MP3 Player

CSEE 4840 SPRING 2010 PROJECT DESIGN

Zheng Lai zl2215@columbia.edu

Zhao Liu zl2211@columbia.edu

Meng Li ml3088@columbia.edu

Quan Yuan qy2123@columbia.edu

I. Introduction

The purpose of the project is to implement a MP3 decoder on DE2 board using Altera FPGA tool chains. Additionally, the decoder extracts the frequency information from the decoded data and uses these data to exhibits a visual effect, "floating histograms", on the screen, which looks similar to those on classic music player. Our work mainly focuses on building the hardware environment required by this project, implementing the Mpeg 1 Layer III decoding process using low level C programming, and some optimization schemes speeding up the decoding computation within the environment of embedded system.

II. Hardware Configuration

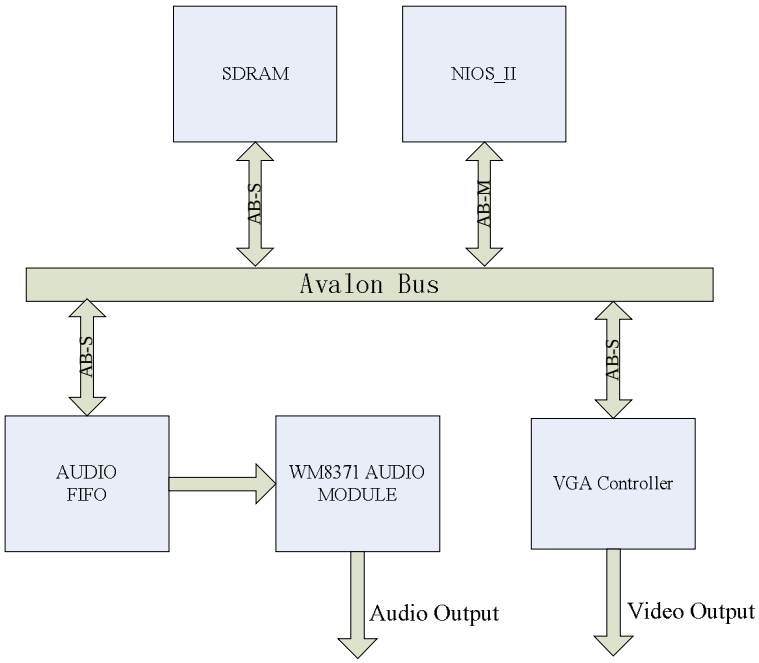


Fig.1 System Architecture

Fig.1 shows the hardware architecture used in this design. Configuration for each module will be discussed as follows.

2.1 Nios II

Due to high computation demand induced by the Mpeg decoding process, we decided to implement the fastest version of NIOS-II available in SOPC environment, accompanied with a embedded multipliers to speed up the large amount of multiplication in the software. Furthermore, a custom instruction used to compute single precision floating point is also added to the system.

2.2 Audio CODEC WM8731

ADC task is to be performed by audio chip on DE2 board. At the end of decoding process for each frame, 36 16-bit PCM samples for each sub-band are acquired. These digital PCM samples are then put into audio control unit and eventually the audio chip on board produce a analog voltage level according to its digital input. Three modules are needed to translate the decoded result to analog audio output.

- WM8731 control unit, serving as an Avalon slave module and providing hardware interface to CPU
- A serializer, which handles the parallel-to-serial conversion of the data
- First-In-First-Out (FIFO) serving as the buffer structure between the WM8731 control unit and serializer
- I2C control unit, used by CPU to configure WM8731
- PPL that generates clock frequency of 11.2896Mhz

WM8731 is configured when system is reset. Since the mp3 file we are decoding uses 44.1KHZ sampling rate, to accurately output an analog audio data, we implement a PLL which takes 50Mhz clock a produce 3 clock frequencies: a 50Mhz "back up" clock to CPU, a 11.2896Mhz clock to WM8731, and a clock with 3ns delay with regard to the main clock. The last clock is required by SDRAM for a stable operation.

Internal control register need to be set up based on the sampling frequency of the mp3 file we are decoding. The resulting configuration is shown in table. 1

Table .1

Left line in	0x0080
Right Line in	0x0280
Left Head Phone Out	0x047b
Right Head Phone Out	0x067b
Analog Path Control	0x08d0
Digital Path Control	0x0a04
Power Down Control	0x0c07
Digital Audio Interface Format	0x0e01

Sampling Control	0x1020
Active Control	0x1201

Note: The first 9 bits indicates the address of a specific control register.

2.3 VGA

A VGA control unit specified in generating the “floating histogram” effect is built in the project. Its functions are described as follows.

2.3.1 Generation of 64 rectangles

First, to show 64 rectangles on screen, the horizontal starting points, horizontal ending points and vertical ending points of each rectangle are fixed.

The following figure shows the horizontal timing of the VGA: (vertical timing is similar)

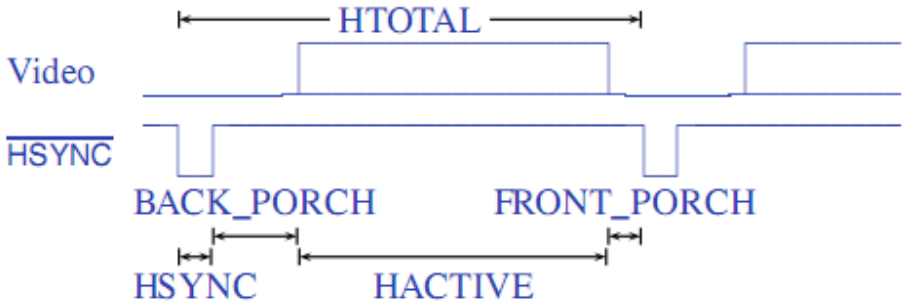


Fig.2

Define a signal of Hcount. At every rising edge of clk25, Hcount add 1, therefore, Hcount works as a horizontal pointer of the scanning position.

