
07,34285,25865,21606,11073,19557,36461,32203,15298,28009,36461,25832,27945,25864,2791
3,21607,32172,15268,23720,21639,21607,30026,36397,36397,25832,23719,40623,15300,34317
,38543,32204,21638,23751,30058,17380,38510,27945,17412,25896,25864,27976,30122,34347,
23782,36493,28008,30121,34348,38574,30154,30154,30122,21638,30122,21638,34283,13186,3
2171,30090,36429,25831,30089,36460,36460,38606,19589,30121,34315,36460,36461,36460,19
524,36461,38541,34283,28009,32203,32235,19524,30122,36461,36460,34315,23815,38574,385
74,25864,32203,34317,25865,23752,21607,23687,36397,30058,27945,32170,34316,27976,3857
4,21702,25928,21669,28008,36525,28041,28008,21669,40718,36428,32266,34380,32234,25863
,28009,19557,19557,30090,32203,32236,34316,27945,32203,36397,44849,27977,21670,23751,
21670,23783,32202,25896,28041,32202,34380,19589,17444,17444,38574,34380,28041,32234,3
0154,32267,32235,34316,28010,40623,30123,34284,23751,23719,46962,32138,32170,36396,42
767,8960,40654,30089,30089,38573,30088,32233,23782,23814,30185,25928,34347,21701,2166
9,21669,30154,

23784,17444,30090,21639,25832,23752,21638,42768,32171,23751,15300,27945,38511,25
865,34285,15300,36398,15235,11009,32172,21639,13187,25864,38510,17380,40623,38543,216
07,25832,23751,23751,17379,34316,34315,21670,23783,30122,25864,21605,23719,32139,3431
6,23720,27978,23720,17348,27978,23720,32172,36365,23719,17380,11041,44881,27978,23719
,34317,34284,15300,13154,30058,19493,42703,32171,25864,38542,38541,27977,19557,30122,
32267,28041,28008,32267,23783,23815,25896,23783,25896,17412,40655,36461,34316,15299,3
4316,34283,32170,19524,25896,23783,38573,30122,25863,38573,36460,34347,30057,30089,30
089,23750,44848,27976,21670,19525,30090,27977,27976,28009,32202,25895,27976,30090,280
09,25864,38510,23719,19526,30059,19526,15235,27945,27945,25864,32203,36429,25863,3012
2,32235,17476,32234,32234,38573,25927,38606,28008,36492,38605,38541,34380,23750,30122
,25896,25863,21670,17412,30122,34348,34349,30058,17412,32203,38542,19525,42736,13218,
25896,25896,28009,28009,34315,32235,21670,17476,28009,30122,34347,25927,28040,32266,3
4379,25928,17411,28009,27977,25864,32203,30090,21606,23719,42736,36397,40623,23751,36
396,8960,17443,23750,28008,34347,25863,30120,25927,28040,28072,30121,30186,28040,2170
1,34315,23815,

13186,34349,34284,27977,32203,27945,32203,36429,25865,30058,25832,21639,34285,36
398,32172,32139,15300,32172,4736,21639,15268,25832,25832,27977,19493,21638,25865,1523
5,17380,32171,23718,21605,25896,13218,34347,23782,25896,28009,21605,38542,27945,27977
,25800,19461,21639,27913,32172,19494,32171,19494,17380,23751,32171,42703,19525,4736,3
2203,25832,19493,27977,27912,25832,36397,38510,34316,36461,36461,13218,27977,23815,34
315,25928,21670,25896,25896,23815,17444,23815,15331,19525,32203,23751,27977,23751,257
99,23751,15299,32203,23718,30122,30154,34348,23815,30122,36428,30122,17411,23783,2378
2,21669,38574,30089,13186,27977,25896,38574,40654,23783,32235,19557,36429,19525,23783
,28009,36397,15300,25832,40656,25865,8961,36397,23719,30122,30122,28009,19557,40687,3
4380,21702,30121,30121,40719,32267,28008,19589,34347,32267,34379,19589,30154,32299,30
122,32267,23783,32203,34316,34349,40655,34284,15300,27945,30090,21638,34284,32203,406
55,38542,32203,25896,27976,19557,21702,34380,38574,32267,21702,32299,21669,32267,4068

6,30089,17444,25896,34348,27945,32203,38543,25832,27977,36397,27945,34284,19492,38574
,17444,17379,32235,27976,32202,25895,34347,34347,32267,23782,25895,32234,38573,32267,
30154,25895,

13219,21671,27977,34284,40655,19525,36429,38542,17413,44849,21639,21606,40656,19
493,23752,38478,13155,27946,19526,8961,17381,19526,21639,32171,32204,30091,25833,1315
5,34252,27945,36429,17379,21637,17411,34348,19524,34348,30122,23751,21638,17412,30090
,44849,23719,23720,23720,19494,27913,34285,19494,25800,32204,23719,25864,23719,11073,
15300,44881,34284,38510,25864,27945,36429,25832,19492,42768,25896,30090,30122,32203,2
8009,15298,17411,25896,23815,36493,27977,30090,17444,21638,27977,23783,25864,34284,27
945,27912,30090,32203,17412,36461,23783,32235,32202,28009,36428,23782,8992,30121,4068
6,21670,34380,19557,15331,32203,28009,42800,30154,30089,32235,15331,40655,21637,32235
,30090,25832,11041,19526,34317,21638,23751,42768,25864,23751,32235,32235,19525,36428,
34315,28008,19589,34380,38573,30121,13250,8992,38573,21637,32234,23782,28008,32266,23
782,23782,30121,38574,42800,40655,25897,34284,21638,25832,19526,25896,34348,34348,364
29,36461,23783,19589,34348,25863,38574,34380,40687,6912,25928,36492,23814,36525,36460
,34380,32235,25864,30122,28010,34316,36429,30090,25865,30090,25800,30090,21605,40622,
27977,32235,25863,23782,32234,32234,38605,32266,23815,25863,25895,34380,28040,28041,3
8573,30121,

17412,19557,21670,34348,25831,23718,25831,23751,15299,15299,30058,32171,27978,21
606,34317,34284,17381,27946,34285,19461,23720,23752,19526,36430,25832,27978,27946,279
13,27978,21606,19525,25831,21637,28009,30122,34315,34348,34348,30090,36396,34251,2583
2,30058,23719,17413,30059,23720,32139,30059,19526,23720,40623,25832,38510,40687,25864
,23751,36461,38542,36429,17412,21605,15299,21606,34283,25832,30057,30057,21638,23751,
40687,30090,42768,21670,36429,34348,28009,25896,40687,38542,30057,25864,21606,23751,2
5832,27945,27977,27978,30058,30090,23783,34316,30090,15298,30089,23750,30089,32235,23
815,25895,21669,25895,34348,32235,36428,40654,36429,19557,30090,25896,28009,25928,300
90,23784,21606,15300,17412,34317,27977,27977,36462,23751,23783,30122,32235,19557,3434
8,21702,34347,30154,25895,30153,30089,19589,13249,34380,32234,23750,34347,32266,30121
,23814,17508,30154,38574,32235,30122,25864,25864,28010,32203,13219,36461,38542,38542,
19557,32202,19557,25896,32202,27976,34347,34380,28041,21669,34347,30186,38605,40719,2
5927,32267,36461,34316,34316,34316,34284,25832,32171,23687,40655,23751,25799,30057,30
089,30089,38606,34347,27976,28040,42799,32202,25895,28008,23782,36460,30154,28009,237
83,28041,21701,

8992,30090,27977,34348,15299,15267,19492,17380,21606,34284,34284,40623,21639,279
45,30091,25865,30059,19526,42769,32204,23720,23752,25832,40623,27978,19526,21607,3425
2,34285,25800,17380,25831,21605,30090,32235,21670,21670,30090,25831,27944,38509,34283
,32171,34285,42737,30059,25833,23720,25833,27946,19526,32139,38510,34317,40687,36462,
34316,32203,38574,38509,17412,15266,6848,27944,36397,32170,27977,25896,19557,15266,34
316,27976,32203,19557,38542,27977,15299,32235,32235,34348,17412,23718,32203,27977,279
77,38542,27945,25865,25865,30058,21606,27977,28009,13218,34315,28009,36461,28009,1955
7,19589,28041,38574,38541,36493,38606,23783,40655,30122,21638,21670,23816,32203,21670
,23751,23752,21606,15267,25832,25864,32203,34316,19557,30090,36461,32235,30122,28041,
21670,34347,38606,23815,30153,30121,17443,23814,40686,32267,28008,30154,32266,28073,2
1701,32267,36492,38606,30089,27977,23783,32203,30123,34316,15299,44881,36429,34348,28
009,34348,30089,30122,40654,23783,34380,32234,25895,38606,32266,34347,30121,30121,280
08,40687,34380,34316,28009,38575,21638,19525,30090,21606,40623,23719,17380,30057,2797
6,30089,34380,34380,36428,34379,30089,30154,34380,32234,30121,34315,23750,25863,25863
,30121,23814,

27977,19525,28009,30090,28009,13186,30090,23751,23783,30122,40656,23752,11041,23
752,34317,34284,19526,32204,23752,32204,32171,19526,30091,32172,34317,4704,30091,4065
6,23720,15268,6784,19493,21606,34316,30089,30122,27977,25864,25831,40654,40655,38509,
27977,30059,34252,21607,34317,30059,11042,25833,17380,19493,38511,38510,40688,28009,2
5864,30122,34283,25831,23751,27945,19493,30025,38510,30058,27945,23686,15299,15267,34
284,27977,32170,28009,27977,30090,21670,21670,25864,27977,19493,30057,30090,25864,174

12,42769,25832,23752,15267,25832,30058,21606,30122,23783,25896,25928,30122,25895,2170
2,25895,36460,32235,30154,34315,40654,27976,38574,25896,25928,21703,21638,28010,23751
,32236,34284,34316,30090,32235,36429,36461,32235,21702,40719,32203,34348,25864,27976,
21670,25863,25863,25895,32234,38573,15298,28041,34315,25928,38606,34379,34379,32299,2
5927,36460,30186,32267,32235,30090,15299,28009,25897,23784,21671,42768,27977,32235,34
348,32267,38574,19524,36428,25863,25895,30154,34315,28008,25895,40686,25927,30186,322
67,38606,23783,30122,36461,32203,21639,25832,34284,32171,30058,34251,30090,34315,3646
0,25895,36428,30154,34347,30121,32202,36492,36460,30121,34347,44848,30122,34283,36428
,36460,32234,

23784,17444,36462,36429,34316,21670,17411,17444,25896,23783,30090,23752,15267,25
832,40656,40624,23752,25865,27978,32171,34284,23752,32204,27978,19494,17381,36398,342
85,8896,23719,21606,27945,34284,23783,32203,30089,21638,32203,32202,19492,23718,32170
,25832,30091,30059,21607,25801,36366,19494,30027,17348,11009,30059,40623,30090,32203,
17412,30122,27977,32171,27945,40623,27945,19525,32171,23751,27977,23751,19493,21670,3
0058,34251,21638,34316,28009,32235,17412,19557,13218,30089,34316,36429,23751,34283,47
36,38510,27945,23784,27945,40656,42768,19493,34316,36461,28009,19557,21670,23782,1955
7,34379,30154,32234,34315,23750,30089,30122,28041,34348,27977,28042,32235,38542,36429
,36461,27977,32203,34316,40655,30122,34348,40687,30122,19557,21670,30090,21702,34380,
13185,34380,19556,30154,44880,38573,27976,30154,17476,32234,23782,34347,30121,34379,1
9621,30153,36493,32267,30122,30090,21605,30090,34316,23784,19526,30090,36461,34348,30
090,40655,34348,17476,21637,32234,27976,38541,27976,21669,25896,32202,34347,34347,238
15,34348,21702,32203,36462,19558,25897,32203,30090,42800,30090,25864,34284,44880,3217
0,28008,32234,30154,30121,28009,32234,32267,36460,30154,28008,42735,42767,40686,40654
,34347,25895,

21639,27977,25832,23751,30090,25864,19589,34380,28041,28041,30090,28042,23752,27
978,32204,27946,21639,32172,23752,36430,40656,23720,17413,27913,17348,23752,27945,385
43,15267,27945,25864,40623,34283,30057,23718,40622,34315,42768,32170,23783,23783,3428
4,21639,25865,42737,25898,8929,34317,17413,32204,21639,19526,23719,21639,23719,30090,
15299,38542,34283,27944,27944,27977,27977,36429,32171,25832,25864,25865,25864,34284,1
9525,19525,32203,34284,30122,25864,28009,30090,21702,30122,34348,32203,30090,40687,13
186,25896,21638,32235,30122,30090,36429,17444,36396,21637,27944,30090,21637,40654,279
43,40686,25927,44912,34315,28008,27976,34347,30122,28009,28009,17412,34316,38542,3431
6,25896,19557,28009,32235,38574,42801,38542,23784,21670,6880,21670,30122,30154,40654,
11104,23782,23814,28008,36460,30121,30121,32267,13218,34380,32235,30154,21669,32234,2
1702,28041,38574,30089,17444,27976,27977,40622,34251,17413,21606,19493,32203,40655,23
751,23783,23751,21702,23783,44913,34348,34316,23783,28009,25896,21702,30089,30121,238
15,25928,25928,32235,21702,25896,28041,36429,32235,40654,19525,25897,34348,42800,2589
6,25896,27976,36428,34380,34315,23783,32235,32234,25863,27976,28009,42800,36460,28008
,28009,19556,

27977,25864,19493,17413,21638,30123,19589,30187,30154,19557,32203,34316,17445,38
543,27946,19461,8961,8961,25832,40656,36430,25865,23687,32171,25832,40591,27945,32203
,23686,36364,27977,40590,30090,25831,23718,36396,38509,27976,34283,15331,30090,32171,
36430,23784,30091,28010,8994,30059,21607,17381,25800,23752,30058,25832,19526,30090,25
864,28009,34316,32203,32203,32171,34348,34284,32203,32171,27978,32203,36397,23687,343
16,27945,44849,13154,21670,34348,30154,32235,28009,32235,32267,32202,23783,44880,2589
6,19557,21702,30122,30122,28041,32235,17444,40622,15298,30090,38509,27976,25863,32234
,40686,40654,42734,36492,34380,32202,38541,38574,28009,23783,23751,32235,36429,34316,
21638,27977,36428,28009,34316,32203,32203,15299,23751,11105,34348,36428,28009,32235,2
5928,23782,30121,30121,21669,23814,27976,32267,30155,34380,30154,28009,28008,21702,34
348,30122,38574,34348,13186,25896,27977,44816,27912,13154,27977,23719,32171,38575,194
93,19558,27977,36429,30123,34348,30122,32236,28009,21606,28009,30089,28041,23783,2170
2,17444,25896,28009,23815,23750,25928,32202,40687,36461,25928,25896,32203,36429,27977
,28009,28009,32235,25896,38541,25863,28009,23783,28008,28009,25863,38573,30122,32235,
27976,23750,

30090,36430,11041,17412,15267,23751,36461,32267,30155,17477,38574,32236,25865,42769,21574,4736,15235,27978,42769,21639,36397,27945,42737,21606,30058,34284,42736,44849,25864,32203,30057,23686,23718,25831,44848,25831,13186,34283,30057,30122,30122,30090,40688,30091,17445,21639,19558,36430,25833,15300,30059,38510,32171,27978,23751,30058,30122,17380,25864,27977,27977,38542,32203,17412,27945,27945,38542,44882,32171,23719,25832,34316,44849,13186,38542,42768,34380,19525,28041,34316,23783,28009,27977,44881,32203,30122,32203,21670,34348,27977,32202,19557,19524,21637,34315,21670,23718,32170,40654,28008,34315,42767,36460,38605,30089,36428,30154,25896,21638,23783,42768,38542,34348,28009,28009,32202,36428,28041,32235,25896,25929,25864,13186,25896,28009,34348,32235,30121,30121,34315,30121,28008,21637,38573,21670,32235,25896,23750,17476,28041,25896,25928,21637,30154,25864,23750,38542,30090,38477,34284,15235,25865,34316,30091,21638,17445,32203,30090,40655,32235,21638,30090,34348,6880,17444,19557,30089,34379,25895,32234,17444,25896,21702,32235,36493,15298,28041,38573,38574,32203,30090,34380,36429,30090,34316,21670,27976,32235,34315,30089,25896,23783,30121,30122,34315,34348,34315,23783,36428,30121,

38542,27978,19493,27977,21638,15299,34348,34380,28042,15331,34316,27945,27978,38543,13187,8896,38511,30026,36398,32204,23719,19493,38543,27977,27978,21606,25865,27978,30090,23751,32170,32138,25864,36429,36396,32170,25864,30089,30089,28009,23751,21638,25864,17445,23719,25897,30123,25865,27945,23719,38543,25865,25865,25864,36429,38542,15299,27977,25864,40655,23783,27977,21638,19557,17380,42736,32203,21606,36397,25832,15300,42704,34316,21670,34316,40655,34380,15331,42768,32267,21670,32235,23783,42768,21670,32203,30123,30122,32203,23751,25896,13250,30057,34348,38542,21638,25831,25864,32202,27975,34314,42831,32234,34315,25928,38573,15363,34315,30122,21670,28009,34315,32203,30122,32235,19589,23750,21670,34348,30154,30155,27977,17444,28041,32267,36428,36428,30089,30121,32234,25863,32267,21637,32234,21702,30122,30089,11072,13218,23783,21702,25863,17476,32235,23751,30122,38509,34316,32203,25864,19493,30090,23751,42736,25897,21671,34348,25896,40719,23816,21670,25896,30089,9024,21670,32235,38573,25895,23814,30121,34347,23815,30154,34380,32234,19556,30153,32234,32234,38574,30154,27977,38542,28009,30090,15299,32235,38574,21670,28009,21638,23750,27976,34348,36461,23782,27976,23750,30121,25863,

27945,23752,30091,21606,15267,23751,19589,17444,25896,15331,38542,30058,21639,23720,6816,13122,38511,17381,23719,34317,34317,32204,32171,38542,34284,15299,25864,19525,19525,19493,44881,38477,38542,25831,27945,34316,25832,21605,34348,21670,23750,25896,21638,15299,21638,11073,23719,27945,23719,38543,40623,30091,25832,34284,36429,38542,21638,25864,19460,42703,23751,27977,13186,34316,15331,40687,36429,25832,34316,25832,34316,25800,23719,36429,13186,38574,27977,19557,36429,17444,23783,23783,25864,38575,13186,13219,30123,34381,32267,17476,36428,17444,32203,42768,42768,27977,25864,17444,32170,30121,34315,38605,28008,36428,28008,36460,28041,32267,34315,19556,36428,40686,36461,30089,30154,25895,9024,25928,32267,32235,38542,23815,19525,27977,38542,40687,32235,28009,30154,32202,25863,32202,23815,25896,32202,27977,25863,11105,23750,25928,34315,34348,21670,36429,25864,17477,27977,34316,25865,19493,23719,30090,23719,42736,32203,19558,27977,17412,34348,25864,21670,25896,32235,19524,34348,36460,30121,23814,32266,36460,19589,30121,36460,30121,28008,32266,30089,32266,32235,36428,32235,25896,28009,32236,42768,25896,30123,40687,15331,34316,19557,23783,32235,36428,25928,30089,25896,30089,30154,21669,

32171,21639,32204,25865,19526,34317,17412,30090,25896,11073,32204,21639,8961,23752,27978,19526,34285,25833,32172,30091,40656,25864,32139,34251,17412,15234,38510,23686,23686,17347,36396,27944,32138,25831,36364,38510,15266,30090,30089,30122,32202,38574,19557,21670,30123,13154,17380,27978,30090,32203,38510,32171,32171,25896,25864,42768,27945,34251,34284,34251,8960,27977,36429,34284,40655,30123,21671,40688,42736,30091,34284,25832,34284,25864,27944,38542,28009,23783,42800,17412,21670,32203,34348,23751,17445,13186,36462,23783,36461,27977,34315,36493,19492,25896,40655,32235,21638,19492,28009,32202,38573,34379,32234,19556,25895,32234,25928,30089,23782,17443,42800,34380,34347,21701,23815,32234,21669,34347,30089,30122,30122,25896,15299,34348,34348,38606,32235,30122,42800,34348,30122,25896,23782,38541,32202,42800,27976,17444,25863,28009,38541,3

6493,32235,28041,28042,23783,21638,30123,27978,23719,27945,32139,19493,44849,19525,21638,13186,21670,38574,25896,27976,27976,25896,25863,38541,34315,28008,21701,36460,38605,19589,32234,32267,32234,34347,34347,34379,32234,30121,28041,38606,36461,23719,36429,36397,34316,32236,34316,25896,30090,27977,13218,32203,38541,19557,28009,28009,27976,25895,27976,

25865,23719,27978,21606,25865,32171,19493,36430,25832,23719,27945,27945,23687,32139,42769,25833,17413,25865,25833,17413,25832,21639,32171,23719,23751,27977,27945,21638,25896,8960,36429,38542,38510,15299,36429,13154,32171,23783,25896,23783,28041,36429,21670,23783,36429,4768,21639,30058,27945,36397,19525,32171,25832,15267,11041,30058,38477,27913,30090,36429,25832,32171,36397,23751,25865,25897,28010,32171,28042,34316,23752,32203,34316,34284,38574,34283,30090,34316,40687,30122,27977,28010,30091,30123,21606,34348,42768,23783,28041,27977,40687,32202,19557,23751,27944,34316,23751,32202,30089,34347,42767,32202,36428,15330,28041,32266,23815,42799,21669,30121,36460,36460,28008,32267,25863,40718,28041,32266,32235,38574,36493,32267,30122,34380,38574,25896,32203,34348,38574,28009,23783,15331,15331,32235,32203,36396,25863,15331,11105,34315,32235,25928,28009,30122,25864,23751,21671,23784,21639,17412,25864,32171,21638,44849,17445,28009,28009,17411,36461,28009,34315,36461,21669,36460,38606,30121,25927,28007,34347,32234,25895,32234,34379,34380,28008,30154,38573,30153,21669,34348,40687,34349,30058,27978,34317,32236,36462,25897,32236,34316,21670,17412,36461,36428,21670,25928,32234,23750,21669,23782,

19526,23752,30091,30058,25865,19493,25832,27945,30026,30091,27978,8961,19493,27946,32172,21607,25832,25865,21607,19526,19526,21639,34316,6848,25832,17412,23751,23751,30090,21638,13186,30122,27977,15267,36429,17380,13218,34316,25896,25896,32267,25896,27977,36461,34348,6848,15267,32171,25865,25864,15267,32203,19493,19461,11009,36365,30025,19493,25832,40623,36397,42768,21606,30090,28010,27945,15332,36429,25865,17412,17380,21638,30090,38510,44848,30057,23783,23783,32203,30090,25864,34316,38543,27977,27945,42768,38575,38574,32235,30122,28041,34348,30090,32170,13186,40655,30057,23751,21670,34348,42767,25895,30154,25895,32202,32267,38573,32266,40718,25895,30089,23782,28041,27976,25928,30154,38541,30121,32234,28009,32235,30122,36429,30122,32203,19589,34316,40655,36461,23783,23783,32203,21670,30089,32203,32203,25863,23783,19524,36461,25896,25928,23783,30154,17412,17445,27977,21638,25864,13186,19493,32204,23783,38542,32235,34316,28009,19557,28009,30154,34315,23782,15330,36428,40654,23782,30121,30153,28041,34347,30122,28009,30154,25895,32234,25863,21669,30121,34380,36461,30122,30155,27945,27978,28010,30091,27978,34316,34316,32203,15299,23783,34316,30122,30122,30122,23783,27976,25928,25895,

36398,23720,19526,23719,11042,27945,27978,25800,32171,34252,17380,8961,23720,27978,30058,32204,38510,40656,21639,17413,17413,19526,23752,23719,17380,25832,27977,23719,42768,32171,13154,21638,32171,19526,30091,36429,21606,25864,27977,34316,32235,15299,30122,38607,40688,30091,23719,21606,30058,19493,17413,36397,27978,8896,30026,30058,25833,17348,32171,34317,32171,34284,36397,19493,28010,27977,30058,42801,25897,15299,21606,23751,25864,38477,36397,23751,19557,25864,23783,21670,17412,28010,44849,27945,19526,21638,40655,32235,40687,40687,19524,46993,36396,17412,19493,36429,32235,13153,19557,34348,32234,25863,25895,28008,34380,34315,19524,32234,32267,17475,25895,23781,36460,30121,17475,28040,34347,34347,30121,32235,38574,23783,30122,19557,30090,15331,38574,30090,30090,15299,30090,38574,32202,34348,27977,19525,27944,36396,25896,19557,21670,23783,15331,19557,6912,13218,19590,32203,21671,17412,25897,30123,32236,32203,32235,32235,19557,36396,34315,30057,32202,25895,32234,36492,25927,23750,34347,40686,32234,38541,40654,36493,38574,23783,44880,25895,32267,38574,38574,42800,27977,19558,19525,30091,21639,25832,21639,30123,36430,30090,27977,30122,36461,30122,44880,23815,32267,30121,32234,23782,

21639,17380,27946,25865,21607,27978,23687,44817,27978,25865,19493,25833,27945,23719,32171,38543,30058,27945,36397,19526,19494,17413,19493,21606,34316,40687,21638,42736,25864,44849,28009,6848,36397,13187,32171,36397,36397,23783,28009,21670,28009,8992,32236,25897,40688,34349,21606,32204,17413,19526,21574,27945,8896,15267,32171,38510,30

026,19461,36365,25864,27978,27945,32171,15299,23751,23719,32204,38510,25897,17380,364
30,25832,30058,38510,15299,27977,34316,27977,15299,21638,28009,36461,40623,23719,2794
5,17412,25896,40655,30090,38541,23750,25928,34348,21605,32203,34284,27945,19493,17412
,25864,32235,32202,25895,25928,30121,38541,32234,23782,30121,23782,32233,30153,34379,
25895,19556,23814,32266,36492,25928,36493,38574,30122,30090,17444,28009,19557,28009,3
8542,21638,25864,19525,32203,25864,32235,23750,23751,25863,32202,17444,23750,13218,34
348,25896,28041,15299,17444,25864,21670,17445,36461,30090,32203,34316,27977,27977,322
03,17412,44880,40622,36428,28008,30089,40686,44912,27976,34379,34380,38574,21637,3646
1,42832,21702,44880,36461,38541,25863,36460,34348,36429,34348,13219,19526,25833,32204
,19558,19526,19526,27978,34284,25896,34349,36461,36461,32235,40687,27976,28009,40654,
36428,23782,

23687,30091,21574,30026,17413,36365,40624,40591,42704,8928,25832,42768,23719,237
19,30058,11073,23719,11041,19493,21606,8929,30026,19493,21638,21638,27945,30122,36461
,42768,28009,25864,512,15300,21606,34317,32171,32204,11073,19590,25929,30155,15299,32
203,34316,38510,19525,27945,25832,17413,15267,27913,27913,32204,38510,34317,34252,173
81,34252,36397,21606,46962,17413,8960,17380,21671,32204,38575,40656,30058,25897,32204
,32138,42736,34284,21605,27977,25864,27977,27977,42768,34348,36461,23751,27945,23719,
21638,28009,25864,21670,17412,21637,23751,21670,21605,40623,32171,34317,23751,21670,2
1638,40687,32235,25895,30121,30153,30153,23814,19556,28072,36492,34379,28040,28040,25
927,25927,28040,21701,21702,28041,25928,38574,28009,17444,21670,38575,30122,21606,343
48,15332,23783,28009,32203,34316,13218,21670,34283,27977,25864,25863,23750,21670,3223
5,30089,28009,21638,21638,32235,28009,27977,32203,32235,23783,25896,34316,32203,32203
,17412,32235,27976,36460,25863,32169,42767,32234,34379,42767,36428,36461,17444,34316,
36428,36461,27976,38574,25895,38541,38574,23783,23751,25896,19525,15332,23752,30091,2
3752,25897,27978,30123,21638,25864,36429,36429,40655,34348,34348,28009,21670,36461,28
041,30089,

36430,36430,25865,25833,27978,36398,42769,25800,19493,32171,27945,42736,36429,23
719,17412,15299,36397,15267,23719,21606,17381,32204,17412,21606,25864,23719,21638,195
25,25896,17412,21638,13154,17412,38510,38510,30058,42736,25864,30090,40655,34348,2163
8,38607,21638,40688,34349,25832,25865,19493,21639,42769,32171,25833,36430,30058,32139
,23720,34284,36430,25832,21606,6848,11041,8928,30090,40623,36462,34317,38542,30058,34
317,32139,34284,36397,19525,32171,23751,19557,36429,34316,25896,38574,28009,32171,280
09,21670,30122,34348,23751,21637,30089,23718,30057,34316,38543,25865,25832,25864,3220
3,17412,36429,34380,19621,34379,30185,21733,25927,19588,38605,36460,30153,21733,25895
,34347,15362,19621,21734,28008,23815,32235,25864,25896,23783,32235,32236,38542,23751,
25864,25864,21638,30122,38574,42800,23783,32170,36429,25831,25896,34348,36461,30089,2
5928,25928,23783,25896,30122,23815,25896,30090,25929,38574,17412,25864,34316,23783,32
203,34315,32203,21637,38541,27976,32202,38573,21669,36428,40719,40687,36461,34315,237
83,28009,36461,32202,36493,38541,23782,34315,21637,23783,32235,21671,32204,27978,2375
2,30058,19558,32236,34317,23752,25896,34348,32203,34348,40655,27977,17444,25896,23750
,27976,36428,

27913,17413,13187,30027,42705,34252,32171,8928,25832,21606,36429,40687,23751,322
03,23783,17444,21638,21606,30058,25833,19494,25833,17380,30058,25832,13154,19526,1529
9,30091,27978,34284,23719,23751,23719,27945,34316,21606,23783,25896,32235,36429,28009
,23815,13186,36429,30122,21606,27945,13154,21638,36430,44849,27913,36397,21607,21607,
15300,36430,17380,30091,25865,17380,13154,38542,27978,38510,21639,30058,36397,36397,2
1638,27913,27977,42736,28009,36429,25831,30090,32235,25896,17379,42800,21637,15331,13
186,21638,25864,36461,17444,21638,34315,28009,27944,32203,27977,27945,15300,23719,301
22,8992,30122,25896,23815,34379,30153,30121,30185,23846,36492,36492,30153,28008,34379
,40718,15330,23814,32267,32266,17476,30154,34348,32235,25864,38575,36462,42768,27978,
25864,21671,25864,34348,42768,42768,30090,42768,38509,23783,27977,42767,38574,30121,2
5895,25928,25863,32235,32202,28041,32235,32203,21670,28009,38607,19557,34348,17412,32
235,40622,34283,23718,44848,34315,40686,36428,30057,21670,32267,38541,25895,32235,237
83,40687,23783,19589,30089,38541,21669,27976,23782,30122,30090,32236,36462,40688,3009
0,25865,30123,25864,34349,21670,34348,32235,32235,28009,36461,32203,25896,21670,28009

,38541,36461,

34285,27913,17413,32140,38511,25833,15267,23751,23719,17412,21670,34348,32235,25
864,30122,13186,30122,23751,30026,25832,27978,27978,17413,44849,30058,6816,30058,2375
1,21638,23752,25865,42769,32235,21638,27945,11041,19525,30122,19557,32235,30122,25928
,13250,19557,27977,30122,19525,17380,6784,25864,38542,27945,17380,42736,34252,21639,2
3752,32204,34285,36430,21639,13187,30090,36430,32171,36430,30058,32204,23719,36365,34
284,25832,38542,44849,36429,30058,17411,38509,32235,15299,21670,25864,25896,23751,321
70,32203,30122,34316,25896,23783,32235,32203,30057,28009,21606,17380,6848,25832,25864
,11106,36461,17444,19588,23814,30153,40750,23781,38637,38637,32266,30185,13249,34347,
40718,30153,36492,36493,36461,23783,32235,25928,21670,32203,32203,40655,34316,28010,2
5864,19525,17412,23751,42768,38542,42768,36461,38574,34315,30122,42800,34348,28008,34
380,38606,32202,36428,34380,25896,30122,28009,27976,23783,42832,32268,36429,25864,301
22,34348,27977,27944,40655,23782,38574,42767,17411,25896,23783,25928,15298,23750,2800
9,36428,19557,21702,34315,28008,25895,34315,27976,32235,28009,36462,34349,38575,38542
,27977,19526,21671,23751,28010,38542,27977,34316,25864,36429,19525,21670,23751,32202,
34348,25895,

36366,21607,32172,27914,15268,13155,25865,34284,32203,19558,15299,32203,34348,11
105,25863,36461,36429,23751,25832,32139,46931,27946,17413,25864,23752,13219,30090,300
58,23752,21606,32171,42801,25896,30123,21638,11105,27977,28009,19557,28041,30122,1955
6,17476,23750,38541,27977,17444,36397,30090,36397,36397,27945,32138,40623,44850,19526
,21574,19526,23752,30091,19493,13154,34284,34284,40623,40656,38510,23719,19526,23752,
34251,21574,25832,40590,46962,38477,21605,40687,30122,21637,38542,17444,38574,27944,3
4316,32170,38574,30122,34315,23783,23751,32203,32203,25864,23751,11041,15267,40623,17
445,4736,23783,28009,32266,32266,21668,34379,30153,32266,21668,32266,21701,28008,3860
5,19556,34379,40686,28041,38606,36461,15331,23783,28009,21670,19558,30155,23751,25864
,32203,19525,27977,23783,40687,32203,25896,25896,32203,42800,28009,40687,34348,25895,
42799,30121,32234,32267,40654,19524,44880,30089,25896,34380,28009,23783,32235,25896,3
0122,38509,13218,34316,40655,25863,27977,28009,38574,34348,30089,25863,25895,25895,25
895,21637,40654,13217,23814,27975,32266,40653,30153,32202,40655,38542,28009,28010,322
03,27977,23751,38575,28009,36429,34316,25896,28009,21670,25864,19557,34348,30122,3642
9,25863,42767,

13123,36398,27946,13187,13123,25833,40623,38574,21638,25896,25864,28009,30122,13
218,11105,25864,32203,30091,34284,27946,34285,19526,17380,34317,17412,21638,30123,322
03,23784,23751,32203,30090,27977,27977,15299,23751,21606,21670,32267,19589,28041,2170
2,17443,32267,19524,11072,23783,40655,30090,36397,38477,30058,42704,34252,32171,21607
,32139,27913,21606,30058,21606,17380,34284,38510,36430,38510,40591,17380,21606,23720,
23719,32171,19525,40622,44881,27944,23751,32203,36428,27944,34316,27976,38542,15299,3
2202,25831,38542,38541,30090,28009,17412,30057,36429,27977,17412,11073,30058,32204,25
832,19558,21670,30122,36460,36492,25895,32234,28040,34347,8992,30121,28040,32234,3646
0,25895,42832,38606,34380,34347,30122,11105,28009,25864,25897,23783,32268,32236,32204
,27945,17445,21670,19525,36429,34348,25896,25896,36461,30090,36461,30154,36493,19589,
36493,32267,36460,38606,38606,23750,44912,30057,25896,36461,30122,34380,32235,32203,3
2203,30057,19525,30057,34283,25864,23750,25896,36461,23750,32235,30121,34347,38573,32
202,34347,34347,23750,23814,36427,36460,42799,40686,25928,40687,27976,19525,23751,301
22,28009,38542,36429,38542,38574,25864,21670,25896,28009,30090,30122,30090,30122,2589
6,28009,34348,

2592,36430,30091,27945,8896,25832,32171,30090,15299,32170,32171,23718,30090,2375
1,32171,34316,30091,19493,34317,23720,21607,19526,13154,21606,6816,25832,25896,40655,
25897,28010,32203,40687,36397,27977,13153,23718,8960,25864,30057,32202,32202,15331,25
863,32234,19556,19589,32203,42832,25864,38510,30058,34317,27913,21574,30091,8897,4270
4,30059,23720,30058,34284,34284,27945,30091,23719,34317,42736,19493,30059,36365,13186
,30090,32171,44881,38509,40655,32170,32170,27977,23751,34283,40622,42768,38574,30122,
25928,28041,32235,25896,28009,30090,17380,27978,23719,6816,17380,36429,27977,23751,25
864,25863,38574,32234,34379,25895,27976,30121,21702,23782,38605,38605,30120,38573,406

86,40654,32234,34380,30089,34316,23751,19557,25864,32203,25897,38543,28010,23784,2589
7,21639,27977,28010,40688,32235,30090,32203,42800,28009,36429,30089,30154,30122,30122
,36493,42800,42832,42800,28041,36428,30090,23750,32235,34316,40687,30122,23751,30058,
17412,17412,27977,23751,25864,23783,21670,25896,30090,38574,34348,42735,42800,27976,3
0154,28041,40719,30089,36492,34347,40686,40686,23782,40654,17412,23750,28009,34315,30
089,32235,25896,42800,36429,32203,30089,23783,30122,30122,25864,25896,30090,25864,237
51,27977,

15235,17413,30091,27946,23752,21606,17380,34316,25864,30122,34284,13186,21638,40
623,30090,21639,25865,30091,23720,17381,30059,27978,30059,32203,13187,25832,25897,301
22,30090,17445,32236,36429,19525,13186,21638,19492,23686,23751,30057,30057,28009,2170
2,36428,30154,30122,30122,42832,36493,17445,42768,40623,21606,8896,8929,19494,19526,2
5800,36430,21639,27945,34349,27978,23719,11073,30058,27945,42736,23752,15267,38510,15
300,21606,34283,34316,40654,30057,42768,32170,23751,36396,32170,34316,40622,42767,322
02,23783,36428,36461,30122,40655,36397,21606,19461,21574,15267,21606,34284,32171,1741
2,30089,23751,30122,25895,21669,19524,23782,34380,21670,36493,30153,38605,30153,30120
,32233,32202,34379,36460,28041,30122,28009,15332,17444,32204,21671,30091,27977,21606,
27945,25865,32171,21638,25896,28010,23783,32235,40687,30090,38542,23783,25863,30089,3
6428,34348,30089,40719,36461,21670,36429,27977,30122,27977,28009,40687,27977,23719,38
510,15331,21670,19493,30090,32235,25864,19557,21670,32203,32202,30122,40622,42799,152
98,17443,25895,38605,34347,36460,25927,38606,40718,21670,30089,13185,42767,25863,3854
1,32202,30089,34315,32202,28041,32235,38574,28009,32235,15299,32203,30090,30090,30122
,19557,27977,

17445,25833,13219,25864,36429,21606,25832,36397,19525,38510,42769,21638,36397,25
832,13187,23752,19526,15300,23753,17381,36398,27979,25865,27978,23719,34317,15332,688
0,30123,34316,25864,17445,36397,25831,30058,23719,42768,25831,30057,19524,19524,34315
,23783,25928,34380,30154,32235,30122,17412,34316,34284,32171,15300,17380,17316,21607,
21607,25800,15300,25865,21606,17380,19493,30090,34284,30058,32203,40623,34284,25864,1
9525,27977,23751,38542,30057,30057,40655,32203,17412,27977,36429,32170,23783,38574,38
541,28008,28041,32235,32267,25896,30123,23751,21639,27978,30058,19526,23752,28009,237
18,32170,23783,38541,23782,23815,28008,28008,21702,23815,38541,30154,42799,25894,3012
0,40685,32266,23782,38606,23783,25863,30122,19525,30090,32203,19525,34284,25897,25864
,15332,32236,34349,21671,30091,25896,23784,30090,38574,30122,42768,30122,19557,34348,
32235,23815,32234,38574,36461,6880,34315,23751,36461,17444,17444,30090,23750,32171,27
977,17412,17412,30058,32203,23783,25864,6880,23783,32202,19557,34315,32202,27976,3008
9,19556,28008,32234,32234,36460,19556,30154,42799,25863,34347,25863,27944,36428,34348
,34348,32202,30122,30089,32235,32235,40655,27977,34348,13218,27977,36429,32203,30090,
23783,27977,

28010,23752,19558,28010,21606,32236,32171,34316,19493,11074,32171,34284,8960,174
13,8928,19493,25833,19494,19494,448,17414,34285,34317,32171,21639,34284,21639,21639,2
8010,28010,25897,34316,21638,42736,42736,40622,32203,13186,27944,11073,19525,34348,38
541,21702,36493,32235,28009,34380,23784,23784,21671,32204,30059,27945,19461,19494,300
59,19526,17413,25865,17380,17413,15332,25864,30058,30058,17412,44881,27977,21638,4062
2,28009,27977,30090,36396,32170,27944,27912,17380,21606,40655,25896,19557,38573,44880
,34380,42832,36461,25896,30154,30123,21638,30058,21639,30091,36397,25864,40622,19525,
32202,19557,28008,27976,36460,34380,25896,13218,30154,36460,38606,30120,25927,32266,3
2266,25895,40719,36493,23750,25896,30090,32235,25896,25897,30122,32203,30058,21671,30
058,38542,32203,25897,30058,25896,23751,28010,32235,27977,30090,30089,17444,38606,364
61,25896,25896,34315,25863,19557,38574,19557,25896,34316,19525,32203,27977,34316,2375
1,21670,19493,42800,32235,23751,23751,17412,23783,30122,30089,25831,25896,38573,30089
,30089,30089,23782,32234,32234,28008,30153,38573,25863,25895,30089,34315,38574,36428,
32202,25896,21637,32202,25863,32202,38542,23751,34316,30122,23783,28009,25864,28009,3
0090,19557,

25897,27945,19526,23752,21638,21671,42736,30091,21606,34284,17380,15300,21606,30
058,23752,23720,30059,30059,15236,480,27947,34318,21639,27946,21639,23752,19493,15332

,17445,28010,36429,25832,32204,36429,44849,38510,13121,21605,38542,17444,25864,21702,30154,34348,28009,21702,13218,36461,21638,34316,19558,30123,34317,25832,19461,30059,19494,19526,27946,23720,17381,17380,17380,19525,30058,23719,21606,34315,32202,36428,36396,30090,25896,11105,30122,30122,36461,32203,21638,23719,32203,34316,11105,34315,34347,30121,40686,36461,32203,34380,30090,13218,19525,32203,34284,32203,27977,38574,13218,40654,23783,17476,32267,40686,32235,23782,23815,30154,36493,30153,32266,34347,34379,34346,25863,38573,28008,28008,25863,32202,28009,32203,30090,17412,34316,32203,21638,30058,30123,34349,27977,19558,32203,25896,21638,23783,32203,23751,25896,23783,30122,36461,21670,21637,25895,25928,30089,34315,23782,30121,34348,17411,38574,25896,38541,23751,28009,19525,36396,36397,32171,15299,8960,15266,32203,40622,19524,30089,38541,28009,27976,34315,13217,19524,30154,23782,30121,34347,30122,30121,27976,40654,34348,32267,19557,19557,19557,34348,25896,23783,27977,23751,30122,36461,30122,28009,27977,27977,28009,25863,

38510,27977,30090,30123,19525,32171,25865,38510,27945,36397,25865,23752,15267,17380,36430,17413,30091,36398,8929,11075,23753,19494,25833,19493,17381,23720,27978,17412,21671,23784,21606,34316,27945,40656,36429,21606,15267,13186,30058,13186,23751,17444,30154,28041,15331,23815,21670,34348,15331,21638,23784,36430,36397,17380,21606,25832,36398,30026,13155,19526,15267,21607,15268,21606,19526,21670,15331,30089,36428,32202,38606,28041,28009,21670,40655,40655,32203,25864,23719,23751,38510,27977,19589,34380,36493,25927,36493,38573,34348,36461,30090,15332,21671,30091,32203,23752,34316,32203,23751,32203,25896,23783,9024,34347,36460,25863,19589,34347,34379,13249,34315,38573,23781,36460,32266,32234,17411,23750,25928,42800,36461,40655,27977,17412,27977,23751,21638,28010,30123,34316,30090,27978,34316,30090,30090,23784,25864,30122,28009,23783,17444,40654,21637,15363,34315,25928,30089,38573,30154,36460,30121,23782,34380,19557,32235,30090,19524,32203,44881,32203,15266,15299,13218,19525,32235,44848,34316,40654,23750,21669,38541,30154,11137,30121,32234,17443,27976,30121,25863,30154,32202,38541,30154,32235,25896,30122,17444,25864,38542,25896,21638,32203,30090,30090,15331,15332,23783,23783,23783,36428,

42769,32236,36462,36462,21671,30123,23719,32171,25832,30090,30091,34284,32171,19526,30058,17381,40656,30026,8961,8961,40592,25833,21639,17413,19461,17381,21606,25865,38510,27945,30058,27945,19493,27977,32171,32171,38542,11073,15299,19557,21670,25929,23815,17412,28009,30122,23815,32267,23751,23783,32203,38575,27977,19525,17348,23654,34284,32107,15235,19526,17413,23752,30091,30058,28010,40655,34315,30122,40686,30089,32202,36461,32202,32202,34316,34348,23784,21638,27978,27977,32203,25897,34348,36460,28008,30121,23782,34380,34380,36461,25929,27977,19525,30122,34284,21638,28010,30122,17444,19589,36461,30089,25895,32234,36428,32234,13185,25928,23814,28008,25928,25895,30121,34347,30154,17475,15298,21670,32202,42799,30089,36461,36428,30090,23751,32235,17445,21670,25897,38542,30123,30123,38575,36429,28010,34316,28010,28009,25896,28009,19557,32202,15331,11105,34347,17443,32234,27976,32202,25863,30121,34315,30089,21669,36461,23783,15298,32203,36429,36396,21605,21573,30057,27945,30090,42735,27976,30121,27944,27944,38574,28041,23782,34347,25896,23783,27976,30089,25895,23782,27976,36461,34348,32235,34316,34348,17444,30122,38574,25864,27977,30090,23751,32235,23751,21670,23783,21638,23783,36428,

25897,32236,32236,34316,13219,34381,27978,27977,40623,36397,34316,40623,40623,25865,15267,19494,25865,23752,13155,27978,30059,19527,4672,25865,38478,27978,21606,34284,36430,30058,25832,15267,30091,42769,19461,42736,25864,19493,15300,27977,19525,30122,30154,30090,25928,23815,25896,32268,30090,25864,44881,40688,27977,17413,15267,17380,25832,32171,27978,30059,17381,27978,34285,38543,36462,44913,25864,34315,28041,32234,25928,34347,40687,25863,32235,27977,23784,30123,11106,32236,25897,30090,25896,40686,38573,32234,9024,32234,40719,34380,21670,25896,34316,32236,27945,21638,36429,32235,19557,15331,13250,34315,36460,23815,30154,34348,23815,28041,19589,25928,21637,34347,34347,30121,28008,19589,25896,34380,34348,36428,23783,34348,38574,25896,28009,30122,17444,23783,25864,30090,23784,30090,30123,30090,23783,27977,30123,23784,28009,28009,21670,30122,25896,17411,17476,21637,30089,32202,21701,27976,32234,28008,23815,23815,32202,28009,25831,36461,32170,36429,38510,27977,32170,30058,25832,27977,19525,32170,30089,300

89,40687,32234,40654,19557,21669,36428,36428,28008,23815,21670,25863,36493,34348,2378
3,36461,36429,15331,44913,38542,25896,34316,34316,25896,28010,23783,19558,25864,28009
,32203,28009,

6912,21671,30123,32236,13251,21671,17380,19493,25864,38542,21606,40623,44881,279
45,19493,19493,23719,13155,21606,25865,38511,27978,25865,32204,32236,25865,21574,1738
1,40623,25832,23719,19494,23752,34284,19493,40624,23720,27945,15267,17445,23751,38542
,21638,19590,21703,25864,30122,32203,21670,28009,36461,36494,17412,25832,27977,21606,
23687,36397,30026,34220,21607,27913,27978,34317,42768,42768,19557,23815,36428,23814,2
3750,38574,34380,27976,23751,19557,27977,17412,25865,40688,23751,30122,30154,38606,32
267,17476,19556,28041,40719,34380,19557,30155,21671,34316,30123,21671,30122,25864,280
09,13218,23783,17411,23815,23782,30089,34348,17411,32235,17476,32267,34347,30121,3649
2,28008,27976,23814,27976,28009,27976,25928,30121,28009,30122,21670,32235,30154,19557
,25896,25896,30122,30155,23751,25864,30090,11105,15331,36461,23751,27977,25896,23751,
34348,34348,17444,32202,21669,32234,32202,34315,34315,21702,21669,19556,40686,32234,3
2234,30154,38574,30122,32235,36396,27977,38509,32203,25799,40622,19492,36396,27944,34
283,42767,27944,30122,23750,25863,40654,36460,21669,30089,13218,32202,34348,32202,322
02,38541,32203,28009,34348,30090,32203,34349,25897,19558,15332,25897,21670,25896,3012
2,36428,25863,

13219,32236,27977,28010,19525,25865,21638,40655,23719,21638,21605,23751,21638,40
687,32171,32171,6816,19526,17413,21639,17380,34252,27978,28011,27946,25898,23752,2586
5,34285,21606,25833,30091,36397,23719,21639,38510,23720,32204,17413,25864,36429,27977
,11106,13219,21703,30090,19525,30090,28009,25896,30090,23751,30122,21606,21639,23719,
13122,34252,30026,23720,27946,19526,21639,19493,32204,32170,21638,30154,34347,19556,3
0121,34347,30121,30154,21637,34348,19557,15331,32203,38575,27977,21670,23816,30121,30
153,34347,25895,25896,38606,28008,30122,28009,25864,42801,25864,23784,25864,34316,322
35,8992,17444,19589,21670,21702,32235,25928,21637,23815,25928,32202,38573,34315,34347
,23782,17476,25895,32235,17411,13250,28041,34315,21669,36461,28009,28009,32235,17412,
23751,28041,28009,25897,36461,27977,15299,13186,25896,30123,34348,21670,28009,34316,2
8009,27976,38573,34315,21669,38573,36460,44912,28008,23782,42831,32267,34380,30121,32
202,32234,30089,32202,23783,30057,27977,38509,36396,32138,30090,19557,34315,25831,343
15,34283,19524,23750,30089,23782,27976,40686,17443,28008,19524,32267,34348,34348,3642
8,21670,32202,17444,28009,32236,27977,17444,17412,21670,21638,25864,40655,17444,32235
,38574,17411,

11073,36429,13186,19525,21639,19525,27977,32235,8928,13186,17379,19525,17380,132
18,32235,21638,17380,34284,27945,25865,23752,32204,38576,30091,25865,17413,21639,4062
4,36430,19493,15300,32139,34317,40591,34317,34317,40623,34317,25832,23719,36397,23784
,23752,23783,25896,28009,32235,28010,25864,32203,27977,30122,27977,21606,30026,23752,
21607,27946,19493,25833,30091,17380,15268,21639,19493,23718,32203,36428,30089,23750,4
0687,36461,28041,34348,17444,27977,36429,32203,32235,25897,25896,17445,19590,30122,28
041,34380,30154,32234,40719,38541,36429,28009,28009,36429,25831,21606,27977,23751,406
55,13186,30122,34348,28009,25895,25896,34348,25864,30122,30154,30089,30121,25895,2800
8,23782,17443,38573,30089,15331,17443,19589,38606,21669,32267,25928,28041,28041,17476
,21637,34316,30122,25929,34316,34381,19558,27977,28009,21670,27977,25896,32235,32203,
19525,32202,36493,42799,28008,42799,25895,42799,21669,34347,34347,30154,30153,40686,2
8008,36460,25927,28008,17444,32234,34315,40655,21637,25864,32203,30089,40654,30089,23
750,34283,27976,30057,23750,23750,32170,40719,21637,23782,28008,28008,28008,32235,322
34,28041,38541,19525,32235,32203,30090,13219,19558,30090,34316,27977,36429,15299,2800
9,34348,21669,

15267,23752,25864,23751,23719,23752,30090,27944,17379,19525,38509,30090,21605,36
429,32171,30090,19525,36397,30091,30058,25832,32172,42737,34349,15300,11042,21574,363
98,25865,19526,11009,27946,30059,30091,36430,36430,36430,27978,30091,30058,32171,1952
6,34284,25864,21606,19557,30122,27977,28009,32235,23751,38574,38542,30090,34284,21574
,27913,21574,27913,30059,27913,25832,19526,23719,19493,21606,25896,36461,25896,17412,
42800,30090,36461,23783,23750,21670,38541,28009,23751,21670,36429,38542,6912,28041,30

122,32267,32267,38606,42800,36461,30122,34348,21638,23783,27945,13218,34348,38542,427
68,30090,32203,25896,23750,32202,21670,36493,21638,28009,21670,27976,36460,34347,2592
8,32202,21702,34380,27977,28009,38574,32234,32267,21669,25896,30154,36460,21669,25863
,27976,38574,40655,23815,38574,34349,19557,32235,15331,15331,21638,32203,32203,28009,
23783,32235,36460,38606,36460,21669,25927,34314,15362,30153,32234,25895,34315,23782,2
8008,32202,25928,32202,30089,34380,42799,34315,30089,30089,44848,32170,36460,27976,25
831,42767,32202,34316,34316,32202,40655,30122,27976,15331,25895,32267,27976,38541,301
21,27976,34348,25896,23751,23751,19525,21638,34316,30090,32235,25896,23751,23751,3226
7,27976,21669,

30123,34317,23784,17445,17381,32204,32203,19557,23751,34316,17380,34284,32235,30
122,34284,13218,34284,34316,21606,21639,36398,42769,30091,30124,11074,13187,36398,236
87,19494,23687,15267,34284,19526,32204,32171,23719,17412,21639,21639,38511,38510,1738
0,30091,13154,21606,25832,23751,40623,34283,30090,28009,21638,32235,32203,21606,19526
,19493,27978,46963,23720,44849,27913,21606,34284,25864,27945,30122,30058,21670,25896,
32203,25864,36461,11105,28009,30121,36428,30122,30090,21638,32203,15331,19557,25896,3
4381,28009,28009,38606,42832,40654,32235,28041,23751,23783,32203,19493,36429,44913,25
864,34283,25864,32203,32202,25896,17412,30122,23751,23751,23750,19524,32234,30121,258
95,34347,32234,32235,19557,25928,38574,30154,34315,30121,30121,25895,30121,34347,3012
1,30121,40687,30154,38606,40719,25896,30155,30090,19557,15299,25896,38542,34315,27976
,36461,30122,38574,32234,34347,15330,30121,32234,11104,25895,34315,19556,25928,30121,
34347,32266,36492,23815,17443,34315,38573,38574,30122,34348,40654,27976,34315,34283,2
1637,38509,23718,34283,27977,23718,40654,40687,23783,21669,34348,30089,25895,32202,34
315,32202,28009,27977,23751,27977,25864,30090,44913,36429,34348,36429,36429,30122,427
67,21637,13218,

42768,40623,15267,19558,34317,19494,23719,25865,36430,21638,21639,44881,30090,32
171,30090,34284,32171,32204,23752,30059,21639,36430,34317,30091,25865,30059,11042,195
26,27946,27978,32172,19461,30091,21606,32204,36430,19493,27945,34285,34285,17348,1315
4,13187,8961,25832,32171,30058,23751,21638,27977,19557,27977,32235,17412,25832,13155,
36398,32204,34284,27913,40591,38478,27945,30058,34284,34284,23752,11073,27977,17412,1
9558,21671,21638,13218,25863,34348,36461,32235,30122,21670,30090,21638,17445,21670,32
235,28041,30089,40686,40687,38541,15331,36461,25831,32170,32235,25864,34348,32235,195
25,34283,34316,30122,38542,21670,23815,23815,15363,21670,15331,19589,36493,38573,3857
3,23750,40686,32267,17444,28041,21637,28041,25895,28041,36460,28041,23782,36493,32235
,28041,38574,21702,34348,30154,28009,28041,32235,23783,23751,32235,30090,28009,28009,
32235,34315,19524,32235,27976,21669,34379,28008,23814,25895,36460,17476,34347,32234,2
8008,28040,36493,8992,15362,32234,36492,40687,34347,34315,34380,38574,21605,34283,321
70,34315,21605,44848,25863,23718,30090,38541,28009,28009,30154,34380,36460,30154,4071
9,32202,28008,25896,27976,32203,25864,32203,38575,38574,42768,42768,38574,25896,30122
,13185,15363,

36397,23751,13154,15300,30091,17381,21639,34284,42769,27978,25832,36397,30091,44
881,34284,42769,40656,21607,25832,25833,15268,30091,28011,13187,27978,34350,19526,153
00,21639,42737,25833,38511,36397,23719,25865,44881,25864,13154,25865,11009,8929,17348
,17348,4672,30090,34284,27912,21573,27977,17412,34315,27944,27976,34316,32172,19526,4
6963,36430,30091,23752,17381,34284,19460,30026,23719,25832,15300,30091,25865,23719,23
784,28009,30090,23783,17379,28008,32202,38508,44848,32203,30090,28009,21638,27977,258
64,30122,32267,32235,28009,23750,28009,34316,21638,30090,30090,32170,34283,30058,1318
6,28009,38574,30090,38574,32267,13218,40687,38574,25864,23815,42767,34347,40686,36428
,28041,13218,25928,25928,32235,19557,38574,17411,32235,38606,23782,21669,42767,27976,
19524,27976,19556,36461,17411,30122,25928,28009,30122,34315,28041,23783,17411,23783,2
8009,30122,13218,13250,32202,19556,32234,32234,38605,15330,40654,19556,21702,23815,23
815,21701,23814,25960,32266,28073,36493,34380,23814,21670,32202,36460,32169,30057,342
83,36428,30089,32170,32138,15266,19492,30089,19524,21702,32267,30153,38606,25895,2170
1,28008,25895,19557,25896,19525,28009,30090,30090,34348,42800,36461,28009,25863,23750
,23751,21638,

25832,17412,19493,23752,8961,30091,21639,40624,38543,15268,21606,38510,30058,364
29,42736,44849,34317,25865,19494,25833,23687,40624,32204,13220,25898,23752,32204,2375
2,30091,27945,27913,42704,19493,30026,15300,40656,21638,17412,19493,13155,17413,32139
,23719,13186,27913,32171,21606,32203,21638,21638,17412,32170,34315,30122,40656,36462,
38511,32171,34252,15299,27945,21574,17348,25800,17380,19526,19493,27978,30091,32171,1
7412,30090,42768,32235,30089,32234,32202,28009,42735,23751,38542,30090,36461,25896,23
751,21670,34348,30154,32234,13185,25896,40719,30122,32203,30089,32203,11040,19492,216
38,34348,32203,38542,38542,36461,15331,38542,34316,25864,28041,19557,36461,38573,3646
0,19556,4768,25895,32234,40686,23782,25895,36428,34380,36460,17476,32234,42799,23815,
23782,25928,27976,23815,17476,28009,28009,30122,36461,38606,32235,27977,11105,25896,2
5863,30122,11105,11072,32234,25895,25863,25895,36460,25895,28009,21669,23815,15330,21
734,30153,17475,32266,36460,25960,30154,25895,19588,30153,34347,38541,40654,32202,342
83,40622,36429,25831,30090,15299,17379,32235,21637,21670,40686,30089,32267,21669,1110
4,27976,28008,30089,38606,28009,27977,25864,27977,28009,38574,32202,27976,25896,23751
,34348,21638,

19558,6848,27978,38478,40591,25800,32140,27946,21639,6816,4768,30123,38510,21671,
38542,42768,25897,40623,19493,17413,30059,15236,15268,21608,19527,17349,27978,17413,2
5832,23751,25865,30058,21606,36430,8961,17413,21639,17380,11073,19525,27978,36429,258
64,17380,32171,27977,19525,30090,25831,11041,17379,34284,34284,32170,27977,34284,3217
1,25832,30058,27945,21639,21606,25800,21639,19526,19461,17413,19493,30058,21670,13187
,25864,36397,25864,25896,23751,19557,25896,34348,21638,28009,38574,32236,36462,25864,
25864,28041,30153,21701,13217,36493,40686,28008,30122,30122,17444,23783,17412,25832,3
0091,23719,27977,30091,30091,27945,17380,19525,21670,23783,17444,36428,34347,30153,30
186,17475,32299,34412,40686,32234,17443,36460,30121,30153,21637,34315,34380,23783,301
54,30154,25863,30122,21638,21670,23783,23783,25928,32267,32235,32203,21670,25864,1741
2,27977,19557,19557,32234,28008,28041,21702,34347,34380,25895,25896,25928,23783,34347
,28008,21669,28008,34380,38541,28008,30121,32234,42799,38541,30056,46993,32170,25831,
36429,30090,23751,32203,19525,21638,23719,28009,34348,42767,32235,21637,13218,23783,3
4347,36428,38574,30122,30090,32203,32202,30122,21670,38509,25831,23751,21670,27977,40
655,27977,

30123,4736,13154,40656,42769,30059,25833,25834,13188,6848,27946,15300,36397,3220
3,25896,32203,34349,17413,27978,19526,30026,15268,15269,19495,21608,21575,23752,17380
,17380,27977,44881,27977,25832,25865,19526,13154,38478,13154,21638,38510,38510,30122,
38542,8960,32203,23751,21605,17412,19525,23751,30090,30090,32171,27977,19525,34315,19
525,21638,25864,27945,21607,25865,23720,21607,8929,21574,15300,25832,21606,27977,1315
4,19525,23719,23719,19525,23783,25896,28009,36461,25896,25864,25896,40623,36397,34284
,36429,34347,30153,17508,30121,38606,32266,21702,23750,19589,11137,40655,6880,21639,2
3751,25832,23719,19526,23720,11042,11009,15267,25896,30122,25864,34348,30089,34347,32
267,30186,34411,32266,30186,42799,34347,36460,25927,32234,30121,32234,32267,25896,343
80,28009,30122,25864,21670,21638,23751,17476,34315,32235,34348,32235,38574,30090,2167
0,36429,23783,19525,32267,32234,30154,23782,40686,23815,30154,36493,21702,25895,30121
,28041,25928,25895,42767,28008,25927,34379,42799,44912,27976,38541,40622,21638,25863,
38542,21605,44848,27977,17380,13186,21638,30057,34284,34316,15331,11105,15298,38541,3
2202,28009,28009,21670,23751,25864,28009,19525,23718,23718,23718,32203,34348,34316,34
317,38575,

25832,15299,27945,34284,40655,34317,30027,25801,32172,17413,17413,15300,38542,23
751,6848,32235,34317,19525,21639,21606,36398,19462,17382,17349,19462,21607,15267,2583
2,13187,27978,27945,32171,23720,32204,30026,40623,36364,21638,34316,34284,34284,17412
,23718,21605,30122,34315,23751,28009,27944,23783,23751,32203,34284,17412,19492,19492,
23751,23718,23718,19525,36398,30026,8929,19462,21639,15268,21639,25832,27977,25864,25
864,30090,21638,27945,28010,34316,32235,32203,30122,30122,19525,32203,36396,30058,194
93,30090,25928,25928,17443,34380,32234,23782,21637,23782,32235,28009,32203,23784,1318
7,30058,21606,21607,23687,34285,2592,19526,13187,32203,34348,28009,36461,25896,36493,
34380,40719,30153,25927,36460,30121,40653,34347,38605,25928,30121,30154,34379,28008,3
2234,34315,25896,34316,36461,25864,23783,21637,40687,38541,34380,30090,34316,27977,32

235,25864,21670,30089,40654,23815,32234,23815,32202,25928,36428,38573,21669,21637,300
89,32235,17476,23783,36460,21637,34347,34347,34314,38573,19524,36428,42767,36396,3431
6,30057,25864,23751,19525,11041,15299,19525,25896,25864,21670,30089,36428,23782,38573
,30089,34315,28009,25863,21637,27976,21670,21670,34316,34316,21638,30090,42768,38542,
32171,30123,

21671,23784,30058,25832,32203,38510,42737,25801,13123,19526,17413,15332,36429,34
348,28009,19558,42769,17413,11074,30091,25833,17349,4704,17349,30059,23720,17381,3005
8,25832,30058,36430,32204,23720,25865,32171,19493,32171,34284,42769,6816,23719,17412,
27977,25831,38509,23718,27976,23718,30089,21638,30057,25864,19525,21638,21638,27976,2
8009,15266,19460,25832,32172,34284,25801,25801,17413,34252,38511,34284,34284,46962,32
171,34284,23719,21638,36461,27977,21638,21638,25864,23783,36429,40655,19525,38542,321
71,32203,34348,25895,23815,34380,38573,23815,25895,32202,32235,36461,25897,25864,2797
7,38510,8961,21639,21606,21574,21607,15300,15267,36429,36429,32235,36461,30122,36493,
28008,28073,19621,19588,28040,38605,38573,25895,30153,28008,25927,30121,34347,34347,3
4315,40687,32235,42768,38542,17412,42768,21670,36461,34348,30122,25864,25864,25896,32
203,23783,25863,32235,27976,30121,34347,32234,23815,38606,40719,25895,28008,13218,216
37,32235,13250,23782,27976,27976,34347,27976,32202,21669,34379,38574,40622,38509,3642
9,25831,21638,25864,15299,19492,23783,27977,27944,23751,36461,36493,36461,25896,36461
,34315,32202,36428,28041,19524,30089,25864,32203,38574,38542,19557,32203,42768,40687,
30122,28010,

19526,17412,23719,38510,25799,32204,42769,25833,13123,32204,25897,27945,44914,32
236,15332,42768,34284,30090,32172,27978,32172,13155,19495,27914,21607,17348,17413,258
32,25864,32204,38510,34284,30059,13155,23719,13187,32139,32171,27945,17444,32235,2794
5,28009,25864,36461,23751,25896,6880,32235,19524,23783,30122,34348,25896,30090,30090,
25864,17412,19525,27945,21606,34284,30059,34317,19494,27946,34317,38510,42768,42801,2
3719,44849,32171,27945,28010,36462,19557,19557,30090,19525,23751,23783,25864,34316,36
461,11105,36461,32235,28041,38573,25928,28009,25928,32267,44913,28009,25896,34348,406
56,38543,27913,21607,19494,25865,42704,6848,25865,38542,30090,23751,30122,32235,28009
,25895,23847,25960,32267,30154,40686,23814,15362,28040,25895,19556,40718,32266,25928,
36493,38541,38542,27977,17412,32170,38574,23783,36493,34316,30122,15299,19525,25896,2
5896,40655,23751,30090,32267,30154,27976,44912,15362,34380,32267,28041,19557,23750,30
089,34348,19524,15331,21637,32202,34315,19556,28008,8992,30089,23718,15298,34315,3005
7,25864,25864,40687,17411,34316,21670,36461,28009,30154,32235,30122,32235,30121,34347
,32234,36492,25895,28041,23782,34380,23783,30089,34315,36461,19557,27977,44880,25864,
27977,21670,

23784,11074,19493,44849,19493,38542,40623,8929,25833,15268,27977,30091,25897,216
38,30090,32203,32203,38575,23752,13155,19494,13123,46931,17381,15300,6816,23719,36397
,21606,13186,23752,21606,15300,2592,23720,17381,23720,30058,25865,27977,27945,32203,1
3186,30122,28009,27977,21670,17412,32235,25864,32235,30122,40687,17412,21670,36429,30
122,23718,17380,34284,25832,34285,30059,27914,21640,25801,27946,23719,36462,34316,195
25,23751,27978,32203,25864,25896,21670,25864,36461,34316,15331,21670,30090,23719,2163
8,21606,34348,25928,21670,23750,17444,28009,38574,28041,38606,32203,28010,36462,36430
,38543,36398,8929,11074,21639,30091,27913,27977,21606,25896,17444,19525,38606,17444,1
7444,23815,25960,28041,30121,40718,17475,30120,32233,30153,32234,28073,30154,15330,36
460,44881,30122,21638,25832,32203,36397,36429,34315,30122,34348,21638,21670,27977,237
83,34348,27977,21670,32234,19589,15362,40719,19556,34380,28009,19557,25896,34348,1958
9,25896,15331,34348,27976,23783,28008,27976,25863,19524,21637,19524,21670,36396,28009
,17412,21670,36429,19492,23750,21605,42767,36461,28009,34315,23782,34347,30154,40719,
30121,40686,15298,34315,42800,30121,21637,32235,36461,34348,32203,36396,27976,27977,3
2235,30123,

27977,19557,21606,19557,27944,34316,25865,4704,17413,27945,15267,19526,34316,258
65,38543,34317,25833,23752,36398,15235,15300,13123,27914,34317,21639,19494,25832,3639
7,21606,21606,30091,13187,17413,8929,23752,23719,21606,25864,27945,28009,25897,23751,
15331,21670,23751,21670,21638,32203,23751,42800,30122,36429,40719,25896,25864,30122,3

4316,30058,34284,42736,27946,17348,23752,21607,27979,15300,19526,19493,30123,23784,19
493,19493,19493,25832,32235,27977,19525,27977,32235,42800,15331,21670,13153,21606,322
03,42736,25896,23783,30090,36461,28041,30122,36461,25928,32235,28041,34316,23751,3217
1,25832,25865,15268,4672,21639,36398,32139,21606,19525,30122,21670,19525,34316,25896,
17444,6880,21702,30122,19589,36492,21668,34379,32298,30153,28040,25927,30121,17443,40
719,38574,19525,8992,32235,27977,36397,27977,23783,32268,38607,32235,28009,30090,2797
7,23751,28009,25896,32235,32202,23815,36492,25895,25928,23815,11105,32202,19524,17444
,19557,23783,34316,28009,21670,23750,34315,21669,21637,28041,23783,19557,30122,25896,
17412,17412,25896,23783,25863,15363,42767,30122,34347,32234,17443,32234,34379,34379,3
2266,32202,25927,42799,32234,30121,32234,38606,32234,34315,36461,42767,32234,42767,42
800,27977,

32236,38575,30058,32203,34316,30090,17380,8929,15300,17380,19526,23719,32204,385
42,25865,23752,19494,32204,34317,36398,25833,23752,30027,32139,25833,34317,30058,4059
1,21606,30091,32171,23752,19493,19526,27945,21639,34284,27978,36462,23783,32203,28009
,21638,15299,21638,34348,19557,30090,34316,36461,30122,25896,28009,19589,30122,40655,
42735,38509,19525,34317,19461,17381,13155,23752,27979,15300,17413,23719,15267,25897,3
4284,17413,19493,27977,38575,38542,34316,21638,27977,36461,32203,15331,11073,30057,34
348,34284,36428,34348,38574,34348,28009,25896,28041,21670,25896,30154,28010,21671,321
71,21639,21574,15300,21606,25833,42704,30058,21606,34284,36429,28010,27977,32235,3857
4,19589,4800,21702,23815,23815,28008,34378,28007,40718,38573,30153,23782,23815,21702,
32267,25896,25896,32203,30058,15267,30122,21670,23815,34348,32235,30090,36429,27977,1
7412,25896,21638,23783,34315,28008,27976,23782,23750,21669,19556,15363,30154,21670,19
557,25896,36461,36461,32235,17476,25896,34315,32202,34348,32267,25928,25863,32202,216
70,25864,23750,28009,36461,19557,19589,40654,23783,38541,19556,19524,25895,28008,3226
6,19556,34380,36460,44912,30154,32202,34315,34380,32202,30154,36428,42832,34348,40654
,42800,34349,

23751,36462,23783,27977,34348,25864,23751,23719,23752,23752,30091,30058,23752,19
526,17413,25865,32172,36398,34252,36398,27946,23720,17413,34285,30026,38510,23687,300
90,27977,23687,38510,27946,19494,36398,27945,30091,42801,15299,34316,34284,30090,3009
0,8960,8960,23751,34348,17412,32235,25928,40655,42832,27977,19589,38542,17444,34349,3
6429,32203,13154,27978,23752,27946,17413,21575,30092,30027,21671,23784,25832,34349,27
978,17380,17412,30058,32203,32203,36429,21670,30122,36461,42767,4768,38542,27977,3005
7,27944,25896,23783,34348,23751,28009,25896,30090,25928,25864,30122,30123,30058,36430
,27978,21574,30026,36365,27913,36430,25800,27945,42736,42800,23751,28009,32235,28041,
15363,15331,8960,23783,21702,28008,25959,25894,25927,32266,13281,15395,23814,34348,32
234,30089,30089,34316,23751,27977,40622,30089,36461,38574,27977,21638,34348,17444,195
25,30122,23783,27977,23750,21669,27976,23750,21701,21702,8992,28009,28009,32235,23783
,36461,34316,36461,21670,15299,25896,25896,30154,36461,42800,36428,30122,40654,27976,
28041,32202,28009,40654,15363,30154,38541,30089,38541,17443,28008,32202,23815,19589,3
4347,36492,36492,42799,28008,36428,34347,40686,32202,19524,42767,42799,32234,34348,38
574,40655,

13187,25897,21638,23751,25864,21670,23719,21606,34317,30058,30058,27978,25865,23
719,23720,30059,44849,27978,25865,36398,23752,27946,32172,32171,30059,19526,32139,258
64,25832,27945,30091,21639,27978,30091,25865,32171,38510,17445,32171,34317,21638,3639
7,19526,27977,23783,42768,25897,28009,32268,32235,30122,23783,30090,34348,28009,27977
,30090,34316,21638,23751,38510,25832,25833,8962,30059,32204,23720,19558,30091,19525,2
3752,15267,23751,30058,25864,25864,23783,17444,27977,30122,38541,27977,44848,19525,30
090,19492,38509,23718,27977,42768,23783,34348,30122,21638,34348,36461,30122,32203,427
36,23719,21639,19461,38510,40656,40623,23687,38510,36429,27977,17412,21670,38574,2797
7,19557,25896,19525,30122,30122,36460,32233,28072,23781,28073,11169,25895,32267,32234
,30154,36461,23750,17412,34316,27976,36429,25896,38574,32268,23783,15331,25896,17412,
21638,23783,23751,25864,23783,17444,23783,25863,32202,30089,17476,23815,38541,38541,4
0654,40655,23783,25864,13218,23783,15331,32202,25928,28041,38541,38606,23783,42767,30
122,38574,32235,32202,36428,19557,32235,32234,19524,28041,21669,27976,28008,28008,301
21,30121,38573,38541,38606,21669,38573,23782,34315,34315,30089,30089,34315,32234,3220

2,32235,34316,

8961,36397,19558,15331,8992,19557,15299,19525,30123,19493,38510,32171,23752,2163
9,13155,30091,27945,32172,32204,32236,23752,32171,23752,25833,15268,19461,32203,27945
,23751,25864,38478,36365,21639,40623,15299,25865,32203,27977,19526,36397,21606,36430,
25865,42769,21638,42768,28010,15332,28010,34316,36461,23815,23751,27977,34316,27977,3
4316,27977,30057,23751,34316,25832,25833,19494,30091,23720,19526,19526,11106,21639,21
639,17413,32204,30058,30090,28009,32203,30090,36429,15331,23783,30089,38541,27977,385
42,44849,15299,17379,28009,38574,28009,25896,32268,23816,19622,30155,34348,34348,3217
1,19525,25832,17380,36397,44882,30058,34284,40623,25864,38542,21606,21638,28009,17412
,21670,23783,34348,32267,25895,38573,32233,23814,36459,30153,25959,30153,25927,25927,
28073,23815,21702,30089,30122,32202,23750,30154,40719,34316,28041,23751,19557,21671,2
3751,34348,23751,30122,27976,23783,19556,28041,36460,32202,25863,25895,40687,36461,40
654,28009,21638,23783,13185,15331,23751,30122,28009,32235,28041,34347,13250,34380,364
93,23750,23782,30089,27976,21637,34380,34348,21637,28008,28008,28008,28041,25895,3015
3,28008,32234,38606,32267,25928,28008,21702,36460,30089,38573,34315,30121,28008,32202
,30090,25864,

32203,27977,13219,17412,21670,19589,32236,25897,34317,25897,25865,32204,30091,17
413,11074,32204,30091,32172,36430,38543,30091,25897,25800,27946,15267,23720,27913,300
58,23719,19493,30058,44882,13187,32204,25832,25897,30090,38574,21606,40624,27945,1315
4,19526,32204,34284,34316,34316,17413,38542,34349,30090,34348,28009,32235,30090,17444
,36461,25896,25831,30057,32203,30058,23752,27978,27913,21607,32204,30091,19525,25865,
25832,34317,32204,30058,44849,36429,32203,34348,25896,19525,21670,38542,23751,25831,3
8574,36429,8928,23719,32203,21670,30090,23751,36461,21735,19557,28009,38542,30123,258
65,27945,27945,21574,30026,38510,27977,30058,30058,32203,38510,32170,44881,19557,2378
3,25928,17476,21637,30089,36493,34379,36492,36524,34346,34411,21734,36492,32299,32267
,23782,34380,30089,34315,30089,32202,32235,32202,40654,30122,40655,32203,23783,28010,
28010,23783,17412,21670,23783,27977,23783,28009,38573,32202,34347,30089,25896,28009,3
2202,21670,28009,25863,19557,25896,25896,32235,32235,30089,25863,23750,23782,23815,38
573,23815,17443,15363,36461,25896,30089,34315,30089,30089,28008,23750,36428,25895,300
89,25895,25928,40719,19588,30121,34347,34379,34315,30154,36428,38573,30089,28008,3642
8,36396,23719,

17413,15332,21638,30123,17476,25896,38542,25832,21639,23784,15268,38510,25800,19
494,25833,21607,25865,30091,32204,25865,19526,25897,23687,44882,13155,21574,23719,363
97,38510,42736,40591,27913,19493,36430,34349,25897,30155,40655,11074,30091,13155,4704
,34285,23687,27978,30091,32204,19526,30123,30123,28009,21638,17444,23751,30122,17476,
32202,32203,19525,27977,30122,30058,25832,27945,36398,25865,30091,23784,30123,32203,2
7945,36397,23719,30058,34316,32236,25864,36461,19524,25896,30090,19557,30090,19525,25
864,40687,19525,8960,25896,25896,19589,36494,21703,28041,28074,30122,30154,28009,3009
0,32235,32171,27945,25800,27978,17413,23719,27977,25832,27977,34316,38574,30122,34316
,30154,25896,30154,30154,34379,30121,32266,40718,32234,32266,19588,40751,34379,34347,
30154,19556,32267,36460,21669,23815,32234,30122,36460,28009,36429,38542,25896,32236,3
2203,19558,11106,30090,32203,25896,30089,32267,28008,25863,32266,30121,17411,32202,36
428,19557,11105,30122,25928,32203,28009,30122,23783,25896,21637,17443,30153,38573,322
34,32234,6912,23782,30154,25896,28008,32202,28008,17411,28009,21637,36460,34347,30121
,19589,38573,30121,32234,34347,38541,32234,25895,38573,30154,42767,28008,21702,36428,
30057,23719,

36398,21639,21671,23751,32235,32235,34349,25832,23784,13219,38543,25865,32204,25
833,42737,21574,11074,23752,36430,25865,13187,15300,32172,21607,21639,27946,34284,342
52,34251,23686,27945,15267,21606,36397,28010,34348,30154,42800,17348,23687,21574,2368
7,34317,32172,25833,23719,32204,27945,28010,27978,36429,25896,25896,30090,25896,19589
,40655,30122,32235,23751,32203,27945,19526,30058,32204,30091,27977,30091,23751,32236,
30091,25833,11041,21606,25865,32203,19557,38542,11105,25864,25896,21638,27976,23751,2
5864,21670,19493,13186,34283,38542,17444,30155,34348,21702,23815,25928,19590,36429,25
896,32203,40623,36397,30026,23720,23719,17412,30058,30058,32170,36396,23783,32202,322

35,30089,34315,21637,28008,32234,28041,28040,21701,19620,28008,19621,32299,34412,3437
9,34379,32234,32267,23782,32234,25896,34347,30154,40686,21670,27976,34316,34316,32203
,32203,21670,13186,30090,30122,27977,23815,27977,15363,30089,23814,34380,17476,27976,
34347,21670,23750,32203,32235,32235,28009,30122,28009,25863,21669,15363,34347,30186,3
4347,28041,15363,21702,25895,32267,34315,15330,34348,23750,25928,27976,34380,32267,25
960,30121,40719,15330,36492,36460,32234,34380,30154,30154,30154,38541,30089,19556,385
73,17412,32203,

19526,21639,32203,19558,28009,27977,21671,8993,42769,34349,38511,27978,27978,237
52,27946,13187,15267,27946,38543,30091,23751,23784,40656,32172,44817,17381,30058,2157
4,38510,19493,27945,25832,27978,36397,27977,28009,21638,30122,21639,15268,36365,32139
,32172,32204,21607,21639,32204,17380,27977,34284,25864,25896,19557,21670,32235,28041,
42767,32202,17411,21638,30090,23751,30058,30058,23752,36430,13186,30090,28009,21671,3
8510,15300,8961,21639,27945,36429,21638,25864,21670,19525,23783,30122,23718,23751,322
03,21605,15267,34284,30090,40687,28009,38574,30154,36526,17476,19622,19557,36494,2163
8,25896,34316,30058,30058,25800,21606,15267,30090,30090,25831,34283,30122,34348,32235
,36461,30089,23782,25895,34379,23750,21701,25895,21669,25927,17475,32299,32266,23847,
25928,36460,34347,28073,30153,19589,28008,28041,36461,21670,21670,30122,34316,36429,3
2203,25896,17412,21638,32235,21703,19525,17412,8960,34315,15297,27976,19556,23815,322
02,30122,36428,23783,34348,36461,36461,34315,28009,42767,23782,19621,32266,25895,3012
1,25928,36525,32234,28009,34380,34348,30121,36493,25895,28008,23815,42799,38606,32234
,32267,30121,32266,34379,36460,38573,38573,32234,23815,36493,34315,34380,21637,32234,
25864,27977,

23752,32204,30091,34316,25897,36494,30123,23784,38543,38543,34284,34284,25833,23
720,13122,13122,15300,40624,25865,34317,40623,30091,25832,34317,21606,17380,15235,342
52,25800,32204,23752,25833,25833,25865,21638,28009,27977,23751,27977,23687,40624,2794
6,19462,27913,21607,19494,30091,15300,11041,38543,21671,25864,19525,19525,28009,25896
,36461,28009,13185,15331,32170,17379,19525,21638,19525,19493,15332,32235,19557,17412,
27978,15235,11042,23752,23784,36462,23751,27977,21638,21670,27977,36428,30057,36461,2
5896,23719,15234,23751,27977,40687,25896,40655,23783,34380,25928,28041,36461,36461,32
203,25832,25864,32171,23720,30058,19493,19493,25832,32203,25864,25896,28009,27976,322
35,30121,19556,34380,32234,40654,13250,30089,27976,28008,21701,32266,32266,23847,2592
7,19621,36492,34412,32234,28008,21669,25895,28008,36461,25928,30122,27976,32203,36461
,30122,19558,23751,19525,28009,30090,17444,13218,11105,25895,23750,30121,30089,27976,
25863,38574,34315,25864,30090,30090,32267,34348,27976,36428,36493,28008,28040,32267,3
0121,28008,36492,34380,34380,38606,36428,28041,30154,30089,23750,23783,38574,32234,34
380,21702,32267,36492,32267,25928,40718,34315,32234,28041,28008,32234,38573,13250,301
22,40654,17444,

15333,21639,36462,34316,25864,27945,21639,17413,36397,30091,21638,38575,27945,25
833,13154,27913,34285,19461,38446,32171,36430,30058,36397,21606,21606,36397,27945,279
45,38511,36398,30059,30059,23720,8961,19526,13187,34316,23752,30026,30026,40591,21574
,21607,25833,27914,17348,30059,21639,17380,42736,32236,25896,36461,6880,23751,32203,3
8574,23783,19589,23815,36428,21637,27944,40654,25864,19492,23719,38542,21638,8896,131
22,30026,25832,32204,23752,25833,23751,25864,21702,30122,34315,23783,30089,32203,2160
6,38543,11041,19493,25896,36429,27977,23750,32267,36461,34348,36461,30122,32203,34316
,19525,19525,27978,19494,21574,23752,15300,23751,32235,23783,15298,19556,21637,17443,
27976,27943,30089,34315,30089,19524,28009,25831,32202,17443,34347,23814,30186,28072,2
1733,32266,34379,34379,32234,34347,25928,32202,30122,25863,27976,38574,27977,38542,28
009,21670,21638,19558,25897,34316,11105,19558,30123,30122,40655,27977,32235,21702,195
25,30122,28009,28009,23751,23815,19557,30154,25896,30122,23783,32267,25928,34347,3226
6,30121,40686,38606,38606,28008,30121,23815,23750,32267,34315,21638,23751,30122,30154
,28041,32267,34380,32299,25927,28040,30121,32234,28040,30186,30153,30154,17411,30089,
36493,21702,

27978,38575,38543,32171,27945,19525,15235,21606,27978,27945,25832,32236,34284,17
380,21574,25833,21575,19461,44817,23719,42736,34316,36461,21638,40623,44849,25865,173

80,23720,34285,21607,27978,27946,11074,19526,25865,21606,34317,32171,30026,25832,2583
3,34252,25865,27978,21607,36430,21606,11074,27945,36429,27977,42768,6848,13218,25896,
32235,30090,30154,34315,38574,36428,32170,34348,34315,13153,34316,40622,38509,19525,6
784,25832,32172,32204,27978,19526,21639,19525,23783,32202,44913,30089,28009,30058,363
65,32171,19526,27977,30090,25864,23783,32203,36428,40687,34348,36461,27977,38541,2797
7,19525,6880,38542,27946,15300,13154,19493,30090,25896,32203,11072,15298,25863,15363,
30121,30121,27943,32202,34315,25864,32202,15298,25863,23750,23782,25927,36492,25927,1
9620,34379,25960,28073,34347,28008,30122,23783,30154,30122,28009,34315,27976,36429,23
783,30123,25896,17412,32204,19493,23719,36430,32171,32235,19525,23751,28009,21638,111
38,15364,19557,30090,25896,30154,32235,21670,25928,30154,13250,38541,32267,32234,3434
8,34380,42832,25960,32299,28041,19556,19557,23783,32203,42767,13154,25864,25864,28041
,36461,36493,38606,32299,25927,32266,32298,28073,36460,30154,28041,23782,30154,28008,
23750,36460,

38543,42737,40623,30091,32171,17348,15235,25800,32204,25832,28010,32236,27945,44
849,30026,25833,10977,21607,30026,27913,36397,36397,34284,25865,42768,40656,38543,131
87,21639,25865,19494,40624,25865,25865,21639,34317,25832,34284,27946,23719,21574,3213
9,23752,32139,21607,32139,30091,17380,13187,19493,25896,36461,28009,15299,19557,23783
,27977,34348,21637,28009,32235,32202,32170,23718,30057,25864,38542,42800,42768,27977,
13154,42736,44849,32139,23752,21639,11074,32203,42768,40687,32202,36461,32170,34316,2
1606,21639,38542,25864,30122,32203,27977,28009,32235,36428,32235,40719,27976,19525,17
444,30090,27977,34317,19461,23719,13187,15299,19558,25864,30089,21637,28008,25896,280
09,30089,32202,30089,25863,30057,25863,34283,17379,21669,32234,30121,30153,42831,1747
5,32266,19588,23814,28008,25895,23815,28041,23783,23783,32202,19557,30122,27977,34348
,28009,42768,40655,21671,38574,17380,25832,34316,30090,30090,15299,25897,28009,28009,
15331,15364,25896,32267,28009,25896,38541,25896,32202,30154,17476,36461,38574,21669,2
7976,34347,38606,21637,25895,32267,25895,21670,25896,30090,27944,17412,34348,23751,32
202,30089,28008,36460,28040,25927,28073,32299,25960,34379,28041,25928,27976,19557,301
54,34347,34379,

30124,42737,34252,32204,36397,19493,27881,23655,30091,30091,25864,19558,36430,38
543,38478,23720,8896,11042,21574,32171,27977,11073,40623,15267,23752,40656,21639,1738
0,30058,27978,25832,32171,34317,21607,13187,44850,36397,30091,17348,21574,17381,15235
,15300,44817,44850,30026,15300,13154,25865,19525,15331,34348,30122,15299,23783,25864,
25864,32235,25896,27976,27977,25896,25864,17411,15331,25896,28009,25831,46961,27944,2
1605,38542,38510,30091,32204,15268,17413,36397,40623,40655,42800,42768,28009,21605,15
299,25832,25832,25864,19558,38542,21637,38542,38574,25928,30122,40687,23783,15331,195
25,38575,34349,28010,27978,32139,21639,17412,15299,23783,30090,30089,30122,32235,2804
1,28041,23782,32235,21605,19492,27944,27976,19524,21702,25895,34347,32266,32266,23814
,23846,17475,25927,25927,30121,21669,32202,28009,25896,21670,23815,25896,23783,32203,
32203,40655,34348,19525,38542,27945,21606,30090,27978,30090,21670,28010,25896,30090,3
0090,4768,25864,28009,19525,32235,32235,30122,30122,36493,28008,30122,34315,11104,280
09,30121,38541,30121,25928,38606,28041,23750,19557,30090,19525,23751,36429,25864,3649
3,30122,25896,32234,25927,30121,23847,28040,28073,30186,28040,30154,23782,13218,25895
,32202,30089,

21640,27946,34285,40656,30059,21607,19494,15203,44817,38510,8928,19525,40656,321
71,42704,15268,21542,17348,34252,30091,21606,19493,19525,17380,21606,19526,32204,2163
9,25865,25832,25832,27945,30058,19493,21639,32204,21639,32204,11042,17348,21574,21607
,21574,36398,36430,23751,23719,27945,34316,13186,28009,32235,21670,25896,28009,25864,
15331,38574,30122,36429,30090,28009,21605,30122,19492,23750,15299,25896,27977,11105,2
1637,34316,36429,27945,36398,13187,27946,23752,27913,34284,36429,36396,21638,21638,19
557,27977,32203,40655,11073,36429,21670,36461,36461,23750,23783,38574,38574,19557,406
55,40687,25897,23751,25832,30059,11041,15300,17412,25864,30122,32267,23783,25863,2586
3,27976,25864,30057,15331,32170,30057,30089,27976,23782,36460,36492,30185,23814,40718
,36524,25959,21701,23814,32266,23815,30121,23815,21702,21670,28009,21637,21670,23783,
30122,36461,32235,17380,40623,17380,17412,23719,30058,27945,25864,25832,25864,28009,2
8009,13218,32203,27977,23750,28009,32202,30154,30122,44912,36493,25928,25895,15363,34

347,25928,38573,30121,27976,30154,30154,30089,27977,32203,19557,27978,34284,28010,322
35,30090,23815,25928,28041,32267,28008,32234,25960,23814,28040,28073,13217,4800,28008
,28040,38605,

17381,32205,17349,32172,23720,11042,21574,19461,30091,17413,4736,32203,30091,322
04,30026,42769,8896,32171,25832,21638,23751,21670,25865,27945,17380,8961,27977,21606,
27945,32171,30058,19493,23719,34317,21607,38543,30058,25832,6784,8896,25833,40592,236
87,27946,13154,34316,30122,23783,27977,19557,17411,15299,23783,30122,34315,38541,2375
1,30122,25864,30122,27977,17444,15267,40655,23751,8960,15331,27976,30122,23783,30154,
36429,32236,27977,27945,11042,21607,15300,19493,23719,25832,32170,23783,36396,38574,2
8009,27977,32235,17412,34348,17444,23783,38541,15363,28009,38541,32235,30089,34316,34
316,13218,6848,21639,17413,13187,17380,28009,21638,34348,32203,19524,27976,34315,1952
5,21605,25896,27944,34316,27944,27944,30121,19556,23814,34379,28040,25895,30153,32298
,28072,28072,34347,36492,19556,25928,28008,30154,23815,28041,23750,21670,32203,30090,
23751,30090,42735,38510,36397,27977,19493,27945,38510,27945,21670,23719,38542,23751,3
2235,21637,40687,21670,32202,28009,25928,36460,25895,28008,25895,27976,32234,25895,28
008,36460,21702,30153,28008,23815,30122,25864,32203,25864,23752,19526,32204,42768,195
57,32203,23751,36461,30154,36492,23814,19588,19621,32266,13282,23815,28041,25928,3012
1,32298,

17382,30060,11042,30059,19494,21574,11010,19494,30059,13187,23752,34316,30090,34
317,17413,32172,19461,23720,19493,17380,19525,30122,32171,21607,19493,6848,4704,13154
,32171,30058,25864,21606,27945,25832,27946,21607,25865,23752,17381,15300,21607,27978,
23720,32171,27977,36364,36397,25864,27977,25864,21605,23751,32170,34348,32235,30122,2
1670,34348,25864,23751,32203,19558,23784,25897,19558,4736,25863,32235,32202,34348,386
06,40719,30122,32203,27946,17413,30091,21574,13155,32139,30090,27945,27977,40655,3223
5,17412,34348,36429,23783,28009,30154,34380,34348,27976,34315,32235,15363,34316,38542
,25896,11138,23783,21639,15332,17413,23751,36429,28010,27977,30090,23783,25864,36461,
19525,13186,28009,21670,32203,23751,34315,36428,25863,21701,40686,25895,30121,40718,2
5927,28072,36460,32234,25895,25928,40718,25896,30154,30154,34380,28009,30121,25896,17
444,17412,27977,38541,25864,44881,34283,19493,21638,34284,30058,21638,32171,23783,300
57,25896,23751,36428,17444,42799,30122,32234,34348,32234,23750,19556,23782,34347,1955
6,25928,28008,21701,34347,30089,27976,30090,25896,32236,23751,27978,13154,27978,40655
,23751,32203,30090,30089,30121,34347,25895,23782,34379,34412,21734,36493,28008,23814,
38637,25927,

11010,11010,19494,19526,25832,40624,19494,34252,25833,11041,27978,32203,38510,25
864,21607,30059,25865,32171,34284,30122,30090,32203,17413,19494,17381,19493,27977,174
12,38509,32203,27977,21605,21638,30090,17413,30091,32172,21607,34253,15300,11009,2163
9,30059,27946,44817,40623,34284,30090,23783,44881,23750,15266,27977,30089,32267,36460
,28009,25863,30090,19558,32236,17412,25897,15299,21638,21638,28009,25864,34315,40687,
36461,40687,25896,30090,32171,25832,25800,44850,13122,30026,36429,25864,25864,36429,3
2202,19557,30122,30090,38606,25928,38542,32267,23783,21702,38574,21670,32267,40687,36
461,19525,30090,34348,28010,25897,32204,32203,28009,32235,27977,11073,23751,25896,322
35,23751,21670,23783,19557,32203,23751,27944,38541,19556,32202,30153,36492,25895,3012
1,17443,38605,32266,34347,19556,30121,34380,36461,32267,30186,25960,28041,25928,32267
,23815,19557,27977,17444,25863,40622,27944,38542,27977,38542,34317,30058,25864,23718,
21638,30057,42800,34315,23783,40654,34315,36461,28041,30122,32235,21702,23783,32235,3
8606,30154,25895,32234,36492,28009,23783,32235,28009,30122,25864,25832,15267,23752,27
945,21638,34284,25864,36429,34315,32234,25928,34412,36493,32299,30121,34347,23847,259
60,32266,30185,

27914,13188,13187,34252,40624,30059,38446,40624,19494,13154,36397,25897,32203,19
526,30058,38510,27913,23687,23719,38542,38574,34316,13187,13187,25865,38510,19525,322
03,36396,21605,19492,19525,21638,36397,23719,21639,21607,32172,36398,19462,13155,3217
2,25833,23752,27945,32171,34348,40687,23751,44881,11105,13153,23783,40654,30089,36460
,30122,21638,23751,23783,32236,21638,28010,11073,19557,25896,21638,25864,34315,38606,
28009,21702,25896,25864,38510,30091,15235,44850,13122,32107,32139,27977,27945,23751,2

3750,34315,30090,32203,30090,23815,36461,21670,23815,28009,38574,21670,38541,38574,38542,23751,28041,25928,27977,38543,32203,30090,19558,28009,32203,6848,23783,23751,30090,32203,13186,30122,27977,30122,30122,23815,34348,23750,38541,21702,28040,23782,15363,32202,36460,21637,28041,19588,32202,38605,32235,21734,36493,28041,30121,21702,40686,28008,27976,32235,17411,17411,34316,27977,34316,27945,42736,32203,32203,27977,19525,30122,38541,44880,40654,28009,25895,32235,36460,30154,23783,32234,23782,17443,30089,40718,28041,19556,32234,36460,32202,25896,30122,28009,27977,23751,30123,13154,17413,21639,30058,30058,27977,36428,42767,23815,28008,32234,32234,30154,34379,28040,23847,25927,30153,32298,

30059,15268,17348,40623,32172,21639,34252,25768,25865,23752,34284,21639,23719,27977,11074,32139,23687,11009,46962,30123,34316,32203,30059,8896,13187,34284,27978,34317,30090,34316,11041,19525,36397,34284,42736,6816,25832,32204,38511,34253,25833,27913,27978,27946,8896,25800,34316,40655,38542,36429,21638,17444,42735,23783,27976,23750,38541,38541,15331,34316,32203,30123,13186,13154,13219,23783,32235,23783,23783,28009,30122,30154,30122,25896,34284,25864,17413,19493,17348,38510,25865,34284,27944,23751,21670,30089,32235,25864,32235,38606,21637,19589,40719,25896,32235,32267,32235,30154,30122,25864,34316,13218,21671,32236,21638,25865,17412,23783,25864,23783,36429,25864,32235,28009,21638,30122,27977,42768,38574,23750,19589,15331,30121,30121,30121,30153,36428,40718,25927,23814,25928,23782,21669,36493,30121,28041,30186,19589,32234,19589,32267,40686,32267,30089,23783,27977,32170,34315,30090,23718,25864,30058,30090,21605,32170,34348,38509,38574,34347,40654,27976,32234,34347,32235,28009,40719,28008,23782,21669,36428,28008,25895,23750,34347,30089,40719,28009,28009,21638,25865,32236,8960,19525,21638,25864,27977,32203,23751,36428,36428,34315,34315,32267,32266,28073,17508,21734,34379,30153,36492,

38479,4672,34317,36397,42768,38542,34284,21574,15300,23752,36462,25864,21639,25864,17380,13154,11074,23752,30091,30090,23751,28010,27945,32204,21607,36398,38543,21606,38510,32203,27945,32171,36429,23751,32204,8961,27945,30091,27978,27946,30059,19461,27946,30026,448,19461,27978,36397,32171,34316,25863,11105,36428,25864,30122,23750,30154,32235,23751,36461,32235,11105,17445,17444,27977,27977,40655,13218,17412,25864,34380,36461,25896,28009,32203,32235,30090,11009,25800,27913,34317,32204,27977,30057,13218,25928,32235,23815,32202,36428,23815,15363,36428,25928,30154,30122,32235,27976,30122,23750,38574,13218,21638,23783,15299,23783,21638,17445,23783,34316,40687,32203,32235,25864,19557,28009,25928,38574,34315,28041,15331,17476,34348,32267,34347,30121,25928,25895,32234,28041,32234,28009,19589,25895,30154,25928,23847,19589,36492,25895,21701,34380,23782,34347,34315,25896,36461,27977,34283,30057,19525,23719,27977,32170,32203,36428,28009,28009,34380,28009,36428,32235,34380,17444,25896,36461,25928,28041,30121,27976,28008,28040,38573,21669,38541,42832,8960,25896,25896,21671,27977,19525,21638,23751,21638,27977,36396,23750,28009,25895,38573,36428,30153,25928,25927,32266,21766,28072,30185,30120,

23720,25833,32171,25832,38542,34283,40591,6848,480,19525,34348,34349,28010,19526,17380,30058,40591,17413,17381,36397,27978,21638,23752,21639,17413,34284,32171,21639,25865,32171,44849,19493,21639,6816,17413,21639,32204,34317,36398,21607,34317,8929,27946,19494,19461,13155,32204,30091,21639,42768,17412,13186,30057,17411,21670,17443,32235,27976,30089,42768,30090,11105,19557,40655,17444,25929,34348,13250,32203,30155,40655,30122,30090,36396,38574,42768,42768,34316,32203,23752,34284,21606,25864,38509,17444,28009,38541,23783,25896,38606,9024,23783,32267,32235,21670,30122,34380,25896,32235,23815,27976,19589,25896,19525,25864,38542,21671,19525,21670,36429,25896,23751,32203,23750,28009,25896,30154,19525,27976,36428,34315,30089,38541,30122,36461,25895,27976,28008,40687,42832,28008,36461,34347,21669,25895,28041,30154,28041,25960,25928,23814,15395,32234,28008,23815,30121,36461,27977,38542,30090,23719,23718,23718,40654,11072,40686,25928,32234,34315,25928,36460,36461,25896,6880,17412,28009,28008,34380,38606,23782,32266,34347,25927,23782,36428,38573,15298,25863,36461,21670,32236,32235,23718,27977,23783,28009,38509,17411,19557,25895,38573,25928,36492,25927,25927,30186,32298,30153,32234,34379,

13187,32171,30058,36429,32267,27976,27945,13186,15300,25896,36462,34381,32236,21671,32171,17413,19526,6816,30059,38543,34349,32171,25865,23719,13187,15300,19493,25833,32172,13155,21639,19493,17380,32171,19526,21606,32203,34284,25832,13154,25865,15268,17381,27946,13155,4704,34285,30091,23720,38510,23751,19525,27944,23718,21670,25863,23750,25896,15298,32235,30089,17412,23783,36461,19589,30090,30122,30122,38542,38542,30058,21638,19525,30090,46993,38509,42768,34348,40656,19526,25865,23784,25832,40623,32235,34348,38541,38541,13218,32235,30089,23750,30154,21637,21670,32235,30089,30089,25896,27976,28008,36461,28041,32203,25897,27977,23783,23784,32235,36461,27977,27976,23783,21670,25896,27976,30154,15331,30122,36428,38542,34348,17412,34316,23751,23750,28009,30089,42800,32235,28009,28009,30089,30122,34380,30122,36460,28041,15362,30121,19588,6912,30121,30089,30153,30154,36428,25928,21670,28009,30090,30122,32235,27977,11072,34347,32267,23815,21702,32234,32235,25928,36461,15299,25896,25896,19589,28008,23782,34380,36492,28040,32234,32202,25862,25895,23782,34348,36429,32235,36494,36429,23783,36461,32203,21638,38542,8960,13218,28008,32267,25928,44912,21701,28040,36492,28073,34379,36460,34379,

11041,25865,36429,27976,19524,25896,32170,25864,25864,19557,25929,25896,38542,36462,13187,21574,21607,34317,36365,44849,32171,30123,34284,36430,23720,19494,34285,27946,34317,17348,17381,30058,21639,21606,17380,21638,30091,21606,36397,27945,19493,11041,34285,38544,4672,11010,38512,44818,21639,30058,36429,27944,36428,17411,34283,27976,23750,30122,21637,30089,23750,21637,36493,34348,32202,36493,21703,30122,25896,32171,19494,42736,19493,15299,40622,34315,36428,40687,36429,38542,21639,30090,23751,21606,21638,32170,25863,36493,15331,21702,32202,25896,34315,27976,17476,34315,34348,27976,23783,32235,42767,25896,25896,32235,34381,30123,32203,23784,36461,32235,28009,28009,25864,25863,30089,36461,30154,23782,36461,28009,36428,25864,23783,34316,19524,23750,38542,32235,28009,30089,28041,21637,23750,36461,30089,23815,30121,28041,21669,32266,23814,28073,34379,36460,32267,34347,30089,32235,19557,28009,32235,30122,27977,15363,17444,30089,42831,32234,19556,28008,28008,28009,38574,30122,32268,28009,25895,25895,25895,40686,42831,30121,30089,32266,27976,30121,28008,34348,32267,38574,28009,30122,34348,42800,34283,23718,38542,17411,23750,30057,36428,38541,32234,32234,32298,28040,32298,30185,36460,34379,

4704,30122,40655,30121,13250,32234,30090,27945,28010,23751,36429,38542,30155,38542,23719,17381,34252,34252,30059,46962,30091,32203,34284,27978,30091,25865,44850,40656,21640,38511,19494,34317,21606,6816,19493,27977,34284,21638,40655,27945,17413,21606,36398,25801,25833,17349,32140,17414,27914,40623,25832,27945,30089,23751,27976,21637,25896,32235,34347,25863,25895,38573,28009,32235,30089,19557,23816,28009,23816,30090,23719,42704,21606,17412,27944,44880,34348,36428,38607,34316,25865,19525,32171,25832,23751,23751,23750,36428,25896,23782,25928,30122,40687,19557,17411,23815,30122,17476,30154,32235,40686,30122,38574,15364,32236,34348,23751,23783,25896,25896,32203,30090,28009,30122,28009,38574,28041,28008,32203,21670,25864,25864,30122,25864,19525,25864,25896,23783,21670,44881,30122,21637,36461,36461,23783,28009,19589,30121,32266,30186,34347,34347,32234,28040,32234,36460,36460,21702,21670,21702,30122,30090,25896,21670,34315,28041,42799,34347,28040,25928,34380,30154,32267,25928,28009,34348,32267,28008,32267,40719,40686,34379,25830,36427,36460,21637,34315,42767,32235,25896,36429,38574,30122,34380,23783,30089,34283,17443,32202,32234,30121,42767,38541,32234,34347,23814,34379,32266,38541,40686,

21639,36429,36429,28009,17443,27976,27977,36429,32236,23784,15364,34316,40655,32171,27978,27913,25865,30091,36365,23719,30091,19526,27978,40656,21606,23752,36431,21639,27979,25833,27946,25833,8961,6848,25832,32171,21606,27977,40655,21606,19526,27978,32204,32172,40592,17382,19462,23721,32140,42736,21606,27977,21638,38541,19524,28009,30122,34347,36428,30121,32234,30154,21669,36428,28008,30089,30154,28009,21671,23751,23687,40591,8928,13186,19525,32235,30154,36461,32235,28010,23751,38542,25832,23719,15234,25896,30090,36428,28009,21670,28009,25896,30122,23783,30122,25928,34348,28009,30154,36493,28008,30121,32235,28042,32235,21702,25896,25864,32203,34316,34348,19557,32203,23783,32235,28009,36428,32267,25928,17412,23783,34348,19557,11105,21670,32235,28009,21638,17444,27976,30122,38574,34315,30154,25928,25895,25928,32266,32266,25895,28040,36

492,36492,32234,30121,32267,32235,19557,23783,25896,30090,40719,30154,28041,36460,343
47,34379,34347,28041,34379,38606,25896,40687,27977,25864,34348,28009,28009,34347,3220
2,27976,28008,30121,30121,30089,30121,36493,38573,25896,21637,32267,36428,19556,34348
,28009,34380,36428,19524,36428,25896,25896,25895,38573,40719,38573,32298,30121,25927,
28008,38573,

15267,19526,25864,23783,25896,30089,38510,25865,27978,21639,13155,23751,23784,38
543,25865,36397,38511,4736,32172,11075,34285,30091,32172,40656,8928,34317,17381,15268
,32172,30091,27946,30091,21606,23719,19493,36430,23784,23784,34284,25832,27978,30059,
23688,25866,42705,19494,19494,30092,21574,32204,17380,13186,32235,40654,27976,25831,2
8041,30154,34347,34347,32266,23782,28008,28008,36493,30089,23783,21638,27977,23719,27
978,32139,13187,34284,25896,36429,23783,28009,32235,27977,38542,25864,19526,19526,216
39,25800,36429,32235,25864,17412,25832,21670,28009,17444,28008,21669,32235,38573,3854
1,21701,28008,23782,32267,34347,42800,17444,17444,21670,30122,34316,30122,21670,25864
,38510,27977,21638,15331,19493,23719,25897,30122,30122,27977,17444,30090,28009,17444,
17444,8992,23751,40687,36461,25863,19557,19557,25863,25896,38541,34379,23781,13249,32
266,23782,21669,19556,34380,25863,23750,36461,27977,23783,36461,32202,30089,38573,322
34,32202,28008,23814,40654,36461,21637,42767,25896,28010,27977,27977,30089,30090,1744
4,15298,21670,23750,34347,25928,30154,32234,30153,25928,13185,38541,32234,17443,34314
,40686,30089,25895,23815,36428,23750,19524,28008,40654,34379,34347,32234,28007,25927,
25927,28008,

17445,27978,38542,40687,34315,28009,27978,21607,21607,8962,19526,23752,25865,300
91,36398,34317,21639,19526,21672,6816,23785,34317,25865,27913,11042,38511,19494,17413
,27946,38478,32171,34284,23752,17412,23719,34284,21639,13187,32204,17413,36398,34285,
21607,32172,25833,21607,34253,23752,21607,32171,15267,23719,42800,38574,28009,25863,2
1637,25863,25863,23782,19588,28072,34379,38638,30122,25896,32202,27977,36397,15300,34
284,11009,19525,34284,17445,32203,28009,15331,38574,30090,36429,23751,23751,30091,194
94,30058,27945,38542,34316,15299,17379,23751,38542,23718,25928,30121,28041,32234,3434
7,15330,28040,34315,30154,34347,25896,19557,21670,23783,34348,30090,30123,30122,27978
,25832,27977,13187,13154,13186,30090,25896,21671,32235,30090,23783,34348,17444,25896,
21702,15299,28009,28009,34315,27976,11104,17476,21669,23782,38573,36460,27975,32233,2
3782,17475,25927,23782,30121,15331,19589,28009,32203,25896,42799,25928,25895,30154,28
007,34379,17443,23750,36493,36461,32235,30090,17444,27977,21638,27977,30122,17412,173
79,21637,25928,25896,28008,28008,23815,25895,25863,30154,30121,28008,32202,19556,3642
7,38573,28008,23750,30089,32235,27976,19524,30122,30154,23750,25927,36492,30153,25894
,19589,30153,

23784,34317,34316,28009,40687,28009,36397,34285,8929,4736,21639,42737,40623,2372
0,36430,30123,21671,32172,15300,17413,4736,25898,21639,21574,19526,21607,25833,23720,
25865,32204,30091,23752,23751,42768,19493,23752,27945,32172,19526,23752,40656,30059,2
1639,23752,17413,34253,19462,6816,23752,27945,15300,19525,32171,42768,30090,34283,195
24,15298,23750,32234,32266,32298,40686,40686,34380,23783,32203,32203,21638,17413,2583
2,13154,32204,32203,17444,28009,25896,17444,38542,34348,27977,28009,32171,30091,30059
,25832,23719,30058,21606,17380,11073,21670,25864,34348,21670,21637,30089,32234,32202,
25895,36492,40654,19589,28008,27976,30122,11105,25864,38542,25896,32235,36429,23719,3
6429,25864,21638,11073,21606,34284,19525,32203,32203,32203,28009,36461,23783,23751,13
251,17412,32203,30090,32202,23750,19524,30121,30154,28041,34347,38541,25895,34379,364
60,34347,32234,34347,28041,19589,28041,25928,32267,25863,32202,17411,40654,21669,2592
7,30153,21701,25863,34315,28008,25896,19525,23783,23751,17444,30090,40655,28009,21637
,25896,25863,17476,23782,32234,30121,30089,19556,32267,30153,23815,36428,34314,38540,
30153,23782,32202,36428,23750,32235,19557,32235,30121,34315,25927,30120,30153,32266,3
2234,36428,

30091,36397,13186,28009,34348,30090,27978,23688,17413,15300,25865,30059,27945,19
526,40624,34317,17445,17445,13219,19526,30123,34317,32139,17380,44849,23720,21607,321
72,15268,13154,32204,32204,17412,36397,32203,30058,27945,32172,17380,25865,34317,1741
3,21639,23720,23752,23752,17381,27945,23752,38510,23719,11073,36429,38510,44880,38509

,32169,21669,38573,32234,34347,28040,30121,30154,28009,32267,25864,30122,17413,17380,
27913,19493,32171,30090,30122,23751,17444,34315,38542,38542,21638,28009,27945,15300,2
1639,19526,19493,27945,19525,21606,32235,23718,21637,38574,27977,30154,32235,34348,30
121,32267,21702,30121,17476,30154,19589,32235,21702,32235,38542,30122,34348,30090,279
45,27977,27977,32171,27945,34316,32203,21670,27977,21638,34316,34348,21670,28009,2375
1,17444,17412,19557,25896,25896,38541,40654,40654,25895,23782,36428,38605,32234,34315
,36492,23814,30121,28041,36460,36493,21702,25896,38574,25863,32267,32234,40686,17411,
21702,34315,34379,19589,30121,25896,32235,13186,21670,17412,17444,27977,38509,25831,2
7944,25928,25863,19556,23815,21669,38573,25928,19589,30154,32202,38541,40686,32234,30
153,28008,27976,28009,38541,23751,25896,21702,28009,23750,25895,25928,34315,30121,280
40,30154,19556,

23752,36462,544,17412,30090,30123,30091,21607,13122,17413,21639,30059,30091,3434
9,32236,23752,17412,13187,23752,21639,38575,25897,36397,34317,46930,25832,36430,25833
,13187,8928,40623,17380,30058,34316,36429,32203,25865,30058,19526,17381,21639,11042,1
9494,27945,19526,30059,23719,32171,13154,34317,21639,11074,25832,40623,44848,36461,42
767,30089,42799,32234,25927,30153,30153,19589,30089,27977,34316,32171,15267,32171,279
78,11074,25832,34284,36429,17412,17412,30122,34348,21670,30090,25864,30058,15267,2375
2,8928,15299,23719,21606,8960,27977,21638,8928,25896,30122,30090,23783,32202,38541,28
009,19589,23782,23750,38606,34348,28009,32203,34348,30122,27977,36462,38574,30122,237
19,30122,25896,36461,27945,30090,19493,27977,32235,27976,34348,36461,30122,23815,1533
1,32235,21670,30122,28009,40654,36428,32234,23782,30121,42831,34347,25895,28040,34347
,36493,27976,21637,40719,38574,17444,23783,30090,30154,38574,36493,19589,21637,21669,
25928,21669,25896,34347,30154,36428,32235,17412,19557,25864,30090,32235,30121,17411,2
1670,32267,30089,19589,28008,38573,19556,38605,40686,36460,34380,32234,28008,28008,27
976,32234,32202,40654,25896,28009,23751,21670,30090,19589,25928,25895,28040,36460,259
28,19556,

19493,36430,11073,19525,19525,17412,30091,17381,21574,13187,15300,19493,17413,38
510,19493,38542,11074,19493,36430,21638,25865,30123,27945,36365,42736,15267,17381,343
17,21639,13154,17413,19557,25864,19525,40623,32171,23752,25865,25865,25865,21607,1315
4,32171,23719,34317,42736,19526,32171,19526,38510,32204,30091,25833,44849,25864,38541
,42767,28008,36492,15362,23782,25928,17443,30121,38541,32202,23751,27977,11074,17413,
23719,30058,19493,44881,27977,23783,21637,23783,42767,15331,28009,32236,27945,15267,4
736,15300,21638,36397,25864,17412,17444,25864,23718,44881,36429,21670,17444,19525,301
54,30122,34380,17444,34315,25928,42800,34315,38574,36461,28009,25896,30122,32203,3854
2,32171,23751,32235,36397,23751,21639,25864,30090,42768,13218,30122,32235,30122,23783
,23783,30090,38574,34316,34348,38541,36460,23783,28008,25928,25928,25895,42767,34347,
34347,32267,25895,28041,38574,40655,25864,21670,25864,34316,25896,34348,19557,27976,2
7976,21670,23782,34380,34348,30089,23750,44880,21638,19557,27977,25863,32202,21670,17
443,28008,28008,30121,28008,34347,28040,21701,28008,34347,32234,32234,23783,32234,301
22,25896,21670,36428,30122,23783,34316,13218,13186,30090,25896,30154,19556,38541,3015
3,30121,32267,

30058,36397,23752,23783,19525,15267,17413,25832,38510,23752,17381,19525,23752,32
236,30058,36397,28010,23784,38542,6880,28010,28010,25832,30090,34284,32171,4736,36430
,23719,25865,13187,30090,40623,21670,34316,19493,21639,27978,34284,27978,25832,13187,
25832,34316,32203,27977,25864,23752,23751,42769,34317,36398,15268,27945,21605,30089,3
8573,19524,38541,30121,28008,21637,30122,38573,32202,21670,25864,30058,15267,11074,36
397,32203,23784,42768,28009,27977,40654,25896,27977,15331,38542,32203,19526,19558,688
0,17413,34284,36429,32203,13218,17412,34316,30057,40623,44881,23783,17412,23783,30090
,30122,30122,36461,23783,23782,38574,34348,38541,40687,38574,32203,27977,21671,40688,
30123,21638,30090,40655,23719,23752,23719,25896,28009,17444,32203,28041,28009,17412,2
1670,28010,25864,32235,32202,34315,36461,34380,23815,34380,25895,32234,32234,38606,40
718,32234,25863,28009,34348,36396,27976,32235,27977,40687,15331,34316,23751,28041,280
09,21638,17444,34348,30122,32235,30090,30122,28009,19525,28009,23750,32234,19556,2378
2,23782,32234,15330,32234,32266,21637,36460,25928,36492,23815,28008,25895,36460,32235
,32235,17412,30090,21670,25897,30090,13186,11073,21638,34348,38574,23750,42800,25895,

36492,32266,

30091,30091,23719,38510,27977,15300,11074,17380,32204,27945,13186,19558,21671,25
864,17412,15299,25832,34316,23719,11073,15332,19493,25832,32171,27977,27945,25865,173
80,36397,25832,36397,32203,34284,27945,34284,23719,27945,21639,34285,27946,25865,8961
,30058,17412,23751,27945,32171,32203,19526,36429,25833,38543,2592,21607,23718,27976,1
3153,34315,25895,25863,23782,32202,36461,32235,32235,21638,19525,25864,25865,32171,30
123,25896,32203,30090,32235,34316,36429,32203,21638,42768,38574,23751,23751,34284,342
84,13154,25864,30058,34284,11073,19493,27945,30090,38509,23751,21670,30090,25864,2586
4,17412,17444,34348,27976,25895,36461,32202,25863,38542,30122,27977,30123,17412,27977
,36429,27977,34284,38542,32203,32204,15299,25896,21670,17412,23783,32235,21702,19557,
30154,30122,25896,30090,23783,23783,30089,32234,25960,21702,23783,23782,30153,38573,3
0121,36428,21670,19557,30089,27977,23783,19557,23783,28009,9024,28009,32203,30154,238
15,21702,17444,30090,25896,25896,28009,30090,30122,23751,30122,32202,32201,28008,3223
4,25927,32234,17443,38573,30121,25863,40686,32234,34347,28008,30121,36460,32235,27976
,23783,17444,30122,21638,15331,19525,17412,30090,27977,27977,28009,32235,36461,21669,
23815,32299,

25832,25865,32204,21638,23719,17380,6848,17412,34317,21671,11073,21671,17380,300
91,4768,13186,21606,15300,13154,15267,30090,36429,27977,25864,40623,32204,21639,19526
,34284,15267,32203,36429,28009,23719,23719,23752,38510,21639,30059,30059,30058,21639,
23751,11073,17380,36396,25864,36397,32171,30091,25865,36430,2624,30058,25831,21670,21
669,36461,17379,11072,25928,34315,34348,32235,25864,17412,38542,30122,30058,38510,364
29,19525,30090,23783,34348,21638,30122,21638,11073,36429,27977,21638,32203,38542,2583
2,21671,19493,30090,34284,19557,30090,23719,28010,30090,25864,28009,34349,23751,25897
,25864,21671,30122,28009,23783,38541,34348,19525,32203,34348,30090,34316,15299,19525,
34316,15299,34316,36396,30090,19525,21606,36462,28009,11073,28009,32267,25896,21670,3
0123,34316,17444,30090,28009,19524,23750,30154,25928,30121,25896,23782,28041,21669,11
104,38573,25895,21637,38574,32170,19525,19525,25896,25896,23783,30122,38574,38542,322
35,21670,17444,28009,21638,28009,23783,25864,25896,21638,23783,30089,32234,38573,1955
6,25895,34347,21669,36460,21702,30089,34315,40686,25927,17475,30121,36460,25863,21670
,19557,21638,25896,19525,21638,25897,28009,27977,32203,21670,30090,30122,25896,17411,
28008,32235,

15267,23752,34284,38510,17413,17380,19526,11074,21671,23752,13186,30091,19526,27
945,6848,30123,13219,11106,19525,38574,42736,38574,30090,30123,21671,17412,23752,2160
6,15267,23752,13154,30058,38574,23751,30058,23752,34317,27978,40624,27978,27945,32171
,32171,25864,32171,32171,27945,11041,19493,19526,38543,27978,6816,32171,21638,21637,2
3751,36428,17443,13185,36428,34348,28009,32202,19557,36428,30057,30090,25832,27945,25
864,15299,21638,19525,36461,23783,30090,21670,17444,25864,27977,28010,34349,30123,174
45,36462,32203,27977,32203,28009,19557,38542,32203,32203,28010,32203,30122,21671,1952
5,32203,32203,34348,23751,17444,34316,36429,19525,23783,40687,40655,30123,23751,23783
,27977,27977,30090,30057,30122,17413,27977,42768,25864,21638,25896,36461,27977,34316,
32203,17445,11073,30090,21670,23718,17444,28041,30186,32267,28041,23782,19556,17476,2
3750,30122,23783,27976,36428,17412,17412,23718,38541,23815,30122,28009,34348,30154,28
009,27977,25864,23783,32203,30122,21638,15331,34348,15331,21637,25863,30153,42831,153
30,28008,19556,32234,34315,19589,25895,34379,28041,34347,13217,30121,32234,19589,1747
6,17412,25864,23751,25864,23751,25896,27977,25897,21671,28009,21638,21638,25896,21670
,34315,27976,

17413,36430,21639,40623,27945,23719,21639,19558,25897,23751,30123,40656,15332,21
639,19558,38575,17413,21639,34284,42768,25832,27977,34349,30123,27945,25864,30123,406
56,6848,17380,15332,28010,34316,27977,36429,23752,21606,30059,30058,30091,27978,40623
,34317,32203,36429,27945,30122,19493,17380,36397,38510,19526,15300,25865,34316,30089,
25864,36428,25863,38574,36428,27976,25928,23750,25864,36461,25864,13218,25864,34316,2
5896,6912,19589,21670,25864,30090,30123,30090,17412,36429,23783,27977,30090,34316,216
38,34316,38542,32235,25864,40655,15331,23784,30090,36462,23784,32203,21638,25864,1533
1,25896,25864,23751,23750,32202,30122,34348,19557,21670,36429,32203,23751,17412,28010

,17412,25896,21638,30122,36461,23719,34284,38542,15299,25896,30090,32235,38542,32236,40687,11041,23784,23751,21638,32203,19524,32234,34412,30121,30154,28008,25895,25928,19589,40719,23782,27976,34315,11072,30154,38541,38574,21670,34315,38574,25896,15363,32235,23815,30122,34316,25864,25896,19525,21670,34348,19557,23750,23782,32234,36460,23750,27976,21670,34315,34347,28008,21637,32202,34380,32234,25895,34379,32266,23750,19589,15331,25896,23816,21670,23783,32235,38542,32235,21670,19557,17444,28009,32235,23783,30154,23783,

32171,27978,8961,32171,27978,34317,21638,21671,17412,28010,32236,36462,17380,21671,23752,23752,30058,23719,38510,30090,13154,17380,34349,25896,21671,25897,15300,25865,27978,23784,25897,15267,25896,34284,32171,25832,15300,27978,32204,32172,13187,40656,27945,32204,32203,34316,25832,32171,21639,25864,23719,13154,21639,17380,30090,19557,19557,25863,36428,32202,28009,23750,21670,28009,25896,17412,19525,17412,27945,32235,17444,17444,21702,19557,25864,30090,40687,30090,30090,25864,30090,21671,30122,28010,21702,25929,34348,36397,23784,32203,13186,25864,32235,25897,32203,27977,21638,11106,34316,28010,34348,34316,28009,34315,21638,36428,25864,15299,40655,28010,30122,32235,32203,25864,27977,34316,28009,19525,21638,23751,23751,8992,28009,34348,23815,28009,38574,34349,15299,15299,19525,15331,38541,23783,34347,30121,23847,25927,30121,21669,32235,25928,38606,19556,38574,25896,13185,40654,36429,25896,30154,36460,34347,23750,15298,38542,21638,32235,25864,19558,25864,25864,30090,27977,28009,19557,32202,34315,36493,19556,36428,27976,30121,32234,28009,15331,25895,34315,34379,38541,30121,30154,28041,28009,15331,25896,15363,23783,32203,34316,36461,25896,27977,25864,23783,34348,38606,23815,21638,19589,

42769,6848,15300,21607,27945,36397,13219,11105,21670,36429,27945,25897,8993,15332,25833,19526,21607,21606,32171,36397,15267,15267,15331,25864,15332,17445,23784,30091,38543,23752,19525,15332,23719,32203,32204,25865,23752,25833,21607,25865,8928,25832,30058,36397,21606,23752,25864,36429,25832,36397,34284,17412,19525,17380,25864,17412,27977,23783,21670,23751,23783,23783,25896,30122,25863,23751,36429,32203,19557,32235,23783,23815,19557,32203,28009,32235,40688,34349,23784,21671,23751,15332,19557,30122,21670,32203,30122,25832,23751,25832,23784,34316,34316,17445,21638,17444,34316,15331,32235,30122,38542,30090,27977,23783,28009,30122,25896,13186,32203,30090,23784,42768,25897,25864,32235,36429,25896,30090,25864,17380,25896,15299,19557,36429,19557,30122,30090,30090,8993,17445,21670,25864,34316,25863,30121,23782,15330,21669,30121,23782,25928,25895,38606,30089,40686,28008,36461,27976,25928,25896,40686,38541,30121,15363,34348,28009,25896,15364,25864,17444,38542,25864,34315,27976,28009,17444,23815,32234,30122,8992,30089,28009,21670,34283,21670,17411,34315,36428,34347,34347,25927,25928,36460,21702,17411,23815,13250,23783,36429,25896,27977,30122,32235,34348,36461,38541,38541,27976,25864,30122,

32204,8961,25865,13187,17413,30058,30123,32203,21671,28010,30090,15299,23751,25897,34284,27946,19526,21607,27978,32204,30091,19558,11041,30090,25897,17445,11074,32236,25865,21639,27945,19525,21639,38510,27978,19493,23720,25865,17380,21639,19526,21606,30058,34284,23752,19526,25832,21638,23719,36397,13186,19493,32171,19525,25864,17412,23751,34348,23751,25864,44881,42768,34316,25896,30090,30122,38574,36429,40655,32235,25928,30154,21702,28041,30122,25864,32236,25897,27977,13219,21670,8960,13251,30090,28041,32267,21638,25864,17413,34317,32236,28010,25864,21670,17412,30123,27977,36461,34316,34316,25864,28009,23751,27976,21638,36461,27977,13186,25864,34316,30090,30123,32203,28010,36461,36429,23751,21670,32203,15331,23783,19525,34315,38542,21670,25896,21671,25929,13186,19558,19558,40688,23719,28009,21669,17443,28041,28008,25895,25895,21669,23782,30154,36460,40687,28008,32235,21702,21670,25928,34379,34380,32234,15330,36461,28009,25928,17444,27977,21670,23783,32235,23783,25896,32203,13186,34348,36428,25896,8960,32235,30089,19492,23750,23815,34380,42767,36428,34379,34315,32234,30121,36460,32235,27976,30089,13250,38574,32235,21638,30089,25864,36461,34348,36428,32202,34348,23750,36429,25896,

17381,17413,15235,13187,15268,30091,42768,17476,19557,34316,30123,21671,34349,21639,21607,32172,21607,27946,38511,42737,27978,15299,8961,17445,25865,32236,6880,32237

,32204,25897,30091,21639,25832,32204,30059,38510,11074,8929,8929,34284,21639,15300,23752,34317,30091,13187,44849,21606,25832,17380,25864,34316,27977,21638,30122,19525,32268,23751,23751,19557,36461,32235,27977,19525,32235,21702,36461,44913,25928,25928,19557,15363,19589,32235,28009,32203,30123,25864,27977,25897,21671,8960,21670,23816,32235,28009,23751,15332,32203,19493,28009,30123,13218,23751,19557,40687,32203,42768,25864,23751,23750,19525,32235,34316,25864,34316,19525,23783,34316,30090,17412,25896,30122,25864,32235,42768,21638,21670,23783,25864,11105,32235,27977,32235,23783,21670,25896,32236,21671,27977,23784,42800,23751,21637,23782,23782,34315,30121,28008,17475,38573,25895,32235,32234,23815,25895,13217,25896,34348,30089,36460,36460,38573,15362,30121,21702,21670,25896,25897,19525,28009,36461,21670,25896,36461,15331,30122,32267,32202,21670,28009,25864,21638,27977,34348,40654,40654,25863,30121,34315,42799,28008,40654,40686,32267,30154,30089,38606,28009,28041,28041,23783,38541,32267,36461,27976,30121,27976,25864,19493,

34317,15300,36365,30059,11041,25865,32236,17444,30122,21638,40655,30123,32203,25865,17413,15300,32204,42737,23720,27978,25865,13154,11073,23784,23751,27978,23720,36462,34317,30090,38575,30091,27945,32204,19526,27913,15268,15300,8961,34317,15300,11042,27945,19493,34317,13154,36397,27978,21638,23719,30090,40655,19525,23751,21671,19558,19558,19525,21638,23751,17412,27977,30090,25864,23815,25864,30090,32235,25864,19589,15298,23815,15299,21638,30090,30090,30122,13186,21638,23784,23784,17412,21638,23783,23783,32203,32235,15332,13219,23719,19558,25864,13186,23783,36397,27977,25896,34316,27944,17379,27944,34283,36461,30122,28009,19525,27977,25896,38542,28009,17412,23751,30090,32203,30090,32203,28009,27976,23751,21638,25864,34316,17444,32235,32235,19558,25897,23784,34284,27977,27977,30090,23783,17412,25863,28008,36492,21669,19556,36460,42799,30121,19524,19556,34315,34380,8992,36460,40719,28008,21669,36460,30089,15331,25896,25896,30122,23783,30090,17412,28009,32235,25864,23750,34315,23783,28009,30089,36428,30089,25864,27977,30122,30090,27976,40654,34315,13218,21637,28040,34379,32234,36493,38574,23783,25863,30154,34347,28008,32202,25896,28009,30122,28009,25864,28009,36429,21605,19525,19557,

32204,15267,32204,17380,34284,27945,21671,34316,34316,27977,30122,30123,23783,40688,17412,30090,25832,36430,21639,13187,32172,30059,19461,19493,40656,25864,32171,40623,27945,30091,36397,32171,42769,36397,17413,25833,27978,6816,27978,32204,19493,21607,30059,13122,13155,11009,36397,32171,30090,21638,25896,32203,15300,19525,21671,19525,21670,17412,23751,25864,19525,17412,28009,25864,15299,17412,28009,30090,15331,28009,28009,25863,21670,21637,27977,36429,19525,13186,27945,25864,27945,17380,13186,19557,19557,32203,34348,17412,15331,28009,25865,15299,11074,36429,36429,38542,32170,21638,36396,21605,40654,42800,27977,27977,23783,17444,36428,27976,32202,23815,21670,34316,34315,32202,21637,32235,34315,23750,30122,19492,36396,23751,15299,40623,30026,25864,19493,19494,42704,30058,25896,28009,34348,17412,21670,28008,34347,23782,13249,38573,30153,28040,17475,25959,21669,30153,34380,28041,38573,25895,25928,36461,19557,17476,27977,32235,34316,21670,25864,27977,19557,32235,25896,23750,32234,42800,28009,30090,34316,30090,23783,25896,36396,30057,30089,36428,32234,17444,23782,27976,17411,28009,32267,25895,25895,19588,21637,34347,38606,32266,34380,27976,23751,25864,21670,30090,30090,17412,30090,19589,

27945,27978,25832,13154,15267,32203,28009,25896,28009,28042,32235,25929,9025,15364,19525,36397,27978,40623,27913,6848,21639,25833,19526,23719,32171,21606,27913,34284,19493,32204,32171,21639,49043,27946,32172,32172,21574,15235,38543,32204,25833,25833,36430,11042,4704,13154,34316,30058,27945,21606,36397,23719,19526,23784,27977,30090,23783,28009,28009,28009,25864,17412,15299,17445,11073,25864,27977,15332,27977,27977,19557,19492,28009,17444,34283,36428,27944,15299,30058,34316,19493,13154,15299,15300,23719,25897,25897,11138,32236,38575,30091,21639,21606,38542,27977,40623,25832,25832,42768,19492,27977,34315,21605,27977,15331,23783,28009,25896,25928,30154,38541,34348,40654,27976,30122,23782,38574,21702,27976,27944,34348,23718,13153,27912,30058,23687,19461,23687,34284,27977,27945,30058,13218,21638,28009,30154,28009,32267,13249,34379,32233,23814,19555,32266,21701,28073,30121,34380,30122,17476,25896,34315,15299,23751,30122,30154,34348,28042,25896,32203,21702,38541,25928,21637,32202,42767,32235,40655,21605,2586

4,34315,27977,42735,27977,21637,32170,27976,17411,27944,23783,30122,36428,36460,28041
,30121,25895,21669,32234,36492,32267,36493,30089,30090,30122,25864,19525,27977,25864,
40719,25928,

21670,30090,21638,13154,21671,19525,34348,34380,34316,30154,25896,23783,13251,11
105,30090,34316,38510,40623,23720,11042,17413,19493,23720,27913,38510,23720,21606,152
67,27945,40656,27978,19525,23719,6848,25865,25800,21639,38510,25832,25897,34284,19493
,21574,19526,6816,13154,40590,30025,25799,21638,40655,27945,19558,19493,13218,28009,3
2235,34348,38542,30122,27977,11073,21638,25864,15299,21671,23784,34316,30090,19557,19
525,21670,34315,13218,27977,25864,21670,34348,32203,27977,15267,11105,21639,27945,258
65,27945,19525,23816,38542,32171,34316,34316,30058,30090,15300,32138,32138,44849,2794
4,23718,30090,23718,23750,25896,27977,32235,34347,32267,32267,30121,32235,32170,42767
,34315,25863,17411,36461,21637,25895,32235,36461,19557,15299,30090,25832,8928,15299,2
7945,32171,25864,32171,15299,21638,27977,23783,23783,30122,40686,30121,32234,28008,30
120,25895,30153,23814,21702,32234,30154,15331,28041,32267,28008,19589,30122,38574,195
25,36493,25928,28009,30090,19557,34315,32235,21670,36461,36460,32170,27944,19525,1955
7,34316,36461,36428,28008,15298,34315,32235,21605,23750,23783,42767,28009,21670,17443
,25927,38573,36492,28040,38541,34347,34315,17444,38542,30122,23751,13186,42767,32235,
42800,23783,

19493,36429,19493,30090,21671,34316,32267,25896,34316,34348,36493,21702,36461,28
042,34348,36462,32171,17413,30091,19526,17413,19461,30091,34284,36430,27978,27945,216
06,25832,27978,42736,19493,19493,13187,21606,32139,25800,32204,17380,30090,36429,2583
2,30091,21638,11074,32171,27945,27977,42735,21605,40655,32203,17413,8960,13186,36461,
27977,30122,38542,34316,30122,21670,23751,19557,13218,15331,32203,27977,23784,17412,2
1670,30122,25863,34316,27977,17412,25864,34283,34284,17412,19525,15299,23719,27978,25
865,32203,25832,40688,32204,32203,23751,27945,32171,27977,23719,40623,21638,19493,385
74,25864,38541,19557,34316,32203,42800,30154,34348,19557,32267,32234,28008,36460,2797
6,8960,23750,19557,25863,32267,21702,34347,36428,25896,21606,34283,19493,21606,23719,
27977,42736,21606,15299,23751,19557,25864,30122,21702,28009,36461,34347,30154,34379,3
4379,25927,30153,19588,21669,36460,21702,21702,40686,28009,28009,23783,25863,40687,23
783,30122,32267,32235,25896,25896,30090,27976,30089,32202,34315,30057,23751,21637,216
37,32235,34347,32235,36428,19524,30057,30057,27976,34348,25863,32202,32170,15330,2167
0,40686,42799,36492,19556,34315,28040,27976,15363,36429,21638,25896,17444,40655,32203
,32235,28009,

25864,27945,42768,36429,30123,40687,36461,36493,36493,34315,38542,25864,34348,38
542,42768,40655,17413,13187,32139,23752,34317,8961,23720,23720,27978,25865,32171,3643
0,32236,25864,38542,17445,23752,19558,21638,27945,23687,27978,23719,36429,25864,32204
,34284,21606,32203,32203,38509,30057,38542,17412,40655,40623,25864,30123,15299,21606,
25832,21606,25831,21638,21670,32203,25864,30090,19557,23751,19589,21702,27977,17444,1
9589,23783,30122,30122,23783,19525,19557,28009,19557,25896,30090,23751,25897,36429,34
316,19526,25897,36462,30123,25897,21606,11041,23720,25833,36397,30058,11073,13154,385
10,19492,32170,21638,34316,38574,38574,19557,28009,23782,30122,25928,27976,34283,3008
9,17411,21637,25896,21702,30122,9024,34347,36461,25863,30122,25896,21671,30058,42800,
25864,40623,21606,28009,27977,23751,28009,36461,21670,36429,38541,17476,36428,23781,2
1669,19588,34346,30153,28040,30186,30186,36460,32234,32234,19589,32234,17476,25896,28
041,30122,28041,36461,28009,25864,25864,25864,30122,32203,25896,32202,25863,25831,385
41,32202,30121,32234,34347,28009,25895,46961,30089,34283,17444,23750,34283,19557,1536
3,38573,32234,28008,21669,17475,23782,28008,25896,25896,27977,32203,30090,36461,34380
,21670,28009,

17444,21638,38574,34316,30122,34316,38542,30089,38574,28009,23751,23751,34348,38
542,32203,34349,15267,17380,30091,25833,44818,4736,15300,27913,36397,30091,38543,2375
1,23784,13154,27977,38543,21606,36430,34317,42769,23719,21606,34284,36429,27977,34316
,32171,17380,44849,25864,30057,30057,36396,23751,34284,38543,40688,46994,32171,30090,
38542,19493,19525,42800,13218,25896,30122,27977,27977,23783,23783,15363,30090,23783,2
5896,28009,30089,19589,19525,30089,30090,25864,17412,28009,32235,36429,40656,32236,25

864,8960,30090,27978,23719,23784,19493,17413,19526,32139,40655,23719,17412,23719,1952
5,25864,34283,32203,27977,25896,32202,21670,25896,30122,19524,23750,17411,25895,25863
,23783,19524,17444,38606,23782,17443,34347,30089,36461,34380,28009,15331,25897,25864,
21606,38510,34316,32203,21638,27977,32171,27977,28009,25896,28009,11137,34347,17475,2
8008,30153,32233,36460,32266,30121,34379,34379,38573,36492,30154,32235,23782,25896,34
348,25928,32235,36429,32202,30122,25896,13218,23783,19525,23751,36428,32202,21605,427
67,27976,28008,34315,36395,32235,32202,32170,27976,25896,8992,15299,27944,15298,17443
,30089,28040,21701,25927,17443,25895,36428,27976,28009,25896,28042,15331,36461,28009,
32203,30122,

15299,34284,30090,17412,44913,23783,23783,21670,28009,21670,17444,34315,25896,27
977,32235,42768,19493,34317,23752,30091,32172,15268,15268,21607,25833,32172,34316,174
45,21638,11106,40688,36461,19558,40688,40655,34316,30058,38543,40623,32203,40623,2378
3,17380,6848,42736,23751,30057,30090,30090,23751,36429,25832,44882,42737,23687,42736,
32171,30090,40623,30058,23783,21670,32235,32202,27977,25928,32267,11105,28008,25863,1
9557,38541,28009,28009,17412,32235,27977,23750,15299,23751,23783,38542,25897,36430,21
671,25897,36430,25865,30058,27945,23752,23752,34317,38510,30058,17380,25800,36364,237
51,15331,36429,40687,17412,11105,25896,23783,28009,38574,27976,32202,32202,25896,2163
7,28009,19557,30122,34348,19589,30154,32266,28041,32234,23783,28009,15331,19589,30122
,25896,27977,34316,25832,11073,34316,34316,30090,32236,19557,17444,17476,34348,19556,
36492,28008,34347,38573,32266,32266,40686,36492,32266,36428,38541,32267,25927,30154,2
5895,15363,30154,30090,27977,19557,28009,13186,25896,21638,30090,28009,32235,30089,36
460,27976,36428,25895,32202,40686,25863,40654,25831,21637,15331,19557,23718,36428,237
83,32202,28008,30121,32234,15330,32234,34380,28009,19557,25864,25896,21670,32267,3226
7,38574,34348,

34284,17380,19525,23718,38542,19557,25928,21670,23751,30089,34348,32235,13186,25
863,36429,34316,17412,34284,30059,23752,27946,27979,11074,17413,21639,30059,34317,237
19,9025,23751,32236,34316,19525,36462,36397,27977,23751,30091,27945,32203,30058,25864
,21638,17380,21606,19525,34348,42735,17412,11073,34284,17380,32172,27946,30026,30090,
34251,32171,19493,13218,32203,13218,34348,30122,17411,25863,28041,17476,30154,34380,3
0122,28009,34348,32202,21638,30122,32235,17444,34348,23783,32203,28009,23751,36429,34
316,40688,21639,36397,40656,25832,34317,30058,40591,25832,25800,19493,25832,36429,131
86,21638,40655,40687,11040,15299,25864,30090,25928,19589,23783,36461,34315,27976,1741
2,36428,34315,28009,28009,23783,42832,32234,28041,32267,19589,30122,25896,25864,36461
,36461,34316,30090,21638,6848,36461,34316,28009,30090,15332,19525,15331,28041,36460,4
0718,38605,34347,36492,32233,28040,42831,32234,28040,25928,30153,32234,32266,21669,30
154,15298,36461,34348,21670,30090,23751,23751,23783,25864,34316,21637,32202,34347,364
60,19556,28008,36460,19524,30089,25863,32202,30089,17411,34283,34316,25863,32267,3642
8,32202,23782,34347,23814,30121,32267,38573,21702,11105,21670,32267,32235,28009,32235
,30089,30154,

25832,21606,32203,21638,34316,19525,28009,25896,34316,27977,32203,32203,8960,237
51,38542,25832,25832,42768,42737,34285,25801,30091,11075,6848,17348,30059,40656,23751
,23751,34316,30123,28010,25864,25864,34284,23752,19493,36430,19525,38510,32171,32204,
38543,17413,19493,34284,32203,36429,23719,11041,25833,11074,30059,34285,36430,21574,3
0025,13186,25864,36396,27977,25896,23751,27976,28009,30154,19556,23814,38605,28041,42
800,40654,36493,27976,32235,32235,30090,21638,34348,30090,27977,17444,30090,28010,364
29,36397,15267,34284,32171,21639,36365,27913,27978,30026,27945,19493,23719,30058,1738
0,28009,38574,34316,13186,28009,32203,30090,32267,17444,23815,32235,36396,23718,15331
,32202,30122,30154,21702,28041,32267,23847,19556,28009,13218,28041,28042,32235,36461,
40719,34316,27977,25864,15298,32203,36429,17444,32171,17412,19557,21638,32202,25928,3
8605,42799,34379,28040,36460,34347,34379,32234,32234,25895,36428,34347,30153,32202,34
315,17411,42800,40687,25896,27977,28009,23751,15332,21671,21671,23783,21702,36461,322
34,23750,28041,32234,17411,27976,32202,19524,38509,27944,36428,36396,30090,40654,3226
7,25896,30089,34347,23814,25895,30121,34380,21670,17444,25896,38574,21702,30122,34380
,28041,21702,

30058,15331,34284,21638,15299,13186,19525,36429,30090,25864,13218,32203,30089,40655,27944,23751,27977,40623,42769,36397,17381,19494,8929,13123,30059,30091,34317,19526,30123,34317,30090,23751,23751,25864,21606,21638,15300,32171,17380,38543,23752,32171,30059,23752,25865,36430,32204,38510,17413,19493,13187,27978,19494,32172,23687,11073,15299,17380,36429,38542,28009,15331,11072,25896,30121,36460,17443,28040,34347,40686,30122,32235,34348,34315,34315,11105,17412,38574,30090,23751,15332,28010,32203,28010,34349,27945,27978,21671,27978,27945,32139,27913,27945,23752,25865,32204,34284,13186,25832,32203,38574,32203,27945,36461,25864,32235,38606,32267,32235,40654,34347,19524,13218,44848,23783,34315,30122,21702,19589,25928,23782,25928,23783,21670,28009,28041,32235,36461,27977,30090,25863,13185,40622,44848,19558,21638,27977,21670,32202,19557,30122,36460,40686,32266,23814,28040,36460,32266,36492,21669,32234,30153,38605,23782,34315,36460,17443,25895,42767,25896,32203,34316,13187,21671,23751,30122,30122,25895,17411,32202,32235,32202,27976,19524,34315,21637,11072,27976,42768,25896,25864,36429,34283,21670,17443,32234,25927,25863,32234,23782,25928,30122,23751,28009,28042,19557,32267,23783,38574,13250,

25865,13187,36429,17444,4768,21638,21606,36397,19493,23751,11105,36461,34316,23751,23751,27977,27977,30090,38543,32171,19494,21607,17382,19494,25833,36430,15300,15300,25865,27977,32204,34284,23752,23784,13154,32204,30026,17348,21606,30091,25832,23752,34317,19493,30091,42769,36397,27978,6848,19493,23719,30091,19494,21639,23719,8928,25864,27945,27944,38542,34348,23751,25896,25896,32267,28008,28041,21702,30089,38573,25863,30122,32235,30122,25863,17444,21638,44880,30122,15331,17444,32203,25864,30123,38542,40688,32171,25864,25832,21639,21607,15300,27978,23719,27945,34284,23752,19525,21670,44881,32203,13218,19557,40687,34316,38606,34348,30122,36493,23782,40687,19524,17444,44880,28009,27976,23783,36461,23815,28041,25928,23815,19557,19556,38574,25928,30090,28009,25896,25864,34283,19557,32203,40655,23751,23784,23751,17444,32203,13218,30121,21637,30153,21701,19556,30153,40685,40685,23781,25894,19588,30121,36460,34380,23750,32266,19556,21669,34380,27976,36461,36461,6848,30090,27978,34316,25864,25896,8992,32234,36460,34347,27976,27976,25863,6848,17411,30089,30090,23718,21638,30122,25896,25863,21669,34315,21701,38573,34347,17476,15363,25863,21670,28009,32235,28041,34348,19589,32234,19557,

30058,11073,36397,15267,8960,13219,30058,32171,17380,23719,27977,38510,25896,17412,30090,38574,32171,27945,25865,27978,27946,23752,23720,13123,21640,27946,6816,27946,27946,21639,38575,28010,30091,19558,21639,38510,23687,19461,21574,19493,30059,30059,21607,32139,40656,38511,38510,30090,4704,6816,30058,27913,19494,21607,25832,17380,28009,36396,34316,34316,25896,28041,38574,8992,28008,34347,38573,25927,25927,38573,27976,32235,19524,13218,30090,19557,25863,30090,21670,34316,23784,19558,21671,21638,40688,38575,15299,32203,27945,8928,21607,13122,17348,21606,23751,27945,34316,27945,36461,15299,19557,21637,36461,42800,34316,36428,32267,25895,23782,21637,38573,25863,27976,40654,34380,19557,38574,30154,28041,38541,25896,25863,19589,30154,30122,32268,30122,36493,27977,34316,30057,17444,21638,34284,21638,23752,21671,36461,19525,17444,36428,19524,32202,17443,23782,38572,36492,32234,17475,30153,11136,40718,38573,40719,30121,32234,19589,17411,34348,23783,23783,25864,8993,36429,32236,27977,21638,32235,17444,30122,30122,21670,28008,38509,27944,21670,40654,34316,34283,21638,21638,25896,23783,19524,19557,30121,34347,34347,30121,21702,28008,23815,19589,28009,25896,23750,30122,32267,30122,25895,

32171,21606,40656,13154,36462,23784,38510,19526,23752,13187,34284,34316,21670,32235,21638,40655,34316,27945,15300,34285,32204,23720,21607,8929,13123,21639,19526,25865,27978,19494,23752,36430,30091,25865,21607,25833,23720,25833,32171,21607,34252,25865,27946,27978,21639,32236,23784,17413,8961,17412,21574,30091,17381,23720,17413,32236,28009,32235,30122,11105,21638,25896,34348,23783,36461,25928,38541,23750,36492,40686,25928,21702,15363,17444,32235,28041,32203,28041,19557,25864,30090,17444,25864,19525,30123,21639,23752,17380,17413,21606,25832,6816,23719,19461,19525,38478,23751,23719,30090,38542,36428,28009,34316,36428,34380,32202,30154,21702,34315,30089,32234,36460,32202,44880,34315,11104,23782,28041,28009,34315,19557,34348,25928,23783,32235,30122,32203,34380,21670,21670,17412,23751,32171,15267,19525,23752,30122,36429,23783,19589,25863,23

783,30121,23814,36492,38573,36460,32266,23782,32234,32234,40654,25927,28040,30121,217
02,25863,23815,32235,23750,21670,27977,25864,38510,28010,23752,23783,34348,23783,2589
6,34348,15331,25863,32235,32170,28009,40622,21637,38542,34316,23718,36396,27977,30089
,21670,32266,34315,32202,34347,21669,15330,32267,21638,21670,21702,25928,23783,32235,
25928,25895,

19493,23719,23752,23719,30123,19526,30059,15268,17381,13154,21606,34316,19557,27
977,27977,34316,25864,13154,17380,36430,23720,17381,15268,13155,11010,19494,27978,279
46,23752,21639,32171,34285,32172,40656,25833,32172,30059,32172,40624,25865,32204,2160
6,21607,17381,8993,25864,17412,17412,25864,42768,13186,8961,30026,17413,30090,32203,2
8009,23751,19557,13186,30122,23783,25928,32267,25895,23815,36460,32202,34347,30089,19
589,36461,23783,25896,34348,32235,32235,34380,21702,28009,34349,21638,30123,25929,280
10,11073,23719,30091,34284,32203,15300,23752,25832,34252,23752,30090,17412,25864,2378
3,28009,25896,36428,30090,19557,32202,32267,32267,28041,25895,30089,28008,34282,30056
,40686,28008,19589,28041,32202,34348,27976,25863,36493,34348,28009,28009,28009,28009,
21670,21703,21702,28009,19525,30090,15299,13186,15267,30090,34316,30089,15331,19557,3
0089,25863,27976,32266,19589,28040,34346,28040,34347,30154,38606,21669,19556,28041,23
815,36493,30089,21702,21670,30090,30122,30090,36429,32204,40655,27977,21670,34348,322
35,32235,19525,28009,27976,23783,38509,34315,32203,42735,32170,15299,40655,25864,2592
8,21702,23782,30154,25927,32266,25927,23782,23783,21670,30122,19557,30122,21670,27976
,30089,21670,

34317,15267,21606,34317,19526,13155,30059,8961,36430,23752,19526,38510,36461,174
12,25864,21605,21639,13154,15300,30059,21639,25833,23720,25833,15236,23688,34253,1741
3,21607,19526,23752,27978,42737,42704,27946,32140,34252,30027,36365,23687,25800,17381
,17380,11041,19526,23751,15331,23783,25864,19492,25832,17412,23752,27945,30058,11073,
19557,25864,25864,25864,23783,25928,19557,25928,28008,28008,19556,34315,30089,21669,2
1637,15363,30122,30122,21702,28009,38574,32235,34316,27977,38542,30122,27977,27977,32
236,15299,19525,36429,42736,30058,34284,32171,32139,32171,32171,21638,25896,32203,216
70,30089,25896,30122,32202,15331,21670,34380,28008,32234,28041,34347,34314,23717,4065
4,32234,28008,23814,32267,34380,34348,27977,17411,40687,32203,32202,19589,21638,19557
,28041,23783,23816,34349,23751,15331,15299,13186,25864,32203,32203,25864,32202,23783,
21669,23782,25895,23814,21669,32266,32266,34347,30121,23815,34347,17411,15330,32266,2
3815,34315,32234,23750,23750,34348,32203,27977,30090,25897,32203,21670,23816,36461,30
122,25864,17444,25864,38542,32203,28009,30057,40655,34283,25864,19492,38509,25864,280
41,23815,21669,25895,28008,21637,28008,28041,21669,32235,36429,25896,36428,25863,2800
9,30154,30154,

23719,15300,32204,38510,23752,8961,17413,40657,32172,25865,30058,27945,40655,152
67,25864,6848,19525,32171,17380,34317,32171,21607,25801,32140,19495,21575,23752,25801
,23752,15268,11074,40656,40624,40624,32172,38511,34285,25833,34284,30026,19493,19494,
21607,11074,23751,15299,15331,23783,23719,30090,36429,30058,27945,34284,30090,8960,13
186,36461,34315,32203,23783,27976,19557,34347,25895,32267,28008,32234,40686,36460,174
76,17476,28009,25896,25896,30154,32235,32235,19590,17444,34348,23816,21671,34316,3857
5,23784,15299,19493,38510,36364,34284,27977,34316,23751,40590,27912,30090,27977,32235
,40655,19524,28009,32235,17444,23815,19589,38573,32234,25895,34379,36460,25863,25895,
25895,21702,21669,32202,32235,42735,27977,13153,40655,30090,25864,21703,21670,19590,3
2235,32203,15331,32235,36461,8928,21638,21606,30090,36397,23719,30090,28009,21638,407
19,23750,25863,17411,23782,32266,42799,19588,32234,30121,23815,19588,23782,19589,2800
8,32267,36493,30122,21670,30122,27977,25864,21671,25864,25897,19557,19525,19525,32235
,19557,17412,36429,36461,17412,32202,32170,42767,27977,25896,21638,30090,21702,30122,
25928,21669,28008,25895,19589,25895,30153,23782,34315,34315,34315,32202,27976,23815,3
4347,28008,

23752,15332,40623,42769,17381,15300,27978,30091,27946,21639,25865,19525,25864,34
284,21670,17412,32171,36397,23752,19493,23719,21639,30091,19462,23720,27946,15300,448
50,27979,21607,17414,36398,25834,23721,36366,42737,27946,21607,25833,13154,19493,2163
9,23752,8961,27978,15300,15332,17412,23784,34316,27977,23784,21670,23751,40622,8928,1

1040,32170,30090,42735,30090,28009,36461,30121,28009,30154,30121,28008,40686,32266,15362,23750,34380,17411,28041,30122,30090,32235,19557,19557,21670,21670,28009,34316,36461,17444,23751,11041,30058,36429,30058,30122,23719,19493,36397,23719,25864,28009,25896,42735,23750,34348,23718,15331,27977,17411,28009,30154,28009,30122,30089,36460,19524,25928,23782,23782,30121,34347,32235,25864,15299,36429,27977,34316,25896,28010,30122,32235,27977,17380,23719,36430,25865,25865,25832,25833,42736,17413,19526,13186,25864,32203,19557,23750,21669,32266,38605,36493,23815,30121,34347,25896,25896,23782,15330,32234,38541,40686,40687,25896,28009,19525,21670,21670,19525,17445,23751,25896,25896,30090,21671,19525,30090,28009,23783,32235,25864,30090,23751,30122,38542,32235,19525,40687,30122,15363,30121,32266,23782,32234,25959,36460,36428,32202,36428,30089,30089,21669,36460,32234,

21671,23784,34349,42801,38543,30091,30091,27978,36430,27978,11073,17380,21638,19525,27977,30057,42768,32203,25865,21606,8961,19526,30059,17413,27946,11074,25865,32139,32172,25833,17381,25834,13156,11043,40657,44818,34317,27978,17413,19526,23719,25865,30059,15268,23784,19526,13187,11073,27977,32236,21638,23751,21670,25864,42736,23718,27977,25864,25896,34283,15298,42767,21670,19524,30154,28008,25927,25895,34379,23782,32234,40686,25928,21637,25928,30090,32203,27977,17412,8960,19493,38542,25832,34316,30122,15267,23719,8928,32171,30058,32171,25832,17412,13154,34316,32203,27977,32203,36428,38541,32170,19524,23686,17412,30057,11073,21638,30122,32202,25896,25895,32267,17476,19621,36460,28041,23815,38606,36428,25864,13218,32203,30090,25896,30122,36462,36462,23719,17380,23752,30090,25864,36397,32204,23752,23720,36430,11041,2592,27977,34349,17412,17444,23783,25863,30089,32202,32234,30153,36492,38605,36428,32235,34315,19556,30089,40686,38573,36428,32202,32235,21670,23783,23783,32203,19525,25864,34316,23751,34316,25864,23751,21638,25864,36429,25896,30090,32235,21638,36461,25864,32203,25896,40687,21637,30089,19524,32266,28008,32266,32234,36492,40686,32202,23782,30122,32235,21670,40654,36460,

25865,32204,19493,32171,36398,25832,17413,32204,40656,13187,36397,15300,30058,19493,27977,34316,42768,32171,17413,30058,19526,27946,19493,30059,27978,4736,36398,38543,30091,23752,21607,27979,17381,416,38543,42769,40656,23752,8961,23752,34284,30091,19526,23720,23720,21639,21606,15267,32171,21606,23784,36429,30090,13218,32203,30058,40623,25832,25831,21605,23718,21670,32202,25863,34380,30121,25928,36460,28040,32266,30153,25927,28008,28041,27977,15331,21670,27977,25864,13186,30058,38542,28009,27945,19525,17412,15300,13186,44881,23751,28009,17380,17412,13186,36429,21670,40655,44881,32235,27977,27944,23751,32138,27945,30090,25831,23783,32202,28009,25896,25895,23815,28008,28041,23782,38573,25928,25895,38606,30122,25928,28041,30122,23783,42768,30155,28010,27977,11105,30090,30122,36429,36461,40623,19526,27978,36430,4704,15300,25832,42768,15331,23783,21670,25896,32235,40687,34347,34347,34347,25863,40686,30121,32266,28008,32267,36493,30154,30154,28008,36428,27976,32267,40655,32235,28009,28010,28010,23783,34316,27977,28009,23783,28009,36461,34316,30122,30090,15299,32235,19557,17444,34283,27976,28009,30089,25928,28008,32267,25927,36460,32266,32266,30121,32234,25895,32235,27976,40654,27943,

13186,21606,23719,30059,21639,25865,13187,23752,30091,25832,40656,27977,34316,23751,17412,23751,25864,30091,30091,32171,30058,25832,15267,40623,27945,27978,44849,27978,36397,19493,32204,21607,19527,2624,40624,40624,32204,21639,15267,25865,21639,40623,38510,27946,30091,30091,30091,23719,32203,17412,21638,28010,23751,17444,30058,30058,38542,23719,34316,15331,11073,17444,32235,32202,32235,23815,34379,34379,30185,23814,34347,32267,28041,40719,15331,15331,27977,30122,17445,21638,25832,32203,21670,32171,17380,19525,21639,23751,32203,30058,21606,19525,11073,21606,40622,25864,25831,34316,23718,27944,21638,21638,40655,32203,34283,44848,25864,32235,30122,23815,21669,23815,36460,32267,28008,34347,25895,23815,42831,32267,36461,30122,25928,21670,38542,19557,27977,30090,25864,28009,30122,44881,30122,32236,17413,32172,36397,4704,21638,25832,34348,23751,34348,27977,25896,32235,36460,23815,32234,34347,27976,25895,25927,36460,30153,34347,32234,25863,34380,25895,28041,36461,32203,42768,34316,23783,30090,27977,30090,32203,32203,25896,23751,30090,38574,36429,27977,25896,30090,34348,25896,25864,40622,32170,32202,32202,21670,25895,32234,38541,30153,30121,23782,28041,34380,34380,32202,19524

,34347,32234,

25832,17380,15300,19526,38511,21606,6816,17413,25832,32204,25865,27945,34284,279
77,6816,11073,30090,27978,32171,27978,25833,17381,27945,42769,17412,25864,38510,30058
,42736,27945,23752,23752,13155,8961,11074,30058,27945,25865,30058,25832,17413,27978,3
0091,21639,30058,25832,27978,19526,27945,21638,32203,30090,19525,15299,25832,23687,15
267,27978,30090,8960,11073,27944,30090,19589,23815,25895,32266,32266,25927,19588,3438
0,30121,32234,34347,15331,23815,36429,32203,23783,32203,25832,23751,17380,21639,27945
,19493,32203,32203,23751,13186,19493,23719,25832,27977,36429,23719,30058,25832,19493,
25864,19493,30057,36397,32171,21605,36461,15298,32202,21637,21637,15331,30154,40686,4
0686,27976,28008,28040,25863,28040,30154,30186,21734,25928,30154,25896,11137,32267,25
896,32203,25896,34348,32267,19557,32235,27978,27913,36397,19526,36429,21638,38510,300
90,32235,34316,32202,36428,32234,17444,28041,28008,34347,21669,40686,38573,23782,3012
1,30121,34347,28008,28008,23815,40687,36461,44880,32203,17444,25864,27977,30122,40655
,30122,23751,28009,30122,28009,32235,17412,27976,38541,36428,23783,34348,30090,23751,
15363,27976,21702,27976,28041,38606,25895,28040,23815,19589,32267,36493,25863,21670,2
1702,38573,

38510,21639,8929,17380,30091,19493,17381,19493,19526,30058,23783,27977,25896,322
03,27977,17412,30058,25832,32171,19526,21606,11074,25832,32171,27945,19493,38542,2583
2,34284,42736,32204,13187,21639,6848,19493,34317,21639,25832,27978,23752,21606,25832,
27946,19494,15300,25833,27945,32204,8960,38543,25832,23719,21670,30090,23719,32171,21
638,19493,6848,13186,17412,23751,30090,19524,28008,17443,23750,32266,23814,30121,3437
9,23815,21702,25863,21669,34315,36461,32203,32236,30122,17412,11074,8960,19493,21606,
30058,36397,40623,25832,19525,27977,32171,34316,30090,36429,21638,38542,17444,13218,3
0058,21670,27977,34316,21638,15331,36429,34348,15299,23815,11072,25928,40686,38606,40
718,19589,32234,23782,40718,21734,23782,34412,28073,28041,25960,25896,21703,30154,280
09,30122,28009,30122,17477,15363,36461,19526,30058,25864,27977,34317,21638,30058,2797
7,34315,27977,32202,28009,19589,25928,19556,28041,36460,38605,30121,34346,21701,36460
,25895,25927,21637,38605,30121,40719,34348,25863,25864,21670,25864,28009,30122,30122,
25896,19557,19525,36429,30090,27977,21638,34316,36461,36428,28041,23783,23783,17379,8
960,27976,30122,38541,25928,32234,30154,30121,23782,28008,23783,40686,25863,17444,174
43,40719,

34284,34317,19461,25865,30059,30059,23752,4736,30058,27978,21638,23719,30057,448
48,19557,30090,17413,27978,25832,27945,23720,17413,13187,25832,27977,21638,30090,3005
8,40623,42768,27945,11074,15300,4736,25865,34316,19493,19493,38575,21671,27978,25832,
23719,25865,17380,23752,32171,30091,11041,21638,19493,32203,19525,27977,36397,23752,1
3154,4704,4736,21574,23719,23718,25864,23751,17411,25895,23750,21669,23782,36459,3857
3,25928,36460,21637,11137,25928,25896,36461,32236,32203,13154,17380,21639,19526,25832
,25865,25864,36396,15299,13186,23718,36397,40655,32203,34284,32203,13186,25832,11074,
17380,21638,25864,42768,21606,23719,21638,38542,21637,17411,21670,17476,32267,40719,2
5895,21669,38572,23782,34347,32234,32267,32267,21702,32267,28042,28009,28042,28009,30
122,32235,30122,21703,17444,23751,30090,17412,27977,15299,19493,32171,25864,17380,300
57,28009,27977,34316,23783,32235,19524,32202,32202,30121,23781,30153,21701,28040,4071
8,30153,32266,23814,23814,30154,36525,30122,25928,23783,23783,30090,28009,17444,30090
,28009,15331,34316,40655,38574,34316,17444,32202,38574,40687,23783,28009,34283,30122,
23783,27976,34315,38574,32202,34315,32235,30121,34347,38606,34315,28008,19556,21670,3
2202,36493,

27945,27945,30026,32139,17381,23752,19494,19526,32171,13186,11105,17412,19525,23
783,19493,8928,30058,19526,23752,30058,17413,11074,25832,30058,27978,23719,36429,3009
0,30090,27945,23752,23719,11041,15300,36398,42736,21606,13187,40655,25865,25865,25832
,32171,25832,34284,21639,15300,23719,19493,11074,25832,25864,19493,32171,32204,30026,
17380,2592,8928,25800,32171,25832,23784,21638,19524,34347,34347,13250,34315,34379,407
18,25928,36493,25928,23782,25895,25896,25896,27977,21670,36397,30091,27945,30058,3431
7,23719,27977,30122,15266,19525,25896,25864,32171,32171,23719,23751,11073,38510,13186
,17380,27945,27945,38542,32203,32170,11105,27977,34348,23783,25896,19556,28041,38541,

15298,28040,34347,30121,30089,28041,21669,23783,30154,23751,19589,30090,28009,21670,32203,27977,28009,23751,32203,21638,25864,21606,32203,13186,13186,36397,38510,30057,19525,21670,30090,36461,30090,32235,23750,30122,32267,23814,28007,32266,21701,25894,38605,38573,36492,28008,23814,32266,34380,25895,25928,28009,30154,32235,21670,11105,27977,27977,21637,30122,34348,44881,34348,23750,17444,40654,19524,42800,34315,23751,40655,25896,27976,32202,34348,30089,23750,34315,23815,36461,32234,32267,34380,25896,28009,38541,30121,

25800,40623,44817,36365,17381,19493,38478,25865,23752,23719,23751,19525,8928,19492,25896,19557,21639,17380,19493,30059,25865,19526,17380,27945,25832,23719,34316,27977,17380,21606,17413,30091,4704,23752,27945,34317,13186,17413,25832,32236,30091,17413,42736,38543,40656,15300,21606,21639,23687,30091,17412,19461,25864,32171,23719,21606,23752,8896,15268,34252,27978,21606,17380,19492,19557,30089,25896,23782,32234,38573,34380,21669,21669,32266,25896,21669,28009,21638,23751,13186,15267,15267,32204,34317,32172,23752,30090,34316,30058,23783,34316,34284,25832,25864,25832,30058,23719,23751,11041,36397,30090,17380,38574,34316,27977,21606,19493,17379,36429,32202,25863,25928,30121,19588,25895,38541,30153,23750,28009,23783,34348,23751,19557,15364,32203,23784,21638,30123,28010,30123,36397,32203,15267,30058,17412,32171,19525,25832,38509,25864,25864,13218,32170,30089,34315,32170,25896,21670,32235,23783,25895,17443,32266,21701,19588,36524,34379,36492,23814,30121,32234,34347,32299,25928,32234,28009,28009,21670,11105,30090,32203,21670,19589,21702,42800,32235,25896,19557,32267,23750,30154,38541,11072,40687,23783,25864,32203,32235,32235,17411,25863,28009,28041,34380,25863,32267,28008,32266,28009,28041,

38543,32204,42704,21574,21574,15235,40623,25865,32171,32203,34284,32171,8960,32203,30058,25864,17380,15267,21606,21607,32172,17413,19526,34284,19493,23751,27945,34284,13187,21639,23720,25865,13155,23720,30059,25832,21639,23719,30090,38542,36430,32171,34284,34317,32171,30091,32171,34284,27946,40656,32204,19493,25832,32203,17381,21607,23720,32171,21639,30026,30091,17380,23751,6848,25896,32203,34315,30122,28041,21637,13217,21669,23815,36493,30121,25896,21670,28009,21638,25896,4736,27945,32171,32204,23720,23719,30090,25864,25864,34316,34283,25864,23783,36429,34284,30058,15299,23752,27945,30058,21606,15267,21638,36397,21638,36429,11073,34316,38510,34348,34348,21637,23783,15330,30121,40686,28008,30121,36461,23815,21670,17411,17412,27977,27977,17444,25897,34316,27945,30091,32203,27978,21638,34284,23783,21606,21638,30090,34283,25864,17379,11073,30090,34283,32235,32203,19524,23783,21637,19556,36460,17507,30121,23814,17442,28040,28007,32298,25927,32266,21701,36492,40719,28008,28073,23815,21702,21638,21638,25896,27977,28009,30090,17412,40655,28009,23783,25896,28041,32235,19589,30122,25896,15299,34316,28009,36429,32203,19557,13218,25896,25863,23750,38573,25896,30089,32267,32267,32234,28008,

36430,34316,32171,17380,13154,13122,36397,36430,27978,30091,27977,25864,21638,32203,34284,23751,25832,21606,27946,17380,32204,11074,21606,25832,21606,11041,30058,30058,30091,25865,27946,27978,21575,21607,27913,27945,40656,15300,27977,36462,34349,21671,19493,32204,21639,15267,38478,27978,32171,46930,25832,25832,21574,25800,19461,30059,17380,19494,27946,38478,25832,11074,19493,2624,27945,21670,23783,30089,23750,21702,15330,13218,25896,38606,28008,19589,25896,21670,17444,23783,21606,25832,19526,23752,23719,27977,25896,32235,23784,21638,25896,28010,32203,25864,32171,32203,21671,34316,32203,36429,27977,15299,15299,32171,27977,34316,8960,19493,38509,28009,38541,25863,21702,23750,40686,30154,32234,25863,42767,25896,25864,23751,25864,25897,25864,15300,19526,30058,27978,30058,23751,27945,23752,32171,25864,21606,21638,23719,36429,32203,17379,8928,27976,21605,32202,23750,23750,21605,19492,25896,34347,25863,25895,23782,25927,23782,25894,25927,32233,32266,25895,34379,36460,28040,30153,21637,25864,23783,23816,28009,25929,32235,34348,13186,36461,19589,21638,32235,32203,36429,15363,30154,21638,19557,34348,36429,38574,21670,15299,23750,25896,23783,34315,34348,25928,32234,30186,28041,32234,32235
};

const int control_panel[]= {

43945,43944,43944,43977,43944,43945,43944,43944,43944,43944,43945,43945,43945,43945,4
3977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,46
057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46089,46089,460
89,46089,46089,46089,46089,46089,46089,46121,46121,48169,48169,48169,46121,46121,4816
9,48169,48169,48169,48201,48201,48201,48201,48201,48201,48202,48202,48202,48202,48202
,48202,48202,48234,48234,48234,48234,48234,48234,48234,48234,50314,50314,50314,
50314,50314,50314,50314,50314,50282,48202,46121,46056,50314,50346,50379,52459,52459,5
2459,52458,50378,52459,52459,50410,50410,50410,50410,50410,52459,50378,50346,50378,50
379,50379,50379,50379,50347,52427,50347,50347,50347,50347,50347,50347,50347,50346,503
46,50346,50346,50346,50346,50346,50346,50347,50346,50314,50282,48234,48201,48169,4612
1,46121,46121,46089,46089,46089,46089,46089,46088,46121,46121,46121,46121,46121,46121
,48169,48169,48201,48201,48201,48201,48169,48169,48169,48169,48202,48202,48202,48202,
48202,48202,48202,48169,46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,4
6121,46121,48169,48169,48169,48169,48169,48169,48169,48169,48169,48169,48169,48169,48
201,46121,
43912,43912,43912,43945,43912,43944,43944,43944,43944,43945,43945,43945,43945,43945,4
3977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,46057,46
057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46089,46089,46089,46089,460
89,46089,46089,46089,46089,46089,46089,46089,46089,46121,46121,46121,46121,46121,4612
1,48169,48169,48201,48201,48201,48201,48201,48201,48202,48202,48202,48202,48202,48202
,48202,48202,48234,48234,48234,48234,48234,48234,48234,48234,50314,50314,50314,
50314,50314,50314,50314,50314,50314,48234,48201,44008,46088,46153,48266,50379,5
2459,52459,52459,50378,50378,50378,50378,50378,50378,50346,50346,52459,52459,52427,50
378,50378,50378,50379,50378,50378,50378,50378,50346,50346,50346,50346,50346,50347,503
47,50347,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50347,50315,5031
5,50314,50314,50314,50282,50282,50282,48234,48234,48201,48201,48201,48201,48169,48169
,48169,48169,46121,46121,46121,46121,46121,46089,46089,46089,46121,46121,46121,46121,
46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,4
6121,46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,46121,48
169,46121,
41864,43912,41864,43945,43912,43944,43944,43944,43944,43944,43944,43944,43944,43944,4
3977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,46057,46
057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46089,46089,46089,46089,460
89,46089,46089,46089,46089,46089,46089,46121,46121,46121,48169,48169,46121,46121,4612
1,48169,48169,48201,48201,48201,48201,48201,48201,48202,48202,48202,48202,48202,48202
,48202,48202,48234,48234,48234,48234,48234,48234,48234,48234,50314,50314,50314,
50314,50314,50314,50314,50314,50315,50315,50315,50314,50379,50314,46121,44040,44008,4
6121,48234,50347,50379,50379,52427,52427,52427,52427,52427,52427,50378,50346,50346,50
346,50346,50346,50346,50346,50346,50346,50378,50378,50378,50378,50378,50378,50346,503
46,50346,50346,50346,50347,50347,50347,50346,50346,50346,50346,50346,50346,50346,5034
6,50346,50346,50346,50314,50314,50314,50314,48266,50314,50314,50314,50314,50314,50314
,50314,50314,50282,50282,48234,48234,48234,50282,48234,48234,48234,48234,48234,48234,
48234,48234,48234,48234,48201,48202,48202,48202,48202,48201,48201,48201,48169,48169,4
8169,48169,48169,48169,48169,48169,48169,48169,48169,48169,48169,48169,48169,48169,48
169,46089,
43912,43912,43912,43945,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,4
3977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,46057,46
057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46089,46089,46089,460
89,46089,46089,46089,46089,46089,46089,46121,46121,46121,48169,48169,48169,46121,46121,4816
9,48169,48169,48169,48201,48201,48201,48201,48201,48202,48202,48202,48202,48202,48202
,48202,48234,48234,48234,48234,48234,48234,48234,48234,48234,50282,50314,50314,50314,
50314,50314,50314,50314,50314,50314,50346,50346,50346,48266,50314,50347,50347,50315,4
8234,46121,44040,46121,48169,48201,48201,48234,50315,52395,52427,52427,52427,52459,52
459,52459,52427,50378,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50346,50346,50314,50314,50346,50346,50314,50314,50314,50314,50314,50314,50314,50314
,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,50282,50282,50282,50282,
50282,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,4

8234,48234,48234,48234,48234,48234,48202,48202,48202,48202,48202,48201,48201,48201,48
201,46121,
43944,43912,43944,43945,43944,43912,43944,43944,43944,43944,43944,43944,43945,43945,4
3944,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,46057,46
057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46089,46089,46089,46089,460
89,46089,46089,46089,46089,46089,46089,46089,46089,46121,46121,46121,48169,48169,48169,4816
9,48169,48169,48169,48169,48169,48201,48201,48201,48202,48202,48202,48202,48202,48202
,48202,48234,48234,48234,48234,48234,48234,48234,48234,48234,50282,50314,50314,50314,
50314,50314,50314,50314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,5
0346,50347,50347,48201,48201,46121,46121,46121,46121,46121,46121,48234,48234,48266,50
314,50347,50347,50379,50379,50379,50379,50379,50379,50379,50378,50378,50378,50378,503
78,50378,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50346,50346,50346,50346,50346,50346,50314,50314,50314,50314,50314,50314,50314,50314
,50314,50314,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,
48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,4
8234,48234,48202,48202,48202,48202,48202,48202,48202,48202,48202,48202,48201,48201,48
202,48201,
43944,43912,43912,41864,43944,43912,43912,43912,43944,43944,43944,43944,43944,43944,4
3977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,46057,46057,46057,46
057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46089,46089,46089,46089,460
89,46089,46089,46089,46121,46121,46121,48169,48169,48169,48169,48169,48169,48169,4816
9,48169,48169,48169,48169,48169,48201,48201,48202,48202,48202,48202,48202,48202,48202
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,50282,50314,50314,50314,
50314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,5
0346,50346,50346,52427,52427,50379,50346,48266,48233,46153,46121,46088,46088,46056,46
088,46089,46121,48169,48201,48234,48234,48234,50282,50314,50315,50347,50347,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50346,50346,50346,50314,50314,50314,50314,48266,50346,50346,50314,50314,50314,50314
,50314,50314,48266,48266,48266,48266,50314,50314,50314,50314,50282,50282,50282,50282,
50282,50282,50282,50282,48234,48234,48234,48234,48234,48234,48234,48202,48234,48202,4
8202,48202,48202,48202,48201,48201,48202,48202,48202,48202,48202,48202,48201,48201,48
201,48201,
43944,43912,43945,43912,43945,43944,43944,43944,43944,43976,43976,43976,43976,43976,4
3977,43977,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46
057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46089,46089,46089,46089,460
89,46089,46089,46089,46089,46089,46089,46121,46121,48169,48169,48169,46121,46121,4612
1,48169,48169,48201,48201,48201,48201,48201,48202,48202,48202,48202,48202,48202,48202
,48234,48234,48234,48234,48234,48234,48234,48234,48234,50282,50282,50314,50314,50314,
50314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,5
0346,50346,50378,50346,50346,50379,50379,52427,52427,50379,50379,50347,50314,48234,48
201,46121,46089,46088,46056,46088,46088,46088,46088,46089,46121,46121,48169,48201,482
01,48202,48234,48234,50282,50314,50314,50315,50315,50315,50314,50314,50314,50314,5031
4,50346,50346,50346,50346,50346,50346,50346,50346,50314,50314,50314,50314,50314,50314
,48234,48234,48266,48266,48266,48234,48234,48234,48234,48234,48234,48234,48234,48234,
48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,4
8234,48202,48202,48202,48202,48201,48202,48202,48202,48202,48202,48202,48202,48202,48
201,48169,
43944,43912,43945,43945,46025,46025,46057,46057,46057,46057,46057,46057,46089,46089,4
6057,46057,46057,46057,46057,46089,46089,46089,46057,46057,46057,46057,46057,46089,46
089,46089,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,460
57,46057,46057,46089,46057,46057,46057,46089,46089,46089,46089,46089,46121,46121,4612
1,46121,48201,48201,48201,48202,48201,48201,48202,48202,48202,48202,48202,48202,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,50282,50282,50314,50314,50314,
50314,50314,50314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,5
0346,50346,50378,50378,50378,50378,50378,50378,50378,50378,52427,52427,52427,52
459,52459,52427,50347,50314,48234,48202,48170,46121,46089,46089,46057,46057,46057,460
57,46057,46057,46089,46089,46089,46089,46089,46089,46089,46121,46121,48169,48201,4820
1,48201,48201,48201,48201,48234,48234,48234,48234,48234,48234,50314,50314,50314,50314

,50314,50314,48266,48266,48266,48266,48266,50282,48234,48234,50282,50282,50282,48234,
48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,4
8202,48201,48201,48201,48201,48201,48201,48201,48201,48201,48201,48201,48201,48201,48
201,46121,
43977,46057,46057,46089,46089,46088,44009,46057,46057,46057,46089,46089,46089,46089,4
6121,46121,46121,48169,48201,48201,48202,48202,48234,48234,48234,48234,50314,50314,50
314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50314,48266,48266,48266,482
65,48233,48233,48233,48201,46153,46153,46121,46121,46089,46088,44040,46057,46057,4605
7,46057,46089,46089,46089,46089,46121,46121,48170,48169,48170,48202,48202,48202,50282
,48234,48234,48234,48234,48234,48234,48234,48234,48265,48265,48265,48265,48265,48265,
50314,50347,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,5
0346,50346,50346,50379,50379,50379,50379,50378,50378,50378,50378,50378,50378,52427,52
427,52459,52459,52459,52459,52459,52459,52459,50411,50378,50378,50378,50378,50282,482
34,48234,48202,48201,48169,46121,46121,46121,46121,46089,46089,46089,46089,46089,4608
9,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089
,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,
46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,4
6121,46121,46121,46121,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46
089,46089,
46057,46057,46057,46057,46057,46056,46121,46121,46122,46121,48169,48202,48202,48201,4
8202,48202,48202,48202,48234,48234,48234,48234,48234,48234,48234,48234,50314,50314,50
314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,503
46,50346,50346,50346,52427,50379,50379,50379,50379,50379,50379,50379,50314,50314,4823
4,48201,48169,46121,46088,46088,46056,46056,46056,46056,46088,46089,46089,46089,46121
,46121,48202,48234,50315,50314,50314,50314,48234,48234,50314,50314,50314,50314,50314,
50314,48234,48234,50314,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5
0347,50347,50347,50346,50346,50378,50379,50378,50378,50378,50378,50378,50378,52427,52
427,52459,52459,52459,52459,52459,52459,52459,52459,52459,50411,50411,50411,52459,524
59,52459,52459,52459,52459,52427,52427,50379,50379,50379,50379,50347,50347,50347,5034
7,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314
,50314,50314,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,
48234,48234,48234,48234,50314,50314,50314,50314,50314,50314,50314,50347,50379,5
0379,50379,50379,50379,50379,50347,50346,50346,50314,50314,50314,50314,50314,50314,50
314,50314,
46089,46089,46089,46089,46121,48169,48169,48202,48202,48201,48201,48202,48202,48202,4
8202,48202,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,50314,50314,50
314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50378,503
79,50379,50379,50379,50378,50378,52427,52427,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,50346,50314,50282,48234,48201,46121,46088,46056,46089
,46056,44009,44009,46057,46089,46089,46089,48234,48234,50282,50314,50314,50314,50314,
50314,50346,50346,50314,50314,50314,50314,50346,50346,50314,50314,50346,50346,50346,5
0347,50347,50347,50378,50378,50378,50378,50379,50379,50379,50379,50378,50378,52427,52
427,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,
46089,46089,46121,48169,48202,48202,46121,46121,48169,48169,48201,48201,48201,48201,4
8202,48202,48202,48202,48202,48202,48202,48202,48234,48234,48234,48234,50314,50314,50
314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,503
46,50346,50346,50346,50378,50378,50378,50378,50378,50378,50378,52426,50378,50378,5041
0,50410,50410,50410,50378,50378,52459,52459,52459,52459,52427,50379,50347,50347,48266
,48234,48201,46121,46089,46089,46089,44008,44008,44008,46089,46089,48202,48234,
48234,50315,50315,50314,50314,50314,50346,50346,50378,50346,50346,50346,50346,50346,5
0346,50346,50346,50378,50378,50378,50378,50379,50379,50379,50379,50378,52427,52427,52
427,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50410,504

10,50410,50410,50410,50410,50410,50410,50410,50410,50410,50410,50410,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50410,50410,5
0410,52459,52459,50410,50410,50410,52459,52459,52459,52459,52459,50410,50410,50410,52
459,52459,
46121,46121,48169,48169,48169,48169,48169,48169,48169,48201,48201,48201,48201,48201,4
8202,48202,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,50314,50314,50
314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,503
46,50346,50346,50346,50379,50379,50379,52427,52427,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,50411,50378,50378,50378,50411,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52427,50347,50314,48266,48169,46121,46089,46056,44008,44008,44008,
44008,46057,46089,48169,48234,48267,50315,50347,50347,50378,50378,50378,50378,50378,5
0378,50378,50378,50346,50346,50346,50378,50379,50379,52427,52459,52427,52427,52427,52
459,52459,52459,52459,52459,52459,52459,52459,52459,50411,50411,50411,50411,52491,524
91,52491,52491,52491,50411,50411,50411,50411,50411,50411,50411,50411,50411,52491,5249
1,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,50411,50411,50411,50411,50411,50411,50411,50411,50411,50411,50411,50411,
50411,50411,50411,50411,50411,50411,50411,50411,52459,52459,52459,52459,50411,50411,5
0411,50411,50411,50411,50411,50411,50411,50411,50411,50411,50411,50411,50411,50411,52
459,52459,
46121,48169,48169,48169,46121,46121,48169,48201,48201,48201,48201,48201,48201,48201,4
8202,48202,48202,48234,48234,48234,48234,48234,48234,48234,48234,48234,50314,50314,50
314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,503
46,50378,50378,50379,50346,50378,50378,50379,50379,50379,52427,52427,50379,50379,5037
9,50379,52459,52459,52459,52459,52459,52459,52459,50411,50411,50378,50378,50378,52459
,52459,52459,52459,52459,52459,52459,52459,50411,52460,52460,52427,50347,48266,48201,46121,
46088,46089,46056,44008,44008,46057,46089,46089,46121,48202,48233,48234,50314,50314,5
0346,52426,52426,52427,52427,50379,50379,50378,50378,52427,52427,52427,52427,52427,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50379,503
79,50379,50379,52459,52459,52459,52459,50379,50379,50379,50379,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50379,50379,5
0379,50379,50379,50379,50379,50379,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,
46089,46121,48169,48169,48169,48169,48169,48169,48169,48201,48201,48201,48201,48201,4
8202,48202,48202,48202,48202,48234,48234,48234,48234,48234,48234,48234,50314,50314,50
314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,503
46,50378,50378,50378,50378,50378,50378,50378,52427,52427,52459,52459,52491,52491,5245
9,52459,52459,52459,50411,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,50411,50378,50378,50410,52459,52491,52427,52427,52459,52459,52459,52459,52427,
50379,50314,48266,48201,46089,46056,46056,46057,46057,46057,46057,46089,46121,48201,4
8202,48234,50282,50346,50347,50347,50379,50378,50378,50378,50378,52427,52427,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,50379,50379,50379,50379,50379,50379,50379,50411,50411,50411,50411,50411,
50411,50411,50411,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,50379,50379,50379,50379,50379,50379,50379,52
459,50411,
46089,46089,46121,46121,46121,48169,48169,48169,48169,48202,48201,48201,48201,48201,4
8202,48202,48202,48202,48202,48234,48234,48234,48234,48234,48234,48234,50314,50314,50
314,50314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,503
46,50346,50378,50378,50378,50378,50378,50378,50378,50378,50378,50378,50410,50410,5037
8,50410,50410,52459,52459,52459,52459,52459,52459,52427,52426,52426,52459,52459,50379
,52459,52459,52459,52459,52459,52459,52459,52458,52458,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52427,52395,50282,48170,48137,46057,46057,46057,46057,46057,4

6057,46057,46057,46121,46121,48233,50314,50346,50378,52458,52458,52427,52427,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50411,50411,50411,52459,524
59,52459,52459,52459,52459,52426,52426,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52426,
46121,46121,46121,46121,46121,46089,46121,46121,48169,48201,48201,48201,48202,48202,4
8202,48202,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48
234,48234,50282,50314,50314,50314,50314,50314,50314,50346,50346,50346,50346,503
46,50346,50346,50346,50378,50378,50378,50410,50410,50410,50410,50410,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,50379,50346,48234,46121,46089,4
4009,46057,46057,46057,46057,46057,43977,46057,46089,48170,48202,50314,50347,52459,52
459,50411,50411,50410,50442,50411,50411,50411,50411,50411,50411,50411,50411,50411,504
11,50411,52459,52459,52459,52427,50378,52427,52459,52459,52459,52459,52459,52459,5242
6,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
458,52459,
46089,46089,46121,48169,48169,48169,46121,48169,48169,48169,48169,48201,48201,48201,4
8201,48201,48201,48202,48202,48202,48202,48202,48234,48234,48234,48234,48234,48234,48
234,48234,50314,50314,50314,50314,50346,50346,50346,50346,50314,50314,50346,50346,503
47,50347,50379,52427,50378,50378,50379,50378,50378,52427,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,50378,52427,52459,52459,50379,5
0315,48234,48202,46089,46057,46057,46057,46057,46057,46089,46089,46056,46089,48169,48
234,48266,50346,50379,52459,52459,52459,52459,50411,50410,52459,52459,52459,50411,504
10,50410,50410,52459,52459,52459,52491,52427,52427,52427,52427,52459,52459,52459,5245
9,50411,50411,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
491,52459,
46089,46089,46121,46121,46121,46121,48201,48201,48201,48201,48201,48201,48201,48201,4
8201,48201,48202,48202,48234,48234,48234,48234,48234,48234,48234,48234,50314,50314,50
314,50314,48234,48265,48266,48266,48266,50314,50314,50346,50347,50347,50347,50346,503
46,50346,50346,50346,50379,50379,50379,50378,50378,50379,50379,52427,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52491,52459,52459,5
2459,52459,50378,52427,50347,50314,48234,48169,46088,44008,43976,46057,46057,44008,44
008,46057,46057,46089,46121,48266,50314,50379,52459,52459,52459,50379,50346,52458,524
58,52458,50410,50410,50411,50379,50379,52459,52459,52459,52459,50379,50379,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
458,50378,
46089,46089,46121,46121,46121,46121,46121,46121,46121,46121,46121,48169,48201,48201,4
8201,48201,48202,48202,48202,48234,48234,48234,48234,48234,48234,48234,48234,48234,50
314,50314,50314,50314,50314,50346,50346,50346,50346,50346,50314,50346,50346,50346,503
46,50346,50378,50379,50378,52427,52427,52427,50379,52427,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459

,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50410,50378,50378,50378,5
0378,52459,52459,52459,52459,52459,52427,50379,50346,50314,48266,46121,46089,46056,44
008,46057,46057,46057,46057,46057,44008,44008,46057,48169,50314,52427,54540,52426,524
59,52459,52459,52459,52459,50411,50378,50378,50378,50378,50411,52459,52459,52459,5245
9,52459,52459,52459,52459,50411,50411,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
458,50411,
46089,46089,48169,48169,48169,48170,48169,48169,48169,48202,48202,48202,48202,48202,4
8202,48202,48202,48202,48234,48234,48234,48234,48234,48234,50282,50282,50314,50314,50
314,50314,48266,48266,48266,48266,50314,50314,50346,50346,50346,50346,50346,50346,503
46,50346,50378,50378,50346,50346,50378,50378,50378,50378,50378,52427,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,50379,52459,52459,50411,50411,5
0411,50411,50411,50378,50410,50410,52459,52459,52459,52459,52459,52492,52460,50347,48
266,48234,46121,46089,44008,43977,43977,46057,46057,46057,44008,43976,43976,48202,482
34,50314,50347,50379,52459,52459,52459,52459,52459,52459,52459,50410,50378,50378,5037
8,50378,50411,50411,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,
46089,46089,46121,46089,46089,46089,48170,46121,46121,46121,46121,46121,46089,46089,4
6089,46121,46121,46121,48170,48170,48170,48170,48169,48169,48169,48170,48170,48169,48
169,48169,48234,48234,48234,48234,50282,50314,50314,50314,50314,50346,50346,50346,503
46,50346,50346,50346,50346,52426,52426,52427,52426,52426,52426,52426,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
0411,50411,50411,52491,52491,52459,52459,50411,50378,50378,50379,50411,50410,50411,50
411,52459,52459,50379,50379,48202,48202,46121,46089,46057,46057,46057,46089,43977,439
77,44009,46089,46121,48201,50314,50314,52427,52427,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50
378,52459,
46057,46057,46057,46057,43977,43977,43977,43976,43976,43976,43976,43976,43976,43976,4
3976,43976,43976,43976,43976,43976,43976,43976,43976,43976,43976,43976,43976,43976,43
976,43976,43976,43976,43976,43976,44008,44008,46057,46056,46056,46089,46121,46121,482
02,48234,50282,50315,50347,50347,50347,52427,52427,52427,52427,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50378,50378,50378,5
0411,50411,52491,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,50411,50378,52427,52459,52427,52427,50347,48266,48201,46121,46089,460
89,46056,44008,44008,46056,46056,46089,46089,46121,48202,48234,50315,52427,52459,5245
9,50378,50411,52459,52459,52459,52459,50411,50411,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
426,52491,
43977,43977,46057,43977,43977,43977,43977,43977,43977,43977,43977,46057,46057,46057,4
6057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46
057,46057,46057,46057,46057,46057,46057,46057,46057,43977,43977,43977,43976,439

76,43976,43976,43976,43976,44009,46057,46121,48201,48234,50314,50347,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52458,52458,52458,52458,52459,5
2459,52459,52459,52427,52427,52459,52459,52459,52459,52427,52427,52427,52427,52427,52
427,52427,52459,52459,52459,52458,52458,52458,52459,52459,52459,52459,52459,52427,503
46,50314,48234,48170,46089,46057,46057,43977,43977,43977,43977,44009,46056,46088,4608
9,48233,48233,48265,50346,50378,50411,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50
378,50378,
46057,46025,44009,44009,46057,46057,46057,46057,46057,46057,46057,46057,46089,46089,4
6089,46089,46089,46089,46090,46090,46090,46090,46121,46121,46121,48170,48170,48170,48
170,48170,48201,48201,48201,48201,48201,48169,48169,48169,48169,48169,48169,46121,460
89,46089,46089,46089,46057,46057,43976,43976,43944,43976,43977,43977,46089,46089,4820
2,50282,50347,52427,52459,52459,52458,52458,52458,52458,52459,52459,52459,52459,50411
,50411,52491,52459,52459,52459,52427,52427,52459,52459,52459,52459,52459,52426,52426,
52426,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,50314,50314,48233,48201,46121,46057,44008,4397
6,43976,43976,43976,43976,43977,46057,46090,46090,48169,48201,48201,50282,50314,50346
,52426,52427,50378,52458,52458,52458,52458,52459,52459,52491,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52426,52426,52427,50379,50346,48266,48234,46153,46
057,46057,
46057,46057,46057,44009,46089,46057,46057,46057,46057,46057,46057,46057,46057,46089,4
6089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46
121,46121,46121,46121,46121,46121,46121,48169,48169,48169,48169,48170,48202,48202,48202,482
02,48202,48202,48202,48202,48202,48202,48202,48170,46121,46089,46089,44008,43976,4397
6,43976,43976,46056,46121,48169,50347,50347,52427,52459,52491,52459,52459,50378,50410
,50411,50411,50411,50411,52459,52459,52491,52459,50411,50411,50378,50378,52459,52491,
52492,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50379,50379,50347,5034
7,48234,48234,48201,46121,46089,46056,44008,43976,43977,43977,43977,43976,43976,43976
,43976,43976,46057,46057,46057,46089,46089,46089,46121,46121,48201,48201,48202,48234,
48234,50282,50314,50314,50314,50314,50314,50314,50314,50282,50282,50282,48234,48234,4
8234,48202,48202,48201,48169,48169,48170,46121,46089,46057,46057,43977,43977,43977,46
057,46121,
43976,46024,44008,43976,46057,43976,46057,46057,46057,46057,46057,46089,46089,46089,4
6089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46121,46121,48
169,48169,48169,48169,48169,48169,48169,48201,48201,48201,48201,48201,48202,48202,482
02,48202,48234,48234,48202,48202,48234,48234,48234,48234,48202,48202,48235,48202,4817
0,46089,46089,44009,43976,43976,43977,44009,44041,46121,48234,50347,50379,52459,52459
,52459,52459,52459,52459,50410,50410,50410,52459,52459,52459,52459,52459,50379,50379,
50379,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52427,50379,50379,48235,48235,48202,48202,48170,46090
,46089,46089,46057,46057,46057,46057,46057,43977,43977,43977,43976,43976,43976,43976,
43976,43976,43976,43976,46057,46057,46057,46057,46057,46057,46057,43977,46057,46057,4
6057,46057,46057,46057,46057,46057,46057,43977,43977,43977,43977,43977,46057,46090,46090,46
057,46121,
46057,46025,46057,46057,46089,46057,46057,46057,46057,46057,46057,46057,46057,46057,4

6057,46057,46057,46089,46089,46089,46089,46089,46089,46089,46089,46121,46121,48169,48
169,48169,48169,48169,48169,48169,48169,48169,48169,48169,48201,48201,48201,48201,482
01,48201,48201,48201,48234,48234,48234,48234,48234,48234,48266,48266,48234,48234,5031
4,50315,50315,48234,48202,48202,46090,46057,43976,43944,43976,44008,46089,46121,50314
,50315,50379,52459,52459,52459,52459,50410,50378,50378,52459,52459,50411,50411,50411,
50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,50410,50410,50410,50410,50410,50410,50410,5041
0,50410,50410,52459,52459,52459,52459,50411,50379,52459,52459,52459,52459,52459,52427
,50379,50379,50347,50347,50346,50314,50314,50314,48234,48233,48202,48202,48170,48169,
46121,46089,46089,46089,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,4
6057,46089,46089,46089,46089,46121,48201,48201,48233,48233,50314,50346,50346,50346,52
460,50346,
46025,43976,43977,43977,46057,46057,46057,46057,46057,46057,46057,46058,46058,46058,4
6057,46089,46089,46089,46089,46089,46090,46090,46090,46090,46090,46090,48170,48170,48
170,48170,48169,48169,48170,48170,48201,48202,48202,48202,48201,48202,48234,48234,482
02,48202,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48233,48233,4823
3,48233,48234,48234,50314,50314,50314,50314,48234,48201,46121,46057,43976,43976,43976
,44008,46089,48201,48234,50347,52459,52460,52459,52459,52459,50410,50378,50410,50410,
50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,50411,50411,50411,50411,50411,5041
1,52459,52459,52459,52459,52459,52459,52459,52459,52458,52458,52458,52458,52458,52458
,52458,52458,52491,52491,52458,52458,52458,52458,52458,52458,52459,52459,52459,52459,
52459,52459,52427,52427,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,50378,50410,52459,52458,52491,52458,52458,52458,52
427,52458,
46025,43976,44009,43977,43977,43976,43944,43977,43977,43977,43977,43977,43977,43977,4
3976,43976,43976,43976,43976,43977,43977,43977,43977,43977,43977,43977,43977,43977,46
057,46057,46056,46056,46057,46089,46089,46089,48170,48170,46121,46121,48169,48202,482
02,48202,48202,48234,48202,48234,48234,48234,48234,48234,48234,48202,50314,50314,5031
4,48233,48233,48233,48233,48233,48265,50313,50314,50314,50314,50282,48202,48202,44009
,43976,43976,43976,43976,46089,48169,48202,50346,52427,52459,52459,52491,52459,50411,
50410,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,50379,50379,50379,50379,50379,52459,52459,5245
9,50379,50379,50379,50379,50379,52459,52459,52459,52459,52459,52459,52459,52459,50379
,50379,50379,50411,50411,50411,50411,50411,50411,50411,52491,52458,52458,52458,52458,
52458,52458,52458,52458,52458,52458,52458,52458,52458,50410,50410,50410,50410,52458,52458,5
2458,52458,52458,52458,52458,52458,52458,52458,50411,50411,50411,50411,50411,50411,50411,50411,50
379,50410,
43976,43976,44008,41928,43976,43976,44008,44008,44008,44008,44008,44008,44008,44008,4
4008,44008,44008,44008,44008,44008,44008,44008,43976,43976,43976,43976,43976,43976,43
976,43976,43976,43976,43976,43976,43944,43976,43977,43977,43977,46057,46057,46057,460
57,46089,46089,46090,48170,48170,48202,48202,50282,50282,50282,50282,48201,48201,4823
4,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314
,48234,48201,46089,44008,43976,43944,43976,44008,46089,48202,50314,50347,52459,52459,
52491,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52491,52491,52491,52459,52459,52459,50411,50411,52459,52459,52459,52459,52459,50379
,50379,50379,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
50410,50410,50410,50410,52459,52459,52459,52459,52459,52459,52459,52459,50410,50410,5
0410,52459,52459,52459,52459,52459,50379,50379,50379,50379,52459,52459,52459,52459,52

491,48298,
50314,50315,50379,50347,50347,50347,50347,50347,50347,50347,50347,50347,50347,50347,5
0347,50347,50347,50347,50347,50347,50347,50347,50315,50314,50314,50314,50282,50282,48
234,48234,48202,48170,48170,46090,46057,46057,43977,43977,43944,43944,43977,43977,439
44,43944,43977,43977,43977,43976,43976,43976,46056,46089,46089,48169,48234,48234,4823
4,48234,48234,48234,48234,48234,48266,48266,48266,50314,50314,50314,50314,50314,50346
,50346,50347,50347,50282,48170,46089,46057,43945,43945,43944,43976,44008,46121,48266,
50346,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52458,52458,52458,52458,52459,52426,52426,5242
6,50378,50378,50410,52459,52459,52459,52459,50410,50411,50410,50410,52459,52458,52458
,52458,52458,52426,52426,52426,52459,52459,52459,52459,52459,52427,52427,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52427,52427,52458,52458,52458,52458,52458,52458,52458,50378,50
378,52491,
52426,50379,50411,52426,52428,50378,50378,50378,50378,50378,50378,50346,50346,50346,5
0347,50346,50346,50346,50346,50346,50346,50346,50347,50347,50347,50347,50314,50314,50
314,48266,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234,48202,48169,460
89,46089,46057,44009,43977,43977,43977,43977,43977,43977,43977,43977,43945,43977,4605
7,46057,46090,48170,48234,50315,50282,50314,50314,50314,50314,48265,48265,48265,50378
,50346,50314,50314,50346,50346,50346,50378,48202,48169,46056,43976,43944,43945,43977,
43945,46057,48170,50283,52427,52459,52459,52459,52459,52490,52458,50410,50411,50411,5
0411,50411,50411,50411,50411,50411,50411,52459,52459,52458,52459,52427,52427,52459,52
459,52459,52459,52459,52426,50378,50411,50411,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
490,50378,
52425,50379,50410,50346,52427,50378,50379,50379,50379,50379,50347,50347,50347,50347,5
0346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50
346,50346,50314,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234,48234,482
34,48234,48234,48234,48170,48170,48170,46090,46089,46057,43977,43977,43977,43977,4397
7,43944,43944,43976,43976,44008,46089,48169,48234,50282,50314,50314,50314,50347,48233
,48266,50346,50346,50346,50346,50346,50346,52427,52427,50347,50314,48234,46121,43976,
41896,43944,43944,43976,44008,46121,48202,50314,50347,50379,52427,52459,52459,52459,5
0411,50411,50410,50411,50411,50411,50411,52459,50411,50378,50378,52459,52459,52459,50
379,50379,52459,52459,52459,50411,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
458,50379,
52425,50411,50411,50346,52427,50410,50378,50378,50378,50378,50378,50378,50378,50378,5
0346,50346,50346,50346,50346,50346,50346,50346,50314,50314,50314,50314,48265,48265,48
265,48265,50314,50314,50314,48233,48233,48233,48233,48233,48234,48234,48234,48234,482
34,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48170,46089,4608
9,46057,43976,43976,43976,43944,43944,43944,43977,44009,46089,48202,50315,50348,50347
,50347,50346,50314,50314,50346,50346,50314,50346,50346,50346,52427,52427,50346,50314,
50314,46121,46089,43976,43944,43944,43976,44008,46057,48202,48234,50347,52427,52459,5
2459,52458,52458,50410,50410,50410,50410,52459,52459,52459,52491,50379,50379,52459,52
459,52459,52459,50379,50379,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,

52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50
410,52459,
52394,52428,50411,52395,52395,50379,50378,50346,50346,50346,50346,50346,50346,50346,5
0346,50346,50346,50346,50346,50346,50346,50346,50315,50315,50315,50315,50314,50314,50
314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,50282,50282,48234,482
34,48233,48233,48233,48201,48201,48201,48202,48202,48201,48201,48201,48202,48201,4820
1,48202,48202,48169,46089,46089,46057,46057,43977,43944,43944,43976,44009,46089,50283
,50315,50346,50346,50346,50346,50346,50347,50346,50346,50346,50346,50346,50378,52427,
52427,52459,50347,48266,48202,46121,46057,43976,43944,43944,43944,44008,46089,48202,5
0314,50347,52427,52459,52459,52459,52459,52459,50411,50378,50410,50411,50411,50411,52
491,52491,50411,50411,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50
410,52459,
43943,43945,43976,43944,43944,43976,43977,43977,43977,43977,43977,43977,43977,43977,4
6057,46057,46057,46057,46057,46057,46057,46057,44009,46057,46057,46089,46089,46089,46
089,46089,46089,46089,46121,46122,48170,48170,48170,48170,48202,48202,48202,48202,482
34,48234,48234,48234,48234,48202,48202,48202,48201,48201,48169,48169,48201,48169,4816
9,48169,48202,48202,48170,48170,48170,48170,48170,46121,46089,46057,43976,43944,43976
,46089,48234,50346,50346,48266,50314,50379,50346,50346,50378,50378,50378,50346,50346,
50346,50378,50378,50378,52427,52427,50347,50314,48202,46089,46057,43976,41895,41864,4
3944,46057,46057,48202,50315,50347,52427,52459,52491,52459,52458,52491,52491,50410,50
378,50378,50378,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50
411,50411,
43911,43913,43944,45992,45993,43976,43944,43944,43944,43944,43944,43944,43945,43945,4
3976,43976,43976,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43
944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43976,43976,43976,43977,439
77,46057,46057,46057,46121,46121,48169,48202,48202,48202,48202,48234,48201,48201,4816
9,46121,46121,48169,48169,48169,46121,46121,48169,48169,48169,48169,48169,46089,43976
,43976,46056,48201,50314,50347,50346,50346,50314,50346,50346,50347,50347,50379,50379,
50379,50346,50378,50378,50378,52459,52459,52459,52427,52427,52395,50315,48202,46089,4
3977,43944,41864,43944,43976,44008,46089,48201,48266,50378,52427,50378,50411,52459,52
459,52459,52459,52459,50410,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
491,50378,
48233,48235,48234,50314,50314,48266,50346,50346,50346,50346,50346,50346,50346,50346,5
0315,50315,50315,50314,50314,50314,50314,50314,50282,50282,48234,48234,48202,48202,48
202,48202,46121,46121,46089,46089,46089,46057,44009,44008,43976,43976,43944,43944,439
44,43944,43944,43944,43944,43944,43944,43976,46057,46089,46089,46121,48201,48201,4820
1,48201,48202,48169,48202,48202,48202,48169,46121,46089,46089,46089,46089,46089,48170
,46089,43944,43976,46089,50315,50347,50314,50379,50378,50346,50346,50346,50346,50347,
50379,50378,50378,50378,50378,50346,50346,50378,50378,50378,50411,52459,52460,52459,5
0379,50315,48234,46121,46056,43976,41895,41895,43944,43976,46057,48234,48266,50347,52
427,52460,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245

9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50
411,50411,
52458,52459,50378,52458,52427,50410,50379,50379,50379,50378,50346,50346,50346,50346,5
0378,50378,50346,50346,50346,50346,50346,48298,50314,50314,50314,50314,50314,50314,50
314,50314,50314,50314,50314,50314,50282,50282,48234,48234,48234,48202,48202,48202,481
70,46090,46089,46089,43977,43977,43977,43944,43944,43944,43976,43976,46056,46089,4816
9,48202,48202,48169,46121,46121,46121,46121,46121,48169,48169,48170,48137,46089,46056
,48169,46090,43944,43944,46121,50346,50346,50346,50346,50346,50346,50346,50346,50346,
50346,50346,50346,50379,50378,50378,50378,50410,52459,50410,50410,50410,50410,52458,5
2458,52459,52459,52459,52427,50347,48234,46122,46058,43977,43945,41896,43944,43976,43
976,46089,48202,50314,50347,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50
410,52491,
46057,46057,46057,46090,46057,46089,46089,46121,46121,48169,48169,48201,48201,48201,4
8202,48202,48234,48234,48234,48234,48266,50314,50282,50315,50315,50314,48234,48234,50
314,50314,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,482
34,48234,48202,48202,48233,48201,48169,46089,46089,43977,43944,43944,43945,43944,4605
7,48170,48169,48202,48202,46121,48169,48169,48169,46121,46121,46089,46089,46089,46057
,48169,46089,46057,43944,43944,48234,50346,50346,50346,50346,50346,50346,50346,50378,
50378,50378,50378,50379,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,50378,50378,50346,46121,46089,44008,43
976,43944,43944,43944,43945,46025,46089,48202,50314,52427,52459,52458,52458,50411,504
11,50411,50411,50410,50410,50410,50410,52460,52459,50379,50379,50379,50379,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52427,52458,52458,52458,52426,52426,52427
,52427,52427,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,50411,50411,52459,52459,52459,52458,52458,52458,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
490,52491,
43944,41896,43912,41864,41831,41863,41864,41864,41864,41864,41864,41864,41863,41863,4
1864,41864,41864,41864,41896,41896,43944,43944,43976,44008,46089,48202,50314,50314,50
314,48233,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,482
02,48202,48202,48202,48201,48202,48202,48202,48202,48202,46121,46089,43977,43944,4397
7,46121,46121,46121,48169,48169,48169,48169,46121,46121,46121,46089,46089,46089,46089
,46089,46089,46057,43944,43944,46121,50347,50346,50346,50346,50346,50346,50346,50378,
50378,50378,50378,50379,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,52459,50411,50411,52459,52459,52459,50411,50378,50378,52491,52459,50379,50
314,48234,46121,46089,44008,43977,43944,43944,41896,43944,44009,46089,46122,48266,503
14,50347,52427,52460,52460,52459,52459,50410,52458,52458,52490,52490,52490,52458,5245
8,50410,50410,50410,50410,50410,50410,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50379,50379,50379,50379,
50379,50379,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,50410,50410,52458,52458,52458,52459,50410,50378,50
314,48234,
44008,44041,46089,46090,46056,46153,46122,46122,46122,46122,46122,46122,46089,46089,4
6121,46089,46089,46089,46057,44009,43977,43944,43945,41864,43944,46088,48234,50314,48
266,48233,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,482
34,48202,48202,48202,48202,48202,48202,48202,48202,48201,48201,48201,46057,43944,4397
6,48170,48169,46121,46121,48169,48169,46121,46121,46121,46089,46089,46089,46089,48169
,44008,46121,46057,43977,43944,44008,50347,50346,50346,50346,50346,50346,50346,50378,
50378,50378,50378,50379,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,52459,50411,50411,50411,50411,50411,52459,52459,52459,52459,52459,52459,52

491,52491,52459,50379,50379,50314,48234,48170,46089,43977,43944,43945,41896,43944,439
44,43944,43977,46057,46090,48202,48203,50346,50378,52426,52426,52459,52459,52458,5245
8,52459,52459,52459,52459,52459,52458,52458,52458,50411,50411,50411,50411,50411,50411
,50411,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,50378,50410,50411,50411,52459,52459,52459,52459,50410,50410,5
2459,52459,52459,52459,52459,52459,52459,52426,50346,50314,48201,46121,46056,44008,43
976,43944,
50379,50379,52427,50380,50346,52459,50378,50378,50378,50378,50378,50379,50379,50379,5
0346,50346,50346,50346,50347,50315,50282,48202,43977,43912,41864,46057,48234,48266,48
266,48266,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,482
02,48201,48201,48201,48202,48201,48201,48201,48201,48169,46121,46121,43944,43944,4605
7,48202,48170,46121,46121,46121,46121,46121,46121,46089,46089,46089,46089,46089,46089
,46056,46089,46089,43977,43944,46056,50347,50346,50346,50346,50346,50346,50346,50346,
50378,50378,50378,50379,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,50411,50411,52459,52459,52459,50410,50378,50378,50
410,52459,52459,52491,52459,52458,52491,52459,52459,52427,50346,50314,48202,46121,460
89,46057,43976,43944,43944,43944,43944,43944,43976,43977,46057,46089,48170,48203,5028
3,50347,50347,50347,50379,50379,52427,52459,52459,50411,50410,52459,52459,52458,52458
,52490,52490,52458,52458,52458,52458,52458,52458,52458,52458,52458,52458,52458,52458,
52458,52458,52458,52458,52458,52458,52458,52458,52458,52459,52459,52459,52459,52459,5
2427,50379,50347,50347,48266,48266,46057,46057,43977,43945,43944,43945,43945,43977,43
944,43976,
52459,50378,50346,50379,50345,50346,50345,50345,50346,50345,50346,50313,48265,48265,5
2426,50346,50314,48233,48169,46056,43943,41863,41864,43912,43977,48202,50314,48266,48
234,48266,50314,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,482
02,48201,48201,48201,48201,48233,48202,48202,48170,46089,43977,43944,41896,43977,4612
1,48202,48169,48169,48169,46089,46121,46121,46089,46089,46089,46089,46089,46089,46089
,46121,46057,46025,43944,43944,48202,50346,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50379,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52491,50411,50411,50
410,50410,50410,50410,50378,50410,50410,52458,52458,52458,52458,52458,52458,52459,524
59,50379,50347,48266,48234,46121,46121,43977,43977,43945,43944,43944,43944,43944,4394
4,43944,43944,43976,43977,46057,46089,46121,48169,48202,48202,48234,48234,50314,50314
,50346,50347,52427,52427,52427,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52427,52427,50379,50379,50347,50346,50347,50314,48266,48266,48202,48202,4
8170,46089,46057,43977,43944,43944,43977,43945,43945,43945,43977,46057,46089,46090,50
282,50314,
52459,50379,52394,52428,52427,50379,50347,50347,50347,50315,50282,48202,48202,48170,4
3977,43944,43944,41864,43944,43944,43944,43944,43976,46057,48202,50314,50314,48234,48
233,48234,50314,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,482
34,48202,48202,48202,48201,48169,46089,46056,43976,43944,43944,43912,44009,46089,4820
2,48169,46089,46089,48169,46089,46089,46089,46089,46089,46089,46089,46089,46057
,48170,43976,43944,43944,43977,50314,50346,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50379,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,52459,50411,52459,52459,52459,52459,52459,52459,52459,50379,50379,50379,52
459,52459,52459,52459,52459,52460,52459,50379,50379,50378,50378,50378,52426,50410,504
10,50411,50411,52491,50411,50411,50411,52427,52427,50346,50314,48202,46121,46088,4400
8,43977,43977,43977,43944,43944,43944,41864,41864,43945,43945,43945,43945,43945,43945
,43945,43977,43976,43976,43977,43977,43977,44009,44009,44009,44009,44009,44009,44009,
44009,43977,43977,43977,43977,43977,43977,43977,43977,43945,43945,43945,41864,41864,4
1864,43944,43944,43944,43977,43977,46089,46089,48169,48234,50314,52426,52458,52459,52
459,52491,
46122,46122,46056,46090,46089,46057,43977,43945,43945,43944,43912,41864,41864,41832,4
3913,43913,41864,41864,43944,43977,43977,46057,48202,48234,50314,50314,48233,48233,48
234,50282,50314,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,481
70,46122,46122,46090,46057,46025,43944,41864,43912,43944,46025,46057,46122,46121,4820
2,48169,46089,46089,48169,46089,46089,46089,46089,46089,46089,46089,46089,46089,46057
,46057,43944,43944,46057,48202,50346,50346,50346,50346,50346,50346,50346,50346,

50346,50378,50378,50379,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,50411,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,52459,50379,50379,50379,52459,52459,52459,52459,52459,52459,52459,52459,524
59,50410,50378,50378,50410,52459,52459,50410,52458,52459,52459,52459,52459,52459,5245
9,52427,52395,50347,50314,48234,48202,48169,46121,46057,46057,46057,43977,43977,43944
,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,
43944,43944,43944,43944,41896,41896,43944,43944,43976,43977,43977,43977,46089,46089,4
6121,48201,48234,50314,50347,52427,52459,52459,52459,52491,52491,52458,52458,50410,50
410,50410,
41832,41897,41831,41832,43912,41832,43912,43912,43912,43912,43912,43944,43944,43944,4
3944,43976,46057,46089,48169,48234,50314,50346,50314,50314,50314,48233,48233,48233,50
282,50282,50314,48234,48234,48234,48234,48234,48234,48202,48169,46121,46121,46089,440
08,43976,41896,41864,43913,43913,43945,43977,46057,48137,48169,48201,48202,46089,4612
1,48201,48169,46121,48169,48137,46089,46089,46089,46089,46089,46089,46089,46089,43977
,41864,43944,43944,48170,52396,50346,50345,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50379,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,52459,52459,52459,50411,50411,50411,50411,52459,52459,52459,52426,52426,52
426,52459,52459,52459,52459,52458,52458,52458,52458,52459,50411,50411,50411,52427,524
59,52459,52459,52459,52459,52459,52427,52459,52459,52459,50411,50379,52459,52459,5245
9,52458,52458,52458,52459,52459,52459,52459,52459,52459,52458,52459,52426,50378,50378
,50346,50346,50314,50314,50314,50282,48234,48234,48202,48201,48201,48201,48201,48201,
48202,48202,48234,48234,50282,50314,50314,50314,50346,50346,52394,52426,50379,52459,5
2459,52459,52459,52459,52458,52458,52458,50410,50410,50411,52459,52459,52459,52459,52459,50
411,52459,
46089,46056,48139,44041,46057,46122,46121,48201,48201,48201,48233,48234,50314,50314,5
0346,50346,50346,50314,50314,50282,48234,48234,48202,48202,48202,48202,48202,48202,48
202,48202,48170,48170,46090,46090,46057,43977,43977,43945,43944,41864,41864,41864,418
64,43944,43977,46057,46089,46121,48202,48202,48169,46121,46121,48169,46121,46121,4612
1,46121,46121,46121,46121,46121,46089,46121,48169,46121,46089,46057,43945,43944,41832
,43944,46121,48234,50346,50346,50346,50347,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50378,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,50379,50379,52459,52459,52459,52
459,52426,52426,52426,52426,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,50410,50410,50410,50410,50410,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459
,52459,52459,52490,52458,52458,52458,52458,52458,52458,52458,52491,52491,52491,52491,
52491,52491,52491,52491,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52426,52426,52426,52458,52458,52
459,52459,
52459,50378,52395,50410,52426,50346,50315,50315,50315,50283,48234,48202,48202,48202,4
6057,46057,46057,43976,43944,43944,43944,43911,41864,41864,41864,41864,41864,41864,41
864,41864,41831,41863,41863,41864,41864,43944,43944,43944,43977,46057,46057,46089,481
70,48170,48202,48202,48202,48202,48202,48201,48201,46121,46121,46121,48169,48169,4612
1,46121,46089,46089,46089,46089,46089,46089,46089,46057,43977,43944,43912,43912,46057
,46121,48234,50314,50346,50346,50346,50347,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50378,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50379,50379,52459,52
459,52459,52459,52459,52459,50411,50411,50411,50411,50411,50411,50411,50411,52459,524
59,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,50379,50379,5037
9,50379,50379,50379,52459,52459,52459,52459,52459,50379,50379,50379,50379,50379,52459
,52459,52459,52459,50379,50379,50379,50379,50379,50379,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52491,52491,52491,52459,52459,50411,50378,50378,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,50411,52459,52459,52459,52459,52459,52459,50411,52
459,52459,
46122,46088,46026,44040,46057,43976,43945,43944,43944,41864,41864,41864,41832,41831,4
3912,43912,43912,43912,43912,43912,43912,43912,41864,41864,43944,43944,43944,43976,43
976,43977,46089,46089,46121,46122,48202,48202,48202,48234,48202,48202,48202,48234,482
34,48202,48201,48201,48201,48169,48169,46121,48169,46121,46121,46121,48169,48169,4816

9,48169,48169,48169,46121,46121,46058,43977,43944,41864,41864,43944,43977,46089,50282
,50282,50346,50346,50346,50346,50346,50347,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50378,50379,50379,50379,50379,50379,50378,52427,52427,52459,52459,5
2459,52459,52459,50378,50411,50411,50410,50410,50410,52459,52459,52459,52459,52459,52
459,52459,52459,50411,50411,52459,52459,52459,52459,52459,52459,50379,50379,52459,524
59,52459,50379,50379,50379,50379,52459,52459,52459,52459,52459,52459,52459,52459,5245
9,52459,52459,52459,52459,52459,52459,52459,52459,50411,50411,50411,50411,50411,50411
,50411,50411,50411,50411,50411,50411,50411,52459,52459,52459,52459,52459,52459,52459,
52459,50379,50379,50379,50378,50378,50378,50410,52459,52459,52459,52491,50379,52459,5
2459,52459,52459,52459,52459,52459,52459,50378,50378,50411,52459,52459,50379,50379,50379,50
411,50411,
41865,41864,43882,39816,41865,41865,41864,41864,41864,41864,43944,43944,43944,43944,4
3944,43945,43977,46025,46057,46057,46089,46090,48170,48170,48202,48202,48202,48234,50
283,50283,48234,48234,48234,48234,48202,48201,48201,48201,48234,48202,48201,48201,482
01,48169,46121,46121,48169,48169,48169,48169,48170,48202,48170,48170,48137,48137,4813
7,46089,46089,46057,46025,43977,43912,43912,43944,43944,43977,46089,48234,50314,50346
,50346,50346,50346,50346,50346,50346,50347,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,52427,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52491,52491,50411,50
411,50411,50411,50411,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52458,52458,52458,52458,52490,52490,52490,5249
0,52458,52458,52458,52458,52458,52458,52458,52458,50410,50410,50410,50410,50410,50410
,50410,50410,52458,52458,52458,52458,52458,52458,52458,50410,52458,52458,52458,52458,
52458,52458,52458,52458,52459,52459,52459,52459,52459,50410,50410,50410,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,50411,50411,50411,50411,50411,50411,50411,52491,52
459,52459,
44008,46088,46058,44040,46121,46122,48202,48202,48202,48202,48234,50282,50282,50282,5
0315,50315,50314,50315,50314,50314,50315,50315,50282,50282,50282,50282,50282,50282,48
234,48234,48234,48234,48202,48202,48201,48201,48201,48201,48201,48201,48201,48201,482
01,48202,48202,48234,48201,48169,48169,48169,48170,48170,46089,46089,46057,46057,4397
7,43944,43912,41864,41832,41832,41864,43944,46089,48170,48234,50314,50314,50346,50314
,50314,50314,50314,50346,50346,50346,50347,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,52427,52427,52459,5
2459,52459,52459,52459,52459,52459,52459,50410,50410,50410,50410,50378,50378,50378,50
410,50410,52459,52459,52459,50378,50378,50378,50410,50410,52459,52459,52459,52459,524
59,52427,52427,52427,52427,52427,52459,52459,52459,52459,52459,52459,52427,52427,5242
7,52459,52459,52459,52459,52427,50379,50379,50379,50347,50347,50347,50347,50347,50347
,50347,50347,50347,50347,50347,50347,50347,50347,50347,50347,50346,50378,50378,50379,
50379,52427,52427,52459,52459,52459,52459,52459,52459,52459,52459,52459,52427,52427,5
2427,52427,52426,52426,52426,52458,52459,52459,52459,52459,50410,50410,50410,50410,52
459,52459,
52459,50378,52395,50377,52426,50346,50346,50346,50314,50314,50314,50314,50314,50314,5
0314,50314,48266,48266,48265,48233,48233,48233,48234,48234,48234,48234,48234,48233,48
233,48233,48202,48202,48202,48202,48202,48202,48234,48234,48202,48202,48202,48202,482
02,48202,46122,46121,46089,46089,46057,46057,43977,43976,43944,43944,41864,41864,4186
4,41864,41864,43944,43977,46057,48169,48201,48234,50314,50314,50314,50314,50314,50314
,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,52427,52427,52459,5
2459,52459,52459,52459,50410,50411,50411,52459,52459,52459,52459,52459,52491,52491,52459,52
459,52459,52459,52459,52459,52492,52492,52491,52459,52459,52459,52459,52427,52427,503
47,50346,50346,50314,50314,50314,48234,48170,48170,46122,46090,46090,46089,46057,4605
7,43977,43977,43977,43977,43977,43976,43976,43944,43945,43945,43945,43945,43945,43945
,43945,43945,43944,43944,43944,43944,43944,43944,43944,43944,41896,41896,43944,43944,
43945,43977,43977,43977,43976,44008,44008,46089,46089,46121,48169,48202,50314,50314,5
0346,50347,52427,52459,52459,54539,50379,52459,52459,52491,52491,52459,52459,50410,52
459,52459,
50412,50346,50315,48330,50346,48266,50346,50346,50346,50346,50346,50314,50314,48266,5
0314,50314,50314,50314,50314,50314,50314,50314,48234,48234,50282,50282,50282,48234,48

234,48234,50282,50282,50282,48234,48202,48202,48202,48202,46090,46090,46089,46089,460
57,43977,43944,43944,43944,43944,43944,41864,41864,41864,43944,43944,43944,43976,4397
6,46057,46089,48202,50282,50282,50315,50314,50314,50314,48234,48234,50314,50314,50314
,50314,50314,50314,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,52427,52427,52459,5
2459,52459,52459,52491,52491,52459,52459,52459,52459,52460,52460,52427,52427,50379,50
347,50347,50315,48266,48266,46121,46089,46089,46089,44009,43976,43976,43976,43944,439
44,43944,41864,41864,41863,41863,41863,41864,41864,41864,41864,41864,41864,41864,4186
4,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944
,43944,43944,43944,43944,43944,43944,43944,43944,43944,43944,43945,43945,43945,43945,
43945,43945,43945,43945,43977,43977,43977,43944,43944,43944,43944,43944,41863,41864,4
3944,43976,43976,46056,46088,46089,48267,50315,50347,50347,50379,52427,52459,52459,50
378,50411,
50347,48265,52395,48330,50314,48299,50314,50314,50314,50314,48266,48266,48266,48266,5
0282,50282,50314,50314,50314,50282,50282,50282,50314,50314,50282,48234,48202,48201,48
169,46121,46089,46089,46057,46056,43976,43976,43944,43944,41864,41864,41864,41864,418
64,41864,41864,41864,41864,41864,41864,43944,43944,43977,46057,46090,48202,48234,5028
2,50314,50314,50314,48233,48233,48233,48234,48266,50314,50314,50314,50314,50314,50314
,50314,50314,50314,50315,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,
50346,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,52427,52427,52459,5
2459,52459,52459,52459,52459,50410,50378,50346,48266,48233,46153,46121,44008,44008,43976,43
976,43944,41896,41864,41863,41864,41864,41864,41864,41864,41864,41864,41864,43912,439
12,43945,43945,43977,46025,46058,46058,46089,46089,48169,48169,48201,48202,48202,5028
2,48234,50282,50282,50314,50314,50314,50315,50315,50346,50346,50346,50346,50346,50346
,50346,50346,50314,50314,50314,48266,48234,48234,48234,48234,48170,48169,48137,46089,
46057,46056,46024,43976,43945,43945,43945,43945,43945,43945,43945,43945,43945,43945,4
3945,43945,43944,43944,43944,43944,41896,41896,43944,43976,44008,46089,48201,48234,50
378,50378,
52460,50346,50314,50379,52427,50346,50347,50347,50347,50315,50314,50314,50314,50314,4
8234,48202,48202,48170,46090,46089,46057,46057,43977,43977,43945,43944,41864,41864,41
864,41864,41864,41864,41864,41864,41864,41864,41864,41864,41864,41864,43944,439
44,43977,43977,43977,46089,46121,48201,48202,48234,48234,48234,48234,48234,48234,4823
4,48234,48233,48233,48233,48233,50314,50314,50314,50314,50314,50314,50314,48266
,48266,50314,50347,50347,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,
50378,50346,50378,50378,50378,50378,50378,50378,50378,52427,52427,52427,52395,50315,4
8202,48170,46089,43977,43945,43912,41864,41832,41864,41864,43912,43945,43913,41864,41
864,41864,43944,43944,43944,43944,43976,43977,46057,46089,48138,48170,48170,48201,482
34,48234,48234,48234,48266,48266,48266,50282,50282,50282,50282,50282,50282,50282,5028
2,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50315,50314,50314
,50314,50314,50346,50346,50346,50346,50346,50346,50378,50378,50347,50347,50378,50378,
50378,50378,50378,50378,50378,50378,50346,50346,50314,48266,48266,48265,48202,46121,4
6089,46057,43977,43977,43976,43944,43944,43944,43944,43944,43944,43944,43945,43945,43
913,43945,
48170,46089,48136,46122,48169,46089,46089,46089,46089,46057,44008,43976,43976,43944,4
1863,41863,41863,41864,41864,41832,41832,41831,41864,41864,41864,41864,41864,41864,43
944,43944,41864,43912,43912,43944,43944,43976,46025,46025,46089,46089,46121,48170,481
70,48202,48202,48202,48202,48202,48202,48202,48202,48202,48202,48202,48234,48234,4823
4,48234,48234,50282,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50346
,50346,50346,50346,50346,50346,50346,50346,50378,50378,50378,50378,50378,50346,50346,
50346,52427,52427,52427,50347,50347,50314,50314,50282,46089,46056,43976,43944,41863,4
1863,41864,41864,41864,41864,41864,43912,43944,43944,43945,43977,43944,43977,46057,46
090,48170,48170,48170,48169,48234,48234,48234,48234,48234,48202,48202,48202,48234,482
34,48234,48234,48234,48234,48234,48234,48234,48234,48233,48233,50314,50314,50314,5031
4,48233,48233,48265,48265,48266,48266,48266,50314,48266,48266,50314,50346,50346,50346
,50346,50346,50346,50346,50346,50346,50346,50314,50314,48265,50378,50378,50378,50378,
50378,50378,50378,50378,50378,50378,50379,50379,50379,50379,50379,50379,52458,52458,5
2459,52459,52459,52427,50379,50347,48234,48234,48202,46121,46089,44009,43976,43976,43
976,43976,

41864,43912,43912,41864,43912,43912,41864,41864,41864,41864,41864,41832,41831,41831,4
1832,41864,41864,41864,41864,43944,43944,43944,43944,43944,43976,43976,43977,46057,46
057,46057,46089,46089,46089,46121,48169,48169,48169,48169,48169,48169,48169,48201,482
01,48202,48202,48202,48202,48202,48202,48202,48202,48202,48234,48234,48234,48234,4820
2,48202,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,50282,48265
,48265,48265,50314,50346,50346,50346,50346,50314,50314,50346,50346,50347,50347,50347,
50347,48203,48170,46090,46057,43976,43944,41864,41864,41864,41864,41864,41864,41864,4
3944,43944,43977,43977,44008,46057,46089,46089,46121,48169,48169,48201,48201,48201,48
202,48201,48202,48202,48202,48201,48201,48233,48233,48233,48234,48234,48234,48234,482
34,48234,48234,48234,48234,48234,48234,48234,48234,48234,48266,48266,48266,48266,48266,4826
6,50314,50314,50314,50314,50346,50346,50346,50346,50346,50346,48266,48298,48298,48298
,48298,50346,48266,50314,50346,50346,50346,50378,50378,50378,50346,50346,50346,50346,
50346,50346,50346,50346,50346,50346,50346,50346,50346,50378,50379,50379,50379,50377,50377,5
0378,52458,52458,52458,52458,50378,52459,52459,52459,52459,52459,52459,52459,52427,50
314,50314,
41832,41832,43880,41865,41832,43912,41864,41864,43912,43944,43944,43945,43977,43977,4
3977,43977,43977,46057,46057,46057,46057,46089,46089,46089,46089,46121,46121,48170,48
170,48170,48169,48169,48169,48169,48169,48169,48169,48169,48201,48201,48201,48201,482
01,48201,48201,48201,48201,48201,48202,48202,48234,48234,48234,48202,48266,48234,4823
4,48234,48234,48266,48266,48266,48266,48266,48266,50314,50314,50314,50314,50314,50346
,50346,50346,50347,50347,50347,50315,50314,52395,50315,50282,48169,46089,43976,43944,
41863,41864,41864,41864,41864,41864,41864,41864,43944,43944,43944,46057,46089,46090,4
6090,46121,46089,48202,48202,48201,48201,48201,48201,48201,48233,48234,48234,48202,48
201,48201,48201,48201,48201,48234,48234,48234,48234,48234,48234,48201,48201,50282,502
82,50282,50282,50282,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,4823
4,48234,48234,48234,48234,48234,48234,48234,48234,48234,50315,50314,50314,50314,50314,50314
,50314,50347,50347,50347,50347,50347,50346,50346,50314,50314,50346,50346,50346,50346,
50346,50346,50346,50378,50378,50378,50378,50378,50378,50379,50379,50379,50379,50411,50411,5
0378,50379,50379,50379,52459,52459,50410,50410,50410,50410,52459,52459,52459,52459,52
491,52491,
43945,43977,46024,43978,43977,43977,43977,46025,46025,46025,46057,46057,46057,46057,4
6057,46057,46057,46057,46089,46089,46089,46089,46089,46121,46121,46089,46089,46089,46089,46
121,46121,46089,46089,46121,46121,46121,46121,46121,46121,48201,48201,48201,48201,482
01,48201,48201,48201,48202,48202,48202,48234,48234,48202,48202,48202,48233,48233,4823
3,48234,48234,48234,48234,48233,48233,48233,50314,50314,50314,50314,50314,50314,50315
,50315,50282,48202,48170,46089,46089,46057,41864,41864,41863,41864,41864,41864,43944,
43944,43912,43944,43944,43977,46057,46057,46090,48138,46121,46121,46121,46121,48169,4
8169,48169,48169,48201,48169,46121,46121,46121,46121,46121,48201,46121,48169,48201,48
202,48234,48234,48202,48202,48202,48202,48203,48203,48203,48202,48202,48202,46089,460
89,46089,46089,46057,46057,44008,44008,43945,43944,43944,43944,43944,43944,43944,4394
4,43912,43912,43912,43912,43944,43944,43944,43944,43944,43944,43944,43977,43977
,43977,46057,46057,46057,46089,46121,48202,48234,48234,50282,50314,50314,50347,50347,
50347,50379,52427,52459,50378,50378,50378,50378,50378,50378,50378,52459,52459,5
0379,50379,50379,50379,50379,50379,52459,52459,52459,52459,52459,52459,52459,52459,50
411,52459,
43977,44009,46024,44010,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,4
6089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46
089,46121,46122,46122,46122,48170,48169,48202,48202,48202,46121,48169,48169,48169,481
69,48202,48202,48202,48202,48202,48202,48202,48202,48234,48234,48234,48234,48234,4823
4,50314,50314,50314,50282,48234,50346,50314,50314,50314,48234,48202,48201,46121,43977
,43976,43944,41864,41864,41831,41831,41831,41864,41864,41864,43944,43944,43977,43977,
46057,46089,46089,46089,46089,46089,46089,46089,46089,48202,48202,48169,48169,48169,4
8169,48201,48201,46121,46121,48201,48202,48202,48202,48202,48201,48234,48202,48202,48
202,48170,46122,46121,46121,46057,46057,43977,43976,43944,41864,41831,41831,41895,418
95,41895,41895,41895,41863,41863,41863,43944,43944,43944,43944,43912,43912,43912,4391
2,43912,43912,43912,43912,43912,43880,41832,41832,43912,43912,41864,41864,41864,41864
,41864,41864,41864,41864,41864,41864,41864,41864,41864,41896,43944,43976,43976,
44008,46056,46089,46089,48234,50282,50314,50346,50347,52427,52427,52459,50346,50378,5

0378,50378,50378,50378,50378,52458,50379,50379,50379,50411,50411,52459,52459,52459,50
379,50379,
43977,43977,46024,44009,46025,46057,46057,46057,46057,46057,46057,46057,46057,46057,4
6057,46057,46057,46057,46057,46057,46057,46057,46089,46089,46089,46089,46089,46121,48
169,48169,46122,46122,46121,46121,46121,46121,46121,46121,48201,48201,48201,48201,482
02,48202,48202,48202,48202,48202,48202,48202,48202,48234,48234,48234,50282,50282,4823
4,48202,48202,48202,48202,48202,46056,46057,43976,43944,43944,41864,41864,41831,41864
,41864,41864,41864,43944,43944,43976,43976,46057,46057,46090,46089,46089,46089,46121,
46121,46089,46089,46089,46121,48169,48169,48169,48169,46089,46089,46121,48169,48169,4
8169,48169,48169,48201,48201,48202,48202,48170,46122,46122,46090,46057,43977,43976,43
944,41895,43943,43943,43944,43976,44008,46088,48169,48202,50282,50315,50315,50314,503
14,50315,50347,50347,50347,50347,50347,50315,50315,50315,50314,50314,50314,50314,5031
5,50282,50282,50282,48202,48202,48202,48169,48169,46089,46057,46057,46057,43977,43976
,43944,43944,41863,41864,41864,41864,41864,41864,41864,43944,41864,41864,43944,43944,
43944,43944,43944,43944,41863,41864,43944,43944,43976,46057,46089,46089,48266,50314,5
0346,50378,50379,52459,52459,52491,52459,52459,52459,52459,52459,52459,50410,50378,52
459,52459,
43945,46057,46024,43978,46025,46057,46057,46057,46057,46057,46057,46057,46057,46057,4
6057,46057,46057,46089,46090,46090,46090,46090,46089,46089,46089,46089,46089,46089,46
089,46089,48169,48169,48169,48169,48169,48169,48201,48201,48169,48169,48201,48201,482
01,48201,48201,48201,48202,48202,48234,48234,48234,48202,48169,46121,46057,46057,4397
6,43944,43944,43912,41864,41832,41864,41864,41832,41832,41832,41864,43913,43913,43944
,43977,46025,46057,46089,46089,46089,46057,46057,46057,46089,46089,46089,46089,46089,
46089,46089,46089,46089,46089,46089,46089,46121,46121,48170,48169,48169,48169,48169,4
8169,48170,48202,48169,46089,46057,43976,43944,41864,41864,41864,43944,43977,46089,48
202,50315,50347,52427,52427,52459,52459,52459,52427,50378,50378,50378,50346,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50346,50346,50346,50346,50314,50314,50314,50314,50314,50314,50314,50314,50282,48234
,48234,48234,48202,48202,48170,48170,46089,46057,46057,46057,43977,43977,43945,43945,
43944,43912,43912,43912,43913,43913,43913,43912,41864,41864,41864,41864,41864,41864,4
1864,43944,43977,46057,46121,48202,50346,50346,50378,52459,52459,52459,52491,52458,52
458,52458,
43912,46057,43977,46057,46024,46057,46057,46057,46057,46057,46057,46057,46057,46057,4
6057,46057,46057,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46
121,46121,46121,46121,46121,46121,46121,48169,48169,48169,48202,48202,48202,48202,482
01,48169,48169,48169,46090,46057,46025,43945,43944,41864,41832,41832,41864,41864,4186
4,41864,41864,41864,41864,41864,43944,43944,43944,43976,43977,46057,46057,46057,46089
,46089,46057,46057,46057,46089,46089,46089,46089,46089,46089,46089,46089,46089,46090,
46090,46122,46121,46089,46089,46121,48169,48169,48169,48169,48169,48169,48170,46090,4
6057,43945,43913,41832,41832,41864,41896,44008,46153,48266,50379,52459,52459,52459,52
459,52427,50378,50378,50378,50379,50379,50379,50379,50378,50378,50378,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,
48202,48201,48201,48201,46089,46089,46057,46057,44009,44008,43976,43976,43944,43944,4
3944,41864,41864,41864,41864,41864,41864,41864,41864,43944,46057,46090,48202,50283,50
347,50379,
43912,46025,43977,46025,46025,46057,46057,46057,46057,46057,46057,46057,46057,46057,4
6089,46089,46089,46089,46089,46089,46089,46089,46121,46121,46121,46121,46121,46121,46
121,46121,48170,48170,46122,46122,46121,46121,46089,46089,46057,46057,43977,43976,439
44,43944,41864,41864,41864,41864,41864,41864,41864,41864,41864,41864,41864,4394
4,43944,43977,43977,43977,46057,43977,46057,46057,46057,46057,46057,46057,46057,46057
,46057,46057,46057,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,
46089,46121,46121,48169,48169,48170,46090,46090,46090,46089,46057,43976,43944,41863,4
1863,43944,43944,46121,48202,50315,52428,52460,52459,52459,50410,52459,52459,52459,52
459,52427,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234

,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,
48202,48201,48201,48201,48202,48202,48202,48202,48202,48202,48202,48169,48170,46121,4
6089,46089,46057,46057,46057,43977,43977,43977,43944,43944,41864,41864,41863,41863,41
895,43943,
41831,43912,41864,43944,43944,43945,43945,43945,43945,43945,43945,43945,43977,43977,4
3977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43
977,43977,43944,43944,43944,43944,41864,41864,41864,41864,41864,41864,41832,41832,418
32,41832,41832,41832,41832,41864,41864,43912,43944,43944,43945,43945,43977,43977,4397
7,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057
,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46089,46121,46121,
46120,46089,46089,46089,46090,46090,46058,43945,43945,41864,41864,41864,41864,43976,4
6121,50314,52427,52459,52459,52459,52427,50411,52459,52459,52459,52459,52459,52459,52
459,52427,50378,50378,50378,50379,50379,50379,50379,50378,50378,50378,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,
48202,48201,48201,48201,48201,48201,48169,48169,48201,48201,48201,48169,48201,48169,4
8169,48169,48169,48169,48169,48169,46121,46121,46089,46089,46057,44008,43976,43976,43
944,43944,
39719,43880,41832,41832,41832,41832,41832,41832,41832,41832,41832,41832,41832,41832,4
1832,41832,41832,41832,41832,41832,41832,41832,41832,41832,41832,41832,41831,41831,41
831,41831,41832,41832,41832,41832,41832,41832,41832,41832,41832,41864,41864,41864,439
12,43945,43945,43945,43977,43977,43977,43977,46025,46025,46057,46025,46025,46057,46057,4605
7,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057
,46089,46089,46089,46089,46089,46089,46089,46057,46089,46089,46089,46089,46089,46089,
46089,48170,46057,43944,41864,41864,41864,41832,41832,43912,46057,48234,52428,54508,5
2492,52459,52459,50411,50411,50410,50410,50410,50410,50410,50410,52459,52459,52459,52
459,52427,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,
48202,48202,48201,48201,48201,48202,48201,48201,48201,48201,48201,48169,46121,48169,46121,4
6121,46121,48169,48169,48169,48169,46089,46121,48169,48169,48169,48170,48170,48170,46
089,46089,
39719,43880,41865,43880,41832,41832,41832,41832,41832,41832,41832,41832,41832,41832,4
1832,41832,41832,41832,41832,41832,41832,41832,41832,41832,41832,41832,41864,41864,41
864,41864,41864,41864,41864,41864,41864,41864,43912,43912,43944,43944,43945,43945,43945,43945,43945,43945,43977,43977,439
77,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43976,43977,4397
7,43977,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057
,46057,46057,46089,46089,46089,46089,46089,46089,46089,46089,46089,46057,46057,43977,
43977,41832,41864,43944,41864,41831,41863,46089,50282,52459,52427,52427,52427,50378,5
0411,52459,52491,50378,50410,50411,52491,52491,52459,52459,50379,52459,52459,52459,52
459,52427,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48234,48202,48202,
48202,48201,48201,48201,48202,48202,48201,48201,48201,48201,48201,48169,48201,48170,4
8169,48169,48169,48170,46121,46089,46121,46121,46089,46089,46089,46089,46089,46089,46
089,46089,
39687,43880,41865,43912,43880,41832,43880,43880,43880,43880,43880,43880,43880,43880,4
3913,43912,43912,43912,43912,43912,43912,43912,43912,43944,43944,43944,43944,43944,43
944,43944,43944,43944,43944,43944,43944,43944,43945,43945,43977,43977,43977,43977,439
77,43945,43944,43944,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,4397
7,43977,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057
,46057,46089,46089,46089,46089,46089,46089,46089,46057,46057,43977,43944,41864,41832,
41832,41864,41864,41896,46089,50347,52460,52459,50378,50378,52458,52458,52458,52459,5
2459,50411,50411,52491,50411,50410,50378,50379,50379,52459,52459,52459,52459,52459,52
459,52427,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50346,503

46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48234,48234,48202,48202,
48202,48201,48201,48201,48201,48169,46121,46121,48169,48169,46121,46121,48170,48169,4
8169,48170,48170,48170,48170,46090,48169,48169,46121,46089,46089,46089,46089,46089,46
089,46089,
39686,43880,41865,41832,43880,43880,41864,41864,41864,41864,41864,41864,41864,41864,4
3912,43912,43912,43912,43912,43912,43912,43944,43912,43912,43944,43944,43944,43944,43
944,43944,43944,43944,43944,43944,43944,43944,43945,43945,43977,43977,43977,43977,439
77,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,46057,46057,4605
7,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057,46057
,46057,46057,46057,46057,46057,43977,43977,41864,41864,41832,41831,41831,41863,43944,
43944,46121,50314,52492,54572,52459,50378,50410,52491,52491,52459,50410,50378,50378,5
0411,52459,52491,52459,52459,52459,52459,52459,52459,50411,50379,52459,52459,52459,52
459,52427,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,
48202,48201,48201,48201,48201,48201,48201,48201,48201,48201,48201,48201,46121,46089,4
6089,46121,48169,48169,48169,48170,46089,46089,46089,46089,46089,46089,46089,46089,46
089,46089,
39718,43912,41865,41832,41832,43880,41864,41864,41864,41864,41864,41864,41864,41864,4
1864,41864,43912,43912,43912,43912,43912,43944,43912,43912,43912,43944,43944,43944,43
944,43944,43944,43944,43944,43944,43945,43945,43945,43945,43977,43977,43944,43944,439
77,43977,43977,43977,43976,43976,43976,43976,43977,43977,43977,46057,43977,43977,4397
7,43977,43977,43977,46057,46057,46057,46057,46057,46057,46057,46057,46089,46089,46057
,46057,43977,43977,43944,41864,41864,41831,41832,41832,41832,43944,46057,48234,52427,
54508,52491,52459,50378,50378,50378,50411,50411,50411,50379,50411,52459,52491,52459,5
2459,50411,50378,52459,52459,52426,50378,50378,50410,50410,52459,52459,52459,52459,52
459,52427,50378,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50346,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,5034
6,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,48234,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,
48202,48202,48202,48202,48202,48201,48201,48201,48201,48201,48169,48169,48201,48201,4
8201,48169,48169,48169,48169,46121,46089,46121,46121,46089,46089,46089,46089,46089,46
057,46057,
39686,43912,43880,41832,43880,41832,41864,41864,41864,41864,41864,41864,41864,41864,4
1864,41864,41864,41864,41864,41864,43944,43944,43944,43944,43944,43944,43944,43944,43
944,43944,43944,43944,43944,43944,43944,43945,43945,43945,43944,43977,43977,43977,439
77,43977,43977,43977,43977,43977,43977,43977,43977,43977,43977,44009,43977,43977,4397
7,43977,46057,46057,46057,46057,46056,46056,46057,46057,43976,43944,43944,41863,41865
,41833,41832,41832,41831,41831,41864,43944,46120,48233,50378,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,50411,50411,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50378,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50314,5031
4,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,
48202,48201,48201,48201,48201,48201,48201,48201,48201,48201,48201,48169,48169,48169,4
8169,48169,48169,48169,48169,46121,46121,46121,46121,46121,46089,46089,46089,46089,46
089,46089,
39718,43912,43913,41865,43912,41832,43911,43911,43911,43912,43912,43912,43912,43912,4
3880,43912,43912,43912,43912,43912,43912,43912,43912,43912,43912,43912,43944,43944,43
944,43945,43944,43944,43944,43944,43944,43944,43944,43945,43977,43977,43977,43977,439
77,43977,43977,43977,46057,46057,46057,46057,46024,46024,43976,43976,46025,46025,4397
7,43977,43945,43945,43944,43944,43913,43913,41832,41832,41832,41832,41832,41832,39751
,41863,43944,46057,48169,50282,50347,52428,50378,52459,52459,52459,52459,50411,50411,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5

2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,50411,50411,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50378,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50314,5031
4,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48201,
48201,48201,48201,48201,48202,48202,48202,48202,48202,48202,48202,48169,48170,48169,4
8169,48169,48169,46121,46121,46121,46121,46121,46089,46089,46089,46089,46089,46089,46
089,46089,
41766,41831,41832,39752,41832,41799,41800,41800,41800,41800,41800,41800,41800,41800,4
3880,43880,43912,43912,43912,43912,43912,43912,43880,43880,43880,43880,43912,43912,43
912,43912,43945,43945,43945,43945,43945,43945,43945,43912,43912,43912,43945,43945,439
45,43945,43945,43945,43913,43881,43881,43881,43881,43881,43881,43881,41832,41832,4183
2,41800,41800,41799,41799,41799,39783,39783,41832,41864,43945,46057,46090,48170,50347
,52427,52427,52459,52459,52459,52459,52459,52491,52491,52459,52459,50378,50378,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,50411,50411,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50378,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50314,5031
4,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,
48202,48202,48202,48202,48202,48202,48169,48169,48169,48169,46121,46121,48170,48170,4
8170,46121,46121,46089,46089,46089,46121,46121,46121,46089,46089,46089,46089,46089,46
089,46089,
41830,41863,41863,39784,41863,41863,41800,41800,41800,41800,41800,41800,41800,41800,4
1799,41799,41799,41799,41799,41799,41799,41799,41831,41799,41799,41799,41799,41799,41
799,41799,41799,41799,39751,39751,39751,39751,39751,39751,39751,39751,39751,39751,397
51,39751,41831,41831,39752,41800,41800,41800,41832,41832,41832,41832,41831,41864,4186
4,43976,46057,46089,48170,48202,50314,50346,50347,50379,52459,52459,52491,52491,52459
,52459,52459,52459,52459,52459,52459,52491,50411,50378,50378,50378,50411,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,50411,50411,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50378,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50314,5031
4,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,
48202,48202,48201,48201,48202,48202,48202,48202,48202,48202,48202,48202,48169,48169,4
8169,48169,48169,48169,48169,48169,48169,46121,46121,46121,46121,46089,46089,46089,46
089,46089,
48202,50347,52427,50347,50346,50347,52426,52426,52426,52426,50346,50346,50346,50346,5
0347,50347,50347,50347,50315,50315,50315,50315,50314,48234,48234,48234,48234,48234,48
202,48202,48202,48170,48170,48170,48170,48169,48169,48169,48169,48202,48202,48202,482
02,48202,48202,48202,48233,48233,48233,50313,50314,50346,50346,50346,52460,52460,5246
0,52460,52459,52459,52459,52459,52459,52459,52491,52491,52490,52458,50410,50409,52491
,52459,52459,50378,50378,50378,50378,50378,50411,50411,52459,52459,52459,52459,50411,
50378,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,50411,50411,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50378,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50314,5031
4,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,
48201,48201,48201,48201,48201,48169,46121,48169,48169,48169,46121,46121,48169,46121,4
6121,46121,46121,46121,46089,46089,46121,46121,46089,46089,46089,46089,46089,46089,46
089,46089,
50346,52459,52491,50411,52459,52459,50410,50410,50410,50410,50410,50410,50410,50410,5
0411,50411,50411,50411,50411,50411,50411,50411,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52458,52458,52458,52458,52458,52458,52458,52458,50378,50378,5037
8,50378,50410,50410,50410,50442,50379,50379,50379,52459,52459,52459,52459,52459,50410

,50410,50411,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,50411,50411,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50378,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50314,5031
4,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,
48202,48202,48202,48202,48201,48201,48201,48201,48201,48201,48201,48169,48169,48169,4
8169,48169,48169,48169,48169,46121,46121,48169,48169,48169,46121,46089,46089,46089,46089,46
089,46089,
50346,50410,52458,50379,50378,52459,50411,50411,50411,50411,50411,50411,50411,50411,5
0410,50410,50411,50411,50411,50411,50411,50411,52459,52459,52459,52459,52459,52459,52
459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,524
59,52459,52459,52459,52459,52459,52459,52459,50379,50379,50379,50379,52459,52459,5245
9,52459,52459,52459,52459,52459,52460,52459,52459,52459,52459,52459,52459,52459,50411
,50411,50411,52459,52459,50378,50378,50378,50411,50411,50411,50411,50411,52459,52459,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,50411,50411,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50378,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50314,5031
4,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48202,48202,
48201,48201,48201,48234,48202,48202,48202,48202,48202,48169,46121,48170,48170,48170,4
8170,48170,48170,48170,48170,46121,46121,46089,46089,46089,46089,46089,46089,46089,46
089,46089,
50379,52459,52491,50443,52426,50411,52459,52459,52459,52459,52459,52459,52459,52459,5
2491,52491,52491,52491,52491,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,52459,50410,50410,50410,50410,50410,50410,52459,52459,50410,50410,50410,50410,504
10,50410,50410,50410,52458,52458,52458,52458,52458,52458,52458,52458,52459,52459,5245
9,52459,52459,52459,52427,52427,52427,50410,50410,50410,52459,52459,52459,52459,52459,50411
,50411,52459,52459,52459,52459,52459,52459,50411,52459,52459,52459,52459,50411,50411,
52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,5
2459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52459,52
459,50411,50411,50378,50378,50379,50379,50379,50379,50379,50378,50378,50378,50378,503
46,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50346,50314,5031
4,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50314,50282,48234,48234
,48234,48234,48234,48234,48234,48234,48234,48234,48234,48234,48202,48202,48201,48169,
46121,46121,46089,46089,44009,43977,43976,43976,43976,43944,41896,41896,41864,41864,4
3944,43944,43944,43944,43977,43977,46056,46056,46056,46056,46057,46089,46089,46089,46
089,46089};

const int mole_shape[] =
{65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63390,56920
,50548,46257,46225,37707,35561,35561,35562,46225,46225,44209,54872,61212,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,61211,50549,37707,29190,24898,24898
,24899,24866,24867,24866,24866,24899,24866,24867,24866,24866,24866,24866,27044,35562,
46257,59066,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,63390,48403,31336,24866,24866,24866,24867,24866,35425
,35395,35397,35396,39653,39622,39621,39621,35396,35396,35396,35426,26978,24898,24866,
24866,24866,29190,42031,59066,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,57018,39853,24866,24866,24899,27010,37569,50240,56640,65056,65056
,54467,45927,45927,45927,45927,45927,45927,45927,45927,50181,65056,65056,60800,52384,
39713,29122,24866,24867,24898,35562,50549,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,59066,35561,24899,24866,24898,37569,52353,65056,65056,65056,65088,65056,65056
,54403,41701,41669,41701,41701,41669,41669,41669,41700,48069,65056,65056,65056,65088,
65056,65056,56640,39681,27010,24866,24866,29190,52694,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6338

9,39886,24866,24866,27010,41825,60800,65056,65056,65056,65024,64993,62882,62851,62819
,50115,39490,39523,39491,39523,39523,39523,39523,37474,45859,62819,62819,62850,64961,
65024,65056,65056,65056,62912,50241,29122,24898,24866,35562,59066,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54742,2704
4,24866,24866,41825,62944,65056,65056,65056,64961,62818,62819,62818,62819,62819,62819
,52195,39522,39523,39523,39555,39523,39555,39523,39554,45859,62819,62819,62851,62819,
62819,62851,64993,65056,65056,65056,65056,48097,27010,24867,24866,44209,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63390,39885,24866,2486
6,35425,58752,65056,65056,65024,64929,62818,62786,62818,62819,62819,62819,62819,62851
,52227,39555,39555,39555,39555,39555,39555,39554,39555,45891,62851,64899,62851,64930,
64993,65056,65056,65056,65056,65056,65056,65056,62944,41825,24898,24866,33384,61244,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,61244,33384,24866,24898,4812
9,65056,65056,65024,64929,62786,62786,62786,62787,62787,62819,62819,62819,64899,62851
,52227,41635,41635,41635,41635,41635,41635,41635,41635,45923,62851,62851,64962,65056,
65056,65088,65056,65090,65257,65257,65257,65124,65056,65056,54497,29122,24866,27044,5
4872,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,59066,27044,24866,29122,56640,6505
6,65056,64992,62754,62754,62786,62787,62787,62787,62819,64867,62819,62851,62851,62851
,52259,41635,41667,41635,41667,41667,41667,41635,41635,47971,62883,64962,65056,65056,
65088,65056,65290,65491,65459,65491,65459,65491,65290,65089,65056,62944,37569,24898,2
4866,50548,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,59066,27044,24866,33282,62944,65056,6502
4,64865,62722,62754,62754,62786,62787,62787,62819,62819,62851,62851,64899,62851,62883
,52259,41667,43715,43715,43715,43715,43715,43715,43715,48003,64963,65024,65056,65056,
65056,65223,65490,65458,65458,65491,65459,65459,65459,65424,65157,65056,65056,41825,2
4866,24898,50549,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65

535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,59066,27044,24866,39681,65056,65088,65024,6278
6,62722,62755,62754,62787,64835,62786,62819,62819,62851,62851,62851,62883,64932,64931
,54339,43715,43747,43747,43747,43747,43747,43747,43747,48003,64994,65056,65056,65056,
65056,65391,65459,65459,65459,65458,65458,65458,65459,65458,65458,65257,65056,65056,5
0273,24866,24866,50548,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,59066,27044,24898,39681,65056,65088,64961,62722,6477
0,62722,62754,62754,62787,62819,62819,62819,62819,64899,64899,62851,64932,64932,64931
,54339,43747,43747,43747,43779,43779,43779,43747,43747,48035,64994,65056,65056,65056,
65056,65458,65491,65459,65458,65490,65458,65491,65458,65490,65458,65491,65224,65056,6
5056,50241,24866,24866,50549,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,63389,29190,24866,37538,65056,65056,64928,62722,62722,6272
2,62754,62786,62786,62787,62818,62819,64867,64867,62851,62883,62883,62884,64932,64964
,54339,43747,45827,45827,45827,45827,45859,45827,45827,50083,65026,65056,65056,65056,
65056,65424,65459,65491,65491,65459,65458,65459,65491,65458,65490,65459,65358,65056,6
5056,65056,50240,24866,24898,56920,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,35561,24866,33314,62944,65056,62880,62690,62722,62722,6272
2,62754,62754,62787,62787,62787,64867,62819,62851,62851,62851,64931,64932,62884,64964
,54371,45827,45859,45859,45859,45891,45859,45859,45828,50115,65026,65088,65056,65088,
65056,65324,65459,65490,65490,65458,65491,65459,65458,65458,65490,65459,65457,65056,6
5056,65088,65056,39681,24898,27044,61244,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,46225,24898,27010,60800,65056,64928,62689,62690,62722,62722,6272
2,62754,62755,62787,62787,62819,62819,62851,64899,62883,62883,64932,64931,64963,64964
,54371,45859,45859,47907,45891,47939,47939,45860,45859,50115,65026,65056,65056,65056,
65056,65122,65425,65458,65459,65458,65459,65491,65459,65491,65459,65459,65358,65056,6
5056,64960,65056,65056,33378,24898,35561,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,59066,24898,24866,52353,65056,64960,60642,62690,62690,62722,62722,6275
4,62754,62786,62787,62818,62819,64867,62851,62851,62851,64931,62883,64964,64964,64996
,54404,45859,45891,47939,47971,47939,47971,47940,45891,50115,64996,65056,65056,65056,
65056,65056,65223,65490,65459,65458,65491,65458,65459,65459,65459,65459,65257,65056,6
5056,62848,62785,65056,62944,26978,24898,46225,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,33384,24866,41825,65056,65024,62690,62658,62690,62690,62722,62722,6275
4,62755,62754,62786,64835,62819,64867,62818,62851,62883,62883,64931,64964,64995,62916
,54403,45859,45827,35459,35458,37539,37507,41699,47939,52195,64996,65026,65056,65056,
65056,65056,65056,65156,65424,65491,65458,65459,65491,65458,65491,65323,65056,65056,6
5056,64897,62722,64865,65056,52352,24866,27044,61243,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,48403,24866,29122,62912,65056,62689,62689,62690,62690,62690,62722,62722,6275
4,62786,62786,64835,62787,64867,62850,62851,62852,64899,64931,64932,64964,64964,48227
,20961,4192,0,0,0,0,0,0,2080,14689,39842,64996,65056,65056,65056,65056,65056,65056,6505
6,65191,65324,65358,65358,65357,65223,65056,65056,65056,65056,62817,62690,62690,64960
,65056,37569,24866,37707,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,63390,27044,24867,50241,65056,64865,62657,62657,62658,62690,62722,62722,62722,6275
4,62754,62786,62786,62819,62818,62819,62819,64899,62883,64931,64931,48227,14689,0,0,0,
0,0,0,0,0,0,0,0,0,2144,35649,65056,65056,65088,65056,65088,65056,65056,65056,65056,65056
,65056,65056,65056,65056,65056,65024,62722,64738,62690,62690,65024,58752,26978,24866,
56920,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,46225,24866,35425,65056,65024,62625,62657,62658,62690,62690,62690,62722,62722,6275
4,62754,62787,64835,62787,62819,62819,62851,62851,64931,60739,23073,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,10528,52480,65056,65056,65088,65056,65056,65056,65056,65056,65056,65056,6505
6,65056,65056,62817,62722,62690,62689,62689,62753,65056,46017,24866,35562,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,27012,24866,54496,65056,62721,62625,62657,62658,62690,62690,62690,62722,62722,6275
4,62754,62786,62786,62787,62819,62819,62851,62851,60707,10496,0,0,0,0,0,0,10597,21098,2
3243,23275,23275,10597,2113,0,0,0,0,0,2144,48256,65056,65056,65056,65056,65056,65088,65
088,65056,65056,65056,65056,62849,62722,62722,62690,62690,62657,62657,64960,62944,270
10,24867,56920,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,505
49,24866,31266,65056,64960,62625,62657,62658,62657,62690,62690,62690,62722,62722,6275
4,62754,62786,62786,62819,62819,62819,62851,60707,10496,0,0,0,0,2113,23243,44405,48631,
48663,50744,50713,50712,48664,48599,29582,8452,0,0,0,0,0,48194,64993,65024,65056,65056,
65056,65056,65088,65056,64992,62786,62722,62722,62722,62690,62690,62690,62657,62689,6
5056,43937,24898,39885,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,

65535,33383,24866,50240,65056,62721,62625,62625,62657,62658,62690,62690,62690,62722,62722,62754,62754,62754,62786,62819,62787,62819,64867,10496,0,0,0,0,16936,44405,50712,52825,54938,54938,54970,54970,54970,54938,52857,52857,50712,25388,0,0,0,0,2144,50306,62851,62882,64929,64961,64929,62850,62786,62754,62754,62722,62722,62690,62690,62690,62658,62657,62657,64897,58752,24866,24866,63389,65535,

65535,61244,24899,27010,62912,62912,62593,62625,62658,62657,62657,62690,62690,62690,62722,62722,62754,62754,62754,64803,62787,62787,64867,27201,0,0,0,0,29582,48631,52825,54938,57051,57083,57083,59164,59164,57083,61309,65535,65535,65535,65503,48599,2113,0,0,0,10496,64899,62819,62819,62819,62787,62787,62786,62754,62722,62722,62722,62722,62690,62690,62657,62657,62657,62657,65056,37538,24898,48371,65535,

65535,48402,24866,39681,65056,62753,62593,62625,62625,62657,62657,62690,62690,62690,62722,62722,62754,62754,62754,62787,62787,62787,54435,0,0,0,0,25388,48631,52825,54970,57083,59164,59196,59196,59196,59196,63422,65535,65535,65535,65535,65535,65535,50712,0,0,0,0,35585,62819,62819,62787,64835,64835,62754,62754,62754,62722,62722,62690,62690,62658,62657,62657,62625,62625,64960,50241,24866,33384,65535,

65535,33384,24898,52384,65024,62593,62593,62625,62625,62658,62657,62658,62690,62690,62722,62722,62722,62754,62754,62786,62787,62787,23041,0,0,0,14790,48631,52825,57051,57083,59164,59196,61277,61277,61277,61309,65535,65535,65535,65535,65535,65535,65535,65535,31695,0,0,0,2112,62819,62819,62787,62786,62786,62754,62754,62754,62722,62690,62690,62690,62689,62658,62657,62626,62625,62752,62912,24866,24866,63390,65535,

65535,63390,24867,24898,65056,62816,62593,62593,62593,62625,62625,62657,62658,62690,62690,62722,62722,62722,62754,62754,62786,62786,58594,0,0,0,0,42260,50744,54970,57083,59164,61277,61277,61277,61309,61309,63422,65535,65535,65535,65535,65535,65535,65535,65535,59164,8452,0,0,0,39778,62818,62787,62787,62754,62754,62754,62722,62722,62722,62690,62690,62658,62657,62625,62625,62625,62625,65056,35457,24866,50548,65535,

65535,52694,24866,31171,48038,56354,62593,62593,62593,62625,62626,62625,62658,62690,62690,62690,62722,62722,62722,62754,62754,62754,39745,0,0,0,16936,48663,54938,57083,59164,61277,61277,61309,61309,61309,61309,65503,65535,65535,65535,65535,65535,65535,65535,65535,61309,29582,0,0,0,18849,64867,62787,62787,62754,62754,62722,62722,62721,62690,62689,62690,62658,62657,62626,62625,62593,58369,48003,35396,24867,42031,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535

,65535,65535,65535,

65535,46225,24898,35396,45927,24898,33282,50050,62593,62625,62625,62657,62657,62658,62690,62690,62722,62722,62722,62754,62754,62786,18817,0,0,0,35921,50776,54970,59164,59196,61277,61309,61309,61309,61309,61309,65503,65535,65535,65535,65535,65535,65535,65535,65535,61309,48631,2113,0,0,2112,62787,62787,62754,62754,62754,62722,62722,62722,62690,62690,62690,62657,62625,62625,54273,37506,24899,33316,39622,24867,35562,65535,

65535,35562,24866,37541,41702,24866,24866,24866,29090,41634,56321,62626,62657,62658,62690,62690,62690,62722,62722,62754,62754,62754,10464,0,0,0,48599,52857,57051,59164,61277,61309,61309,61309,61309,61309,61309,63422,65535,65535,65535,65535,65535,65535,65535,65535,59163,50712,12743,0,0,0,56450,62787,62754,62754,62722,62722,62722,62690,62690,62689,62658,58434,45858,31170,24866,24866,24866,29059,43846,24898,24866,65535,

65535,63389,31335,24867,41734,35428,24866,24866,24866,24866,24866,24898,29090,41666,50114,60578,62690,62690,62690,62722,62722,62754,62755,0,0,0,10597,48631,52889,57083,59196,61277,61309,61309,61341,61341,61341,61309,61309,65503,65535,65535,65535,65535,65535,65535,63422,54971,50745,23275,0,0,0,48098,62754,62754,62754,62722,62722,62722,62690,54274,43778,33250,24898,24867,24866,24866,24866,24866,24899,43846,26979,24899,59066,65535,

65535,59066,29190,24898,24866,45927,33316,24898,24867,24866,24898,24866,24866,24866,24898,26947,29026,35330,43778,50082,56386,62722,64770,62754,0,0,0,10597,48632,54938,57083,59196,61277,61309,61309,61341,61341,61341,61309,63390,63390,65503,65535,65535,65535,65535,63422,59164,57051,52825,23243,0,0,0,46017,62754,62754,58530,52194,45890,35426,26978,26978,26946,24898,24866,24899,24866,24866,24867,24866,24866,39621,31140,24867,27012,50549,65535,

65535,59066,27044,24867,29059,41734,45927,29091,24866,24898,24866,24899,24866,24866,24866,24898,24898,26946,26978,29026,29027,29059,29090,37442,37474,0,0,0,10597,48632,54937,57083,59196,61277,61309,61309,61309,61309,61309,61309,65535,65535,65503,63390,63390,63390,61309,59228,59164,57051,52825,23275,0,0,0,29153,39555,31106,31139,29058,29058,26978,26978,26946,24898,24866,24866,24866,24866,24866,24866,24898,24866,37509,43814,33284,24898,24898,50549,65535,

65535,61212,27044,24866,33284,43846,35396,26947,24866,24898,24866,24898,24866,24866,24866,24898,24866,24866,26946,26978,26978,29027,29059,29059,29090,29059,0,0,0,8452,48631,52857,57083,59196,61277,61309,61309,61309,61309,61309,61309,65535,65535,65503,61309,61309,61309,61277,59196,59164,55002,52825,23275,0,0,0,20737,31138,29091,29058,29058,29027,26978,26946,26946,24898,24898,24866,2486

6,24866,24866,24898,24866,24866,24898,33252,41766,37509,24866,24867,52694,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,35562,24866,37569,58594,29058,24898,248
66,24866,24898,24866,24866,24866,24899,24866,24867,24866,24898,24898,26946,26978,2697
9,29026,29058,29058,29058,4160,0,0,0,44406,52857,57051,59164,61277,61309,61309,61309,6
1309,61309,61309,63390,63390,63390,63390,61309,61277,61277,59196,57084,54970,50712,10
597,0,0,0,26978,29090,29059,29059,29026,26978,26979,24898,24898,24866,24867,24899,2486
6,24866,24866,24866,24867,24866,24867,24867,24898,52227,48129,24866,27044,63389,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,61212,24867,26978,62880,62496,58336,41633,269
78,24866,24898,24866,24866,24898,24866,24866,24866,24866,24866,24898,24899,26946,2697
8,26978,29059,29058,29091,8256,0,0,0,31727,50776,54971,59164,59196,61277,61309,61309,6
1309,61309,61309,61309,63390,61309,61309,61309,61309,61277,59196,57083,54938,46486,0,
0,0,2080,26978,29058,29058,29058,26978,26978,26946,26946,24866,24866,24866,24866,2486
6,24866,24866,24866,24866,24899,24866,37473,56257,62496,64960,35425,24866,48403,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,48403,24866,39714,62880,62464,62464,60416,603
84,47873,29058,24867,24866,24866,24866,24867,24866,24866,24867,24866,24866,24898,2697
8,26978,26979,29027,29058,16577,0,0,0,10597,48631,54938,57051,59164,61277,61277,61309,
61309,61309,61309,61309,61309,61341,61309,61309,61277,59196,57116,54971,50777,27469,0
,0,0,8256,29058,29059,29026,26978,26947,26947,24898,24898,24866,24866,24866,24866,2489
8,24866,24866,24898,26978,41601,56289,62496,62496,60416,64800,50240,24867,37707,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65437,50549,35561,24866,43937,64960,60384,62464,60416,604
48,62496,62496,54176,41602,29090,24898,24898,24866,24866,24866,24866,24866,24899,2489
8,26946,26946,26978,29058,24866,0,0,0,0,35921,50745,54970,57083,59196,61277,61277,6130
9,61309,61309,61309,61309,61309,61309,61277,59196,59164,57051,52858,46518,2113,0,0,0,1
6577,29059,26978,26978,26978,26946,24898,24898,24866,24898,24866,24866,24866,24899,26
978,39522,50017,62528,62528,62496,62496,62496,60416,62720,54497,24866,29190,50548,633
90,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,50581,35562,24866,24866,24866,24866,39681,65056,62528,60416,62432,624
64,60448,60448,60481,62528,62528,56257,45825,35298,24866,24866,24899,24866,24899,2489
8,24898,26946,26978,26979,26978,8288,0,0,0,8452,44438,52825,54971,57083,59164,59228,59
228,61277,61309,61277,61277,61277,59229,59196,59164,57051,54938,50744,16936,0,0,0,2080
,24866,29027,26978,26978,26946,24898,24898,24866,24866,24866,24866,31202,41634,50017,
62561,62529,62528,62528,62496,62496,62496,62464,60448,64928,52384,24867,24866,24898,2
4866,33384,50548,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,54743,29190,24866,24866,24866,24866,29091,24866,26978,60896,64960,60416,62464,624
64,62464,62496,62496,62497,62528,62561,60513,62561,62560,52129,45826,37410,31202,2486
6,24898,26914,26946,26979,26978,22753,2048,0,0,0,16936,48632,52857,54970,57051,59164,5

9196,59197,59197,59196,61277,59196,59164,59131,57051,54938,50744,29614,0,0,0,0,16545,26979,26978,26978,26946,24898,24866,26978,35330,43746,52161,60481,62593,62561,62561,62561,62561,62528,62529,62496,60448,62464,62464,62688,65056,35425,24866,29059,24866,24866,24866,24867,29190,52694,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,39885,24898,24866,24866,29123,37509,43846,43879,29123,24899,39681,65056,64960,60416,60416,62464,62496,62496,60448,60480,62528,62529,60512,62561,62560,62593,62593,62593,62625,52161,47970,43778,37474,35394,31170,12448,0,0,0,0,16936,46486,50777,52890,55002,57083,57083,59164,57116,59131,57083,57051,54970,52857,50711,27469,0,0,0,0,6208,26978,35394,35362,43778,45890,54210,60545,62625,62625,62593,62593,62593,62561,62561,60512,62560,62528,62529,62496,60448,62496,62464,62656,65056,50241,24866,24898,43846,43846,37509,29124,24898,24898,24866,39885,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,39886,24898,24866,29091,41702,45895,45927,45927,45895,41734,24866,24866,39713,65056,65024,62624,62464,60448,62496,60448,60480,60480,62529,62561,62528,62593,62593,62593,62625,62593,62625,62625,62625,62626,62657,62658,58433,20768,0,0,0,0,8452,35921,50712,50809,52858,54938,54970,54970,54970,54938,52857,50744,42293,12743,0,0,0,0,10400,54144,62690,62657,62657,62658,62658,62658,62625,62625,62593,62593,62593,62593,62561,62561,62529,62528,62528,60448,60448,60448,60448,64864,65056,50240,24898,24866,31172,43847,45894,45927,45895,41702,29059,24898,24866,39885,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,48371,24866,24898,33348,45927,45927,45894,45927,45895,39622,39622,24899,24866,24866,37570,62944,65056,64928,62528,62464,60448,60448,62529,62528,62528,62561,62528,62561,62593,62593,62593,62593,62625,62625,62658,62657,62657,62657,58304,20768,0,0,0,0,0,10597,31727,48631,50712,50744,50744,50712,48631,40146,16936,0,0,0,0,0,6208,47872,62657,62657,62657,62657,62625,62625,62657,62625,62593,62593,62593,62593,62561,62561,62560,62529,62529,62528,62496,60448,62720,65024,65056,43937,24867,24866,24866,24898,39621,45927,45927,45894,45927,45894,35365,24898,24866,48402,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63390,27045,24898,33348,43879,45927,45927,43846,35396,26979,31204,27011,24898,31204,26946,24867,27010,45985,65056,65056,64896,62592,62496,62496,62528,62528,60512,62560,62529,62561,62561,62593,62593,60545,62593,62625,62625,62626,62625,62625,56224,18752,0,0,0,0,0,0,0,8452,10597,10597,10597,0,0,0,0,0,0,0,10368,52032,60545,62658,62657,62658,62625,62625,62625,62593,62593,62593,62593,62592,62561,62561,62529,62529,62529,62528,62560,62752,64992,65056,54496,29122,24898,24866,33284,29124,24866,26979,26979,33348,43846,45894,45927,43847,33348,24867,27044,63389,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,48403,24866,29091,41734,45927,45959,43879,29091,31204,37541,39622,29123,24898,39653,37509,29091,24866,24866,29122,45985,62912,65056,64992,62720,62560,62497,62529,62529,62561,60512,62560,62561,62593,62593,62593,62593,62625,62625,62626,62625,62625,56256,33312,4128,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,22880,52064,60545,62657,62626,62625,62625,62625,62625,62625,62593,62593,62593,62561,62561,62561,62561,62528,60480,62656,62848,65056,65056,50241,35425,24866,24898,26979,35428,39622,33316,24866,29091,37542,31203,29091,43814,45895,45927,41734,29059,24866,48403,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,33384,24898,35397,43879,48039,46024,31236,35397,45927,37573,37509,24866,29124,39622,39621,39621,35396,26979,24866,24866,24866,39713,52385,65056,65056,64960,62784,62656,62528,62561,62560,62561,62561,62561,62593,62593,62593,62625,62625,62625,62626,62657,58401,52000,27040,2080,0,0,0,0,0,0,0,0,0,0,0,18752,47840,58336,62625,62657,62626,62625,62625,62625,62593,62593,6

2593,62593,62593,62561,62560,62560,62592,62720,64896,65024,65056,58657,41825,26978,24866,24867,24898,31236,37542,39622,39653,37541,24866,24898,37541,43815,33317,31171,45927,45927,43814,35396,24866,33384,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24899,24866,39622,45960,48104,45992,31204,48104,48104,37542,26979,24866,35396,39653,39654,39622,39622,39621,33317,29091,24866,24898,24866,27010,39681,50240,58752,65056,65056,62880,62784,62657,62592,62560,62561,62593,62593,62593,62593,62625,62593,62625,62625,60545,58336,54112,37440,20768,14528,0,0,0,0,6240,16640,31168,47840,58272,60513,62657,62625,62625,62625,62625,62593,62593,62593,62593,62593,62593,62656,62752,62848,64992,65056,62944,52384,41825,29122,24866,24898,24866,26979,33284,37573,39621,39621,39622,39622,37573,24866,24866,35397,45959,43879,31139,41734,45927,43814,39621,24866,24866,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,61212,24866,27011,41734,45992,48136,45992,39654,50217,48169,45992,29123,24866,39590,39654,43847,43814,39654,39621,39622,39621,37541,31204,26947,24866,24866,24866,24898,29122,39681,48129,54496,65056,65056,65024,64928,62816,62752,62689,62657,60545,62593,62593,62625,62625,62625,62625,60448,58304,56192,56160,56128,54080,56160,56160,56224,58368,62593,62626,62625,62625,62625,62625,62593,62593,62657,62689,62752,62816,64928,64960,65056,65056,56640,50240,41825,31265,24866,24866,24866,24898,24898,29123,35429,39654,39622,39622,39654,43814,43879,41734,39621,29091,24898,33316,45959,45927,35429,41734,45927,43815,39686,29091,24866,59066,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,61244,24866,26979,43847,45992,50217,50249,50250,50250,50250,50217,29124,26979,39621,41702,45927,45927,45927,43879,43847,39654,39621,39622,39622,35461,33316,29091,24866,24899,24898,24866,24866,24866,31265,37569,43937,50240,54496,60800,65056,65056,65056,62912,64960,62912,62816,62817,62816,64864,62816,64865,62816,62817,62817,62816,62816,62816,62816,62816,62880,64960,62912,65024,65056,65056,62944,54496,52353,43936,39681,33378,24867,24866,24866,24866,24866,24866,27010,31204,35429,39622,39653,39621,39654,43814,43879,45959,45959,45959,41734,39622,31236,24866,27011,48040,45959,43879,45927,45895,41766,43847,27011,24866,61212,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24866,24899,46024,39686,50218,50282,50282,50282,50282,48169,24866,31204,39621,41734,45927,45927,45927,45959,45959,45991,45959,43846,41734,39622,39654,39653,39654,35429,33316,29091,26979,24866,24867,24866,24866,24867,24898,24898,24866,24866,35426,35426,35426,43937,45984,43937,43937,43937,43969,48129,54496,54496,50241,43937,43937,43937,43937,43937,45985,37538,35426,33378,27010,24866,24866,24866,24866,24866,24898,24866,24866,24898,29091,31236,35396,37541,39622,39653,39622,41734,43847,45959,45992,45959,45959,45959,45959,45959,43847,39621,35429,24898,24898,43879,45992,45927,45927,45895,39654,46024,24866,24898,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,37707,24866,41832,41766,41767,50250,52363,52363,52330,43944,24866,33348,39622,43814,45927,45927,45927,45959,45959,45959,45991,45991,45992,45992,45959,43879,41766,41734,39654,39622,39653,39654,35429,33348,31236,29091,29059,24898,24866,24898,24866,24898,24866,24866,24866,24898,24866,24866,24866,24866,24898,24866,24866,24866,24866,24866,24866,24866,24866,24898,24866,24867,24866,24899,24866,29059,29059,31236,33316,35428,37573,39622,39622,39622,39654,41767,43847,45960,48072,48072,46024,45992,45992,48040,45991,45959,45959,45927,45927,39654,39654,24866,24866,41735,48040,45959,43847,41701,41799,41832,24867,37707,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,52694,24866,26979,48170,41799,39654,43911,45992,45992,35429,24866,37509,39622,43878,45927,45927,45959,45959,45959,45992,45991,45991,48040,48040,48072,48072,48072,48072,48104,45992,43911,43879,41766,41734,39621,39653,39622,39622,39654,37573,35429,35429,35429,31236,31236,33284

,31236,33284,33284,31204,29091,29091,29091,33284,33284,33284,31236,31236,31236,35396,35429,35429,37541,39654,39621,39622,39654,39622,39654,41767,43878,43911,45992,48104,48104,48105,48072,48104,48072,48072,48072,48040,45992,45992,45992,45960,45959,45959,45959,39654,39622,27011,24898,33348,43847,41766,39654,43847,48170,26979,24866,52727,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,37708,24898,27044,46057,50282,46024,48137,52331,33349,24866,39622,39654,45927,45927,45927,45959,45959,45991,45991,45992,48040,48040,48072,48072,48072,48104,48072,48105,48104,48105,48136,48136,48136,48137,48136,46024,48072,43911,43911,43880,41799,41767,41766,41766,41799,39653,39622,39654,39621,39622,39622,39622,39654,39622,39622,41767,41766,41766,41766,41766,43911,43912,43911,48104,48072,48105,50185,50185,48137,50185,48137,48137,48136,48137,48104,48104,48072,48072,48072,48072,46024,48072,45992,45991,45991,45959,45959,45959,41734,39622,31203,24866,31236,48137,46024,50250,46025,27043,24866,37707,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63390,33384,24898,24866,29092,31237,31236,26979,24866,29091,39622,41734,45927,45927,45959,45959,45959,45991,48040,45991,48040,48072,48072,48072,48072,48104,48104,48104,48105,48105,48137,48137,48137,50185,50217,50217,50217,50217,50217,50217,50217,50218,50249,50249,50250,50250,50249,50250,50250,50218,50250,50250,50249,50250,50250,50217,50249,50249,50250,50249,50249,50217,50249,50217,50217,48169,50217,50217,50217,48137,48137,48137,48105,48137,48105,48104,48104,48072,48072,48072,48072,46024,45992,45991,45991,45960,45959,45959,41766,39621,33316,24867,26979,31236,31236,29091,24866,24866,33384,63390,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,46225,27044,24866,24866,24866,24866,24866,31203,39622,41734,45927,45927,45927,45959,45991,45991,45992,48040,48040,48072,48072,48072,48072,48072,48104,48105,48136,48137,48137,50185,50185,50185,50185,50217,50217,50217,50217,50249,50249,50249,50249,50249,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50249,50249,50249,50249,50217,50217,50217,50217,50217,50185,50185,48137,48137,48137,48136,48105,48104,48104,48072,48072,48072,48072,48072,48040,45992,45991,45991,45959,45959,45959,43846,39622,35429,24866,24866,24866,24866,24866,27012,46225,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,59066,56920,56888,50548,24898,33316,39622,43814,45927,45927,45959,45959,45991,45991,45992,48040,48072,48072,48072,48072,48104,48104,48104,48104,48136,48137,48137,50185,50185,50185,50217,50217,50217,50217,50249,50249,50249,50249,50249,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50250,50249,50249,50249,50249,50249,50249,50217,50217,50217,50217,50217,50185,48137,50185,48137,48136,48136,48105,48104,48104,48072,48072,48072,48072,48072,45992,45992,45991,45959,45959,45927,39622,39654,24898,24867,48402,56920,54872,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,50549,24866,35429,39621,43846,45927,45959,45959,45959,45991,45992,45992,48040,48072,48072,48072,48072,48104,48104,48104,48136,48137,48137,48137,50185,50185,50217,50217,50217,50217,50249,50249,50249,50249,50250,50250,50250,50250,50250,50250,50250,50250,50282,50282,50282,50282,50282,50282,50250,50250,50250,50250,50250,50249,50249,50249,50249,50217,50217,50217,50217,50217,50185,48137,50185,48137,48136,48105,48104,48104,48072,48072,48072,48072,48072,48040,45992,45991,45959,45959,45959,39654,39621,26979,24898,46225,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,44079,24898,39622,39654,45927,45927,45959,45959,45959,459

91,45992,48040,48072,48072,48072,48072,48104,48104,48104,48104,48136,48137,48137,4813
7,50185,50185,50217,50217,50217,50249,50249,50249,50249,50249,50250,50250,50250,50250
,50250,50250,50250,50282,50282,50282,50282,50282,50282,50282,50282,50250,50250,
50250,50250,50250,50249,50249,50249,50249,50217,50217,50217,50217,50185,50185,48137,4
8137,48137,48136,48105,48104,48104,48072,48072,48072,48072,48040,45992,45992,45991,45
959,45959,41734,39654,29123,24866,39885,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,35562,27011,39654,39654,45927,45959,45959,45959,45959,459
91,45992,48040,48072,48072,48072,48072,48104,48104,48104,48136,48137,48137,48137,5018
5,50185,50217,50217,50217,50217,50249,50249,50249,50249,50250,50250,50250,50250,50250
,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,50250,50250,
50250,50250,50250,50250,50249,50249,50249,50249,50217,50217,50217,50217,50185,50185,4
8137,48137,48136,48105,48104,48104,48072,48072,48072,48072,48072,48040,45992,45991,45
959,45959,41766,39621,33284,24866,33384,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,27012,29124,39654,41734,45927,45959,45959,45959,45959,459
92,48040,48040,48072,48072,48072,48072,48104,48104,48104,48136,48137,48137,48137,5018
5,50217,50217,50217,50217,50249,50249,50249,50249,50250,50250,50250,50250,50282,50282
,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,
50282,50250,50250,50250,50250,50249,50249,50249,50249,50217,50217,50217,50217,50185,4
8137,48137,48137,48136,48105,48104,48104,48072,48072,48072,48072,48040,45991,45991,45
991,45959,43847,39622,35397,24866,24866,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,63389,24866,33284,39621,41766,45927,45959,45959,45959,45991,459
92,45992,48040,48072,48072,48072,48104,48104,48105,48136,48136,48137,48137,50185,5021
7,50217,50217,50217,50249,50249,50249,50249,50250,50250,50250,50250,50282,50282,50282
,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,50282,
50282,50282,50250,50250,50250,50250,50249,50249,50249,50249,50217,50217,50217,50185,4
8137,48137,48137,48136,48137,48104,48104,48072,48072,48072,48072,48040,45992,45992,45
991,45959,43879,39622,37509,24866,24898,59066,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,54872,24898,35396,39654,43846,45927,45959,45959,45959,45991,459
92,48040,48072,48072,48072,48072,48104,48104,48105,48136,48137,48137,48137,50185,5021
7,50217,50217,50217,50249,50249,50250,50250,50250,50250,50250,50282,50282,50282,50282
,50282,50282,50282,50282,52362,52362,52362,50282,50282,50282,50282,50282,50282,50282,
50282,50282,50282,50250,50250,50250,50250,50249,50249,50249,50217,50217,50217,50217,5
0185,48137,48137,48137,48137,48105,48104,48104,48072,48072,48072,48072,45992,45992,45
991,45959,45959,39654,39622,24866,24866,52726,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,48403,24898,37509,39622,43879,45959,45959,45959,45991,45991,480
40,48040,48072,48072,48072,29285,16739,8353,0,0,0,0,6273,14659,27205,43944,50249,50249,
50250,50249,50250,50250,50281,50282,50282,50282,50282,50282,50282,50282,50282,5
0282,52362,52362,52362,52362,52362,50282,50282,50282,50314,52330,52330,50282,50282,50
282,50250,50249,50250,50250,50249,48137,31398,16771,8385,0,0,0,0,4193,14659,27204,4394
4,48104,48072,48072,48072,48040,45992,45991,45991,45959,39654,39622,27011,24866
,46224,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,42031,24898,39622,39622,45927,45959,45959,45959,45991,45992,45992,48072,48072,43912,4192,0,0,0,0,0,0,0,0,0,0,18819,43976,50249,50250,50249,50250,50282,50282,50282,50282,50282,50282,50282,50282,52362,52362,52362,52362,52362,52362,52362,52362,52362,50282,50282,52362,50282,50282,50282,50282,50282,50282,50250,50249,46089,25125,2112,0,0,0,0,0,0,0,0,0,0,35558,48104,48072,48072,48072,48072,48040,45991,45991,45959,41734,39622,31204,24866,37708,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,35562,27011,39654,39654,45927,45959,45959,45991,45991,45992,48040,45992,48072,35526,0,0,0,0,8385,10498,12514,12546,0,0,0,0,0,0,25124,50250,50282,50250,50282,50282,50282,50282,50282,50282,50282,52362,52362,52362,52362,52363,52363,52363,52363,52363,52362,52362,52362,52362,50282,52362,50282,50282,50282,50282,50282,50282,37703,2113,0,0,0,0,0,8386,12546,10498,10466,2112,0,0,0,23012,48104,48072,48072,48072,48072,48040,45992,45991,45959,41766,39622,33284,24866,33384,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,27044,29091,39622,41734,45927,45959,45959,45991,45991,45992,48040,48072,48072,48072,20899,14659,33446,48137,48137,48136,48137,50185,50185,39784,20932,2112,0,0,0,18819,50250,50250,50282,50282,52330,50282,50282,50282,52362,52362,52362,52363,52363,52363,52363,52363,52363,52363,52363,52362,52362,52362,50282,52330,52362,50282,50282,50282,52330,31398,0,0,0,0,16803,35526,50249,50217,50185,50185,50185,48137,39751,18787,14626,43944,48104,48072,48072,48072,48072,48040,45992,45991,45959,43846,39622,35396,24866,24867,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63389,24866,33284,39622,41766,45959,45959,45959,45991,45992,45992,48072,48072,48072,48072,48104,48104,48137,48136,48137,48137,50185,50185,50217,50217,50217,46088,20932,0,0,0,20964,50250,50282,52330,50282,50282,50282,52362,52362,52362,52363,52363,52363,52363,52363,52363,52363,52363,52363,52363,52362,52363,50282,52330,50282,50282,50282,33543,0,0,0,8385,43945,50250,50249,50217,50250,50217,48169,48137,50185,50185,48136,48137,48105,48104,48104,48072,48072,48072,48040,45992,45991,45959,43846,39654,37509,24866,24866,57018,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54872,24866,33316,39622,43846,45959,45959,45959,45991,45991,48040,48072,46024,48072,48072,48105,48105,48104,48105,50184,48137,48137,50217,50217,50217,50250,50217,50249,31398,0,0,0,37703,50282,50282,52362,50282,52362,52362,52362,52363,52363,52363,52363,52363,52363,52363,52363,52363,52363,52363,52363,52362,52363,52331,50315,52330,48169,2112,0,0,18851,50250,50250,50249,50250,50249,50217,50217,50217,50185,50185,48137,48136,48104,48137,48104,48104,48072,48072,48072,48072,48040,45992,45959,45959,39621,39654,24898,24898,54872,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,50548,24898,35429,39622,43847,45927,45959,45991,45991,45992,48040,48072,46024,48072,48104,48104,48104,48105,48137,48137,50185,50185,50185,50217,50217,50249,50249,50250,50282,23076,0,0,4224,50282,50282,50314,52362,52362,52363,52363,52363,52363,52363,52363,52363,52363,52363,52363,52363,52363,52363,52362,52363,52363,52362,52330,20964,0,0,12546,50282,50250,50250,50250,50249,50218,48201,50249,50217,50217,50185,50185,48137,48136,48104,48104,48104,48072,48072,48072,48072,48040,45992,45991,45959,39654,39621,29091,24866,44209,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,46257,24866,39622,39621,45927,45959,45959,45991,45991,45992,48040,48072,48072,48072,48104,48104,48104,48104,48137,48137,50185,48169,50217,50217,50217,50217,50250,50249,50250,50250,6273,0,0,37704,52330,52330,52362,52363,52363,52363,52363,52363,52363,52363,52395,52395,52395,52395,52395,52362,52362,52362,52363,52363,50314,52362,50282,2112,0,0,44008,50282,50282,50250,50250,50249,50218,50249,50217,50217,50185,48137,48137,48137,48136,48105,48104,48104,48072,48072,48072,48072,48040,45992,45991,45991,41734,39621,29091,24899,42031,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,37707,24898,39622,39654,45927,45959,45959,45991,45991,45992,46024,48072,48072,48072,48104,48104,48105,48136,48137,48137,50185,50217,50217,50217,50217,50249,50249,50250,50250,50281,33543,0,0,41896,52330,52330,52330,52363,52363,52363,52362,52363,52362,52363,52363,52395,52395,52395,52362,52363,52363,50314,50314,52362,52363,50314,52363,6273,0,18851,50282,50282,50282,50250,50250,50249,50249,50249,50217,50217,50217,50217,50185,48137,48136,48136,48104,48104,48104,48072,48072,48072,48040,45992,45991,45959,41734,39654,33284,24898,35561,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,35561,29091,39622,41734,45959,45959,45959,45991,45991,45992,46024,48072,48072,48072,48104,48104,48105,48136,48137,48137,48137,50217,50217,50217,50249,50249,50249,50250,50249,50282,50282,41864,44009,52362,52362,50314,52363,52363,52363,46057,39654,35429,31237,31236,31236,31237,31237,31236,33349,37606,43912,52362,52363,52362,52363,52363,52363,48170,37704,52362,50282,50282,50282,50250,50250,50250,50249,50249,50217,50217,50217,50185,50185,48137,48137,48136,48105,48104,48104,48072,48072,48072,48040,45992,45991,45959,43814,39621,33316,24866,29190,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24867,31171,39654,41734,45959,45959,45959,45991,45991,48040,46024,48072,48072,48072,48104,48104,48104,48136,48137,48137,48137,50217,50217,50217,50249,50249,50249,50250,50250,50282,50282,52330,50282,50282,50314,52330,52363,39687,27043,24866,24898,24866,29026,31139,31139,31138,31138,29059,24866,24866,24866,26979,35430,50282,52362,52362,52362,52330,52362,52330,50282,50282,50282,50282,50250,50250,50249,50249,50217,50217,50217,50185,50185,48137,48137,48136,48105,48104,48104,48072,48072,48072,48040,48040,45992,45991,43847,39621,37509,24866,24898,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63389,24866,31236,39622,41766,45959,45959,45959,45991,45992,48040,46024,48072,48072,48072,48104,48104,48136,48137,48137,48137,50185,50217,50217,50217,50249,50249,50249,50250,50282,52330,50282,52330,50314,52330,52330,52363,35429,24866,26946,35267,41539,52135,60881,63290,65404,65404,65371,60979,56426,45700,39427,29059,24866,29124,50250,50315,50314,52362,50314,52362,50282,50282,50282,50282,50250,50250,50249,50249,50249,50217,50217,50217,50185,48137,48137,48136,48105,48104,48104,48072,48072,48072,48072,48040,45992,45991,43879,39654,37541,24898,24866,57018,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54873,24866,33348,39622,41766,45959,45959,45959,45991,45992,48040,46024,48072,48072,48072,48104,48104,48136,48137,48137,48137,50185,50217,50217,50217,50249,50249,50250,50250,50282,52330,50282,52330,50314,50315,52363,41832,24866,31107,41476,45669,47780,52103,60849,65338,65404,65404,65339,60980,54313,49860,47748,43589,35236,24866,35494,52363,52362,50314,52331,50314,50282,50282,50282,50282,50250,50250,50250,50249,50249,50217,50217,50217,50185,48137,48137,48137,48105,48104,48104,48072,48072,48072,48072,45992,48040,45991,45959,39654,39622,24866,24866,56920,65535,65535,65535,65535,655

35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54742,24866,35429,39654,43847,45959,45959,45959,45991,45992,48040,46024,48072,48072,48104,48104,48104,48136,48137,48137,50185,50185,50217,50217,50249,50249,50249,50250,50250,50281,50250,52330,50282,52330,52396,56851,44078,24866,37316,43525,43556,45636,47780,49859,49891,51971,51971,51939,49859,47780,45668,43588,41509,43525,24898,33448,61144,54509,52363,52362,52330,50282,50282,50282,50282,50250,50250,50249,50249,50217,50217,50217,50185,48137,48137,48137,48105,48104,48104,48072,48072,48072,46024,45992,45992,45991,45959,39622,39621,27011,24898,46355,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,46225,24866,37541,39622,43879,45959,45959,45959,45991,45992,48040,48072,48072,48072,48104,48104,48105,48137,48137,48137,50185,50185,50217,50217,50249,50249,50249,50250,50250,50282,50250,52330,52330,54608,65437,65535,48468,24866,39428,41477,43557,41509,43556,43588,45636,45669,45668,45668,45636,43589,43557,43525,43525,43588,24898,37805,65535,65502,58965,52330,52363,50282,50282,50282,50282,50282,50250,50250,50249,50249,50217,50217,50217,50185,48137,48137,48137,48137,48105,48104,48072,48072,48072,46024,45992,48040,45959,45959,39654,39654,29091,24866,46225,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,42031,24866,39622,39622,45927,45959,45959,45959,45991,45992,48072,48072,48072,48072,48104,48104,48105,48137,48137,48137,50185,50185,50217,50217,50249,50249,50249,50250,50250,50282,50282,50282,54673,65535,65535,65535,59197,27045,33218,45700,43525,43525,43525,43557,43525,43525,41477,41477,43525,43525,43557,43525,43588,41539,24866,50646,65535,65535,65535,61112,52362,52362,50282,50282,50282,50282,50250,50250,50250,50249,50249,50217,50217,50217,50185,48137,48137,48137,48105,48104,48104,48072,48072,46024,45992,45991,45991,45959,41702,39621,31204,24898,37707,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,35562,24898,39654,39654,45927,45959,45959,45991,45992,45992,48040,48072,48072,39686,43912,48104,48105,48137,48137,50185,50185,50217,50217,50217,50249,50249,50250,50250,50282,52329,50282,50282,65470,65535,65535,65535,65535,40114,24866,35298,45731,43589,43557,43525,43525,43525,43525,41508,43525,41477,43557,45700,41570,24866,33547,61342,65535,65535,65535,65535,54574,52362,52362,50282,50282,50282,50250,50282,50249,50249,50250,50217,50185,50217,48137,48137,50185,48072,39654,48105,48104,48072,48072,48072,48040,45992,45991,45959,41766,39622,31236,24898,35562,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,35561,29091,39622,41702,45959,45959,45960,45991,45991,45992,48039,48072,39622,24866,37573,48104,48104,48137,50185,48136,48137,50217,50217,50217,50249,50249,50250,54542,56754,58998,61210,61210,65503,65503,65535,65535,65503,61277,37936,24931,29058,43682,47780,43557,41477,41477,43525,41477,43556,45700,45762,33250,24866,33547,54938,65535,65535,65535,65535,65535,63291,63258,59031,56819,54607,50282,50250,50249,50250,50249,50217,48170,50217,48169,50185,48137,48137,43879,24866,33316,48104,48072,48072,48072,48040,45992,45991,45959,41766,39622,33316,24898,29190,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24899,29091,39654,41702,45927,45959,45959,45991,48039,45992,48040,41766,24866,37509,48104,48104,48136,48137,48137,50185,48169,50217,50217,50217,54575,59031,63325,65503,65503,65503,65503,65503,65503,65535,65503,65503,65503,65503,61277,40114,29223,24866,31138,43682,47844,45700,45699,47811,47811,33251,24898,27045,37903,54938,63455,65503,65503,65503,65503,65503,65503,65503,65503,65503,65503,65503,59097,56721,50250,5

0250,50217,50217,50217,50217,50185,48137,50185,48136,43879,26978,35461,48072,48072,48
072,48039,45992,45991,45959,43847,39622,37509,24866,24867,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,24866,31236,39622,41734,45927,45959,45959,45991,45992,45992,48071,31172,312
04,48104,48072,48105,48104,48136,50185,48137,50217,50217,54608,61211,63423,65502,6550
3,63455,65471,65503,65471,65503,65470,65502,65503,65503,63422,65503,63422,63390,46551
,35855,29190,24866,26978,35331,37410,29059,24866,27045,33677,44373,59164,63455,65502,
65502,65503,65502,65503,63422,65503,63422,65503,65503,65503,63422,65502,65502,61276,5
6852,50217,48169,48202,48137,50185,50185,48105,48105,48105,37542,26978,45992,48072,48
072,48072,45992,45991,45959,43879,39622,35428,24866,24866,63389,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,61212,24898,33284,39622,43814,45959,45959,45959,45991,45992,48072,46023,24866,396
86,48072,48104,48104,48136,48137,48105,48137,50315,61145,63422,63422,63422,63422,6342
2,63422,63422,63422,63423,63423,63422,63422,63423,63422,65470,63422,63422,63422,63422
,59164,46518,38033,31434,27109,27045,31401,35790,44373,54970,65470,63422,63422,63423,
63422,63422,63422,63422,63422,63422,65470,63422,63422,63422,65470,63422,63422,63423,6
3422,63323,52494,50217,50217,48169,50184,48137,48105,48136,46024,24866,39686,48072,48
072,45992,45992,45991,45991,43879,39622,39654,24866,24866,54872,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,56920,24866,33316,39654,41766,45927,45959,45959,45991,45992,48039,41734,24898,438
47,48072,48104,48104,48105,48136,50153,52462,63357,63423,63390,63422,63390,63422,6342
2,63422,63422,63422,63422,63422,63422,63422,63422,63422,63422,63422,63422,63422,63422
,63391,63422,61276,50744,44405,44373,48632,57083,63422,63422,63390,63422,63422,63390,
63390,63422,63422,63422,63423,63422,63422,63422,63422,63422,63422,63422,63422,63422,6
3422,63422,63422,56787,48169,50185,48137,48137,48105,48104,48104,31204,35461,48072,46
024,48072,45992,48040,45991,45927,39654,39654,24866,24866,56920,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,56920,24866,35429,39621,43847,45959,45959,45959,45959,45992,45992,48072,24866,417
99,48072,48105,48104,48104,48137,52462,63389,63390,63389,63390,63390,63390,63390,6339
0,63390,63390,63390,63390,63390,63389,63358,63389,63390,63390,63390,63390,63390,63390
,63389,63390,63389,63390,54873,50581,63389,63390,63390,63389,63390,63390,63389,63390,
63390,63390,63390,63390,63358,63390,63390,63390,63389,63390,63389,63390,63390,63389,6
3390,61342,63390,63389,56853,50184,50185,48137,48137,48105,48104,24931,39622,48072,48
072,48040,45992,45991,45959,45959,39621,39621,26946,24866,50548,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,48403,24898,35429,39621,43846,45927,45959,45991,45991,45991,48040,48040,26979,375
74,48072,48104,48104,48105,50283,61276,61309,61310,61309,61309,61309,61309,61309,6130
9,61309,61309,61309,61309,61309,61309,61309,61310,61309,61309,61309,61310,61309,61309
,61309,61310,61310,63422,44177,35561,65503,63390,61310,61309,61309,61310,61309,61309,
61309,61309,61309,61310,61310,61309,61309,61309,61310,61309,61309,61309,61309,61310,6
1309,61309,61310,61309,61309,52495,48137,48137,48136,48104,45992,24866,43847,48072,48
072,45992,45991,48040,45959,45959,39621,39622,29092,24866,44177,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,46225,24866,37509,39621,43846,45927,45959,45991,45992,48040,45992,48072,33284,249
31,50250,48104,48104,48104,56854,61309,61309,61309,61309,61309,61309,61309,61309,6130
9,61309,61309,61309,61309,61309,61309,61309,61309,61309,61309,61309,61277,61309,61309

,61309,63455,63389,46225,27044,24866,39885,59066,65503,63422,61277,61309,61309,61309,
61309,61309,61309,61277,61309,61309,61309,61309,61309,61277,61309,61309,61309,61277,6
1277,61309,61309,61277,61309,61243,50250,48137,48105,50249,33349,29091,45992,48072,48
072,48040,45991,45959,45959,45959,39622,39622,29059,24866,46257,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,46225,24866,39622,39622,43846,45927,45959,45959,45991,45991,45992,46024,41734,269
79,29156,50282,50250,50283,61277,61277,59228,59229,59229,61277,61277,59196,59229,5922
9,59229,59229,59229,61277,59229,61277,61277,59196,59197,61245,59196,61309,63390,65535
,59066,39885,24898,29222,37903,38033,31369,24866,35562,50549,65535,63422,61310,59228,
59229,61245,59196,59229,59229,59229,61277,61277,61277,61277,61277,61277,61277,61277,5
9228,61277,59228,59229,59196,61277,52528,50217,52362,35462,24898,37509,48072,48072,46
024,46024,45992,45959,45992,45959,41734,39622,29091,24866,39853,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,39885,24866,39622,39622,45927,45927,45959,45959,45959,45992,48008,48040,45991,354
61,26979,26979,41832,56787,63390,61309,59196,59164,59196,59196,59196,59196,59196,5916
4,59164,59196,59196,59196,59196,59196,59196,61277,61309,63422,65503,57017,48403,29190
,24898,31369,35855,42292,52857,57083,44373,38001,31466,24931,27044,39885,52694,65535,
63422,63358,61309,59164,59164,59196,59196,59196,59196,59196,59196,59196,59196,59196,5
9164,59196,59196,59196,61276,61341,61144,46025,31236,24898,33349,43879,48104,48072,48
071,48040,45992,45991,45959,45959,41734,39622,33317,24866,35562,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,35562,24866,39622,39622,45927,45927,45959,45959,45991,45992,48039,45992,46024,459
27,37542,31172,24866,29190,42031,54872,65535,65503,63422,63422,61341,63390,61342,6339
0,63390,63389,63390,63390,63454,65503,65535,61212,50548,39885,29190,24866,29255,33677
,40114,44373,55003,63423,65535,65535,65503,59196,46518,42260,35790,29353,24931,27044,
37707,48371,59066,65535,65503,63422,63390,63390,63390,63390,63390,63390,61309,61309,6
1342,63422,65470,65535,59066,48402,33384,24867,27043,37509,41767,48104,48072,48072,46
024,48040,45992,45991,45959,45959,41734,39654,33284,24866,35562,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,35562,27011,39622,39654,45927,45927,45959,45959,45959,45991,45992,48040,48072,480
72,45959,39653,37705,31466,27109,24866,24866,31368,35562,42031,46257,46225,46225,4622
5,46225,46225,46257,46225,37708,35561,24866,24866,24867,29223,25159,35921,42260,46583
,57083,63422,65503,65535,65535,65535,65535,65535,63423,59196,48632,42292,42259,23079,
31336,24931,24866,24866,31336,35562,46257,46225,46225,46225,46225,46225,46225,46225,4
6257,35561,35562,24899,24866,24931,31401,37772,39655,43879,48104,48104,48072,48072,48
072,48040,45991,45992,45959,45959,41734,39622,33284,24866,35561,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,33545,29059,39622,39621,45927,45959,45959,45959,45960,41734,26978,43846,48072,481
04,48105,50514,48599,44340,42259,40146,37903,33547,33547,29223,29223,29222,29222,2922
2,29222,29222,29222,29223,31401,33547,33709,37968,42259,42260,31663,38034,54970,59197
,61309,63422,65503,65535,65535,65535,65535,65503,63422,61309,59229,57051,44405,23211,
44406,42228,38033,35855,33547,31434,29223,29223,29223,29223,29223,29223,29223,29223,2
9223,31401,33547,35790,40081,40179,40212,46485,48565,48202,48072,48104,45992,33284,31
236,48040,45992,45959,45959,45959,41767,39622,33284,24866,31367,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,35561,29059,39622,39654,45927,45927,45959,45959,45959,45927,29091,24931,43847,481

05,50613,50777,52825,52857,52825,50712,46519,44406,42292,40211,40212,40180,42260,4017
9,42260,40179,42260,40211,44373,46518,50712,54938,59132,61341,42260,35921,54970,59164
,61309,63390,63423,65503,65535,65535,65503,65503,63422,61310,59196,54970,44405,29582,
63357,59261,54970,50744,46518,44405,42260,42260,42260,40179,40179,42260,42260,42260,4
0212,42260,44406,46486,48631,50744,52825,52825,50745,50679,48235,45992,29091,24931,43
847,45991,45992,45992,45959,45959,41766,39622,37509,24866,24898,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,24867,29091,39621,39654,45927,45927,45959,45959,45959,45959,45960,29091,26978,441
76,50745,52824,52857,52889,54938,54970,55002,57051,57083,57084,59164,59164,59196,5922
8,59228,61277,61277,61309,61309,61309,61309,63390,63390,63390,42292,29582,52857,57083
,61277,61310,63422,63454,65503,65503,63455,63422,61309,61277,57084,52890,38002,31727,
63390,63390,63390,61309,61309,61309,61309,61309,61277,59229,59196,59196,59164,57115,5
7083,57083,57050,54970,54970,54938,52857,52825,50777,50744,48533,29092,24899,45927,48
104,46023,45992,45991,45959,45959,41766,39622,37509,24866,24866,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,24866,29091,39622,39654,45927,45927,45959,45959,45959,45992,46023,46024,33316,248
66,42064,52825,52858,54938,54971,57051,55003,57083,57083,59164,57116,59196,59196,5922
9,61277,61309,61309,61309,61309,63390,63390,63390,63390,63390,52825,25356,50744,54970
,59164,61277,61309,63390,63390,63390,63390,61309,61277,59164,54970,52825,31727,40147,
63390,63390,63390,63390,63390,61309,61309,61309,61309,61277,61276,59196,59196,59164,5
7116,57083,57083,57051,54970,54970,54938,52857,52857,46420,27077,29123,45992,48137,48
104,48072,45992,45959,45959,45959,41766,39622,37509,24866,24866,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,24866,29059,39654,39621,43847,45927,45927,45959,45959,45991,48072,48104,48137,333
48,24866,37805,52890,54970,54970,57051,57083,57083,59132,59164,59196,59228,61245,6127
7,61277,61309,61309,61341,63390,63390,63390,63390,63390,63422,59196,21162,44405,52825
,54970,57115,59196,61277,61309,61309,59229,59196,57083,54970,52825,48631,27469,46486,
63390,63422,63390,63390,63390,63390,61341,61309,61309,61309,61277,61245,59196,59196,5
9164,59164,57083,57051,57051,54970,54938,54906,44242,27012,29124,48137,50217,48169,48
104,48072,45992,45992,45959,45959,41766,39621,37509,24866,24866,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,24866,29091,39622,39622,41702,45927,45927,45959,45959,48072,48104,48136,48137,502
49,39719,24866,33481,52824,55003,57051,59131,57116,59164,59196,59197,59196,61277,6127
7,61309,61309,61341,61342,63390,63390,63390,63422,63422,63422,63422,27469,35953,48599
,50744,52857,54970,57051,57083,57083,57051,54970,54938,50744,48599,44373,21162,57083,
63422,63422,63390,63390,63390,63390,63390,61342,61309,61309,61309,61277,61245,59196,5
9196,57116,59164,57083,57019,57019,54970,39918,24898,35430,50249,50250,50250,48137,48
136,48072,45992,45959,45959,45959,41734,39654,35429,24866,24866,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,24866,31236,39622,39622,39622,41766,45959,45959,45992,48072,48104,48136,50185,502
49,50282,43912,26979,29158,50647,57083,57115,59164,59196,59197,61276,61276,61309,6130
9,61309,61342,63390,63390,63390,63422,63422,63422,63422,63422,63422,42292,23275,38034
,44405,48631,50744,52825,52857,52857,52825,50744,48631,46486,40147,27469,29582,63422,
63422,63422,63422,63390,63422,63390,63390,63390,61310,61309,61309,61309,59229,61245,5
9196,59164,57116,57083,57083,52857,33514,24866,39719,50315,50282,50282,50218,50185,48
136,48104,48072,45959,45959,43846,39621,39621,35429,24866,24866,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24866,31236,39622,39622,39622,39621,43815,45959,45992,48072,48104,48137,50217,50250,50250,50282,48202,31236,24866,41999,57116,59164,59196,52791,61276,61277,61309,61309,63390,63390,63390,63422,63422,63422,63422,63422,63422,65503,65503,59164,33840,25388,21162,21162,25356,29582,29582,29582,29549,27469,23210,21162,25324,29614,50744,65503,65503,63422,63422,63422,63422,63390,63390,63390,63390,61309,61309,61277,61309,61276,54937,54905,59196,57115,48533,27012,26979,46025,52395,52363,52330,50282,50250,50217,50152,48104,48039,45992,43879,39653,39622,39622,35429,24866,24866,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24866,31236,39621,39654,39622,39622,39622,45927,48039,48104,48136,48137,50217,50249,50282,52330,52362,52395,39655,24866,31303,37740,24866,24866,31303,59228,61309,63389,63358,63390,63422,63422,63422,63422,63422,63422,65503,65503,65503,65503,57083,46486,40179,35953,35953,35953,35953,35953,35953,35953,35953,40147,44373,52857,65503,65503,65503,65503,63422,63422,63422,63422,63422,63390,63358,63390,61310,61309,61277,42031,24866,24866,33449,37740,24866,33349,50314,52427,52395,52395,52331,52330,50250,50217,48137,48104,48072,45959,39653,39654,39622,39622,35429,24866,24866,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24866,31236,39654,39622,39654,39621,39621,39622,45959,48104,48137,48169,50249,50250,50282,52362,52363,52395,52395,48138,39755,39885,61244,63389,29190,39984,61342,63390,63358,63390,63422,63422,63422,63422,65503,65503,65503,65503,65503,65503,65503,65503,65535,59196,59196,57083,52857,52857,54938,59196,59196,63422,65503,65503,65503,65503,65503,65503,65503,63422,63422,63422,63422,63389,63390,63390,61342,61309,54938,24866,59066,63389,46224,37707,43912,52427,52427,52395,52395,52395,52362,50282,50282,50217,50185,48105,46024,41734,39654,39621,39622,39622,35429,24899,24866,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24866,29091,39622,39622,39622,39622,39622,39621,39622,45960,48137,50217,50250,50282,50282,52330,52363,52395,52395,52395,54607,65535,65535,63390,29190,35725,63389,63390,63390,63422,63422,63422,63422,63422,65503,65503,65503,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,65503,65503,65503,65503,65503,63422,63422,63422,63422,63390,63390,63390,61309,50745,24866,59066,65535,65535,58997,54475,52427,52395,52395,52395,52395,52363,50282,50250,50249,50217,48072,39654,39622,39654,39621,39654,39621,35429,24866,24866,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24866,29091,39654,39653,37541,37541,39622,39654,39653,39653,45960,50217,50250,50282,52362,52363,52363,52395,52395,52427,52427,61013,65535,39885,24932,42260,63390,63389,63390,63422,63422,63422,63422,65503,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,65535,65503,65503,65503,65503,65503,65503,65503,63423,63454,63390,63390,63389,61342,52825,29288,29190,65535,63290,52427,52427,52427,54475,52395,52395,52395,52363,52362,50282,50250,48072,39654,39622,39621,39621,37509,39622,39622,35429,24898,24866,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,24866,29091,39621,39622,33251,27011,37573,39622,39622,39621,39622,43879,50250,52330,52362,52362,52395,52395,52427,52395,52427,54509,65502,63390,27012,37805,61342,61342,63390,63422,63422,63422,65503,65503,65503,65503,65503,65535,65503,65503,65503,65503,65503,65503,63422,63422,63423,63422,63390,63390,61342,50646,24866,50549,65535,56720,52395,52427,52396,52395,54475,52395,52394,52363,50315,50282,45959,39621,39622,39622,39622,29155,29059,39654,39621,37508,24866,24866,65535,65535,65535,655

35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,27012,29091,39621,39622,39590,29123,27011,37541,39654,39622,39654,39622,41766,481
69,52362,52395,52395,52395,52395,52427,52427,52427,61112,65535,59066,24866,46419,6339
0,63390,63422,63422,63422,65503,65503,65503,65503,65503,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65503,65503,65503,65503,65503,63422,63422,63422,63390,63390,59163,27012,48403,6
5535,65437,52427,54443,52427,52395,52427,52427,54443,52395,52395,50250,41767,39622,39
621,39622,39622,29091,26979,37509,39621,39654,37509,24866,24866,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,35561,29091,39622,39621,39621,39622,29124,26979,41799,43847,39621,39621,39621,396
22,45992,52395,52395,52395,52395,52427,52427,52395,58965,65535,65535,44177,27045,5708
3,63422,63422,63422,63422,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65503,65503,65503,65503,63454,63422,63422,63390,63390,37740,35562,65535,6
5535,63290,52427,52428,52395,52427,54443,52427,52395,52363,45992,39654,39654,39622,41
734,43879,29124,27011,37541,39622,39622,39622,35429,24867,24899,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,35561,29091,39654,39621,39622,39621,39654,35725,24931,33349,48137,43847,39621,396
54,39622,41767,50249,52395,52427,52427,52395,52395,31303,56920,65535,54872,24898,5074
4,63422,63423,63422,65503,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65503,65503,65503,65503,65471,65471,63422,63422,59164,29222,46225,65535,6
1243,35562,41832,52427,54475,52427,52395,52427,48137,41734,39653,39622,41734,48137,39
655,24866,31434,39785,39621,39622,39622,39621,31236,24866,33383,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,35561,24898,39622,39622,39654,39622,39655,42260,38001,29222,26979,39719,48202,417
99,39654,39621,39621,43912,50282,52428,52427,52427,59000,39885,31336,39853,24866,4863
1,63422,63423,63422,65503,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65503,65503,65503,65503,63455,63423,63422,63422,57083,29223,35562,37707,3
1336,56921,54542,52427,52427,50347,48137,41734,39622,39622,39686,46024,45993,29124,27
044,37903,42259,39851,39621,39654,39621,39654,33284,24898,35561,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,37707,24867,39622,39622,41734,41734,39753,42259,42260,40179,35726,27044,29091,418
32,48302,43879,39654,39621,39654,46024,50282,54674,65535,65535,63390,39885,27109,5282
5,63422,63422,63422,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65503,65503,65503,65503,65502,63423,63423,63422,59164,33579,29190,61244,6
5535,65535,61178,48137,45992,39654,39622,39622,41799,48204,46058,31268,24931,33579,40
146,40212,40211,39917,39654,41766,39622,39653,33316,24867,35562,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,46257,24866,37541,39622,41734,45927,43978,42260,42228,40211,40179,42260,35725,270
44,29125,37805,48434,46255,44110,39917,37968,31336,37707,50548,56920,27044,35725,5916
4,63422,63422,63422,65503,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65503,65503,65503,65503,63455,65470,63423,63390,63390,42227,24866,50549,5

2726,42031,31303,35789,39982,41997,44175,48434,42031,29223,24867,31401,38033,40180,42
259,40212,40212,42029,45927,43846,39621,39622,29059,24866,41901,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,46225,24866,35428,39622,41702,45927,46090,48632,46518,42292,42228,42260,40211,402
12,35725,29223,24866,29222,39951,46420,48631,48631,46421,35659,24866,29223,46518,6342
2,63422,63423,63422,63422,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65503,65503,65503,65503,63455,63422,63422,63422,63390,54938,35725,24866,3
1303,44210,48664,48631,48533,42064,33481,24898,27045,33579,40114,40212,40212,40211,40
211,44405,48599,46223,45927,41766,39622,39622,29123,24866,46225,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,52694,24866,35429,39622,39654,45927,46059,48632,50712,50744,48599,44406,42260,401
80,40212,42260,40050,33644,29223,24866,24866,29222,31303,24866,31466,40146,57083,6339
0,63423,63422,63422,63422,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65503,65503,65503,65503,63422,63422,63422,63422,63390,61342,48631,33677,2
4931,29158,31336,24866,24866,27109,31434,37968,42228,40179,42260,42227,42292,46518,48
632,50680,48631,46255,45927,41734,39621,39621,27011,24898,48371,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,54872,24898,31236,39622,39622,45927,46059,48599,50711,50744,52825,52857,52825,486
31,44405,40179,42259,40212,40212,42227,35855,37903,35855,40146,42260,55002,63390,6342
2,63390,63422,63422,63422,65503,65503,65503,65503,65503,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65503,65503,65503,65503,65503,63423,63422,63422,63390,63390,61342,61310,52825,4
4372,37904,35855,37903,40146,40212,40179,40211,42260,44437,48631,50744,52825,50777,50
744,50711,48631,46255,45894,39654,39654,39654,24866,24898,54872,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,63389,24866,31204,39622,39654,43814,46059,48632,50711,50744,50777,52857,54938,549
38,54970,54970,52857,48632,46486,46486,46486,46486,46518,52858,61276,61309,63390,6134
2,63390,63422,63422,63422,63422,65503,65503,65503,65503,65503,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65503,65503,65503,65503,65503,63422,63423,63422,63422,63390,63390,63390,61309,61309,6
1309,57083,52889,50712,50712,50744,52857,54938,57018,54970,54938,52858,52857,50744,50
712,48631,48599,46222,45927,39654,39621,37509,24866,24866,57018,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,24898,29091,39621,39622,41766,46059,48631,50679,50744,50744,52825,52857,549
38,54970,57051,57051,57051,59164,59164,59196,61277,61277,61309,61309,61309,63390,6339
0,63390,63422,63422,63422,63422,65503,65503,65503,65503,65503,65503,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,
65503,65503,65503,65503,63422,63422,63422,63422,63422,63390,63390,61310,61309,61309,6
1277,61277,59197,59196,59164,59164,57083,57051,54971,54970,54938,52857,52825,50744,50
712,48663,48599,46222,43814,39654,39622,35429,24898,24899,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,35562,24898,39622,39654,39654,46058,48631,48631,50712,50744,52825,52857,549
38,54971,54971,57051,57083,57083,59164,59196,59196,61277,61277,61309,61309,61341,6339
0,63390,63422,63422,63422,63422,63422,65503,65503,65503,65503,65503,65503,65535,65535

```
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,65503,
65503,65503,65503,65503,63422,63422,63422,63422,63422,63390,63390,61310,61309,61309,6
1277,59229,59196,59164,59164,57083,57051,55003,54970,54938,52889,52857,52825,50712,50
712,48631,48599,46223,43814,39621,39622,33284,24898,33384,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,42031,24866,37541,39621,39654,46025,48599,48631,50712,50744,52825,52857,528
57,54938,54970,57051,57051,57084,59164,59164,59196,59197,61277,61309,61309,61341,6134
2,63390,63390,63422,63422,63422,63422,63422,65503,65503,65503,65503,65503,65503,65503
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,65503,65503,65503,
65503,65503,65503,63422,63422,63422,63390,63390,63390,63389,61341,61309,61309,61277,6
1277,59197,59196,59164,57116,57083,57083,54970,54970,54938,52858,52825,50744,50744,48
664,48631,48598,46190,41734,39621,39622,29123,24866,37708,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,
};

//15 different main color
const int xcolor[] =
{0,65535,63488,64512,65504,34784,2016,2032,2047,1055,31,30751,63519,63503,33808};

//clock image
const int clock1[] = {

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,63390,57051,33808,8484,0,0,23275,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,23243,0,0,8484,33808,57051,63390,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63
390,38066,8484,0,0,0,0,0,0,6371,61309,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,61309,6371,0,0,0,0,0,0,8484,38066,63390,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,42260,12
678,0,0,0,0,0,0,0,0,0,16904,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,16904,0,0,0,0,0,0,0,0,0,12678,42260,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,8452,0,0,0,0,0,0,
0,0,0,0,0,0,31727,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,31727,0,0,
0,0,0,0,0,0,0,0,0,0,8452,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,46518,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,50744,65535,65535,65535,65535,65535,65535,65535,65535,50744,0,0,0,0,0,0,0,0,0,0,0,
0,0,46518,65535,65535,65535,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,29614,0,0,0,0,0,0,0,0,0,0,0,0,2113,21
130,29582,38034,48631,65535,65535,65535,65535,65535,65535,65535,65535,48631,38034,295
82,21130,2113,0,0,0,0,0,0,0,0,0,0,0,0,0,29614,65535,65535,65535,65535,65535,65535,65535,655
35,

    65535,65535,65535,65535,65535,65535,65535,31727,0,0,0,0,0,0,0,0,0,0,0,8452,44373,549
38,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,54938,44373,8452,0,0,0,0,0,0,0,0,0,0,0,33808,65535,65535,65535,65535,65535,
65535,65535,
```

65535,65535,65535,65535,65535,65535,52857,0,0,0,0,0,0,0,0,0,8452,61277,65535,65535
,65535,65535,65535,61277,38034,23243,10565,2113,0,0,2113,10565,23243,38034,61277,65535
,65535,65535,65535,65535,61277,8452,0,0,0,0,0,0,0,0,0,0,52857,65535,65535,65535,65535,655
35,65535,

65535,65535,65535,65535,65535,65535,12710,0,0,0,0,0,0,0,0,12710,48631,65535,65535,6
5535,57051,46518,19017,0,0,0,0,0,0,0,0,0,0,0,0,0,0,19017,46518,57051,65535,65535,65535,486
31,12710,0,0,0,0,0,0,0,0,0,12710,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,59164,8484,0,0,0,0,0,0,0,35953,65535,65535,65535,5282
5,25388,6371,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6371,25388,52825,65535,65535,65535,35953,0,
0,0,0,0,0,0,8484,59164,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,63422,12678,0,0,0,0,0,6371,50712,65535,65535,65535,2
1130,2145,0,0,0,0,0,0,0,0,0,6371,10597,10597,6371,0,0,0,0,0,0,0,0,0,2145,21130,65535,65535,6
5535,50712,6371,0,0,0,0,0,12678,63422,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,23275,0,0,0,0,16904,65535,65535,65535,44405,0,
0,0,0,0,0,2113,8484,33840,61309,65535,65535,65535,65535,65535,65535,65535,65535,61309,3
3840,8484,2113,0,0,0,0,0,0,44405,65535,65535,65535,16904,0,0,0,0,23275,65535,65535,65535,
65535,65535,65535,

65535,65535,65535,65535,65535,65535,61277,0,0,0,12710,65535,65535,63390,29614,0,0,
0,0,0,12678,27501,63390,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,63390,27501,12678,0,0,0,0,0,29614,63390,65535,65535,12710,0,0,0,61
277,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,8452,0,16936,65535,65535,59164,16904,0,
0,0,0,2113,40147,61309,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,61309,40147,2113,0,0,0,0,16904,59164,6553
5,65535,16936,0,8452,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,42260,6339,65535,65535,63422,6339,0,0,0,
0,19049,63390,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,23243,23243
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63390,19049,0,0,0,0,6339,6
3422,65535,65535,6339,42260,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,63422,65535,65535,0,0,0,0,0,46486,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,14823,0,0,14823,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,46486,0,0,0,0,0,65535,65
535,63422,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,63422,10597,0,0,0,2145,4863
1,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,2145,0,0,214
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,48631,2145,
0,0,0,10597,63422,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,29614,0,0,0,0,52825,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,52825,0,0,
0,0,29614,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,48631,0,0,0,0,46518,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,465
18,0,0,0,0,48631,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,63422,0,0,0,0,42260,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,42260,0,0,0,0,63422,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,23275,0,0,0,27469,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,27469,0,0,0,23275,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,61309,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,61309,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,19017,0,0,0,48599,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,48599,0,0,0,19017,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,54970,6371,0,0,12678,63422,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63422,12678,0,0,6371,54970,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,40147,0,0,0,40147,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,40147,0,0,0,40147,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,23243,0,0,8452,57051,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,57051,6371,0,0,23243,65535,65535,65535,65535,65535,

65535,65535,65535,65535,59196,0,0,0,21130,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,21130,0,0,0,59196,65535,65535,65535,65535,

65535,65535,65535,65535,54970,0,0,0,38066,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,38066,0,0,0,54970,65535,65535,65535,65535,

65535,65535,65535,65535,40179,0,0,0,61277,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61277,0,0,0,40179,65535,65535,65535,65535,

65535,65535,65535,65535,25388,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,4258,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,25388,65535,65535,65535,65535,

65535,65535,65535,65535,12710,0,0,6339,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6339,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,6339,0,0,12710,65535,65535,65535,65535,

65535,65535,65535,65535,6339,0,0,19017,65535,65535,65535,65535,48599,44405,46486,46486,46486,46486,46486,46486,46486,46486,46486,46486,46486,44405,50744,4226,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,19017,0,0,6339,65535,65535,65535,65535,

65535,65535,65535,65535,32,0,0,25356,65535,65535,65535,29614,0,0,0,0,0,0,0,0,0,0,0,0,0,32,0,0,0,4258,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,25356,0,0,32,65535,65535,65535,65535,

65535,65535,65535,65535,0,0,0,27501,65535,65535,65535,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2113,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,27501,0,0,0,65535,65535,65535,65535,

65535,65535,65535,65535,0,0,0,25388,65535,65535,65535,21162,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,31727,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,25388,0,0,32,65535,65535,65535,65535,

65535,65535,65535,65535,4258,0,0,19049,65535,65535,65535,65535,38034,33840,33840,33840,33840,33840,33840,33840,33840,33840,33840,33840,33840,33840,33840,33840,33840,33808,40147,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,19049,0,0,4258,65535,65535,65535,65535,

65535,65535,65535,65535,10597,0,0,8452,65535,8452,0,0,12678,65535,65535,65535,65535,

65535,65535,65535,65535,23243,0,0,32,65535,32,0,0,23243,65535,65535,65535,65535,

65535,65535,65535,65535,38066,0,0,0,63422,65535,63422,0,0,0,38066,65535,65535,65535,65535,

65535,65535,65535,65535,52857,0,0,0,42292,65535,42292,0,0,0,52857,65535,65535,65535,65535,

65535,65535,65535,65535,59164,0,0,0,23243,65535,23243,0,0,0,59164,65535,65535,65535,65535,

65535,65535,65535,65535,65503,16936,0,0,10565,61277,65535,61277,10565,0,0,16936,65503,65535,65535,6553

5,65535,

65535,65535,65535,65535,65535,38034,0,0,0,42260,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,42260,0,0,0,38034,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,50744,4258,0,0,19017,65503,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65503,19017,0,0,4258,52825,65535,65535,65535,65535,6
5535,

65535,65535,65535,65535,65535,65535,14791,0,0,0,59164,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,59164,0,0,0,14791,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,57051,0,0,0,0,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,0,0,0,0,57051,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,16904,0,0,0,33840,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,33840,0,0,0,16904,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,59196,0,0,0,0,48599,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,48599,0,0,0,0,59196,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,40179,0,0,0,4226,57083,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,57083,2145,0,0,0,40179,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,23275,0,0,0,4226,59196,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,59196,4226,0,0,0,23275,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,59164,6339,0,0,0,8484,57051,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,57
051,8484,0,0,0,6339,59196,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,59164,0,0,0,0,4226,528
57,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,52857,4226,
0,0,0,0,59164,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54938,0,0,0,0,0,
35921,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,35921,0,0,0,0,0,54938,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

```
        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,52857,84
52,0,0,0,0,8484,50744,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,50744,8484,0,0,0,0,8452,52857,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,57
083,19049,0,0,0,0,0,21162,40179,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,40179,21162,0,0,0,0,0,19049,57083,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,33840,0,0,0,0,0,0,8484,21162,46518,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,46518,21130,8484,0,0,0,0,0,0,33840,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,61309,10565,0,0,0,0,0,0,0,0,6371,16936,23243,23243,16936,6371,0,0,0,0,0,
0,0,0,0,10565,61309,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,42292,16904,2113,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2113,16904,
42292,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,48599,35953,8484,0,0,0,0,0,0,0,0,0,0,0,0,8484,
35953,48599,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,63390,59196,50712,31695,19017,848
4,2113,0,0,2113,8484,19017,31695,50712,59196,63390,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535
};

const int heart[] = {

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65308,62870,62
```

675,62675,62902,65340,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65275,62837,62675,62675,62902,65405,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65210,60009,57344,57344,57344,57
344,57344,57344,57344,57344,60009,65080,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62870,5987
9,57344,57344,57344,57344,57344,57344,57344,57344,62220,65405,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65405,59684,57344,59392,59392,59392,59392,59
392,59392,59392,57344,57344,57344,55296,57409,62902,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62642,57344,57344,5734
4,57344,57344,59392,59392,59392,59392,59392,59392,59392,57344,59944,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,62642,57344,59392,59392,59392,59392,59392,59392,59
392,59392,59392,59392,59392,59392,57344,57344,55296,57604,65503,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65243,57376,55296,57344,57344,5939
2,59392,59392,59392,59392,59392,59392,59392,59392,59392,59392,59392,57344,65048,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,62415,57344,59392,59392,59392,59392,59392,59392,59392,59
392,59392,59392,59392,59392,59392,59392,57344,57344,57344,55296,62740,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,60302,55296,57344,57344,57344,59392,5939
2,59392,59392,59392,59392,59392,59392,59392,59392,59392,59392,59392,57344,62870
,65535,65535,65535,65535,

65535,65535,65535,62902,57344,59392,59392,59392,59392,59392,61440,61440,61440,61
440,61440,61440,61440,59392,59392,59392,59392,59392,57344,57344,55296,60042,65535,655
35,65535,65535,65535,65535,65535,65535,57636,55296,57344,57344,59392,59392,59392,5939
2,59424,61472,61472,61472,61472,61440,61440,61440,59392,59392,59392,59392,59392,57344
,65340,65535,65535,65535,

65535,65535,65535,57344,57344,59392,59392,59392,59392,61440,61440,61440,61440,61
440,61440,61440,61440,61440,61440,59392,59392,59392,59392,57344,57344,55296,57669,655
35,65535,65535,65535,65535,65535,57376,57344,57344,57344,59392,59424,59457,61505,6153
7,61537,61537,61537,61537,61537,61505,61472,61440,61440,59392,59392,59392,59392,57344
,57441,65535,65535,65535,

65535,65535,62545,57344,59392,59392,59392,59392,61440,61440,61440,61440,61440,61
440,61440,61440,61440,61440,61440,61440,61440,59392,59392,59392,59392,57344,57344,575
71,65535,65535,65535,65535,57344,57344,57344,59392,59392,59457,61537,61570,61602,6160
2,61635,61635,61635,61602,61602,61570,61537,61505,61472,61440,59392,59392,59392,59392
,57344,65015,65535,65535,

65535,65535,57474,57344,59392,59392,59392,61440,61440,61440,61440,61440,63488,63
488,63488,63488,63488,61440,61440,61440,61440,61440,61440,59392,59392,59392,57344,573
44,57636,65535,65535,57344,57344,57344,59392,59424,59489,61570,61602,61635,61667,6374
8,63748,63748,63748,63715,63715,63683,63650,61570,61537,61505,61440,59392,59392,59392
,57344,59944,65535,65535,

65535,65535,55296,57344,59392,59392,59392,61440,61440,61440,61440,63488,63488,63
488,63488,63488,63488,63488,63488,63488,61440,61440,61440,61440,59392,59392,59392,573
44,57344,60204,57734,57344,57344,59392,59457,61570,61602,61635,61700,63748,63780,6381
3,63813,63813,63813,63813,63780,63748,63715,63683,61602,61537,61472,59392,59392,59392
,57344,57344,65535,65535,

65535,65080,55296,57344,59392,59392,59392,61440,61440,61440,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,61440,61440,61440,59392,59392,59392,57344,57344,57344,57344,59392,59457,61570,61602,61667,61700,63813,63845,63878,63878,63878,63878,63878,63878,63845,63813,63780,63715,61635,61570,61537,59424,59392,59392,57344,55296,65535,65535,

65535,62805,57344,57344,59392,59392,61440,61440,61440,61440,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,61440,61440,61440,59392,59392,59392,57344,57344,59392,59457,61570,61635,61667,63780,63845,63878,63910,63943,63975,64008,64040,64040,63975,63910,63878,63813,63780,63715,61635,61570,61505,59392,59392,57344,55296,65275,65535,

65535,62740,57344,57344,59392,59392,61440,61440,61440,61440,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,63488,61440,61440,61440,59392,59392,59392,59457,61570,61602,61667,63780,63845,63910,63943,64008,64105,64170,64203,64235,64203,64138,64073,63943,63878,63813,63748,61635,61602,61537,59392,59392,57344,57344,65178,65535,

65535,62805,55296,57344,59392,59392,61440,61440,61440,63488,63488,63488,63488,63488,63488,63520,63520,63520,63520,63520,63488,63488,63488,63488,63488,63488,61440,61440,61440,59392,59392,59424,61537,61602,61667,63780,63845,63910,63975,64073,64203,64268,64333,64365,64365,64365,64268,64170,64073,63910,63845,63780,61667,61602,61537,59424,59392,57344,55296,65275,65535,

65535,65080,55296,57344,59392,59392,59392,61440,61440,63488,63488,63488,63488,63488,63520,63520,63520,63520,63520,63520,63520,63520,63488,63488,63488,63488,63488,61440,61440,59392,61505,61570,61635,63748,63845,63910,63975,64073,64235,64365,64463,64528,64528,64528,64463,64398,64268,64138,63975,63878,63813,61700,61635,59522,59424,59392,57344,55296,65503,65535,

65535,65438,55296,57344,57344,59392,59392,61440,61440,63488,63488,63488,63488,63520,63520,63520,63520,63520,63520,63520,63520,63520,63520,63488,63488,63488,63488,61440,61440,61537,61602,63715,63813,63878,63943,64073,64235,64365,64495,64593,64658,64690,64658,64593,64495,64365,64203,64040,63878,63813,61700,61635,61570,59457,57344,57344,55296,65535,65535,

65535,65535,55296,57344,57344,59392,59392,61440,61440,63488,63488,63488,63488,63520,63520,63553,63553,63553,63553,63553,63553,63520,63520,63520,63488,63488,63488,61440,61570,63683,63748,63845,63910,64008,64203,64365,64495,64625,64723,64820,64820,64755,64658,64528,64398,64235,64040,63878,63845,61700,61635,59522,59457,57344,57344,57571,65535,65535,

65535,65535,57734,57344,57344,59392,59392,61440,61440,61440,63488,63488,63488,63520,63520,63553,63553,63553,63553,63553,63553,63553,63520,63520,63520,63488,63488,63488,63650,63715,63780,63878,63943,64105,64268,64430,64593,64723,64853,64950,64918,64820,64690,64528,64365,64203,64008,63910,63845,61700,61635,59522,59424,57344,57344,60204,65535,65535,

65535,65535,62512,55296,57344,59392,59392,61440,61440,63488,63488,63488,63488,63520,63520,63520,63553,63553,63553,63585,63585,63553,63553,63553,63553,63520,63520,63488,63488,63650,63748,63813,63910,64008,64170,64333,64495,64658,64788,64918,65015,64918,64788,64658,64495,64333,64170,63975,63878,63813,61700,61635,59489,59424,57344,55296,62967,65535,65535,

65535,65535,65405,55296,57344,57344,59392,59392,61440,61440,63488,63488,63488,63520,63520,63520,63553,63553,63585,63585,63585,63585,63585,63553,63553,63553,63520,63520,63488,63488,63683,63748,63845,63910,64040,64203,64365,64528,64658,64788,64853,6488

5,64820,64723,64593,64430,64268,64105,63943,63878,61765,61667,61602,59489,59392,57344
,55296,65535,65535,65535,

65535,65535,65535,57441,57344,57344,59392,59392,61440,61440,63488,63488,63488,63
488,63520,63520,63553,63585,63585,63585,63618,63618,63585,63585,63585,63553,63553,635
20,63520,63520,63683,63780,63845,63943,64040,64203,64365,64495,64625,64723,64755,6475
5,64690,64625,64495,64333,64170,64008,63910,63845,61732,61635,59522,59457,57344,57344
,59944,65535,65535,65535,

65535,65535,65535,62577,55296,57344,59392,59392,61440,61440,63488,63488,63488,63
520,63520,63553,63553,63585,63585,63618,63683,63715,63683,63650,63585,63585,63553,635
53,63520,63520,63715,63780,63878,63943,64040,64203,64365,64495,64593,64658,64658,6462
5,64560,64463,64365,64235,64040,63943,63878,63813,61700,61602,59489,59424,57344,55296
,63000,65535,65535,65535,

65535,65535,65535,65535,55296,57344,57344,59392,59392,61440,61440,63488,63488,63
488,63520,63520,63553,63585,63585,63650,63748,63780,63780,63748,63650,63585,63585,635
53,63520,63520,63683,63780,63845,63910,64008,64203,64365,64495,64560,64560,64528,6449
5,64430,64333,64203,64073,63975,63910,63845,63748,61635,61570,59457,57344,57344,57344
,65535,65535,65535,65535,

65535,65535,65535,65535,60074,57344,57344,59392,59392,61440,61440,63488,63488,63
488,63520,63520,63553,63553,63585,63650,63748,63845,63878,63845,63780,63683,63585,635
53,63553,63553,63683,63780,63845,63910,64040,64203,64365,64430,64463,64463,64398,6433
3,64268,64170,64040,63975,63910,63845,63780,61667,61602,59489,59424,57344,55296,62577
,65535,65535,65535,65535,

65535,65535,65535,65535,65438,55296,57344,59392,59392,61440,61440,63488,63488,63
488,63488,63520,63553,63553,63585,63618,63748,63845,63943,63943,63878,63780,63650,635
85,63553,63553,63683,63748,63813,63943,64073,64203,64333,64365,64365,64300,64203,6413
8,64073,64008,63943,63910,63845,63780,63715,61602,61570,59424,57344,57344,55296,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,59977,57344,57344,59392,59392,61440,61440,63488,63
488,63488,63520,63520,63553,63553,63585,63715,63845,63975,64040,63975,63878,63748,636
18,63553,63585,63650,63715,63813,63975,64138,64268,64333,64268,64203,64105,64040,6400
8,63975,63943,63878,63845,63748,63715,61602,61570,59457,59392,57344,55296,62480,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65503,55296,57344,59392,59392,61440,61440,63488,63
488,63488,63488,63520,63553,63553,63585,63650,63780,63910,64073,64105,63943,63813,636
83,63585,63585,63618,63683,63845,64008,64170,64300,64300,64203,64105,64008,63943,6391
0,63878,63845,63813,63748,63683,63650,61537,61505,59392,59392,57344,55296,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,60172,57344,57344,59392,59392,61440,61440,63
488,63488,63488,63520,63520,63553,63585,63585,63715,63845,63943,64040,64008,63878,637
48,63618,63585,63585,63683,63845,64008,64170,64235,64170,64073,63975,63878,63878,6384
5,63813,63780,63715,63683,63618,61537,61505,59392,59392,57344,55296,62707,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,55296,57344,59392,59392,61440,61440,61
440,63488,63488,63488,63520,63553,63553,63585,63618,63748,63845,63943,63975,63910,637
80,63650,63585,63553,63683,63845,63975,64073,64073,64008,63943,63845,63813,63780,6374
8,63715,63683,63650,63585,61505,61472,59392,59392,57344,57344,57441,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,62740,55296,57344,59392,59392,61440,61

440,63488,63488,63488,63520,63520,63553,63553,63585,63650,63748,63845,63878,63878,637
80,63683,63618,63585,63683,63813,63910,63943,63943,63878,63780,63715,63715,63683,6368
3,63650,63618,63553,63520,61440,61440,59392,59392,57344,55296,65243,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,57571,57344,57344,59392,59392,61
440,61440,63488,63488,63488,63520,63520,63553,63553,63585,63650,63748,63813,63813,637
48,63683,63618,63585,63683,63780,63813,63813,63780,63715,63650,63650,63650,63618,6358
5,63553,63520,63488,61440,61440,59392,59392,57344,57344,60074,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65503,55296,57344,59392,59392,61
440,61440,63488,63488,63488,63488,63520,63553,63553,63553,63585,63618,63683,63715,637
15,63650,63618,63585,63650,63715,63715,63683,63618,63585,63585,63553,63553,63520,6348
8,63488,63488,63488,61440,61440,59392,59392,57344,55296,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,62642,55296,57344,59392,59
392,61440,61440,63488,63488,63488,63520,63520,63553,63553,63553,63585,63585,63618,636
50,63618,63618,63585,63618,63650,63618,63585,63585,63553,63553,63553,63520,63488,6348
8,63488,63488,61440,61440,59392,59392,57344,55296,65145,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,57669,57344,57344,59
392,59392,61440,61440,63488,63488,63488,63520,63520,63553,63553,63553,63585,63585,635
85,63585,63618,63585,63585,63585,63585,63585,63553,63553,63553,63520,63520,63488,6348
8,63488,61440,61440,59392,59392,57344,57344,60172,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,55296,57344,59
392,59392,61440,61440,61440,63488,63488,63488,63520,63520,63553,63553,63553,63553,635
85,63585,63585,63553,63585,63585,63553,63553,63553,63553,63520,63520,63488,63488,6348
8,61440,61440,61440,59392,59392,57344,57441,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65210,55296,57
344,59392,59392,61440,61440,63488,63488,63488,63488,63520,63520,63520,63553,63553,635
53,63553,63585,63553,63553,63553,63553,63553,63520,63520,63520,63488,63488,63488,6348
8,61440,61440,59392,59392,57344,55296,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62512,55
296,57344,59392,59392,61440,61440,63488,63488,63488,63488,63520,63520,63520,63553,635
53,63553,63585,63553,63553,63553,63553,63520,63520,63520,63488,63488,63488,63488,6144
0,61440,59392,59392,57344,55296,62967,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,59
879,57344,57344,59392,59392,61440,61440,63488,63488,63488,63488,63520,63520,63520,635
20,63520,63553,63520,63520,63520,63520,63520,63488,63488,63488,63488,63488,61440,6144
0,59392,59392,57344,57344,60334,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,57409,57344,59392,59392,59392,61440,61440,63488,63488,63488,63488,63488,63520,635
20,63520,63553,63520,63520,63520,63520,63488,63488,63488,63488,63488,61440,61440,5939
2,59392,59392,57344,57766,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,55296,57344,59392,59392,59392,61440,61440,63488,63488,63488,63488,63488,63488,63520,63553,63520,63520,63488,63488,63488,63488,63488,63488,61440,61440,59392,59392,59392,57344,57441,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65373,55296,57344,59392,59392,59392,61440,61440,63488,63488,63488,63488,63488,63488,63520,63488,63488,63488,63488,63488,63488,61440,61440,61440,59392,59392,59392,57344,57344,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65145,55296,57344,59392,59392,59392,61440,61440,61440,63488,63488,63488,63488,63488,63520,63488,63488,63488,63488,63488,61440,61440,61440,59392,59392,59392,57344,55296,65470,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62870,55296,57344,59392,59392,59392,61440,61440,61440,63488,63488,63488,63488,63488,63488,63488,61440,61440,61440,59392,59392,59392,57344,55296,65308,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62707,55296,57344,59392,59392,59392,61440,61440,61440,61440,63488,63488,63488,63488,61440,61440,61440,61440,59392,59392,59392,57344,55296,65178,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62610,55296,57344,59392,59392,59392,59392,61440,61440,61440,61440,61440,61440,61440,61440,59392,59392,59392,59392,57344,55296,63032,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62545,55296,57344,59392,59392,59392,59392,61440,61440,61440,61440,61440,61440,59392,59392,59392,57344,57344,55296,63000,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62577,55296,57344,57344,59392,59392,59392,59392,61440,61440,59392,59392,59392,59392,57344,57344,55296,63000,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62675,55296,57344,57344,59392,59392,59392,59392,59392,59392,57344,57344,55296,65145,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,62805,55296,57344,57344,57344,59392,59392,59392,59392,57344,57344,57344,55296,65243,65535,65535,65535,65535,65535,

```
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65080,55296,57344,573
44,57344,59392,59392,57344,57344,57344,55296,65405,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65275,55296,573
44,57344,57344,57344,57344,55296,57344,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,573
44,55296,57344,57344,55296,57506,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,57539,55296,55296,59912,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,59879,60269,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535
};

const int coin[] = {

        65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65501,65466,65364,63280,632
13,63210,63177,63209,63210,63245,65328,65397,65467,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
```

,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65534,65434,63247,63144,63074,60992,61024,61024,63072,63073,63105,61057,61056,63104,63104,63104,63104,63104,63174,65361,65534,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65468,65331,63112,60960,60960,60993,60993,61026,63074,63105,63137,63137,63137,63137,63169,63137,63137,63137,63105,63137,63137,63136,63104,63173,65430,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65468,65329,63077,60960,60993,60993,60994,63073,63105,63169,63137,58850,56643,54499,54436,54436,54468,56611,58787,61058,65281,63201,63169,63137,63137,63136,63104,63137,65396,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65502,65365,63110,60992,60993,60993,63073,63105,63137,56675,52228,45798,41511,41511,41511,41479,41479,41479,41447,41447,41447,39367,41511,47973,54468,63105,65281,63169,63137,63137,63104,63170,65465,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65434,63213,63040,63073,63074,63074,63169,60962,52260,43623,43591,43591,43591,45734,47973,50181,52324,54436,54436,52356,52260,48005,45766,41447,39399,39399,39368,45830,56643,65313,63201,63137,63137,63104,63242,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65366,63142,61024,63073,63073,63169,60962,50117,43623,43623,43654,50149,56611,63042,63137,63105,63105,63105,63105,63106,63137,63137,63169,63201,63169,58819,52292,43654,39367,39367,41543,54563,65313,63169,63137,63136,63136,65501,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65332,63106,63072,63073,63105,63169,52324,45703,45703,47910,56611,63137,63137,63105,63073,63042,60994,60994,60994,63042,60994,60994,61026,61026,63074,63073,63105,63169,65281,58850,48037,39367,39368,43654,60962,65281,63137,63137,63104,65430,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65502,65330,63105,63105,63105,63169,60930,47878,45702,47878,58755,65249,63105,63073,63074,61026,61025,60993,63041,63042,63042,60994,60994,63042,60994,61026,61026,61026,63074,61026,61025,63105,65249,63106,48005,39368,39336,52292,65313,63137,63137,63104,65395,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65534,65331,63104,63137,63137,63201,56643,45734,45734,54468,63201,63137,63073,63073,63073,63074,63074,63074,61026,60993,63041,63042,63042,60994,60994,60994,63042,60994,61026,61026,63074,63074,61025,63074,63106,65249,58850,43623,39336,45798,63201,63169,63137,63104,65395,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65332,63137,63137,63137,65281,54436,45734,47910,60994,63169,63105,63073,63105,63105,63073,63073,63073,63074,63074,61026,610

26,61025,63041,63042,63042,60994,60962,63042,60994,61026,60994,61026,63074,63074,6102
5,63074,63137,65281,48037,39336,41575,63137,63137,63137,63072,65431,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65366,63171,63136,63137,65281,54404,47814,50
117,63201,63137,63105,63105,63105,63073,63073,63105,63105,63073,63073,63074,63074,630
74,61026,61026,60993,63041,63042,63042,60994,60994,63042,60994,61026,61026,61026,6307
4,61026,61025,63106,65281,54468,39336,41543,63137,63137,63137,63072,65501,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65401,63208,63136,63137,63233,56579,47814,52228,65
281,63137,63137,63137,63105,63105,63105,63105,63073,63073,63105,63073,63073,63073,630
74,63074,63074,61026,61026,60993,63041,63042,60994,60994,60994,63042,60994,61026,6102
6,63074,63074,61025,61025,63201,56675,39336,43623,63201,63137,63105,63105,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65501,65295,63136,63169,63201,58818,47846,52228,65313,63
137,63137,63137,63137,63137,63137,63105,63105,63105,63105,63073,63073,63105,63073,630
73,63073,63074,63074,63074,61026,60993,63041,63042,63042,60994,60994,60994,63042,6099
4,61026,61026,63074,63074,61025,63169,56675,39336,45830,65281,63105,63104,63242,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65332,63169,63169,63169,63137,49958,52165,65313,63137,63
137,63137,63137,63137,63137,63137,63137,63105,63105,63105,63105,63105,63073,63105,631
05,63073,63073,63073,63074,63074,61026,61026,60993,63041,63042,63042,60994,60994,6304
2,60994,60994,61026,61026,63074,63074,63169,54468,39336,52324,63201,61057,61024,65465
,65535,65535,65535,65535,

65535,65535,65535,65434,63210,63168,63169,65313,52196,49989,63201,63169,63169,63
137,63137,63137,63137,63137,63137,63137,63137,63137,63105,63105,63105,63105,63073,630
73,63105,63105,63073,63073,63074,63073,63074,61026,60993,60993,63041,63042,63042,6099
4,60994,63042,60994,61026,61026,61026,63074,65249,48037,39336,60962,63137,63105,61058
,65535,65535,65535,65535,

65535,65535,65535,65298,63200,63201,63233,58786,49925,60930,63201,63169,63169,63
169,63169,63137,63137,63137,63137,63137,63137,63137,63137,63137,63105,63105,63105,631
05,63073,63073,63105,63073,63073,63073,63074,63074,61026,61026,61025,63041,63041,6304
2,60994,60994,60994,63042,60994,61026,61026,63074,63169,43623,43686,65249,63106,63072
,65396,65535,65535,65535,

65535,65535,65434,63209,63200,63201,63265,52101,54404,65313,63169,63169,63169,63
169,63169,63169,63169,63137,63137,63137,63137,63137,63137,63137,63137,63105,63105,631
05,63105,63105,63073,63073,63105,63073,63073,63073,63073,63074,61026,61026,60993,6304
1,63042,63042,60994,60994,60994,60994,60994,61026,63105,58787,39336,56611,63137,63073
,63073,65535,65535,65535,

65535,65535,65331,63200,63233,63265,58723,49989,63201,63201,63169,63169,63169,63
169,63169,63169,63169,63169,63137,63137,63137,63137,63137,63137,63137,63137,63137,631
05,63105,63105,63105,63073,63073,63105,63105,63073,63073,63074,63074,63074,61026,6102
6,60993,63042,63042,63042,60994,60994,63042,60994,61026,63169,47973,43623,65249,63073
,60992,63317,65535,65535,

65535,65501,65291,63200,63233,65345,52133,56611,63265,63201,63201,63201,63169,63
169,63169,63169,63169,63169,63169,63169,63137,63137,63137,63137,63137,63137,63137,631
37,63137,63105,63105,63105,63105,63073,63073,63105,63073,63073,63073,63074,63074,6307
4,61026,60993,60993,63041,63042,60994,60994,60994,63042,63074,60962,39367,56643,63105
,63073,60996,65535,65535,

65535,65400,63237,63232,63265,60930,52069,63233,63233,63201,63201,63201,63201,63201,63169,63169,63169,63169,63168,63169,63169,63169,63137,63137,63137,63137,63137,63137,63137,63136,63104,63105,63105,63105,63105,63073,63073,63105,63073,63073,63073,63073,61025,61025,61025,60993,63041,63042,63042,60994,60994,60994,63137,47973,45830,63169,63074,60992,63356,65535,

65535,65298,63232,63233,65377,56484,56515,65345,63201,63233,63201,63201,63201,63201,63201,63201,63169,63168,63241,63240,63209,63169,63169,63137,63137,63137,63137,63136,63141,63212,65259,63177,63106,63105,63105,63105,63105,63073,63105,63073,63075,63112,65191,63078,63042,60993,61026,60993,63041,63042,63042,60994,63073,58755,41415,61026,61026,63040,61102,65535,

65535,65260,63232,63233,65345,52101,60994,63265,63233,63201,63233,63233,63201,63201,63201,63168,63202,65364,65222,65220,65252,60992,63169,63169,63169,63137,63136,63178,65321,65312,65312,65344,65349,63139,63105,63105,63105,63105,63073,63078,65351,65472,65504,65504,65442,63075,60994,61026,61026,60993,63041,63042,63042,63137,43623,54468,63105,61025,58819,65535,

65468,63239,63264,65313,61026,52132,65377,63233,63233,63233,63201,63201,63233,63233,63200,63278,65328,65152,65184,65184,65280,48160,65281,63169,63169,63169,63173,65320,65312,65344,65440,65376,65376,65376,63105,63137,63105,63105,63075,65350,65504,65504,65504,65504,65504,63042,63074,63074,61026,60993,63041,63041,63105,52228,47973,63105,63042,58848,63391,

65400,63236,63264,63297,58754,56515,65345,63233,63233,63233,63233,63233,63201,63200,63274,65188,65152,65184,65216,65216,65312,50272,65281,63169,63169,63168,65259,65344,65344,65312,41760,61056,65408,65440,61024,63137,63137,63105,63143,65504,65504,65504,48129,61120,65504,65504,61088,63074,63073,63074,61026,61026,61025,63073,58723,43591,63137,60994,63040,59095,

65366,63265,63265,65377,56547,58818,65345,63233,63233,63233,63233,63233,63233,63232,63240,65184,65248,65280,65249,65248,65312,50272,65281,63169,63169,63169,65320,65344,65440,48160,61057,63171,65440,65440,63296,58849,63137,63105,65317,65504,65504,52544,60961,63075,65504,65440,65440,56737,63105,63074,63074,63074,61026,61025,63105,41447,63042,63042,63040,58960,

65332,63296,63265,65377,56388,61058,63297,63265,63265,63233,63233,63233,63233,63232,63272,65312,60928,37506,65380,65248,65344,50304,63265,63201,63169,63172,65347,65376,65472,48193,65281,63138,65410,65440,65504,56705,63169,63106,65442,65504,65504,52449,63169,63073,65377,65440,65472,54625,63105,63073,63073,63074,63074,63074,63169,45702,58786,63073,63040,56779,

65296,63296,63297,65409,54276,63233,63265,63265,63265,63265,63233,63233,63233,63233,65345,48160,41824,65315,65283,65280,65376,50336,65313,63201,63201,63174,65376,65408,65472,50305,65249,63137,65379,65472,65504,56768,63169,63107,65504,65472,65504,52449,63169,61057,65314,65408,65440,52544,63106,63105,63105,63073,63073,63074,63137,47973,56611,63105,63073,54663,

65294,63296,63297,65377,54244,65377,63265,63265,63265,63265,63265,63265,63233,63233,63233,63265,65377,63235,65283,65312,65376,50368,65313,63201,63201,63174,65408,65408,65408,52481,63201,63137,65379,65504,65504,56800,63169,63108,65504,65472,65504,52417,63169,63105,65250,65376,65408,50432,63137,63073,63073,63105,63073,63073,63137,50149,54468,63105,63105,54564,

65262,63296,63297,65377,56324,65409,63297,63265,63265,63265,63265,63265,63265,63233,63233,63233,63233,63235,65315,65344,65408,52416,65313,63233,63201,63205,65440,65440,65376,54593,63233,63169,65379,65504,65504,56800,63169,63107,65472,65440,65440,52449,63169,63105,65218,65344,65408,50368,63137,63105,63105,63073,63105,63105,63137,52292

,54436,63137,63106,52482,

65262,63264,63297,63329,56387,65409,63297,63297,63265,63265,63265,63265,63265,63265,63265,63233,63235,65315,65344,65440,52448,65313,63201,63233,63204,65472,65472,65408,54625,63233,63169,65379,65504,65504,56768,63169,63107,65440,65408,65408,50369,63201,63137,65218,65344,65376,48288,63106,63105,63105,63105,63073,63073,63137,54372,54468,63137,63106,52450,

65295,63264,63265,63329,56387,65409,63297,63297,63297,63297,63265,63265,63265,63265,63265,63265,63265,63234,65347,65376,65440,52480,65345,63233,63233,63204,65504,65504,65472,54625,63233,63169,65347,65472,65504,54624,63201,63107,65440,65376,65408,50337,63201,63137,65186,65312,65376,46112,63169,63137,63105,63105,63105,63105,63137,52324,56579,63137,63137,50338,

65297,63232,63233,63329,56451,63329,63297,63297,63297,63297,63265,63297,63265,63265,63265,63265,63265,63234,65346,65408,65472,52480,65345,63233,63233,63171,65504,65504,65504,54593,63233,63169,65346,65440,65504,50400,65281,63138,65344,65376,65408,48289,63201,63137,65186,65280,65344,41856,65249,63137,63137,63105,63105,63105,63169,52260,56675,63137,63137,50339,

65333,63232,63233,63297,58595,63265,63265,63265,63297,63297,63297,63297,63265,63297,63265,63265,63265,63234,65378,65408,65504,52513,65345,63233,63233,63233,65441,65504,65504,56737,63265,63139,65409,65440,65504,48193,65281,63169,65281,65344,65376,50369,65249,63106,65217,65248,65344,37633,65281,63137,63137,63137,63137,63105,63169,50117,60898,63105,63137,48294,

65399,63201,63233,63265,58722,63137,63265,63265,63265,63265,63265,63297,63297,63297,63265,63265,63265,63234,65378,65440,65504,54593,65345,63233,63233,63233,65377,65504,65504,58913,63232,65157,65440,65408,65344,46145,65313,63201,65249,65312,65344,52512,63233,65060,65248,65248,65248,35521,65281,63137,63137,63137,63137,63137,63201,47941,63169,63105,65184,48266,

65467,63171,61153,63265,60930,60962,63265,63265,63265,63265,63265,63297,63297,63297,63297,63297,63297,63234,65410,65472,65504,54593,65345,63265,63233,63233,63233,65472,65440,65440,65157,65314,65408,65472,46144,58945,63233,63201,63201,65280,65280,65280,65091,65153,65216,65312,39776,52577,63201,63137,63137,63137,63137,63137,63201,49989,63201,63105,65216,46223,

65535,63141,61152,63233,61089,60802,63265,63233,63233,63265,63265,63265,63265,63265,63297,63297,63297,63234,65442,65504,65504,54625,65345,63265,63265,63265,63233,63264,65472,65440,65440,65408,65472,58976,39745,65377,63201,63201,63201,63169,65280,65280,65248,65248,65312,58848,24961,65377,63169,63137,63137,63137,63137,63169,60962,52292,63201,63137,56768,52661,

65535,65228,61152,61153,63265,58658,63265,63233,63233,63233,63233,63265,63265,63265,63265,63297,63297,63297,63296,59040,61152,52513,65377,63265,63265,63265,63265,65313,56801,58912,65312,61056,46144,37569,65377,63233,63233,63233,63201,63233,52545,52448,63072,58816,39712,22817,65313,63169,63169,63169,63169,63137,63137,63201,56579,58786,63169,63201,46112,61277,

65535,65365,61120,61153,61185,60834,61121,63233,63233,63233,63233,63233,63233,63265,63265,63265,63265,63297,61121,56801,58881,61057,63297,63265,63265,63265,63265,63265,65345,58945,48289,48257,56833,65409,63233,63233,63201,63201,63233,63233,65313,56865,41921,39841,52577,65409,63201,63169,63169,63169,63169,63169,63233,52165,63201,63137,65281,37600,65535,

65535,65502,63107,61121,61153,61089,60866,63233,61153,63201,63233,63233,63233,63233,63233,63265,63265,63265,63297,65377,65377,65345,63297,63297,63265,63265,63265,632

65,63265,63265,65345,65345,65313,63233,63233,63233,63233,63233,63201,63201,63233,6323
3,65345,65345,63233,63201,63169,63169,63169,63169,63169,63169,63169,63137,54340,63201
,63169,63136,39820,65535,

65535,65535,65228,61120,61121,61185,60770,61185,61153,63201,63201,63201,63233,63
233,63233,63233,63265,63265,63265,63265,63265,63297,63297,63297,63265,63297,63265,632
65,63265,63265,63265,63265,63233,63233,63233,63233,63233,63233,63233,63233,63201,6320
1,63233,63201,63201,63201,63201,63201,63201,63169,63169,63169,63201,56515,61026,63169
,65249,46112,56921,65535,

65535,65535,65433,61089,61121,61122,60993,60961,61153,61153,61153,61153,63201,63
233,63233,63233,63233,63233,63265,63265,63265,63265,63297,63297,63297,63297,63265,632
65,63265,63265,63265,63265,63265,63265,63233,63233,63233,63233,63233,63233,63233,6323
3,63201,63233,63233,63201,63201,63201,63201,63201,63169,63169,63233,54340,65281,63169
,65345,31265,65535,65535,

65535,65535,65535,63144,61088,61121,61154,60802,61153,61121,61153,61153,63201,61
153,63201,63233,63233,63233,63233,63233,63265,63265,63265,63265,63297,63297,63297,632
97,63297,63265,63265,63265,63265,63265,63265,63265,63233,63233,63233,63233,63233,6323
3,63233,63201,63201,63233,63233,63201,63201,63201,63201,63233,56611,61026,63201,65249
,52544,44177,65535,65535,

65535,65535,65535,65433,61056,61089,61121,61057,60930,61153,61121,61153,61153,61
153,63201,61153,63201,63233,63233,63233,63233,63233,63265,63265,63265,63265,63297,632
97,63297,63265,63297,63265,63265,63265,63265,63265,63265,63265,63233,63233,63233,6323
3,63233,63233,63233,63201,63201,63233,63201,63201,63201,63201,56515,65313,63169,65377
,29088,65535,65535,65535,

65535,65535,65535,65535,65194,61088,61089,61121,60897,61089,61121,61121,61121,61
153,61153,61153,63201,63201,63233,63233,63233,63233,63233,63265,63265,63265,63265,632
97,63297,63297,63297,63265,63265,63265,63265,63265,63265,63265,63265,63233,63233,6323
3,63233,63233,63233,63233,63233,63201,63233,63233,65345,56515,63233,63201,65313,50368
,44145,65535,65535,65535,

65535,65535,65535,65535,65503,61025,61089,61089,61121,60866,61121,61122,61121,61
121,61121,61153,61153,61153,61153,63201,63233,63233,63233,63233,63233,63265,63265,632
65,63265,63297,63297,63297,63297,63265,63265,63265,63265,63265,63265,63265,63265,6326
5,63233,63233,63233,63233,63233,63233,63233,63265,58850,60995,63233,63201,65345,24896
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65365,61024,61089,61089,61091,60930,61121,61121,61
122,61121,61121,61121,61153,61153,63201,61153,63201,63233,63233,63233,63233,63233,632
65,63265,63265,63265,63297,63297,63297,63265,63297,63265,63265,63265,63265,63265,6326
5,63265,63233,63233,63233,63233,63233,63265,63136,58791,63264,63233,65441,33376,54841
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65227,58976,61057,61089,61061,60994,61121,61
089,61121,61122,61121,61121,61121,61153,61153,63201,63201,63201,63233,63233,63233,632
33,63265,63265,63265,63265,63297,63297,63297,63297,63297,63265,63265,63265,63265,6326
5,63265,63265,63265,63233,63233,63265,63232,58761,65344,63233,65345,50400,33514,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,63109,59008,61058,61057,63144,60993,61
089,61089,61121,61122,61121,61121,61121,61153,61153,61153,63201,63201,63233,63233,632
33,63233,63233,63265,63265,63265,63265,63265,63297,63297,63297,63265,63265,63265,6326
5,63265,63265,63265,63265,63265,63232,58827,65346,63233,65345,61056,22786,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,61027,59008,61058,59008,63177,60994,61089,61089,61089,61121,61122,61121,61121,61121,61153,61153,61153,61153,63201,63233,63233,63233,63233,63265,63265,63265,63265,63265,63297,63297,63297,63297,63265,63265,63265,63297,63200,60941,65379,63232,65345,65313,20608,65503,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,61026,59008,61058,59008,63178,61029,61088,61089,61089,61089,61121,61122,61121,61121,61121,61153,61153,63201,61153,63201,63233,63233,63233,63233,63233,63265,63265,63265,63265,63297,63297,63297,63297,63297,63265,63296,63136,61072,63297,63264,65345,65345,20608,59132,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,58914,59008,61058,59008,61159,63146,61024,61089,61089,61089,61089,61122,61122,61121,61121,61121,61153,61153,63201,63201,63233,63233,63233,63233,63233,63265,63265,63265,63265,63297,63297,63297,63297,63296,61030,63248,65344,63265,65377,63232,18528,56986,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,56772,61056,61058,59009,61091,63246,61061,61056,61089,61089,61089,61089,61122,61122,61121,61121,61153,61153,61153,61153,63201,63233,63233,63233,63233,63233,63265,63265,63265,63297,63296,63169,63152,65386,63296,63297,65441,56864,18496,59131,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54600,58976,61058,58977,58976,63208,65294,61061,61056,61088,61089,61089,61121,61122,61121,61121,61121,61153,61153,61153,61153,63201,63233,63233,63233,63233,63232,63264,63201,63215,63312,63297,63296,65377,65505,44064,20642,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54705,52480,61121,61058,58977,59008,63242,65328,63210,61058,61056,61088,61088,61121,61121,61121,61121,61121,61153,61153,61153,61152,63200,63200,63200,63207,63249,65392,63266,63264,63297,65441,63233,24896,33515,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61245,46084,54688,61121,61058,59009,58976,61125,63278,65328,63245,63208,61092,61090,61088,61088,61088,61088,61089,61123,63207,63244,63282,65362,63306,63232,63232,63265,65409,65409,37728,18528,54873,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54741,41825,52576,63169,61090,61057,59008,58976,61091,63208,63276,63311,65361,65361,65361,65361,65361,63310,63274,63237,61152,61152,63200,63265,65409,63265,39840,18496,44145,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,52661,39748,44032,56864,63201,63170,61089,61089,61056,61056,61056,61088,61088,61088,61088,61120,61120,63233,63265,65377,65377,50464,31296,20609,44177,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,61212,44110,35522,39712,48288,54720,61088,63233,63233,63265,65313,65313,65313,65345,58977,52544,41920,29152,20640,35628,54873,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

```
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65471,54775,44176,37707,355
27,33412,31298,31265,31233,29187,29190,33482,41967,50549,61277,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535
};

const int indicator[] = {

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,48437,377
74,31336,29126,22786,27045,29223,37741,46323,61277,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,48404,22853,18464,35424,481
28,41732,41733,43846,41734,41701,48160,35488,20576,20675,44145,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,46291,16384,31232,58688,65248,65120,651
21,45891,39524,39555,39556,41667,65121,65088,65248,60800,37536,14336,42032,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65470,22788,29056,63008,65152,64961,62787,62787,649
31,43778,39522,39554,39522,39555,64963,62819,62882,65056,65120,65056,33312,18530,6121
2,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,56921,14336,48192,65248,64897,62755,62787,64867,62819,649
63,45859,41603,41635,41635,41667,64964,65056,65056,65223,65392,65390,65216,56544,1433
6,50549,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,54840,16384,60864,65088,62722,62754,62786,62819,62851,62851,649
96,45891,43715,43747,43715,43747,65121,65056,65157,65525,65491,65491,65525,65287,6502
4,18496,48404,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,59066,14336,65088,64992,62658,62754,62787,62819,62819,64899,62883,649
96,47971,43747,45827,43779,45827,65120,65056,65257,65492,65459,65458,65458,65526,6518
9,65152,18496,50549,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,
```

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,14336,58752,64992,62658,62722,62754,62787,62819,62851,62851,64932,65028,48003,45827,47939,47907,45860,65120,65056,65090,65525,65458,65491,65458,65492,65256,65056,65056,14336,61276,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,27047,45984,65088,62626,62690,62754,62754,62786,62819,62851,62851,64963,65124,54276,47939,45891,47940,54180,65187,65088,65024,65222,65525,65492,65524,65492,65088,64992,64929,54496,18562,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,52695,22720,65248,62625,62658,62722,62754,62786,62787,62818,62852,64995,65092,31458,0,0,0,0,0,23106,65088,65152,65024,65056,65190,65222,65056,65056,64929,62626,65120,31200,44178,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,20643,58752,62753,62626,62690,62690,62754,62787,62787,62819,64995,56547,0,0,0,4194,8452,4258,0,0,0,46176,65216,65056,65056,65056,65088,65056,62690,62690,62657,65088,16384,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54840,24832,65152,62561,62658,62690,62690,62722,62754,62787,64899,56547,0,0,12678,48599,57083,57084,57083,52824,16936,0,0,46114,65057,65024,64992,62881,62722,62690,62690,62626,64960,33344,48404,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,35596,48160,64769,62626,62657,62690,62722,62722,62754,64835,65027,0,0,25388,59164,57116,59196,59164,63390,65535,65535,46486,0,0,64931,64867,62755,62754,62722,62722,62690,62657,62625,56608,24901,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,16384,65184,62561,62625,62657,62658,62690,62722,62754,64899,37665,0,8485,59164,59164,61277,61277,61309,65535,65535,65535,65535,21162,0,23041,64963,62786,62754,62722,62690,62690,62657,62593,65152,16416,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61212,22720,43813,54209,64769,64737,62690,62690,62722,62722,64930,8384,0,42292,59164,61277,61309,61309,63422,65535,65535,65535,65535,57083,0,0,65027,62754,62722,62722,62721,64769,64801,58401,39555,26946,52727,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,50581,29090,31204,20642,24898,43746,58434,64834,64834,64834,64995,0,0,57083,59164,61309,61309,61309,61309,65535,65535,65535,65535,59164,4290,0,64867,64866,64866,64866,60546,45826,27010,20674,22786,35363,42031,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54872,18464,43846,26979,24866,24866,22754,20706,24834,35362,43778,58434,0,32,57051,59196,61309,61309,61309,63390,65503,65503,65503,59228,59164,8452,0,50146,45890,37474,24866,22754,22786,24866,24866,22786,43782,20576,48436,65535,65535,65535,65535,65535,65535,65535,65535

,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61212,18464,43844,22787,22786,24866,24866,24866,24866,26946,26947,26946,31107,0,0,57051,59164,61309,61309,61309,63390,65503,61309,61277,59196,59196,4258,0,26946,26979,26946,26946,24898,24899,24866,24866,24866,20707,41733,20640,54807,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,35564,50208,64576,49921,26978,20675,22786,22786,24866,24866,26946,26978,31139,2048,0,40147,59164,61277,61309,61309,61309,61309,61277,59196,50744,0,0,33187,29026,26946,24898,24866,22818,22786,20707,24866,45793,64576,56544,24966,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,56985,39821,20642,58720,62400,62496,64640,58368,41633,29058,18626,20706,22787,24866,29026,14529,0,4258,59164,59164,61277,61309,61309,61309,61309,59196,61277,12678,0,10336,29027,24866,22786,20706,18626,26978,39554,56289,64672,64576,62400,64992,18529,39821,56920,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65503,27012,18496,31171,31172,35489,65120,60320,62464,62528,64640,64672,64672,54209,45858,37442,31170,29026,0,0,19017,59196,59164,59196,59196,59196,59164,61277,27469,0,0,26914,31170,35362,43746,52130,64673,64704,64672,62560,62496,60352,64896,43937,29060,31170,18496,27012,65503,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,20608,31171,48039,48071,45927,18530,39745,65216,64608,60384,60448,62529,62561,62593,64705,64737,64769,64866,58368,0,0,4258,40147,59164,59164,59196,44405,8485,0,0,49984,64898,64769,64769,64737,64673,62593,62560,62529,60416,62528,65120,46113,16450,37509,50120,48039,31171,20608,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,37740,29058,50120,41734,29091,31204,29123,26948,22786,54528,65120,64768,62496,62497,62529,62560,62593,62625,62625,64770,56256,0,0,0,0,32,0,0,0,0,49952,64802,62657,62625,62593,62593,62561,62529,62496,64736,65088,56640,27042,22755,35429,24834,29124,41702,48039,29058,37740,65535,65535,65535,65535,

65535,65535,65535,65535,65535,18496,43814,48104,31236,45959,24866,37509,41734,35397,22755,20706,41825,56672,65056,64896,64736,64608,62561,62561,62593,64706,64673,37408,8288,0,0,0,6176,29120,64608,64738,62593,62593,62593,62561,64736,64864,65056,58752,43937,22818,20675,33316,41702,39686,20673,43847,31204,45959,41734,18496,65535,65535,65535,65535,

65535,65535,65535,65535,65535,18464,45992,50217,48105,52362,22786,41702,45959,43814,41734,37509,29060,22755,18562,29185,41825,50336,58688,64928,65024,64960,64928,64929,64864,64800,64768,64800,64864,64929,64960,64960,65024,64960,58688,52384,41825,31297,18594,20674,27012,35429,41734,43846,45959,41734,24866,39654,43846,43846,43879,18496,65535,65535,65535,65535,

65535,65535,65535,65535,65535,18528,43944,50185,52427,52362,24866,41734,45927,45959,48007,45991,43879,43846,39654,33317,29092,24899,22755,18594,16450,24898,27041,31233,31265,31297,39713,33377,31265,31265,29121,24898,18530,18562,22723,24867,29060,33284,39621,43847,45927,45992,48072,45959,45959,43846,29123,33316,48104,43814,46024,18528,65535,65535,65535,65535,

65535,65535,65535,65535,65535,44144,26913,48137,48137,43880,29091,43846,45927,45959,45959,45992,48072,48072,48104,48137,48104,45992,45959,43879,43847,39622,37509,35429,33316,33317,31172,33284,33316,35429,37509,39622,41798,43911,45960,45992,48104,5018

4,50185,48104,48072,48072,45992,45991,45959,43847,33316,24898,48137,48137,26913,44144
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,39853,16384,22720,20608,35396,43847,45959,45
959,48040,45992,48072,48072,48072,48136,48137,50185,50217,50217,50250,50282,52330,523
30,52362,52362,52362,52362,52362,52330,52362,52330,50282,50249,50249,50217,50185,4813
7,48105,48104,48072,48072,45992,45991,45960,45927,37541,22754,22752,16384,39853,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,37708,35428,45927,45959,45
959,45991,48072,48072,48104,48104,48136,48137,50185,50185,50217,50249,50249,50250,502
50,50250,50250,50250,50250,50250,50250,50250,50249,50249,50249,50217,50217,50185,5018
5,48136,48105,48072,48072,48072,45992,45959,45959,39654,20576,59131,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,27045,39653,45959,45959,45
991,48040,48072,48072,48104,48104,48136,48137,50185,50217,50217,50249,50249,50250,502
50,50282,50282,50282,50282,50282,50282,50250,50250,50250,50249,50249,50217,50217,5018
5,48136,48105,48104,48072,48072,48040,45992,45959,41734,22752,52694,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,20608,41734,45959,45959,45
991,45992,48072,50152,50217,52297,52297,50249,50217,50217,50249,50249,50250,50250,502
82,50282,50282,50282,50282,50282,50282,50282,50282,50250,50250,50249,50249,50249,5229
7,52297,52297,50185,48104,48072,48072,45991,48039,41767,26946,41999,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,61277,24832,41766,45959,45959,45
991,48072,52265,37670,25092,23012,25092,37671,54442,52330,50250,50250,50282,50282,502
82,50282,50282,52362,52362,50282,50282,50282,50282,50282,50250,52330,56522,39783,2512
5,23012,25092,35590,52297,48072,48072,45992,45992,43846,31171,33481,65535,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,52727,29057,43847,45959,45991,45
992,50184,14626,0,0,0,0,0,0,31398,56523,50250,50282,50282,50282,52362,50282,52362,52362
,50282,50282,50282,50282,50282,54475,37703,0,0,0,0,0,0,4225,52265,48072,48072,45991,459
27,35396,24867,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,44144,33251,43847,45959,45991,48
040,48072,43879,35591,56490,54410,56490,46024,14658,0,20932,56523,52330,50282,52362,5
2362,52363,52362,52362,52363,52362,50315,52330,54475,31398,0,6305,43944,56522,54410,5
6490,41831,39751,50153,48072,48040,45992,45959,37541,16384,65535,65535,65535,65535,65
535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,35595,35396,45927,45959,45991,48
040,46024,48104,50185,48137,50185,50185,50250,56523,27205,0,39784,52394,52362,52362,5
2363,52363,52363,52363,52363,52362,52363,48169,0,18819,58635,52330,50249,50217,
50217,50217,50185,48104,48072,48072,48040,48040,39654,20576,61244,65535,65535,65535,6
5535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,27109,39621,45959,45959,45992,48
072,48072,48104,48104,48137,50185,50217,50217,50250,58635,0,10498,56523,52362,52427,5
4507,54507,54508,54508,54507,52427,52395,54475,23045,0,58635,50282,50249,50249,50217,
50217,48137,48104,48104,48072,48072,48040,48040,41734,22720,54775,65535,65535,65535,6
5535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,22753,41702,45959,45959,45991,46
024,48072,48104,48136,48137,48137,50217,50249,50250,50282,52362,52362,52363,52363,354

94,24931,22720,24768,24768,22851,33381,50250,52395,54443,52362,50282,50282,50250,5024
9,50217,50185,50185,48136,48105,48104,48072,48040,48040,41766,26913,46290,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,22688,41734,45959,45991,48040,46
024,48072,48104,48137,48137,50217,50217,50249,50250,50282,52330,50282,50249,22754,372
83,49924,65499,65535,65534,56197,39395,22721,48137,52329,52362,50282,50282,50250,5024
9,50249,50217,50185,48136,48105,48104,48072,48040,48040,43814,29090,39821,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,61211,24865,43814,45959,45959,48040,48
072,48104,48104,48137,50185,50185,50249,50249,50250,50250,50216,56688,56952,28864,435
57,43588,45601,47648,47649,45636,43557,31009,50581,58899,50216,50282,50282,50282,5025
0,50249,50217,48137,48136,48105,48072,48072,45992,48040,43847,31203,31336,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,52727,29025,43846,45959,45991,48040,48
104,45960,48105,48137,50185,50217,50217,50184,50184,50151,54575,65535,65535,22818,456
34,43525,41445,41445,41445,43525,47715,22720,65535,65535,58900,48103,50216,50217,5024
9,50249,50217,50217,50185,45992,50185,48072,46024,45991,43879,35396,24931,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,48371,31170,43847,45959,45991,48072,35
461,33316,50217,48137,48137,48104,48136,54673,61144,63389,65535,65535,65535,59262,208
35,35232,47747,45636,47747,37312,20770,52890,65535,65535,65535,65438,61177,56787,5021
7,48135,50217,50217,50217,37574,33316,50185,48072,45992,45927,37509,18528,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,42031,33283,43879,45959,45992,48104,26
979,50217,48104,48104,48038,56852,65535,65535,65503,65503,65471,65471,63422,65535,655
35,40114,22949,28993,22916,35856,61375,65535,63454,65470,63422,65503,65503,65535,6553
5,58999,48103,48137,48137,52297,31172,45992,48072,45991,45959,39653,18464,65535,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,37708,35396,45927,45959,45992,45991,31
204,50185,48104,48039,61276,63487,63390,63390,63390,63390,63390,63390,63390,63390,634
22,65535,63487,50744,61375,65535,63422,63390,63390,63390,63390,63390,63390,63390,6339
0,63455,63423,48169,48136,50185,37542,41767,48072,48040,45959,39654,18496,63357,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,31336,37508,45927,45959,45991,48104,24
899,52330,45958,56984,63422,61309,61309,61309,61309,61309,61309,61309,61309,61309,633
90,65535,63422,22720,61244,65535,63390,61309,61309,61309,61309,61309,61309,61309,6130
9,61309,61342,61244,48038,52329,29092,48072,48072,45991,45991,39654,22688,57018,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,29190,39621,45927,45959,45991,48072,37
509,29091,52395,65535,61309,59261,61277,59229,59229,59229,61309,61342,65535,65535,526
62,33416,25029,48664,27207,29158,48435,65503,65535,61342,61309,59229,59229,59229,5922
9,61277,61309,63487,56687,31268,31236,50185,46024,45992,45991,41734,24800,52694,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,24867,39621,45927,45959,45991,48104,48
104,37475,24964,33416,46257,52661,54872,54840,54840,56920,50548,41966,31270,22851,337
10,50809,65535,65535,65535,54971,35823,22884,29189,41933,48435,56952,54840,54840,5484
0,52694,48338,35561,24965,35363,50152,50185,48072,45992,48039,41702,24801,52661,65535
,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,27011,39653,45959,45959,45959,22818,50151,52660,50810,46519,37936,31531,29386,29386,29386,29353,33677,42227,54970,33808,61342,65502,65535,65535,65535,65503,63390,31727,55036,44340,33710,29353,29386,29386,29386,31467,35856,44438,48697,52726,50216,27011,41766,48040,45959,41734,26913,46322,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,20544,41734,45959,45959,48040,43847,24932,52923,55003,57051,57116,59229,61342,63422,63455,65503,65503,65503,65535,31695,57051,61277,63422,65470,63422,61277,57051,29582,65535,65503,65503,63455,63455,63422,61310,59229,59164,54971,54970,55036,27143,41734,48136,45991,45991,41734,29026,44144,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,20576,39654,43847,45959,45991,48137,48104,22786,50712,59196,59131,59164,59196,61277,61309,61341,63390,63390,65535,40147,44405,57051,59228,61309,59228,57051,50712,31727,65535,63390,63390,61341,61309,61277,59196,59196,57083,57116,54938,24899,46024,50282,48104,45960,45959,41734,26946,44177,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,22688,39654,39621,45959,48072,48104,50249,52394,26945,42064,63487,59196,63454,61309,61310,63422,63422,63422,65503,61277,21130,29614,35953,38066,38001,29614,21130,54938,65535,63422,63390,63390,61309,61309,63423,59131,61375,46421,24833,52363,52362,50217,48104,48040,43847,39653,26978,44177,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,22688,41734,39622,39622,48104,48137,50249,52362,54540,37541,31271,41933,35627,65535,63390,63422,63422,63422,65503,65535,61309,50712,46486,42260,44405,48631,59164,65535,65503,63422,63422,63390,63390,63455,42064,39852,33449,33315,56620,52395,50282,50249,50185,45960,37573,39654,26946,44177,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,20576,41734,39622,39622,39621,50217,50282,52330,52395,54475,63158,63422,25030,65535,63422,63422,63422,65503,65503,65535,65535,65535,65535,65535,65535,65535,65503,65503,65503,63422,63422,63390,65535,33612,56985,65402,54474,52395,52395,52362,52330,46024,39589,39621,39654,26978,44177,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,20576,41734,29091,35428,39654,37541,48169,54443,52395,52427,50249,65535,29158,61374,63422,63422,65503,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,65503,65503,63422,63390,65535,29190,65535,52362,52395,52427,52427,52427,43879,37541,37509,26979,41766,29026,44145,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,27011,39653,41702,29190,35396,45959,37509,43879,54507,54508,52362,50548,65535,31466,65535,63422,65503,65503,65503,65535,65535,65535,65535,65535,65535,65535,65535,65503,65503,63423,65535,42096,56953,56953,48136,54507,54540,48169,39621,43847,37476,29190,41702,41702,24865,46322,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,24899,39653,39621,42063,38002,27044,41766,43912,39588,48072,54475,63357,41999,33546,65535,65503,65503,65503,65503,65535,65535,65535,65535,65535,65535,65535,65535,65503,65503,65503,63455,65535,42064,37707,59099,56687,46023,39621,41799,41798,31204,35823,42194,39589,39686,24833,50581,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,31303,37509,43813,46321,42325,42325,33710,29256,37674,44209,42227,44144,29125,50777,65503,63422,65503,65503,65503,65535,65535,65535,65535,65535,65535,65535,65535,65503,65503,65503,63454,65503,57116,27045,44112,40114,44274,37707,29255,31499,40245,42293,44307,45893,39654,20640,56953,65535

```
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,35627,35396,41701,48402,52857,50744,46
518,42293,35953,31434,27174,27207,40114,65503,63422,63422,65503,65503,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65503,65503,63422,63422,65535,5071
2,27175,29223,29321,35888,42325,46518,48632,50745,48500,43749,41734,18528,61277,65535
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,42063,33251,41701,48402,50744,52825,54
970,57083,54971,52890,54971,57116,63454,63390,63422,63422,65503,65503,65503,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65503,65503,63422,63422,63390,63390,6342
2,63423,57148,55035,57084,57051,54938,52825,50712,46452,41701,39621,16416,65535,65535
,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,54808,26945,39620,46321,50744,50744,52
857,54970,57051,59164,59228,61277,61309,61341,63390,63422,63422,65503,65503,65503,655
03,65503,65535,65535,65535,65503,65503,65503,65503,63422,63422,63390,63390,61310,6130
9,59229,59196,59164,57051,54970,52857,50744,48664,48467,39620,35396,24931,65535,65535
,65535,65535,65535,65535

};

const int bottle1[] ={

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,61277,48599,29582
,14823,8452,4226,65,32,2113,4258,8484,19017,35921,50744,61309,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,63422,40179,8452,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,2113,14823,57083,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65503,61277,44373,10597,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4
226,29582,54970,63422,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,61277,40147,10597,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,214
5,27469,54970,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
```

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,63390,48599,16936,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2145,29582,57051,65535,

65535,63422,50744,21162,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8484,35953,59164,65535,

65535,63390,46518,19049,2113,0,0,0,0,0,0,0,0,0,0,2080,12515,29289,45998,52272,54353,54320,50159,37644,20902,6241,0,0,0,0,0,0,0,0,0,0,0,6371,33840,59164,65535,

65535,65503,48599,19049,2113,0,0,0,0,0,0,0,0,10467,35531,52239,56368,56368,56368,56368,56368,56368,56368,56368,56368,54319,43917,20902,2080,0,0,0,0,0,0,0,0,6339,38034,63390,65535,

65535,63390,40147,8484,0,0,0,0,0,0,0,0,8354,33450,50159,56368,56368,56368,56368,56400,56400,56368,56368,56368,56368,56400,54287,43885,14596,0,0,0,0,0,0,0,0,2113,27469,57051,65503,65535,

65535,65503,52825,23243,2113,0,0,0,0,0,0,0,6209,43885,54320,54320,56368,56368,56400,56400,56400,56400,56400,56368,56368,56368,56400,56368,54352,50158,22918,0,0,0,0,0,0,0,0,6339,35953,63422,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65

535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63422,3803
4,6371,0,0,0,0,0,0,0,2080,43950,54288,56368,56368,56368,56368,56400,56368,56400,56400,56
400,56400,56368,56400,56400,56368,56368,56368,52240,14563,2080,0,0,0,0,0,0,0,27469,5916
4,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63390,3384
0,2145,0,0,0,0,0,0,0,6241,50127,56368,56368,56368,56368,56400,56400,56400,56400,56400,56
400,56400,56400,56400,56368,56368,56368,56400,56368,54288,29289,4193,0,0,0,0,0,0,0,25356,5497
0,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61309,2961
4,32,0,0,0,0,0,0,2080,14595,52207,56368,56368,56368,56400,56400,56400,56400,56400,56400,
56368,56368,56400,56368,56400,56400,56368,56368,56368,41838,6274,0,0,0,0,0,0,0,14823,44
405,63422,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,61277,2958
2,0,0,0,0,0,0,0,0,10402,52208,56368,56400,56368,56368,56368,56400,56400,56400,56400,5640
0,56400,56400,56400,56400,56400,56400,56400,56367,33547,4225,0,0,0,0,0,0,0,0,12678,40147
,61277,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,59164,29614,4258
,0,0,0,0,0,0,0,0,4128,48111,56368,56368,56368,56368,56368,56368,56400,56400,56368,56400,
56400,56400,56400,56368,56400,56368,56368,56336,29257,4161,0,0,0,0,0,0,0,0,32,12678,3806
6,61277,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,57083,31727,8452,0,0,0,0,0,0,0,
0,0,0,32,33483,52207,54320,56368,56400,56400,56400,56400,56368,56400,56400,56400,56400
,56400,56368,56368,56368,56400,52240,16676,2080,0,0,0,0,0,0,0,0,0,0,0,8484,35953,61309,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,63390,38034,8484,0,0,0,0,0,0,0,0,0,0
,0,20837,50094,56368,56368,56400,56400,56400,56400,56400,56400,56400,56400,56400,5640
0,56400,56400,56368,56368,54288,25031,4161,0,0,0,0,0,0,0,0,0,0,0,0,8484,40179,63422,65535,

65535,

65535,65503,48631,19017,2145,0,0,0,0,0,0,0,0,0,0,6241,29224,50191,56368,56368,56368,56368,56400,56400,56368,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56368,56367,56368,54320,41772,18757,4160,0,0,0,0,0,0,0,0,0,0,4226,21130,54938,65535,

65535,65503,42292,10597,32,0,0,0,0,0,0,0,0,2080,12548,37644,54320,56400,56400,56368,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56368,56368,56368,56368,56400,56368,56368,52304,33483,12515,2048,0,0,0,0,0,0,0,0,0,8484,54938,65503,65535,

65535,54938,19049,2145,0,0,0,0,0,0,0,0,2080,14628,39692,54320,56368,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56368,56368,56368,56368,56368,56368,56368,56368,56368,52239,35530,12547,2080,0,0,0,0,0,0,0,0,2113,14823,61309,65535,

65535,65503,65503,59196,12678,32,0,0,0,0,0,0,0,0,10467,33450,52239,56368,56368,56400,56400,56400,56368,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56368,56400,56400,56400,54352,54352,56368,56368,52239,29224,6241,0,0,0,0,0,0,0,0,0,14791,57051,65503,65535,

65535,63390,12710,32,0,0,0,0,0,0,0,0,2080,18789,43885,54288,56368,56368,56368,56368,56400,56368,56368,56368,56368,56368,56368,56368,56336,43853,18821,2048,0,0,0,0,0,0,0,0,21162,54938,63422,65535,

65535,63390,50712,4226,0,0,0,0,0,0,0,0,8354,33417,54287,56368,56368,56368,56368,56368,56368,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56400,56368,56336,56368,56400,56400,56400,56400,54352,54352,56400,54320,52240,33450,8354,0,0,0,0,0,0,0,0,10597,46518,63390,65535,

65535,65503,57051,10597,0,0,0,0,0,0,0,0,10402,33450,54320,56368,56368,56368,56400,56368,56400,56368,56400,56368,56368,56400,56368,56400,52207,33450,10467,0,0,0,0,0,0,0,0,21162,52857,63422,65535,

65535,65503,59164,33840,2145,0,0,0,0,0,0,0,6241,31337,54288,56368,56400,56368,56368,56368,56368,56400,56368,56400,56368,56400,56368,56368,50094,29257,6306,0,0,0,0,0,0,0,6371,33840,59196,65535,

65535,65503,57051,25356,32,0,0,0,0,0,0,32,16741,41772,56336,54320,56400,56368,56368,56368,56368,56400,56368,56368,56368,56368,52239,39692,14660,0,0,0,0,0,0,0,2113,27469,59164,65535,

65535,63422,50712,12710,0,0,0,0,0,0,0,6241,29224,50126,56400,56368,56368,56368,56368,54352,56368,56368,56400,56368,56368,56368,56400,56400,48047,27176,4193,0,0,0,0,0,0,0,21130,52857,65503,65535,

65535,61309,46486,10597,0,0,0,0,0,0,0,8354,31337,50127,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,58416,58416,58416,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,56368,58416,56368,56368,56368,52175,33417,10435,0,0,0,0,0,0,0,14791,44405,61309,65535,

65535,59196,38066,6371,0,0,0,0,0,0,2080,18854,43885,54320,56368,54287,41772,16709,0,0,0,0,0,0,0,6339,33808,59164,65535,

65535,65503,59164,33808,4258,0,0,0,0,0,0,2080,20903,43918,54321,54352,54352,54352,54352,54352,54352,54352,54352,54352,54384,54352,54352,54352,54352,54352,54352,54352,54352,54384,543

84,54352,54352,54352,54352,54352,54352,54352,54352,54352,54352,54352,54352,54352,5435 2,54352,54352,54352,54384,54352,54352,54352,52304,43950,18854,0,0,0,0,0,0,0,2113,27501,5 7051,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655 03,57083,31695,4226,0,0,0,0,0,0,0,2080,6209,8289,6241,6209,6241,6241,6241,6241,6241,6241, 6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6 241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,6241,62 09,2080,0,0,0,0,0,0,0,2113,27501,57051,65535,65535,65535,65535,65535,65535,65535,65535,6 5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63390,48631,211 62,32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8452,33808,57051,65503,65535,65535,65535,65535,65535,65535 ,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535, 65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61277,42260,12710,32,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2113,25388,54938,65503,65535,65535,65535,65535,65535,65535,6553 5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,65535,65535,65535,65535,65535,65535,65503,48599,19049,2113,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2113,31695,61309,65503,65535,65535,65535,65535,65535,65535,6553 5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,65535,65535,65535,65535,65535,65503,42292,12678,32,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,2113,23243,54938,65503,65535,65535,65535,65535,65535,65535,65535,655 35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,65535,65535,65535,65535,65535,63422,31727,32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,10565,38034,61309,65535,65535,65535,65535,65535,65535,65535,65535,65535, 65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,65535,65535,65535,65535,63422,46518,16936,0,0,0,0,0,0,0,0,4096,18658,228 50,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898,2489 8,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898,24930,24930,24930,24898 ,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898,24898, 24898,24930,24930,24930,24930,24930,24930,24930,24930,24930,24930,24898,24898,24866,1 4496,4128,0,0,0,0,0,0,0,6339,29582,57051,65503,65535,65535,65535,65535,65535,65535,6553 5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65 535,65535,65535,65535,65535,65535,65535,63390,35921,6371,0,0,0,0,0,0,0,0,8288,45730,64224, 64256,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,6 4288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64

288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,642
88,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,6428
8,58048,29024,6208,0,0,0,0,0,0,4226,25356,52857,65503,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,63390,35921,6371,0,0,0,0,0,0,0,10336,49826,6425
6,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288
,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64320,
64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,6
4288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64
288,58048,29057,6208,0,0,0,0,0,0,4226,25356,52857,65503,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,63390,35921,6371,0,0,0,0,0,0,0,8288,47810,64256,
64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,6
4288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64
288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,642
88,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,6428
8,58048,31072,6208,0,0,0,0,0,0,4226,25356,52857,65503,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,63390,35921,6371,0,0,0,0,0,0,0,8288,47810,64288,
64288,64256,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,6
4288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64
288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,642
88,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,6428
8,58048,26977,6208,0,0,0,0,0,0,4226,25356,52857,65503,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,63390,33840,6371,0,0,0,0,0,0,0,8288,47810,64256,
64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,6
4288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64
288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,642
88,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,64288,6428
8,58048,29024,6208,0,0,0,0,0,0,4226,25356,52857,65503,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,63422,46518,16936,0,0,0,0,0,0,0,0,2048,12448,186
88,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,2073
6,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736
,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,
20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,20736,18624,1
0336,2048,0,0,0,0,0,0,0,6339,29582,57051,65503,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,61309,44405,16904,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,10565,38034,61309,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65503,52825,25388,4258,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4258,25356,50744,63422,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63422,44373,14791,32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,14791,44373,63422,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,46518,16904,2113,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2113,16904,46518,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61309,35921,8452,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,8452,35921,61309,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,59164,35921,6371,0,0,0,0,0,0,0,0,0,0,0,4160,14723,29512,37930,40043,40074,40074,40074,40074,42123,42123,42123,42123,42123,42123,42123,42123,42123,42091,42091,42091,42091,42091,42091,42091,42091,42091,42091,42091,42091,42091,42091,42091,42091,42091,40043,42091,40074,42123,42123,42123,42091,42091,40075,42091,42091,42091,42091,42091,42091,40075,40074,40074,40074,40074,40075,42091,40043,42091,42091,42091,42091,42091,40043,40043,37930,29543,14787,2144,0,0,0,0,0,0,0,0,0,0,6371,35921,63422,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,42260,12710,2113,0,0,0,0,0,0,0,0,0,6305,27399,52718,63280,65327,65328,65328,65327,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65328,65327,65328,65328,65328,65327,65327,65328,65328,65327,65327,65327,65328,65297,52719,27399,6337,0,0,0,0,0,0,0,0,2113,14823,61277,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,12678,2113,0,0,0,0,0,0,0,2112,25287,54798,65295,65328,65328,65360,65328,65360,65328,65327,65327,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65328,65328,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65360,65328,65328,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65328,65295,54798,25319,2113,0,0,0,0,0,0,0,4258,33808,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,2145,0,0,0,0,0,0,0,0,27399,56942,65295,65327,65327,65327,65327,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65328,65328,65328,65328,65327,65327,65327,65328,65360,65327,65327,65327,65327,65328,65327,65327,65327,65327,65327,65327,65327,65360,65328,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65295,58991,27399,32,0,0,0,0,0,0,32,10565,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,32,0,0,0,0,0,0,0,2112,33704,63215,65296,65327,65327,65327,65327,65328,65327,65327,65328,65328,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65360,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65327,65327,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65328,63216,33704,2112,0,0,0,0,0,0,0,4226,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4192,35784,63216,65328,65327,65328,65327,65328,65327,65327,65327,65328,65328,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65360,65327,65328,65328,65327,65328,65328,65327,65327,65328,65328,65327,65327,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4224,35816,63248,65328,65328,65327,65327,65327,65328,65327,65327,65327,65327,65327,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65328,65327,65327,65360,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65327,65327,65327,65327,65327,65327,65360,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65328,65327,65327,65328,65327,65327,65360,65360,65360,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4192,35816,63248,65328,65327,65328,65328,65327,65328,65328,65328,65328,65360,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65327,65327,65327,65360,65327,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65327,65327,65327,65327,65360,65328,65328,65328,65328,65328,65327,65327,65360,65327,65327,65328,65327,65327,65327,65327,65328,65327,65328,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4192,35784,65296,65327,65327,65328,65360,65328,65327,65327,65327,65327,65328,65328,65328,65328,65327,65328,65328,65327,65328,65328,65327,65327,65328,65327,65327,65328,65327,65327,65328,65327,65328,65360,65328,65328,65328,65328,65328,65328,65328,65360,65360,65328,65328,65328,65327,65327,65360,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4224,35784,63248,65328,65328,65328,65328,65328,65327,65327,65327,65327,65327,65328,65328,65328,65328,65360,65328,65327,65327,65327,65328,65328,65327,65327,65327,65328,65327,65327,65360,65327,65328,65328,65360,65360,65360,65360,65328,65328,65328,65328,65328,65328,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65360,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4224,35816,63248,65328,65327,65328,65328,65327,65328,65328,65328,65328,65327,65327,65327,65327,65360,65328,65327,65328,65328,65327,65327,65327,65327,65327,65328,65327,65327,65327,65328,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328

,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,2080,4193,4193,4193,2145,4193,4193,4193,4193,4193,4193,4193,4193,4193,4193
,6337,21061,44299,61103,65295,65327,65327,65328,65328,65327,65327,65328,65328,65327,6
5327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65
328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,653
28,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6532
7,65327,65360,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4193,29513,54863,65295,65295,65328,65327,6532
7,65327,65327,65327,65327,65327,65327,65328,65328,65328,65328,65328,65328,65360,65328
,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,18949,48460,63215,65327,65327,65328,65328,
65327,65327,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65
328,65328,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,653
28,65328,65328,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10530,40009,63183,65328,65327,65327,65327,
65328,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65328,65360,65360,65360,65328,65328,65328,65328,65328,65328,65
328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,653
28,65328,65328,65360,65328,65328,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14755,44265,63214,65327,65327,65328,65327,
65327,65327,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65
328,65328,65328,65328,65328,65328,65328,65360,65328,65328,65328,65328,65328,65328,653
28,65328,65328,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4192,25287,50637,63280,65295,65295,65327,6532
7,65327,65327,65327,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65327,65327,65328,65327,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,4193,6337,8385,6337,6369,6369,8417,8418,8418,8418,8418,8418,8418,6369,6369
,10498,18981,37962,56910,65327,65327,65327,65328,65327,65327,65327,65327,65328,65328,
65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6

5328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65
328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,653
27,65327,65360,65327,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4192,35816,63248,65327,65328,65328,65328,65328,65328,65328,65328,65328,653
28,65328,65328,65328,65328,65327,65327,65295,65296,65328,65327,65327,65327,65327,6532
8,65328,65328,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65360,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65327,65328,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63215,65327,65328,65328,65328,65328,65328,65328,65328,65328,653
28,65328,65327,65327,65327,65327,65328,65327,65327,65327,65327,65327,65327,65327,6532
7,65327,65327,65327,65328,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65360,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65327,65328,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35784,63248,65328,65328,65328,65328,65328,65328,65328,65327,65327,653
28,65328,65327,65327,65328,65327,65327,65328,65328,65327,65327,65328,65327,65327,6532
7,65328,65328,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,65360,65328,
65328,65328,65328,65360,65328,65360,65328,65328,65328,65328,65328,65328,65360,65360,6
5360,65328,65328,65327,65327,65327,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63248,65328,65327,65328,65328,65327,65328,65328,65327,65327,653
27,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65328,65327,65328,6532
8,65328,65328,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,65360,65328,65328,65328,
65328,65328,65328,65328,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65327,65327,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35784,63248,65327,65327,65328,65327,65328,65328,65328,65360,65328,653
27,65327,65327,65327,65328,65327,65328,65328,65327,65327,65328,65328,65327,65327,6532
7,65327,65328,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65327,65327,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63248,65328,65327,65328,65327,65360,65360,65360,65327,65327,653
28,65327,65327,65327,65327,65328,65328,65327,65327,65360,65328,65328,65327,65327,6532
7,65327,65328,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65327,65327,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4192,35816,63248,65328,65327,65328,65327,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65327,65327,65327,65327,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65327,65328,65327,65327,65328,65327,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4224,35816,63248,65328,65327,65327,65327,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65327,65360,65327,65327,65328,65328,65328,65360,65360,65327,65327,65327,65328,65360,65328,65328,65328,65328,65327,65327,65328,65327,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4192,35816,63248,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65328,65327,65328,65328,65360,65328,65328,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65327,65327,65328,65327,65327,65328,65327,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4192,35784,63248,65327,65327,65327,65328,65327,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65327,65327,65328,65328,65327,65327,65327,65328,65327,65360,65327,65360,65328,65360,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65328,65327,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4224,35784,63248,65327,65328,65328,65328,65327,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65328,65328,65328,65327,65327,65328,65328,65328,65328,65328,65327,65327,65327,65328,65327,65327,65328,65327,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,0,32,32,2080,2080,2080,2080,2080,2080,2080,2080,2080,2080,2080,2080,2080,2112,8385,23205,48493,63248,65327,65327,65328,65327,65327,65327,65328,65328,65327,65327,65328,65327,65327,65328,65327,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,23206,52717,63247,65327,65328,65327,65327,65327,65328,65328,65328,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,6

5328,65360,65328,65327,65327,65328,65327,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,12610,40041,63215,65328,65295,65327,65327,65327,65328,65328,65328,65327,65328,65328,65328,65328,65327,65327,65327,65328,65327,65328,65327,65327,65327,65327,65328,65327,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8450,37897,63183,65327,65327,65327,65327,65327,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,65360,65360,65328,65328,65328,65360,65328,65328,65328,65328,65328,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65328,65328,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4160,33705,63217,65327,65327,65328,65328,65328,65328,65328,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65360,65328,65327,65328,65328,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6305,33738,58992,63248,65327,65328,65328,65327,65328,65327,65327,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,65328,65328,65328,65328,65328,65360,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4192,33736,63216,65327,65327,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65327,65327,65327,65328,65327,65327,65327,65328,65328,65327,65327,65327,65328,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,65328,65327,65327,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4224,35816,63248,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65328,65328,65327,65327,65328,65327,65327,65328,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65360,65360,65328,65327,65327,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0,0,0,0,0,0,0,4224,35784,63216,65328,65328,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65327,65327,65327,65327,65328,65328,6532

7,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65328,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63248,65327,65327,65327,65328,65327,65328,65328,65328,65328,653
28,65328,65327,65327,65328,65327,65328,65328,65327,65327,65328,65328,65327,65327,6532
7,65327,65328,65327,65327,65328,65328,65328,65360,65360,65360,65328,65328,65360,65360
,65360,65360,65360,65328,65328,65328,65360,65328,65360,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65328,65360,65360,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63248,65328,65327,65328,65328,65327,65328,65327,65327,65327,653
27,65327,65327,65327,65328,65327,65328,65327,65327,65327,65328,65360,65327,6532
8,65328,65327,65327,65327,65327,65328,65328,65360,65360,65328,65328,65328,65360,65360
,65360,65328,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65360,65360,65328,65328,65328,65360,65360,6
5328,65328,65328,65328,65328,65328,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35784,63248,65328,65327,65327,65327,65328,65327,65327,65328,65328,653
28,65328,65327,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65327,65327,65327,65328,63248,35816,4192,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4192,35816,63248,65327,65327,65327,65327,65327,65327,65327,65328,65328,653
28,65328,65327,65327,65327,65327,65327,65327,65328,65328,65327,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63248,65327,65327,65328,65328,65327,65327,65327,65327,65327,653
27,65327,65327,65327,65328,65327,65328,65328,65327,65327,65327,65327,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35784,63248,65327,65327,65327,65327,65328,65327,65327,65327,65328,653
28,65328,65327,65327,65328,65328,65327,65327,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65

535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35784,63248,65327,65327,65327,65327,65327,65327,65327,65327,653
27,65327,65327,65327,65327,65328,65327,65327,65328,65328,65327,65327,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63248,65328,65327,65328,65327,65327,65328,65328,65328,65327,653
27,65328,65328,65328,65327,65328,65328,65327,65328,65327,65327,65327,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65360,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65360,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63248,65328,65328,65328,65328,65328,65328,65328,65328,65328,653
28,65360,65328,65328,65328,65328,65328,65328,65327,65327,65327,65327,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,65360,65360
,65360,65360,65328,65328,65328,65328,65328,65328,65328,65360,65360,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5360,65360,65360,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,2080,12675,25286,25286,25286,27334,25286,27366,25286,27366,27366,25286,252
86,25286,25286,25286,29478,40041,54798,63216,65327,65295,65327,65327,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,
65360,65360,65360,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4225,33738,61168,65295,65328,65328,65328,65
327,65327,65327,65328,65327,65327,65328,65328,65328,65328,65328,65360,65360,653
60,65360,65360,65360,65360,65328,65360,65360,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65360,65360,65327,65328,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8418,35816,63183,65327,65327,65327,65328,6
5328,65327,65328,65327,65327,65328,65328,65328,65328,65328,65328,65360,65328,65
328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,653
60,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65327,65328,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8418,35816,61135,65327,65327,65327,65327,6
5328,65327,65328,65328,65327,65328,65328,65328,65328,65328,65328,65328,65328,65
328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,653

28,65360,65360,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65327,65328,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,23206,54765,65295,65328,65327,65327,65327,
65328,65328,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65
328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,653
28,65328,65328,65327,65328,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2048,4225,19012,46412,63248,65327,65327,65328,65
327,65327,65327,65327,65327,65328,65328,65327,65328,65328,65328,65328,65328,653
28,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65327,65328,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,2144,27399,50574,50637,50637,50605,50637,52686,52685,52686,52686,52685,526
86,50638,52653,52653,54798,61104,63248,65327,65327,65327,65327,65327,65328,6532
7,65327,65328,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65328,65327,65328,63248,35816,4192,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35784,63216,65327,65327,65328,65327,65327,65327,65327,65327,653
27,65327,65327,65295,65295,65328,65328,65327,65328,65328,65328,65327,65327,65328,6532
7,65327,65328,65328,65328,65327,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65328,65327,65328,63248,35816,4192,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63215,65327,65326,65327,65359,65327,65327,65327,65327,65327,653
27,65327,65327,65327,65327,65327,65296,65296,65328,65327,65327,65328,65328,65327,6532
7,65327,65328,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65327,65328,65360,65328,63248,35816,4192,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63248,65328,65328,65327,65327,65327,65328,65328,65328,65328,653
28,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65360,65360,65360,65360,65328,65328,65328
,65328,65328,65328,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328,
65360,65328,65328,65328,65328,65328,65328,65328,65360,65360,65360,65360,65328,65327,6
5327,65327,65327,65327,65327,65328,65328,63248,35816,4192,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4192,35784,63248,65328,65327,65327,65328,65327,65328,65328,65328,65328,653

28,65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65360,65360,65360,65360,65360,65360,
65360,65360,65328,65360,65360,65360,65328,65328,65328,65328,65328,65327,65328,65328,6
5328,65327,65327,65327,65327,65327,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35816,63248,65328,65328,65328,65327,65327,65328,65328,65328,65360,653
60,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65328,65328,6
5328,65327,65327,65327,65327,65328,65328,65296,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35784,63248,65327,65328,65327,65328,65327,65328,65328,65328,65328,653
28,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65360,6536
0,65360,65360,65360,65360,65360,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65360,65360,65360,65328,65328,65328,65328,65328,65328,65328,65360,
65360,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,6
5327,65327,65328,65328,65360,65328,65328,63248,35816,4192,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4192,35784,63247,65327,65327,65327,65327,65327,65328,65328,65328,65328,653
28,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65327,65328,65328,65328,65327,65327,65327,65328,6
5327,65327,65327,65328,65327,65328,65328,63248,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,0,0
,0,0,0,0,0,0,4224,35784,63216,65327,65328,65328,65328,65327,65328,65328,65328,65328,653
28,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65360,65360,65328,65328,65328,65328,65328,65328,
65360,65360,65328,65328,65328,65328,65328,65328,65360,65327,65328,65328,65328,65327,6
5327,65327,65328,65328,65327,65327,65327,63247,35816,4224,0,0,0,0,0,0,0,0,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,21
45,0,0,0,0,0,0,0,64,31624,63215,65296,65327,65327,65328,65327,65328,65328,65328,65328,65
328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,653
28,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6532
8,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328
,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,
65327,65327,65327,65327,65327,65328,65328,63216,31624,2080,0,0,0,0,0,0,0,0,6339,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,10
597,2113,0,0,0,0,0,0,0,8418,35784,61135,65327,65328,65327,65327,65328,65328,65328,65328,
65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,6
5328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65
328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,653
28,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,6532

7,65327,65327,65327,65327,65327,65327,61134,35816,8418,0,0,0,0,0,0,0,2145,21130,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,23275,4226,0,0,0,0,0,0,0,64,19013,48461,63215,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65328,65327,65327,65327,65327,65327,65327,65327,65360,65360,65360,65360,65360,65360,65360,65360,65360,65327,65327,65327,65327,65327,65327,65327,65327,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65328,65327,65327,65327,65327,65327,65327,65327,65327,65328,65327,65327,65327,65327,65327,65327,65327,65328,65327,63215,48492,21029,32,0,0,0,0,0,0,0,6339,40147,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,50744,19017,2145,0,0,0,0,0,0,0,0,2080,12643,35849,56911,65296,65296,65296,65327,65328,65327,65327,65327,65328,65327,65328,65328,65328,65327,65327,65327,65328,63248,56911,35850,12643,2112,0,0,0,0,0,0,0,0,2145,19049,61277,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,52857,19017,2145,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2145,19017,52857,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,42260,10597,32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,10565,42260,63422,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,54938,27469,4258,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4258,25388,54938,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,54938,27469,4258,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4258,27469,54938,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,59196,38066,12678,32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,12678,38034,59196,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,61309,48631,31695,16904,8452,2145,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2145,8484,19017,33808,52825,63390,65535,
};

```
const int evil[] = {

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,63487,65535,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,63487,63487,63487,63487
,61439,61439,61439,61439,61439,61439,61439,61439,61439,63487,63487,63487,63487,63487,
65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,63487,63487,61439,61439,59359,57246,50973,44667,42586
,40473,38393,38361,38361,38393,36313,38361,38393,40474,42586,44699,48892,55166,61439,
61439,63487,63487,63486,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,63487,63487,61439,61407,53053,44634,38393,36313,34265,34266,34298,34299
,34331,34299,34332,34332,34331,34332,34332,34332,34331,34299,34298,34266,36313,36281,
38393,44634,53053,59359,61439,63487,63487,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,63487,61439,59391,48860,40441,36281,34265,34298,34331,32284,32284,32284,32284,32285
,32284,32285,32285,30237,30269,30269,30269,32285,32285,32285,32316,32284,32284,32284,
34332,34299,36282,36280,40473,48892,59391,61439,63487,65535,65535,65503,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

    65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,63487,63487,5939
1,53086,38425,34232,34298,34331,34332,32284,32285,30269,30237,30269,30269,30269,30269
,30269,30270,30269,30270,30270,30270,30270,30270,30269,30269,30269,30269,30238,30269,
```

30269,32285,32284,32251,34331,36345,34232,40505,57279,61439,63487,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,63487,63487,61439,59391,46780,3420
0,34298,34331,32284,32284,32284,32317,30237,30269,30270,30269,30270,30269,30270,30270
,30269,30270,30269,30270,30269,30269,30270,30270,30270,30238,30269,30269,30270,30269,
30270,30270,30237,32285,32284,32316,32283,34331,34266,36312,53085,61439,63487,63487,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,65535,63487,59326,46746,36247,36345,3433
1,32284,32285,30237,30237,30237,30269,30269,30270,30237,30269,30269,30269,30270,30270
,30270,30269,30269,30269,30269,30269,30269,30270,30238,30269,30270,30269,32285,30269,
30269,30270,30270,30269,30237,30270,30237,30236,32284,34330,36312,38327,57213,61439,6
3487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,63487,63487,59359,46780,34233,32219,34332,32284,3228
5,30269,30270,30237,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,
30270,30270,30270,30270,30270,30270,30270,30270,30269,32317,32284,34332,34265,38360,5
5198,61439,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,63487,59391,51005,36280,36346,34332,32284,32284,30269,3026
9,30269,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,
30270,30270,30270,30270,30270,30269,30270,30270,30269,30269,32284,32284,32284,34298,3
4233,46747,59327,63487,65503,65502,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,63487,57279,40506,34265,34331,32284,30269,30269,30269,30269,3026
9,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,
30270,30270,30270,30270,30270,30270,30270,30270,30269,30237,32317,30269,30237,3
4332,34298,38361,55166,61407,65502,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,63487,63487,50973,36280,34331,32284,32285,30269,30270,30270,30269,30269,3026

9,30269,30270,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,
30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,30270,30269,30270,30269,3
0269,32284,34331,34265,46779,59359,63487,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,63487,59327,44634,34265,34332,32285,30237,30270,30270,30270,30270,30270,30270,3027
0,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,
30270,30270,30270,30270,30270,30269,30269,30270,30270,30270,30270,30270,30270,30270,3
0269,30269,32285,32284,34266,38426,57246,61439,63487,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,614
39,55166,38393,36346,32284,32285,30238,30238,30269,30269,30269,30270,30270,30269,3026
9,30269,30270,30270,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,
30270,30270,30270,30270,30270,30269,30270,30269,30270,30270,30270,30269,30269,3
0269,30269,30269,30269,32284,34331,38393,53086,61439,63487,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,63487,61439,509
73,36313,34331,32285,30269,30269,30270,30270,30269,30269,30269,30270,30270,30270,3026
9,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,
30270,30270,30270,30270,30270,30269,30269,30269,30270,30269,30270,30269,3
0269,30269,30270,30269,32317,32284,34331,36313,48892,61439,63487,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,61439,46780,363
13,34331,32285,30238,30269,30269,30269,30269,30269,30269,30269,30270,30270,30269,3026
9,30269,30269,30269,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30270,30270,30270,30270,30269,30269,30270,30270,30270,30270,30270,
30270,30270,30270,30270,30270,30269,30269,30269,30269,30269,30270,30270,30237,30269,3
0270,30269,30269,30269,30237,32285,32285,34331,34233,46780,59391,63487,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65503,65535,65535,63487,61439,44666,34266,343
32,32284,32317,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269
,30270,30270,30270,30269,30269,30269,30269,30269,30269,30269,30270,30270,30270,30270,
30270,30270,30270,30270,30269,30269,30269,30270,30269,30269,30269,30269,30269,3
0270,30270,30269,30270,30270,30237,32286,32285,34332,34265,44666,61407,63486,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65

535,65535,65535,65535,65535,65535,65535,65503,65535,63455,59359,42586,36346,32252,323
17,30269,30269,30238,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30269,30269
,30269,30269,30269,30269,30270,30270,30269,30269,30269,30269,30270,30270,30270,30270,
30270,30270,30269,30270,30270,30270,30270,30269,30270,30270,30269,30270,30270,30270,3
0269,30269,30269,30269,30270,30269,30269,30269,32284,32251,36346,44634,59391,63487,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65534,63487,61407,42554,34298,32284,30237,302
69,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30269,30269
,30270,30269,30269,30269,30270,30270,30269,30269,30269,30269,30270,30269,30270,30270,
30270,30270,30270,30270,30270,30270,30270,30270,30269,30270,30270,30270,30270,30270,3
0269,32285,32285,30269,30269,30270,30269,30269,30269,32285,34332,36314,42651,61407,63
487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,63487,61439,42586,34298,32316,30237,30269,302
70,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30237
,30269,30270,30269,30270,30269,30270,30270,30269,30269,30237,30237,30237,30269,30270,
30270,30237,30237,30270,30270,30238,30270,30270,30270,30269,30270,30269,30270,30238,3
2317,32284,34332,32285,30237,28221,30269,30269,28222,30238,32285,32284,34298,46779,61
439,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,63487,61439,44666,34266,32284,32317,30269,30270,302
69,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30302,30237,30237
,30237,30269,30270,30270,30270,30270,30270,30270,30237,32285,30237,30237,30269,30269,
30269,30237,30269,30269,30237,30269,30238,30270,30269,30269,30269,30237,30205,32285,3
4300,21523,17329,34299,32284,30236,30269,30237,30270,30270,30270,32285,32284,34232,48
924,61439,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,63487,61439,46812,34266,32284,32317,30269,30269,30270,302
69,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30237,32317,30236
,32316,32317,30269,30270,30269,30269,30270,30270,30238,30237,30237,30237,30269,32317,
30269,30237,30237,30237,30269,30237,30238,30269,30269,30269,30237,32285,32284,34267,2
1490,2342,2309,10827,36282,34299,32284,32285,30237,30270,30269,30269,30269,32283,3426
5,53118,63455,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,61439,50973,36281,32283,30269,30269,30269,30269,30270,302
69,30270,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,32284,32284,30106
,32284,32284,32316,30269,30270,30270,30269,30238,30237,30237,32317,30236,32317,30269,
32317,32285,32285,30204,30237,30270,30270,30269,30269,30269,32317,32285,34299,21489,2
341,4323,4291,4356,2375,29942,34332,32252,30237,30269,30270,30238,30270,30269,32283,3
6248,59327,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,63487,55198,36281,34332,32284,30238,30270,30269,30269,30269,30269,30269,30270,30269,30270,30237,32253,32252,36314,17231,2408,25716,36346,32284,30238,30269,30269,30237,30270,30238,30237,32252,13169,34299,32285,32286,32285,32186,19508,32284,32285,30237,30270,30270,30237,32284,34331,23504,2308,4291,6339,6338,6339,4323,4357,19345,34266,32284,30236,30269,30269,30238,30237,32285,34331,38394,59359,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,59359,38393,34331,32284,32285,30269,30269,30270,30269,30270,30270,30269,30269,30237,30269,32284,34299,15053,2309,2309,2342,21457,34299,32284,30269,30269,32285,30238,30270,32317,34299,2473,27862,34300,32285,34299,19410,8878,34299,30236,30237,30270,30270,32284,36347,21490,2308,4324,4291,6338,8386,6338,6339,4291,2277,10860,32153,32283,30236,30237,30237,30237,32317,32284,34298,44699,61439,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61439,42586,34298,32284,30269,30269,30269,30270,30269,30269,30237,32284,34265,10924,2276,4324,4291,4323,2309,15118,34266,32284,30237,30269,30269,30269,32284,36281,2342,6601,32120,38395,38329,2407,2440,34201,32284,30269,30269,32284,34299,23570,2277,4292,6339,6339,6338,6338,6338,6339,4291,4324,2276,4553,27894,34332,32284,30237,30269,30269,30237,32283,34232,53085,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65534,65535,63487,50973,36313,32284,30269,30270,30270,30269,30269,30270,30269,30269,30237,32284,32251,32153,10826,2308,6338,6338,6338,6306,4291,2276,10794,32153,36380,32284,30269,30269,32316,36281,2310,2276,6568,23439,10695,2244,2277,23538,34299,34299,34331,34331,23603,4357,6340,6339,6339,6338,6338,6338,6370,4291,4291,2276,2309,2375,2506,17428,30204,30269,30269,30270,30270,32284,34331,38361,57279,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65534,63487,59327,36313,34299,32285,30270,30270,30238,30237,30269,30270,30269,30269,30269,32284,34299,34169,8714,4324,6338,6338,8386,8386,6338,6339,4323,2276,4488,27764,36379,32284,30236,32284,34200,2310,4291,4324,4291,6339,6338,4324,4422,29910,38394,36281,19312,2342,4324,6306,8354,8387,6338,6338,6370,4323,2276,6503,21457,32055,34201,32153,25913,28059,30269,30269,30270,30269,30269,32284,34298,44699,61407,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,61439,42554,34299,32284,32285,30270,30238,30269,30270,30269,30270,30237,30237,34332,34168,6569,4325,6339,6306,6338,6338,6338,6338,6339,6371,4291,4324,2309,15053,34201,34331,36346,34103,2277,6371,6339,6338,6338,25355,23243,4324,4389,10729,4422,2276,4324,4291,6338,6338,6338,6338,6338,4323,2276,12972,36249,34332,34300,32284,32285,30237,30237,30269,30270,30269,30269,30269,30269,32284,34233,

55166,61439,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61
439,53118,36313,32284,30269,30238,30270,30237,30237,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30269,32284,30007,4488,2277,2276,4291,4291,6339,6339,6338,6338,633
8,6338,6338,4291,4291,4292,4455,15085,27730,14954,4291,38033,44372,8419,6306,29581,147
58,6371,4323,4291,4291,6339,6339,6339,6338,6338,6338,6338,6371,2276,12973,34299,32284,
30269,32317,30237,30269,30269,30269,30269,30270,30269,30269,30269,30270,32285,34331,4
0441,59391,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,61
439,36280,34299,30269,30269,30269,30237,30269,30269,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30237,23767,2441,261,2244,2243,2243,2275,6371,6339,6339,6339,6338,
6338,6338,6338,6338,6339,4292,4324,2276,4291,6338,8419,8451,6339,6339,6338,6338,6338,6
338,6338,6338,6339,8387,6339,6339,6338,6339,6339,6340,6535,36249,34332,30237,32285,302
69,30269,30237,30269,30270,30237,30270,30269,32317,30269,30270,30237,34332,36312,5305
3,61439,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65503,65535,63487,46
812,34299,34332,30269,30269,30270,30269,30270,30269,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,32317,28025,27993,32153,32088,30007,27829,19409,8647,196,4324,633
9,6338,6338,6338,4290,6338,6339,6338,6338,6370,6338,8386,8386,6339,6339,6338,6338,8386,
6338,6338,6338,8386,6338,4291,4323,4324,4323,4291,4292,2277,25748,32251,32317,30269,30
269,30270,30269,30269,30269,30270,30269,30269,30269,30269,30270,30270,30270,32285,343
00,36249,59359,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,59391,36
248,34331,32285,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30269,30236,32284,32284,32284,32251,34299,36379,36314,31990,4455,
4292,6338,6338,6338,6338,6338,6338,6338,6338,6370,6338,8386,6338,6339,6339,6338,8386,6
338,6338,6338,6338,6339,6371,2276,2309,6568,6568,2342,229,10893,34267,32284,30237,3027
0,30270,30270,30270,30270,30270,30270,30270,30269,30269,30269,30270,30270,30270,30269
,32284,34266,50973,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,61439,46779,34
298,32316,30237,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,3027
0,30270,30270,30270,30270,30269,30269,30269,30269,30269,30237,30269,32285,32252,34169
,2342,4323,6339,6339,4323,4323,4324,6371,6371,6338,6338,6338,6338,6339,6339,6338,6338,6
338,6339,6338,6338,4291,2276,25650,36281,36347,36347,34266,34202,34267,32284,30237,30
237,30270,30270,30269,30270,30270,30270,30270,30270,30269,30269,30270,30270,30270,302
70,30269,30237,34332,36248,59359,63487,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65534,63487,59359,36313,34
331,30237,30270,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30269,30269,30270,30269,30269,30270,28190,30237,34299
,17198,2276,4324,2276,4487,17198,15052,4357,4323,6339,6338,6338,6339,6339,6338,8386,83

86,6338,6339,4291,4324,2244,8779,36314,32284,32285,32285,30204,32252,32285,30237,3026
9,30269,30238,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,30270
,30270,30270,30270,32284,36346,50973,63487,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,48892,36346,32
284,30270,30270,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30269,30270,30270,30269,30270,30237,30270,30270,30270,30270,30270,28221,32317,34332
,25716,262,262,25651,34266,34299,36347,32022,4390,4324,4323,4291,4324,4324,6339,8386,6
338,6339,4324,4389,2309,6634,34201,32284,30269,30238,30270,30269,30237,32285,30270,30
269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,302
69,30270,30270,30238,30237,34331,38360,61407,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,61439,38393,34331,32
317,30270,30237,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30237,32285,32284
,25814,2538,30040,34299,32285,32285,32284,34331,23571,2278,2309,23505,31957,6536,2276,
4291,4291,4324,6568,32120,34168,34299,32284,30269,30269,30238,30270,30270,30270,30270
,30238,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,
30269,30269,30269,30270,30270,30269,30237,32284,36314,55134,63487,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,55134,34265,32284,30
270,30270,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30269,30270,30270,30270,30269,30270,30270,3027
0,30270,30270,30270,30269,30270,30270,30269,30270,30238,30269,30269,30237,30269,32285
,28090,28058,32284,32285,30237,30269,30269,32284,34300,10959,23669,34299,34299,32120,
6633,229,2309,2407,30007,34332,32252,32285,30269,30269,30269,30269,30269,30269,30270,
30270,30237,30270,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,3
0269,30269,30269,30270,30270,30270,30237,30237,34299,42586,61439,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,61439,44634,34299,30236,30
269,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30269,30269,30269,302
69,30270,30270,30270,30269,30269,30237,30237,30269,30269,30269,30237,30270,30269,3026
9,30269,30269,30269,30237,30270,30269,30269,30269,30269,30269,30269,30269,30269,30269
,30269,30237,32285,30237,32317,30269,30269,30269,32285,32252,32285,32285,32284,32284,
34234,15151,10957,30008,34299,32317,30237,30269,30269,28222,30269,30237,30269,30237,3
0237,30237,30269,30269,30269,30269,30269,30269,30269,30269,30269,30269,30269,30269,30
270,30269,30269,30269,30269,30269,30269,30237,30269,32316,34265,59327,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,59359,36280,32284,30269,30
269,30269,30270,30269,30270,30270,30270,30270,30270,30270,30269,30269,30237,30269,302
69,30269,30237,32285,32285,32285,32284,34332,34331,34332,32284,32284,32284,32285,3023
7,30269,30269,32285,30238,30270,30270,30269,30269,30269,30270,30270,30238,30270,30270
,30270,30270,30270,30270,30269,30270,30270,30270,30270,30270,30270,30269,30269,32284,
32284,34332,32284,32285,32285,30269,30237,30269,30270,30270,30269,32317,30236,32284,3
2284,32284,34332,34332,32284,32284,32284,32285,30269,30269,30237,30237,30237,30269,30
269,30270,30269,30269,30269,30270,30269,30269,30269,32285,34266,50973,63487,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,55166,36313,32284,30269,30
269,30269,30270,30270,30270,30270,30270,30270,30269,30270,30270,30269,30269,30269,302
69,32285,32252,34332,36315,32088,27829,23603,21490,19409,21555,25748,32055,34234,3433

1,32283,32316,30269,30269,30269,30269,30270,30269,30269,30270,30270,30269,30270,30270
,30270,30270,30270,30270,30270,30269,30270,30270,30270,30269,30270,30270,30269,30269,
32317,32285,32285,30237,30238,30270,30269,32285,32285,32285,32284,34331,36347,32088,2
7862,23603,23571,21522,23603,27862,32088,36315,34332,34332,30204,32317,30237,30269,30
270,30270,30269,30269,30269,30270,30269,30269,30269,32285,34331,42554,61439,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,63487,63487,42554,34298,32317,30237,30
269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30238,32285,32284,343
32,34299,27829,15086,6601,2342,2309,2276,2276,2276,2276,2308,2341,4487,12940,23635,342
66,34332,32284,32285,30237,30269,30270,30269,30269,30237,30269,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,30270,30269
,30237,30270,30270,30270,32285,32284,34331,34299,27830,15118,6600,2342,2309,2308,2276,
2276,2276,2309,2343,6569,15053,25748,34299,32284,32285,32285,30237,30270,30270,30269,
30269,30270,30269,30270,30270,30269,34332,36248,59359,63487,65535,65535,65535,65535,6
5535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,63487,61439,34233,34331,30237,30269,30
269,30269,30270,30269,30270,30270,30270,30269,30270,30270,30269,32285,32252,36379,214
91,6601,2309,4324,4292,4324,4291,6339,6339,6339,6339,4290,4323,4292,4324,2277,4487,172
31,34233,34300,32284,32285,30270,30270,30269,30269,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30269,30270,30270,30237
,30237,30270,30237,32252,36379,23538,6633,2309,2276,4291,4291,4291,6339,4291,4291,4291
,4291,4323,4323,4323,2309,4520,19410,34266,34332,32284,30237,30269,30269,30270,30269,3
0270,30270,30270,30269,32284,34234,53053,63487,65535,65535,65535,65535,65535,65535,65
535,65535,

65535,65535,65535,65535,65535,65535,65535,63487,59391,34233,32284,30269,30270,30
269,30269,30270,30270,30270,30270,30270,30237,30237,32285,32285,34332,27862,6601,2277
,4324,4291,6338,6339,6339,6338,6338,6338,6338,8386,6338,6339,6339,6339,6339,4323,2276,6
503,23603,34299,32284,30269,30269,30270,30269,30270,30270,30270,30270,30270,30270,302
70,30270,30270,30270,30270,30270,30270,30270,30269,30269,30269,30269,30237,30269,3231
7,30237,34299,27861,6601,2309,4323,6339,6339,6339,6339,6338,6338,6338,6338,6339,6338,6
338,6370,6370,4323,4323,2277,6601,25748,34331,30236,30269,30269,30269,30269,30270,302
70,30269,30270,30237,34331,44699,61439,65535,65535,65535,65535,65535,65535,65535,6553
5,

65535,65535,65535,65535,65535,65535,65535,61439,53020,34298,32284,30270,30270,30
269,30269,30270,30270,30269,30270,30269,30237,30237,30269,32252,15217,294,2276,6371,6
338,6338,8386,6338,6338,6338,8386,6338,6338,8386,6338,6339,6339,8386,6338,6339,6339,42
92,2309,10892,34332,30237,30269,30269,30269,30269,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30269,30269,30269,30236,
30236,15184,294,2276,4291,6338,6338,6338,8386,6339,6338,6338,6338,6338,8387,6338,6338,
6338,6338,6338,6306,6340,4292,261,15151,34332,30237,30269,30269,30269,30270,30269,302
69,30269,30269,32284,36280,61439,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,61439,42487,34299,32284,30238,30270,30
269,30269,30270,30270,30269,30270,30238,30270,32285,32317,32252,8780,261,6372,6338,63
38,6338,6339,6338,6338,6338,6338,6338,6338,6338,6338,6338,6338,6338,6338,6339,633
9,4324,262,10991,32285,30270,30270,32286,30270,30270,30269,30270,30270,30270,30270,30
270,30270,30270,30270,30270,30270,30270,30269,30269,30269,30270,30269,32284,322
52,8779,2277,4324,6338,6338,6338,6339,6338,6338,6338,6338,6338,6338,6338,6338,6338,633
8,6338,6338,6307,6339,6339,2244,262,15281,32284,30269,30269,30269,30269,30270,30269,30
269,30270,32284,34167,59359,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,61439,36280,34332,32285,30270,30270,30
269,30269,30269,30270,30269,30237,32285,30237,30270,30237,32284,34299,4520,4324,6338,
6338,6338,6339,6338,8354,8354,8354,8386,6338,6338,6338,6338,6338,6370,6338,6338,6339,6

339,2276,8747,34299,32317,30237,30270,30238,30269,30269,30238,30269,30269,30269,30269
,30269,30269,30269,30270,30270,30270,30270,30269,30237,30237,30270,30269,30237,32317,
32285,36314,4456,2276,6339,6338,6338,6306,6339,6338,6338,6338,6338,6338,6338,6338,8355
,6339,6339,6338,6339,6339,4291,2309,10892,32218,30237,30237,30269,30237,30269,30269,30
269,30269,30269,30269,34265,55165,63487,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,61439,34200,32316,30237,30270,30270,30
269,30269,30269,30270,30237,30237,32285,30269,30237,30237,32317,34300,34265,4455,2275
,6339,6306,6338,6338,8386,8386,6338,6339,6338,6338,6338,6338,6338,6338,6370,4290,4291,4
291,8746,34233,32284,30269,30237,30237,30237,30269,30269,30238,30269,30269,30269,3026
9,30270,30238,30270,30270,30270,30269,30270,30269,30238,30237,32285,32285,32285,30237
,32285,34299,34201,2342,4324,6339,6338,6338,6338,6370,6338,6338,6338,6338,6338,6338,83
86,6338,6338,6338,6339,6340,2276,10859,34234,32284,30237,30237,30237,30269,30237,3026
9,30270,30269,30269,30269,34331,48924,63487,65535,65535,65535,65535,65535,65535,65535
,

65535,65535,65535,65535,65535,65535,63487,61439,36281,32316,30269,30269,30270,30
269,30269,30269,30237,30237,32317,34299,17395,32218,32284,32285,32317,34332,32087,442
2,4324,6339,8386,6338,6370,6338,6338,6339,6370,6338,6339,6338,6338,8386,6371,4323,2276,
8713,34201,34332,32317,30237,32285,34332,32252,30236,30237,30238,30237,30269,30269,30
269,30269,30270,30270,30270,30270,30270,30269,30269,30269,32252,34332,17363,34299,322
84,32285,32285,34331,32054,2342,4292,6339,6338,6338,6338,6338,6338,6338,6338,6338,6338
,6370,6370,6370,6371,6339,2276,8746,34266,34332,32284,30237,30236,32284,32284,32285,30
238,30270,30269,30269,30269,34331,42554,63487,65535,65535,65535,65535,65535,65535,655
35,

65535,65535,65535,65535,65535,65535,65535,57246,36313,32316,30269,30269,30270,30
269,30269,30270,30269,32285,32284,29975,2342,10892,34267,32285,30237,30236,34332,2789
4,2342,4291,6338,6338,6338,6338,6338,6338,6370,6339,6339,6338,8386,8386,4291,2276,6633,
34201,32284,32285,30237,32285,32251,19475,23603,32284,32285,30238,30270,30269,30269,3
0269,30270,30270,30270,30270,30270,30270,30269,30269,30269,32252,27829,295,13038,3429
9,32284,30237,32285,34331,27797,2309,4291,6338,6338,6338,6370,6338,8386,6338,8386,6338
,6338,6338,6338,4291,2244,8714,34233,32285,32285,30237,32317,34332,17363,25750,34300,3
0238,30270,30269,30237,30269,34300,34200,61439,63487,65535,65535,65535,65535,65535,65
535,

65535,65535,65535,65535,65535,65535,65535,50973,34298,32284,30237,30270,30269,30
269,30269,30269,32285,32252,34300,4521,4324,2276,15183,34331,32284,30237,32285,34332,
23635,2309,4323,6339,6306,6338,6338,6338,6339,8355,6339,6338,8386,6339,4324,6569,32153
,32283,32285,30269,32317,34299,21555,294,2375,32153,34332,30237,30269,30269,30269,302
69,30269,30269,30269,30269,30270,30270,30270,30269,30236,34266,6568,2277,2309,19344,3
4299,30269,30237,32284,34299,21522,2309,4291,6339,6338,6338,6338,8386,6338,6338,6338,6
338,8386,6339,4324,6633,32186,32284,30269,30237,32284,34299,21490,295,4489,34234,3225
3,30269,30269,30269,30269,32252,32056,61407,63487,65535,65535,65535,65535,65535,65535
,

65535,65535,65535,65535,65535,65535,63487,48827,34298,32284,30237,30270,30270,30
269,30237,30269,32284,34332,21523,2277,6339,4291,2309,21490,34299,32284,32285,32285,3
4331,21457,2309,4323,6339,6306,8386,6338,6338,8355,6307,6307,6339,4292,6536,34169,3433
1,32285,30269,32316,34299,23635,2342,2276,2276,17166,34299,32285,30270,30270,30269,30
269,30270,30270,30269,30270,30270,30270,30270,32285,34331,19409,2276,4291,4323,2309,2
5651,34331,32284,32285,32284,34331,19312,2277,4323,6371,6338,6338,6338,6338,8387,6339,
8386,6307,4292,6601,34234,32284,30236,30237,32285,36380,21490,229,2277,2277,21393,343
00,30237,30237,30269,30269,32284,32088,59326,63487,65535,65535,65535,65535,65535,6553
5,

65535,65535,65535,65535,65535,65535,63487,44666,34299,32317,30270,30270,30270,30
269,30269,30270,32284,34298,6568,4323,6306,6339,4324,2309,27796,34299,30204,32285,322

84,34299,15184,2309,4324,6339,8386,8386,6338,6339,6339,4291,2276,4520,32121,34300,3228
5,32285,32252,34299,25683,2309,4323,4291,4323,2277,34233,32316,30237,30270,30270,3026
9,30270,30269,30270,30269,30238,30270,32285,32285,34266,4520,4324,6338,6338,4324,4390,
27861,34299,32285,32317,32284,36347,15085,4324,4291,6339,6306,6338,6338,6338,6339,429
1,4324,6601,32153,32316,32285,30237,34332,36379,23570,2309,4323,4291,4324,4422,36314,3
2285,30237,30269,30269,32284,32153,57246,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,63487,44667,34299,32285,30237,30270,30269,30
269,32317,30270,32284,23700,2309,6339,6339,6339,6339,4292,4423,32056,32252,30269,3228
5,32284,34331,15019,2276,6339,6338,8386,6338,6338,6339,4323,6535,32087,32283,32285,302
37,30237,34332,23668,2342,4323,6338,6338,4291,2276,19344,32283,30237,30270,30270,3026
9,30270,30270,30269,30270,30270,30270,32286,32285,23603,2310,4323,6338,6338,6339,4324,
4455,32088,32284,30269,30237,34332,36314,14954,4324,4291,6338,6338,6338,6338,6339,230
8,6600,32120,32284,30269,30269,32285,36379,23571,2310,4323,6338,6338,4291,2276,23603,3
2284,30269,30237,30237,32284,32185,57214,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,63487,44698,34330,32316,30270,30270,30269,30
270,30270,32284,34331,10892,4292,6338,6339,8355,6338,6338,4324,6502,34232,32284,30237,
30237,32284,36347,10860,2276,4322,8385,8386,6339,2276,4488,30007,32284,30269,30237,32
285,34299,25715,2373,4291,6370,6338,6339,6339,4292,2407,34330,32284,30269,30270,30269,
30238,30269,30269,32285,30238,28221,32284,36347,10827,4324,6338,6338,6338,6370,6339,2
308,6600,36314,32284,30301,30269,32252,36281,8714,2277,6339,8354,8386,4291,4324,4552,3
2088,32252,30237,30237,32284,34331,25650,2341,4291,6338,8386,6338,6339,4324,6600,3429
8,32285,30270,30269,32316,32218,55165,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,63487,44666,34330,32317,30270,30238,30237,30
270,30270,32284,34265,4455,4324,6338,6339,8355,6338,6370,4323,4324,8713,34266,32284,32
317,32284,34331,34233,8714,4324,6339,6339,4324,4455,32022,34331,32285,30237,32284,343
32,25749,4422,4324,6338,6338,6338,6338,6338,4291,228,30039,32316,30269,30270,30269,302
70,30237,30269,32285,30238,30269,32316,32153,2374,4324,6338,8386,6370,6338,6339,4323,4
324,10795,34299,32284,32317,32284,34299,34201,6569,4324,6339,6339,4324,6503,32087,343
31,32285,32317,32285,34299,23636,4390,2276,4291,6338,8386,8386,6339,4291,196,34201,322
85,30270,30269,32284,34298,53085,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,63487,44667,32251,32316,30270,30238,30269,30
269,30270,32284,27894,2309,6339,6338,6338,8387,6338,6338,6339,4291,2276,12973,34299,32
252,32284,32317,34300,32120,6536,2277,2276,2342,29975,34299,32284,30237,30237,34332,2
7829,2374,4292,6339,6338,6338,6338,6338,8386,6339,2308,21490,34332,30269,30269,30269,3
0238,30269,30269,32285,30238,30270,32284,27829,2309,4291,8386,8386,6370,6370,6339,633
8,6339,2245,15086,34299,32284,32285,32284,34300,30008,4455,2276,2277,2375,32088,34332,
32284,30237,32284,34299,27764,2342,4292,4291,6339,6338,6338,6338,6338,6339,2244,27828,
32252,30269,30269,32284,34299,53085,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,63487,48860,34331,32284,30270,30270,30269,30
269,30237,34332,21489,2276,6339,6338,6338,8387,6338,6338,6338,6338,4259,2277,17232,342
99,32284,30269,32285,34300,29975,2408,327,30007,32284,32284,30237,30237,32252,29942,2
374,4324,6339,6338,6338,6338,6339,6338,8386,6338,4324,10924,34331,32317,30269,30270,30
270,30269,30269,30237,30238,30237,32251,19377,2276,4291,8386,8386,6370,6339,6339,8386,
6338,4291,2309,19377,34299,32285,30269,30237,34331,27894,327,2408,32088,34300,32286,3
2317,32316,34331,27861,2342,4292,6338,6338,6338,6307,6339,6338,6338,6338,4324,17199,32
251,30269,30269,32284,34299,55165,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,63487,51005,34298,32284,30270,30270,30269,30
269,30269,34331,15086,2276,6338,6338,6338,8387,6338,6338,6338,8386,8386,4291,2309,2152
3,34299,30237,30269,32285,32252,25815,27960,32251,30269,30237,30237,32252,30008,2342,
4324,6370,6338,6338,6338,6339,6339,6338,8386,6338,4323,4552,34299,32284,30269,30270,30
238,30269,30269,30237,30238,30237,34299,12973,4324,6338,8386,8386,6338,6339,6338,8386,
6338,6339,4323,2309,23668,34332,30237,30237,32285,32251,25782,30008,32252,32285,30238

,32285,32283,27829,2341,4324,6339,8386,8386,6338,6339,6339,6338,8386,6338,4323,8746,34299,30269,30237,32284,34267,55165,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,63487,57279,34266,32284,30269,30269,30269,30270,30269,34299,10826,4324,6338,6338,6338,8387,6338,6370,6338,6338,8386,6338,4291,2310,29941,32284,32284,32285,32285,32252,30204,32285,30269,32317,34332,29975,2342,4356,6371,6338,6370,6338,6338,6370,6339,6338,6338,6338,4324,4423,34298,32284,30269,30269,30270,30269,30269,30237,30269,30237,34331,8714,4323,6338,8386,6338,6338,6338,6338,6338,6338,6338,6370,4324,2342,30007,32284,32317,32317,32285,32252,32284,32285,30237,32285,34332,27893,2341,4324,6338,8386,8386,8386,6338,6338,6338,6338,6338,6370,4323,4488,34267,30269,30269,32284,34234,55166,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,59359,34233,32316,30269,30270,30269,30269,32285,36347,8713,4324,6338,6338,6338,8387,8386,6338,6338,6339,6338,6338,6338,2243,2342,32153,34331,30237,30270,30269,30237,30237,32317,34332,30007,4390,4292,6338,6338,6338,6339,6338,8386,6338,6338,6339,6338,6338,4291,2342,34234,32284,30269,30269,32286,30269,30269,30237,30269,30269,34331,6601,4323,6339,8386,6338,6339,6338,8386,6339,6338,6338,6338,6339,4324,2406,32185,32251,32285,30270,30269,30269,30237,30237,34332,29942,2342,4323,6338,6338,6338,6370,6338,6338,6338,6338,6338,6338,6338,4323,4423,36315,32285,30269,32284,32153,57246,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,61439,34200,32284,30269,30270,30237,30269,32285,36347,6633,4324,6338,6338,6338,8387,8386,6338,6338,6339,6370,6338,6370,4291,2276,2440,32186,32317,30237,30237,30269,30237,32284,30041,2440,4324,6338,8386,6338,6338,6339,6338,8386,8386,6338,6339,6338,6338,4324,2310,34234,32284,30269,30269,32285,30269,30269,30269,30237,30269,36379,6601,4324,6339,6338,6338,6339,6338,8386,6339,6338,6338,8386,8386,6339,2276,4553,34299,32285,30270,30270,30269,30237,30236,27959,2407,2308,6339,8386,6338,6339,6338,6338,6338,6370,6338,6338,6338,6338,4323,4423,34267,30237,30269,32252,30041,59327,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,61439,38327,34299,30237,30269,30269,30237,32285,34331,8713,4292,6338,6338,6338,8387,8386,6338,8355,8386,6338,6338,6339,4323,4324,2375,32121,30236,30269,30269,30270,30269,32316,15281,262,4291,8386,8354,6338,6339,6339,6339,6338,6338,6338,6338,6338,6338,4291,2374,34266,32285,30270,30237,30269,30237,30270,30270,30269,32284,36346,6601,2276,6339,8354,8386,6338,6338,6338,8386,6338,6338,6338,6306,4291,2309,2408,32154,32284,30269,30269,30269,32285,32285,13072,2342,4323,6338,6370,8386,6339,6338,6338,6338,6338,6338,6338,6338,6338,4291,2407,34267,32285,30237,32283,32120,59359,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,61439,44633,34331,32317,30269,30269,30238,30237,34331,10859,4324,8386,6338,6338,8387,8387,6338,6338,6338,6338,6338,6339,4324,4390,32088,34299,30237,30269,30270,30270,30270,32285,34299,13005,2276,6339,6338,6338,6338,6338,8387,6338,6338,6338,6338,6338,6338,4323,4423,34267,32285,30238,30269,30269,30237,30270,30270,30270,32285,34299,8747,4324,6339,8354,6338,6338,6338,8386,6338,6338,6338,6338,6339,4324,4455,30040,34332,32285,30269,30269,30269,30269,32285,36315,10860,2276,4291,6370,6338,6339,6338,6338,6338,6338,6338,6338,8386,6338,4323,4552,34299,32285,30269,34363,34201,61439,63487,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,63487,57246,34266,30236,30269,30269,30237,32285,34331,17166,4292,6338,6338,6338,8387,6338,6338,6370,6338,6338,6339,4324,4390,29941,32252,32285,30269,30269,30269,32285,30237,32285,32285,36314,10795,2244,4323,6338,6338,8386,8386,8386,6338,6338,6338,6371,6338,4324,6601,34299,32285,30270,30269,30269,30269,30270,30269,30270,32284,34299,13037,4324,6339,8386,6338,6370,6338,8386,6338,6339,6339,6339,4324,4422,29975,32284,30237,30269,32285,32317,32317,32285,30269,32284,34233,8682,2276,4323,6339,6339,6338,6338,6338,6338,6338,6338,6338,6306,4323,8811,36380,32253,30236,34331,42587,63487,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,63487,61439,34232,32316,30237,30269,30

269,30237,32284,23570,2277,6339,6338,6338,8387,6339,6338,6338,6338,6339,4291,2342,2782
9,34331,32285,30237,30269,32284,30074,34300,32284,32285,30237,32252,34234,6633,4324,4
290,6338,6338,8386,6338,6338,6338,6338,6339,6338,4324,13005,34299,32285,30269,30270,30
270,30269,30270,30238,30270,30269,34332,21458,4324,6339,6338,6338,6370,6338,8386,8354,
6338,6339,4324,4422,29942,32252,30237,30237,32285,34332,27993,32252,32284,30237,30269
,34332,34168,6568,2276,6339,6338,6338,6338,8386,8386,8386,6338,8387,6339,2276,17296,34
300,32285,32317,34330,50973,63487,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,63487,63487,38360,34299,32285,30269,30
269,30269,32284,30007,2309,6339,6338,6338,8355,6339,6338,6306,6339,2243,2341,25748,322
51,30268,32285,32285,34332,30040,4554,13070,34299,32284,32317,32317,32252,34201,4455,
4292,6339,6338,6338,6338,6338,6338,6338,6339,6338,2309,23636,34332,30269,30269,30270,3
0270,30269,30269,30269,30270,30269,34332,29943,2277,4291,6338,6338,6370,6338,6338,835
3,6338,4324,2342,27862,34300,30237,30237,30237,32252,30008,4521,17297,34299,32284,302
69,32285,34300,32056,2374,4323,6338,6339,6339,6338,6338,8386,6338,6339,4291,2276,27894
,34332,30237,32284,34232,55166,63487,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,63487,50972,34267,32285,30236,30
269,30269,32285,34266,4488,4292,6338,6339,8387,6338,6338,6339,4324,2309,25684,34299,32
285,30269,32285,34332,32088,6568,2277,2277,17231,34331,32284,32285,32317,32252,32055,
4423,4324,6339,6339,6338,6338,6338,6338,6339,6371,196,32120,32284,30269,30269,30269,30
269,30269,30269,30269,30302,30269,32252,34267,4455,4292,6339,6338,6338,6370,6338,6338,
4292,2277,27829,34331,32285,30238,32285,34300,32087,4488,2309,2277,21425,36347,32285,
32285,32285,34332,29974,2341,4324,6339,6339,6338,6338,6338,6338,6339,4292,229,34265,32
285,30237,32284,34199,61439,63487,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,63487,59358,34201,34333,30237,30
269,30270,32285,34332,15086,4324,4290,6339,8387,6338,6338,4291,260,25715,34299,34332,3
2285,30237,34332,32153,6600,4324,6339,4323,2277,21425,36347,32284,30269,30269,34331,2
7861,2342,4323,6339,6339,6338,6338,6338,6338,4323,6601,34299,32285,30269,30269,30270,3
0269,30269,30269,30269,30269,30269,32285,34332,13005,4325,6339,8386,6338,6370,6370,43
24,229,27829,34331,32316,32316,32285,34299,32120,4488,2308,4323,4324,2277,23538,36379,
32284,32285,30269,32251,27797,2310,6371,6339,6338,6338,6338,6338,6371,4292,10827,3637
9,32284,32285,32284,40505,63455,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,61439,40506,34299,32285,30
270,30270,30237,32284,27894,2310,4291,6338,6338,6339,4291,2276,23635,34331,32284,3228
5,32285,32252,34201,6600,4324,6339,8386,6339,4324,2341,23570,36347,32285,30269,32317,3
4299,25683,2309,4324,6339,8386,6338,6339,6339,2277,23603,32284,30269,30269,30269,3026
9,30269,30269,30269,30269,30237,30269,30237,32284,27829,2309,6339,8354,6338,6371,4323,
261,25716,34300,32285,30269,32284,34331,34201,4488,2276,4290,6338,6338,4324,2310,2776
3,36379,30204,30269,32285,34299,21523,2276,4324,6339,8386,6338,6338,4291,2244,27829,34
332,30237,32285,34298,48924,63487,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,53053,34266,32316,30
238,30269,30269,32316,34331,8682,4324,6338,6338,4291,2276,23538,34298,32284,30237,302
37,34332,32186,4519,2277,6371,6338,6338,6338,4323,2275,2341,27764,34332,30236,30269,32
252,34299,21425,2309,6371,6338,6338,6339,4323,2342,34234,32284,30270,30238,30270,3026
9,30269,30270,30270,30269,30270,30269,30270,32285,34299,6633,4324,8386,6338,4323,228,2
3603,34299,32252,30237,30269,34332,34200,4488,2277,4324,6338,6338,6338,4291,4324,2342,
27862,34300,30237,32317,32284,34299,19311,2277,6340,6338,6338,4290,4291,4488,34266,32
252,30237,32284,34200,57246,63487,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,59359,36313,34332,32
285,30270,32317,32285,34332,27861,2309,4291,4323,2277,21457,34298,34332,32285,32317,3
4300,34233,4553,4324,4323,6338,8386,8386,6338,6338,4323,4324,4358,29942,32251,32284,32
317,32285,34299,17199,2276,4323,6339,6340,4324,21425,36379,30236,30269,30270,30270,30
269,30270,30270,30270,30270,30270,30270,30270,30237,34332,25684,2308,4291,4324,229,23

570,34299,32284,32285,30269,32284,34201,6601,4324,4291,6339,6338,8386,6338,6338,6339,4
324,2375,32055,34300,32284,32285,32285,36346,15086,2276,6339,6338,4323,2276,25683,342
99,32285,30269,34332,38426,61407,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61439,48860,34266,32
284,30269,30269,30270,30236,34331,8746,2308,2277,17296,34299,32284,32285,30237,32284,
34234,6633,2276,6339,6339,6306,8386,8386,8386,6338,6338,6339,4324,2342,34168,32284,322
85,32285,32285,34299,10892,2276,4324,2276,6535,34234,32252,30269,30270,30269,30269,30
269,30270,30270,30270,30269,30270,30270,30270,30269,30237,36379,8746,2309,197,19409,3
4299,32285,32317,30237,32284,34233,6601,4324,6338,6338,6338,6305,8386,8386,8386,6338,6
339,4324,4455,34201,32252,32285,30238,32284,34299,10827,2276,4323,2276,6634,34266,322
84,32317,30236,34266,48892,63487,65534,65535,65535,65535,65535,65535,65535,65535,6553
5,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,59359,36313,34
331,32317,30269,30238,30237,30268,32185,2376,17231,34331,30236,30237,32317,32252,3426
6,6601,4324,6339,8355,6338,6338,6370,6338,6338,6338,8354,8386,6339,4324,2374,34233,322
84,30269,30237,32252,36379,8714,2277,230,29877,34332,32285,30269,30270,30269,30269,30
269,30270,30270,30270,30269,30270,30270,30270,30269,30270,34332,32120,295,17329,34299
,32284,30270,30237,32252,36314,6601,4324,6339,8386,6338,6338,6338,6338,6338,6338,6338,
8386,6339,4292,4487,36314,32252,30237,30270,32284,36314,6568,228,294,29975,32284,3026
9,32317,34332,34233,57279,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,61439,48892,36
346,32284,30269,30270,30270,30269,32284,27895,34299,32284,30269,32285,32252,34234,868
1,4324,4291,6338,6339,6338,6338,6338,6370,6338,6339,8386,8386,6338,6338,4324,4520,3423
3,32284,32317,30269,32284,34234,6634,17265,34299,32285,30237,30270,30270,30269,30269,
30270,30270,30270,30270,30270,30270,30270,30270,30270,30237,32285,34332,27960,34364,3
2284,30237,30237,34365,36314,6633,4357,4291,8386,6338,6338,6338,6338,6339,6339,6338,63
38,8386,6338,6339,2244,6601,34266,32252,30237,30237,32251,34265,4553,21490,34332,3228
5,30269,32316,34298,44666,61439,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,61439,38
361,34331,30268,30269,30270,30237,30237,32317,30236,32317,32285,32252,36314,8681,4356
,6339,6339,6338,6338,6338,6338,6339,6339,6338,6338,6339,8386,6338,6338,6339,2276,8714,3
4233,32251,32285,30269,32284,32186,34300,32285,30269,30270,30270,30269,30269,30269,30
269,30270,30269,30270,30270,30270,30270,30270,30238,30270,30237,30237,30236,32284,323
17,32285,34332,34266,8714,4324,4259,6338,6338,6338,6338,6338,6338,6338,6338,6339,6339,
6339,6338,6338,6339,2276,8779,36314,32284,32285,32317,32284,30106,32252,32284,30237,3
0237,34332,34232,57246,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65534,63487,53
086,34266,32284,30269,30269,30237,30270,28221,30269,30237,32284,34267,8714,4324,6339,
8386,6338,6338,6370,6338,6338,8386,8386,8386,6338,6339,6339,6338,6338,8386,6339,2276,1
0859,34265,32316,30270,30269,32252,30237,30270,30269,30270,32286,32285,30269,30269,30
270,30270,30269,30270,30270,30270,30269,30270,30269,30237,30269,30270,30237,30237,322
85,30236,34266,8747,2276,6339,6338,6370,4258,6339,6338,6338,8386,8386,6338,6338,6339,6
339,6370,6338,6338,4323,2309,15052,34299,32284,30269,30238,32285,32285,30269,30237,32
285,34299,42586,61439,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63486,63
487,44634,36347,32284,30237,30269,30270,30269,30269,30269,34332,8812,261,4323,6338,63
38,6338,6339,6339,6338,6338,6338,6339,6339,6339,6338,6338,6339,8354,8386,6339,4291,227

6,15085,34299,30237,30237,32285,30237,28222,30269,30269,30269,30269,30269,30269,30237
,30269,30269,30238,30237,30270,30269,30237,32285,32285,30269,30269,30269,30269,32317,
32284,10925,2309,4324,6306,6338,6306,6339,6339,6339,6338,6338,6339,6339,6339,6339,6338
,6338,6338,6338,6339,4291,2276,15184,34299,30237,30270,30269,32285,30269,30270,32252,3
4200,57246,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,59359,36281,34332,32284,30237,30270,30270,30270,32317,32284,23701,4456,2244,6371,
6338,6339,6339,6339,6338,6338,6338,6338,6338,6339,6338,6338,8386,8354,6306,4291,4324,2
276,2408,25880,30270,30238,30269,30269,30270,30269,30237,30269,30269,30237,32285,3228
5,32284,32284,32284,32285,30237,30269,30270,30237,30238,30270,30269,30269,30269,30237
,32285,23668,2408,2277,6372,6338,6338,6338,6338,6338,6338,8386,6338,6338,6338,6338,633
8,6338,6338,6338,6339,2276,229,2473,28025,30237,30269,30237,30269,30238,32317,34331,44
667,61439,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,61439,53086,34233,34331,32285,30238,30270,30269,32285,32285,34300,34201,15054,230
9,2276,6371,6307,6339,6338,6338,6338,6338,6338,6338,6338,6338,6339,6339,4291,4356,2277,
10859,30007,32284,30238,30270,30269,30269,30269,30237,30270,30270,32317,32316,34331,3
4298,30007,30040,34234,34332,32284,32317,30237,30238,30238,30270,30270,30269,30269,30
269,30269,34331,32121,13005,2309,4323,4290,6370,6338,6338,6338,6338,6338,8418,6338,633
8,6338,6339,6339,4323,2276,2309,12973,30041,34332,32285,30270,30269,32285,32285,34332,
34232,57311,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65534,65535,65535,65503,65
535,63487,61439,46780,34265,32283,30269,30270,30270,30269,30269,32285,32253,34268,299
75,13005,4422,2244,4324,4324,4323,4323,6339,4291,4291,4323,4323,4324,2244,2309,10859,2
7829,36347,34300,32285,30270,30269,30269,30269,30269,30238,30269,32285,34332,34234,17
232,4488,2342,2342,4520,17264,34233,32251,32317,30269,30269,30270,30270,30269,30269,3
0269,30269,32317,32284,34300,29975,13037,4422,2243,4324,4324,4323,4323,4291,6339,4291,
4323,4323,2276,2244,2342,12972,27894,34299,32284,32285,30238,30270,30270,32285,32284,
34233,51005,61439,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,63487,59359,42586,34298,32284,30269,30270,30269,30269,30269,32285,32285,322
84,34299,32153,23603,13005,6568,2342,261,229,2276,2309,261,4519,10892,21490,30007,3429
9,34332,32285,32285,30238,30270,30270,30269,30269,30269,30270,32317,34332,32121,8682,
2277,4292,6339,4291,2276,2309,6633,34169,32252,32317,30269,30237,30269,30238,30270,30
270,30270,30269,30237,32285,32284,34299,32120,23603,12973,6601,2342,261,229,228,2309,2
341,6568,12973,23603,32120,36347,34332,32285,30269,30269,30269,30270,30269,32284,3429
9,42586,61407,63487,65534,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,63487,57279,38426,34331,32284,30237,30269,30269,30269,30269,30269,302
69,30269,32284,34332,34299,34266,32121,30007,29975,29942,30007,32120,34233,34299,3430
0,32284,32317,30237,30238,30269,30269,30269,30269,30269,30269,30269,30269,32284,34298
,8714,4325,6339,8355,8355,6339,6371,4323,4324,6536,34201,32284,32285,32285,30270,30270
,30270,30269,30269,30269,30269,30270,30269,30269,30236,32252,34299,34266,32153,30040,
29975,29975,30007,32153,34266,34299,34332,32284,32284,30237,30269,30269,30269,30269,3
0269,30236,34331,36345,59327,63487,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,63487,57279,36280,36379,32284,32285,30269,30270,30269,30269,302
69,30269,30269,32285,32317,32284,32284,32284,32284,32284,32284,32252,32284,32285,3023

7,30269,30269,30270,30238,30270,30269,30269,30269,30270,30270,30269,30269,32251,17231
,2276,6339,6339,6338,8387,6339,6338,6338,6339,4325,8682,34266,32285,32285,30269,30270,
30270,30270,30238,30270,30269,30238,30269,30269,30269,32317,32285,32284,32284,32284,3
2284,32284,32284,32284,32285,30237,32285,32317,30269,30269,30270,30269,32285,32
284,34300,34233,53053,63487,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,61439,55166,36314,34332,32285,30238,30270,30269,30269,302
69,30269,30237,32285,30270,30269,30237,30237,30269,30269,30269,30269,30269,30270,3027
0,30269,30269,30270,30270,30269,30269,30269,30237,30270,30270,30269,32252,36347,2375,
4323,6338,6338,6338,8355,6339,6370,6337,6306,4260,2277,19377,34332,32253,30269,30270,3
0270,30269,30270,30270,30270,30237,30269,30269,30269,32285,30237,30269,30269,30269,30
269,30269,30270,30269,30269,30270,30270,30238,30269,30269,30269,30270,30269,32285,343
32,34267,48925,61439,65535,63487,65535,65535,65535,65535,65535,65535,65535,65535,6553
5,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,63487,55166,36281,34332,30204,30270,30270,30237,302
69,30269,30269,30238,30269,30269,30269,30270,30269,30269,30270,30269,30269,30270,3027
0,30269,30270,30270,30270,30270,30269,30269,30270,30270,30269,30269,32252,25749,2277,
6339,8354,6339,6338,4291,4291,6339,6338,6338,6339,4324,4455,34234,32284,30237,30269,30
270,30269,30269,30269,30270,30269,30269,30269,30269,30269,30270,30270,30270,30269,302
69,30270,30270,30270,30270,30270,30270,30270,30269,30269,30269,30270,30237,34332,3631
3,46779,61439,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,63455,53150,34233,34332,32284,30237,30269,302
69,30269,30269,30237,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30269,30270,30270,30270,34331,13005,2244,
6339,6306,6371,4323,2276,2275,4324,6339,6338,6339,4324,2309,27927,32284,30237,30270,30
270,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,302
70,30270,30270,30269,30238,30270,30269,30269,30269,30269,30269,30269,32284,36379,4050
6,59359,63487,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65534,65534,61439,55198,36313,36379,32252,30269,282
22,28221,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,34331,8747,2276,6
339,6339,4324,228,19278,27730,8681,2308,4291,4323,4324,2277,27927,34332,30237,30270,30
270,30269,30269,30270,30269,30269,30270,30270,30270,30270,30269,30269,30270,30270,302
70,30270,30270,30270,30269,30237,30269,30269,30269,30269,30269,32284,34298,40539,5935
9,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,63487,61439,57278,36280,36379,32284,302
37,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,32286,34332,8812,2277,4
324,2309,2375,29942,34299,34299,36314,13005,229,2277,2277,6601,32219,32284,30237,3027
0,30270,30270,30270,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270
,30270,30270,30270,30269,30237,30269,30269,30269,30237,32317,32284,36346,42619,59359,
63487,65534,65534,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65

535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,59326,38361,34299,322
84,32316,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,32285,21621,4521,
2408,10860,32121,34331,32284,32285,32252,34299,19377,6634,12973,32121,34332,32285,302
37,30269,30270,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30269,30269,30270,30270,30269,32316,34332,34299,44666,59359
,63487,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,59359,42586,342
65,34331,32285,30237,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30270,30237,30270,32285,32284,34267
,34267,34300,32284,32285,32318,30237,30269,30236,34332,34331,32251,34332,32285,30237,
30270,30269,30270,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,3
0270,30270,30270,30270,30270,30269,30270,30269,30269,32284,34331,36313,44764,61407,63
487,65535,65534,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,61439,488
92,34297,32284,30238,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,3027
0,30270,30270,30270,30270,30270,30270,30270,30270,30237,30270,30270,30237,32285,32285
,32253,32285,30237,30237,30237,30269,30269,30237,32316,32285,32317,30237,30269,30270,
30270,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,3
0270,30270,30270,30270,30270,30269,30269,30237,32284,34331,36281,50972,61439,63455,65
535,65503,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65503,65535,65535,65535,614
39,44700,32251,30238,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,3026
9,30269,30270,30270,30269,30269,30270,30270,30269,30270,30237,30269,30270,30238,30237
,30237,30237,30270,30237,30269,30269,30269,30269,30269,30237,30270,30270,30270,30270,
30270,30269,30269,30269,30270,30269,30269,30270,30270,30270,30269,30270,30270,30270,3
0270,30270,30270,30270,30270,30269,30269,32285,34299,36346,53118,61439,63486,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,53085,342
66,30269,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,3026
9,30269,30269,30269,30237,30237,30269,30270,30269,30269,30269,30269,30238,30238,30270
,30238,30269,30269,30270,30269,30269,30269,30269,30269,30270,30270,30270,30270,30270,
30270,30269,30270,30269,30270,30269,30269,30269,30269,30237,30269,30270,30270,30270,3
0270,30270,30270,30270,30270,30269,30237,32285,32185,55198,63487,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,63487,57279,34233,32284,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,3026
9,30269,30269,30269,30269,30269,30269,30269,30270,32317,30269,30269,30269,30269,30270
,30270,30270,30238,30270,30270,30269,30270,30269,30269,30269,30269,30269,30269,30269,
30269,30269,30269,30270,30270,30269,30269,30270,30269,30269,30269,30269,30269,30270,3
0270,30270,30270,30270,30270,30269,30269,34333,36313,59359,63487,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,59359,34265,32284,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30269,30270,30270,30270,30269,30269,30269,30269,30269,30238,32318,30238,30269,30269,30270,30270,30270,30270,30269,30269,32317,30269,30269,30269,30269,30269,30269,30269,30269,30270,30270,30270,30269,30269,30269,30269,30270,30270,30269,30270,30270,30270,30269,30269,30269,34333,38394,61439,65535,

65535,59359,34233,32284,30269,30269,30269,30270,30270,30270,30270,30269,30269,30270,30269,30269,30270,30269,30237,30270,30269,30269,30237,30270,30269,30269,30269,30269,30270,30238,30270,30269,30237,32285,30237,30237,30270,30270,30269,30269,30237,30269,30269,30269,30237,30270,30269,30237,30269,30270,30270,30237,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30269,30268,34332,42555,63487,65535,

65535,59359,36313,32284,30269,30269,30270,30270,30270,30270,30270,30269,30269,30270,30270,30238,30237,30237,32317,32252,32252,32284,30237,30270,30270,30270,28221,30269,30270,30270,30269,30269,32284,34300,32252,32285,30237,30270,30270,30269,30269,30269,30269,30270,30238,32285,32284,32284,34300,32284,32285,30269,30269,30269,30270,30269,30270,30270,30270,30270,30270,30270,30269,30269,34332,42555,63487,65535,

65535,59359,36313,32284,30269,30269,30270,30270,30270,30270,30270,30269,30269,30270,30270,30237,32285,32285,27993,23668,23636,21588,32284,30270,30270,30237,30269,30269,30270,30270,30269,32284,25847,25684,23636,23734,32252,30237,30238,30270,30269,30270,30270,30270,30269,32252,27927,29975,29975,32219,32284,30237,30269,30270,30269,30269,30270,30270,30270,30270,30270,30238,30269,30269,34332,42587,63487,65535,

65535,63487,57246,34233,32284,30269,30269,30270,30270,30270,30270,30270,30270,30269,30269,30270,30269,32285,32284,17362,261,2309,262,34299,32285,30237,32285,30269,30270,30238,30269,30269,34364,10894,2277,2278,6634,34267,32285,30269,30269,30269,30270,30270,30269,30236,34266,2375,2309,230,25782,32284,30270,30270,30270,30269,30270,30270,30270,30270,30270,30270,30270,30269,30237,34332,40539,61439,65535,

65535,63487,55166,34266,32284,30269,30269,30270,30270,30270,30270,30270,30270,30269,30269,30237,30238,30237,34332,19345,2276,2276,261,34298,32285,30238,30269,30270,30269,30269,30270,30269,34363,12940,4324,4324,6601,34298,32285,30237,30269,30269,30270,30238,30269,32285,34266,2342,4323,228,27829,32284,30269,30270,30270,30269,30270,30270,30270,30270,30270,30270,30270,30270,30237,34333,40506,61439,65535,65535,65535,65535,65535,65535,6

5535,65535,

65535,63487,53085,34266,32285,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30237,30270,30269,34332,19345,2276,4324,261,34298,32285,30270,30269,30270,30269,30269,30270,30269,34363,12972,4356,4324,6601,34298,30269,30270,30270,30269,30270,30270,30269,32285,34266,2342,4323,2276,27829,32284,30270,30269,30270,30269,30270,30270,30270,30270,30270,30270,30270,30270,30237,34332,40506,61439,65535,

65535,63487,53053,34266,32285,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30237,30238,30269,34332,19345,2276,4324,261,34298,32285,30237,30269,30270,30269,30269,30270,30237,34363,12972,4324,4324,8648,34298,32285,30270,30270,30269,30270,30270,30269,32285,34266,2342,4291,228,27828,32284,30270,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30269,30269,34332,38425,61439,65535,

65535,63487,50940,34298,32285,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30270,30238,30269,34332,19345,2276,4292,228,34298,32285,30238,30269,30237,30269,30269,32286,30237,34363,12972,4356,4323,6601,34298,30269,30270,30270,30269,30270,30270,30237,32285,34266,2342,4323,228,27829,34332,30270,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30269,30237,34332,38393,61439,65535,

65535,63487,48860,34298,30237,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30237,30238,30269,34332,19345,2276,4292,229,34298,32285,32286,30269,30269,30269,30269,32286,30269,34363,12972,4324,4324,6601,34298,30269,30270,30237,30269,30270,30270,30269,32285,34266,2342,4323,228,27829,32284,30269,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30269,30237,34332,38425,61439,65535,

65535,63487,48892,34298,32285,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30237,30238,30269,34332,19345,2276,4292,229,34298,32285,32286,30269,30237,30269,30269,32286,30269,34363,12972,4324,4324,8649,34298,30269,30270,30269,30269,30270,30270,30269,32285,34266,2342,4323,228,27829,32284,30269,30269,30269,30269,30270,30270,30270,30270,30270,30270,30270,30270,30269,34332,38425,61439,65535,

65535,63487,48860,34299,32285,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,30237,30238,30269,34332,19345,2276,4292,229,34298,32285,32286,30269,30237,30269,30269,32286,30237,34363,12972,4324,4324,6601,34298,30269,30270,30269,30269,30270,30270,30269,3

2285,34266,2342,4323,228,27829,32284,30269,30269,30269,30269,30270,30270,30270,30270,
30270,30270,30270,30269,30269,34332,40474,61439,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,634
87,46780,34299,32285,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,3023
7,30238,30269,34332,19345,4324,4292,229,34298,32285,32286,30269,30270,30269,30269,302
70,30237,34363,12972,4324,4324,6600,34298,32285,30270,30269,30269,30270,30270,30237,3
2285,34266,2342,4323,228,27829,32284,30270,30270,30269,30269,30270,30270,30270,30270,
30270,30270,30269,30270,30237,34332,38425,61439,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,48827,34298,32285,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,3023
7,30238,30269,34332,19345,4324,4292,228,34298,32285,32286,30269,30270,30269,30269,302
69,30237,34331,12940,4323,2276,6600,34266,32317,30270,30269,30269,30270,30270,30269,3
2285,34266,2342,4323,228,27829,32284,30270,30270,30269,30269,30270,30270,30270,30270,
30270,30270,30270,30269,32285,34331,40441,61439,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,634
87,48892,34298,32285,30270,30270,30270,30270,30270,30270,30270,30270,30269,30269,3023
7,30238,30269,34332,19376,2276,4291,228,34298,32285,32286,30269,30270,30269,30269,302
70,30237,34331,12972,4324,4324,6600,36314,32317,30270,30269,30269,30270,30270,30237,3
2285,34266,2342,4323,228,27829,32284,30270,30270,30269,30269,30270,30270,30270,30270,
30270,30270,30270,30269,30269,34331,40473,61439,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,634
87,50940,34298,32285,30269,30270,30270,30270,30270,30270,30270,30270,30269,30269,3023
7,30238,30269,34332,19345,2276,4292,228,34298,32285,32285,30269,30270,30269,30269,302
70,30237,34331,12972,4324,4324,6600,36314,32317,30270,30269,30269,30270,30270,30237,3
2285,34266,2374,4323,228,27829,32284,30270,30270,30269,30269,30270,30270,30270,30270,
30270,30270,30270,30270,30269,34331,42587,61439,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,634
87,51005,34266,32285,30269,30270,30270,30270,30270,30270,30270,30270,30269,30269,3023
7,30238,30269,34332,19345,2276,4292,228,34298,32285,32285,30269,30270,30269,30269,302
70,30237,34331,12972,4324,4324,6600,34298,32317,30270,30269,30269,30270,30270,30269,3
2285,34266,2374,4323,228,27829,32284,30270,30270,30269,30269,30270,30270,30270,30270,
30270,30270,30270,30270,30237,34331,44699,61439,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,634
87,55198,34265,32284,30269,30269,30237,30270,30270,30269,30270,30270,30269,30269,3027

0,30238,30269,34332,19345,2276,4292,229,34298,32285,30270,30269,30269,30269,32317,302
70,30237,34331,12972,4324,2276,6601,36314,32317,30270,30269,30269,30270,30270,30237,3
2285,34266,2342,4323,228,27829,32284,30269,30270,30269,30269,30270,30270,30270,30270,
30270,30270,30270,30270,30269,34299,46779,61439,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,59359,36313,34331,30269,30269,32317,30270,30269,30269,30270,30270,30269,30269,3228
5,30238,30269,34332,19345,2277,4324,229,34298,32285,30237,30269,30270,30269,30269,302
70,30269,34331,12972,4324,2276,6601,36314,32285,30270,30269,30269,30270,30270,30269,3
2285,34266,2342,4323,228,27829,32284,30270,30270,30269,30269,30270,30270,30270,30270,
30270,30270,30270,30238,32284,34266,50972,63487,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,61439,40474,34299,32317,30269,32317,30269,30270,30270,30269,30270,30269,30269,3027
0,30238,30269,34332,19345,2277,4292,229,34299,32285,30270,30269,30270,30269,32317,302
70,30269,34331,12940,4324,2275,6601,34298,32317,30270,30270,30269,30270,30270,30269,3
2285,34266,2374,4323,228,27829,32284,30269,30270,30269,30269,30270,30270,30269,30270,
30270,30270,30270,30269,32284,34233,55166,65535,65534,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,63487,48860,34266,32317,30269,30269,30269,30269,30269,30269,30270,30269,30269,3027
0,30238,30269,34332,19345,2277,4324,229,34298,32285,30269,30269,30269,30269,30269,302
70,30269,34331,10924,2276,2308,6601,34266,32285,30238,30269,30269,30270,30270,30269,3
2284,34266,2342,4323,229,27829,32284,30269,30269,30269,30269,30269,30269,30269,30270,
30270,30269,30269,30269,34332,34233,59359,63487,65535,65535,65535,65535,65535,65535,6
5535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,59359,36279,32284,32285,32317,30237,30269,30269,30269,30269,30269,30269,3027
0,30269,32285,32253,19443,4488,4488,2474,34299,32284,30237,30269,30269,30269,30270,30
269,30237,34332,13104,4456,4456,8813,34267,32284,30269,30269,30270,30270,30269,30269,
32316,32218,4586,4456,2440,25814,32284,30237,30270,30269,30237,30237,30270,30270,3027
0,30269,30237,30269,30237,34299,42554,61407,65535,65535,65535,65535,65535,65535,65535
,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,
65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,61439,50972,34298,34331,30237,30237,30269,30269,30270,30269,30269,30269,3026
9,30269,30236,32316,32283,34266,34267,32251,32317,30237,30270,30238,30269,30270,30270
,30270,30269,32285,34300,34267,34299,32251,32284,32285,30238,30270,30270,30270,30270,
30269,30269,32284,32251,34234,34267,34332,32285,30237,30270,30270,30270,30269,30238,3
0270,30270,30269,30237,32284,32284,34233,53085,63487,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65

```
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,63487,61439,44699,36313,34331,32284,32317,32285,30269,30269,30269,30269,3026
9,30269,30269,32317,30269,30236,32285,32317,30237,30269,30237,32286,30269,30237,30237
,30237,30269,30236,32284,32316,32316,32317,30237,30269,30269,30237,32285,32286,32285,
30237,30237,32317,32284,32316,32284,32284,32285,32317,30269,30269,30269,32285,32285,3
2285,32284,32284,34331,34331,34266,44698,61407,65502,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,

      65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,63487,61407,48860,36313,34298,34331,34331,32284,34332,34332,34332,3433
2,34332,34364,34332,32284,32316,34332,32284,32284,32316,32284,34332,32316,34332,34332
,34332,32316,32284,32283,34332,32284,32284,32316,34332,34332,34332,34331,34332,34332,
34332,34332,34332,34331,34363,34332,34331,34331,34331,34299,34299,34299,34298,34266,3
4233,34233,34232,36280,38361,46780,61439,63487,65535,65535,65535,65535,65535,65535,65
535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,65535,655
35,65535,65535,65535,65535,65535,65535,65535,
};
```

edge_calculation.py

```
import csv



ins=open('clock_white.txt','r')
temp = csv.reader(ins)
f=open('clock_white_result.txt','w')
count = 0
for line in temp:
      for element in line:
            if element =='0':
                  res = 'clock_white_result[ '+str(count)+' ]<=0;\n';
                  f.write(res)
                  count = count +1
            elif element =='1':
                  res = 'clock_white_result[ '+str(count)+' ]<=1;\n';
                  f.write(res)
                  count = count +1


f.close()
```

edge_cal_2.py

```
#!/user/bin/env python
f = open("evil.txt","r")
lines = f.readlines()
r = open("resEvil.txt","w")
counter = 0
Cnt =-1
```

```
lineCnt = -1
flag = "0"
outStr = ""
for line in lines:
    lineCnt = lineCnt +1
    print lineCnt
    x = line.split(",")
    Cnt =-1
    flag = "0"
    outStr = ""
    for y in x:
        Cnt = Cnt +1
        if(y=="65535" and flag =="0"):
            counter = counter +1
            flag ="1"
        elif(flag == "1" and y!="65535"):
#             print lineCnt
#             print "-firstone-"
#             print Cnt
            flag = "2"
            outStr = outStr + str(lineCnt)+", start: ," + str(Cnt) +","
        elif(flag == "2" and y=="65535"):
#             print lineCnt
#             print "-secondone-"
#             print Cnt
            flag = "0"
            outStr = outStr + str(lineCnt)+", end: ," + str(Cnt) +","
    outStr = outStr + "\n"
    r.write(outStr)
f.close()
```

image_convert.m

```
%Jian LU 16bit color map conversion

X = imread('oops','jpg');

height = 50;

width = 100;

[Y,map] = rgb2ind(X,65536);

R = X(:,:,1);

G = X(:,:,2);

B = X(:,:,3);



R1 = bitshift(R,-3);

G1 = bitshift(G,-2);
```

```
B1 = bitshift(B,-3);


% NEW(:,:,1)= R1;
% NEW(:,:,2)= G1;
% NEW(:,:,3)= B1;
%
% image(NEW);




OUTPUT = uint16(R1)*2^11 + uint16(G1)*2^5 + uint16(B1);


dlmwrite('oops.txt',OUTPUT);


R2 = reshape(uint8(bitshift(bitshift(OUTPUT,0),-11)),height,width);
G2 = reshape(uint8(bitshift(bitshift(OUTPUT,5),-10)),height,width);
B2 = reshape(uint8(bitshift(bitshift(OUTPUT,11),-11)),height,width);


NEW(:,:,1)= bitshift(R2,3);
NEW(:,:,2)= bitshift(G2,2);
NEW(:,:,3)= bitshift(B2,3);


image(NEW);
%
% R1 = bitand(R,hex2dec('1f'));
% G1 = bitand(G,hex2dec('3f'));
% B1 = bitand(B,hex2dec('1f'));
%
% NEW(:,:,1)= R1;
```

```
% NEW(:,:,2)= G1;

% NEW(:,:,3)= B1;

%

% image(NEW);
```