When Hcount=HSYNC+HBACK_PORCH+RECTANGLE_HSTART, where RECTANGLE_HSTART represents the horizontal starting point of the rectangle, it means the pointer is moving into the horizontal area of rectangles in the histogram. When Hcount=HSYNC +HBACK_PORCH+RECTANGLE_HEND, where RECTANGLE_HEND represents the horizontal ending point of the rectangle, it means the pointer is moving out of the horizontal area of rectangles in the histogram. During the clock cycle and the clock cycles following, the area on the screen will black out.

Having the same function with Hcount, signal Vcount works as a vertical pointer of the scanning position. When Vcount=VSYNC+VBACK_PORCH+RECTANGLE_VSTART, where RECTANGLE_VSTART represents the vertical starting point of the rectangle, it means the pointer is moving into the vertical area of rectangles in the histogram. When Vcount=VSYNC +CBACK_PORCH+RECTANGLE_VEND, where RECTANGLE_VEND represents the vertical ending point of the rectangle, it means the pointer is moving out of the vertical area of rectangles in the histogram. During the clock cycle and the clock cycles following, the area on the screen will black out.

The necessary condition for a point on screen to light up is

Hcount=HSYNC+HBACK_PORCH+RECTANGLE_HSTART, and

Vcount=VSYNC+VBACK_PORCH+RECTANGLE_VSTART

Which means the pointer is in both the vertical area and horizontal area of rectangles.

The 64 rectangles in the histogram are all realized in this way.

2.3.2 Controlling the height of each rectangle with the frequency information from the output of the frequency analyzer.

The frequency information is acquired through the decoding software. Specifically, inputs of the function IMDCT (inversed modified discrete cosine transform) are considered to be frequency information for the corresponding PCM samples. We bring these inputs to the VGA controller and adjust the heights, in reality, the vertical starting points for each rectangles based on these inputs.

According to the situation that the active area of the screen is 640*480, the ending point of each rectangle in the histogram is chosen to be 380. Therefore, the frequency information input should be from 0 to 380.

2.4 SDRAM

Since our decoding process bases on a huge array which represents the bit-stream in the MP3 file, the resulting .elf file would occupy a large amount of memory. As a result, SDRAM, which has larger capacity than SRAM, is necessitated by the design.

The Altera DE2 board contains an SDRAM chip that can store 8 Mbytes of data. This memory is organized as 1 M*16 bits*4 banks. The SDRAM chip requires careful timing control. To provide access to the SDRAM chip, the SOPC Builder implements an SDRAM Controller unit. This unit generates the signals needed to deal with SDRAM chip.

All of the control signals, except the clock, can be provided by the SDRAM Controller that can be generated by using the SOPC Builder. For proper operation of the SDRAM chip, it is necessary that its clock signal, DRAM_CLK, leads the Nios II system clock, by 3 nanoseconds. As mentioned in the part of WM8731, this clock is accomplished by using a phase-locked loop (PLL) unit, which is generated from Quartus II Mega function called ALTPLL.

III. MP3 Decoding Algorithm

The MP3 decoding algorithm contains several steps and the flow chart of the algorithm is in the below:

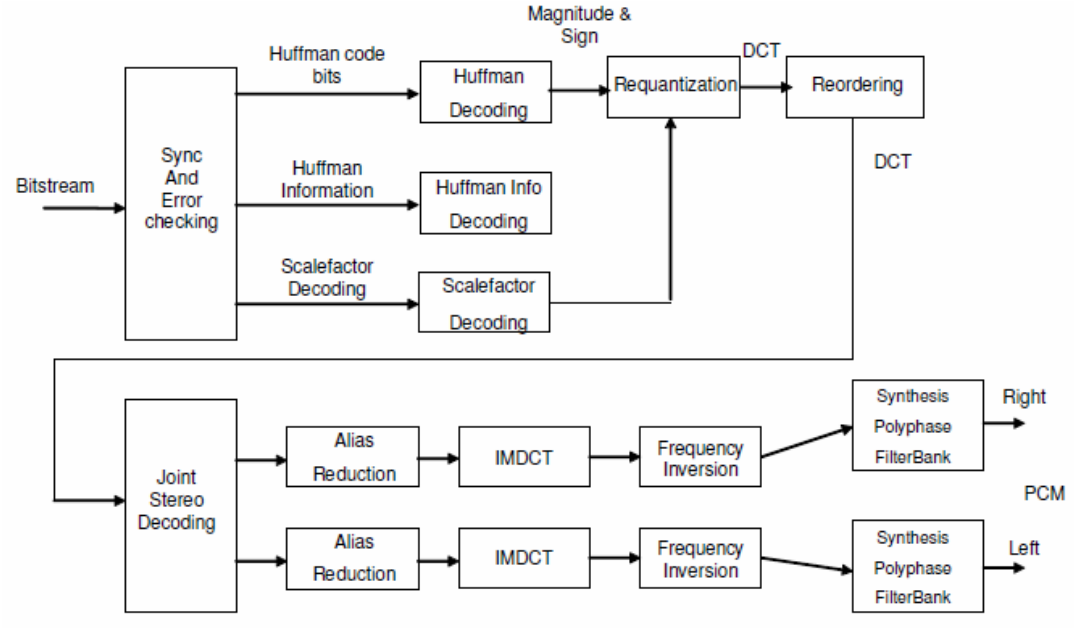


Fig.3 Flow Chart of MP3 Decoding

3.1 MP3 Data Structure

The bit stream data in MP3 file stored is frame by frame. And each frame has the following four parts: Header, Side Information, Main Data, and Ancillary Data.

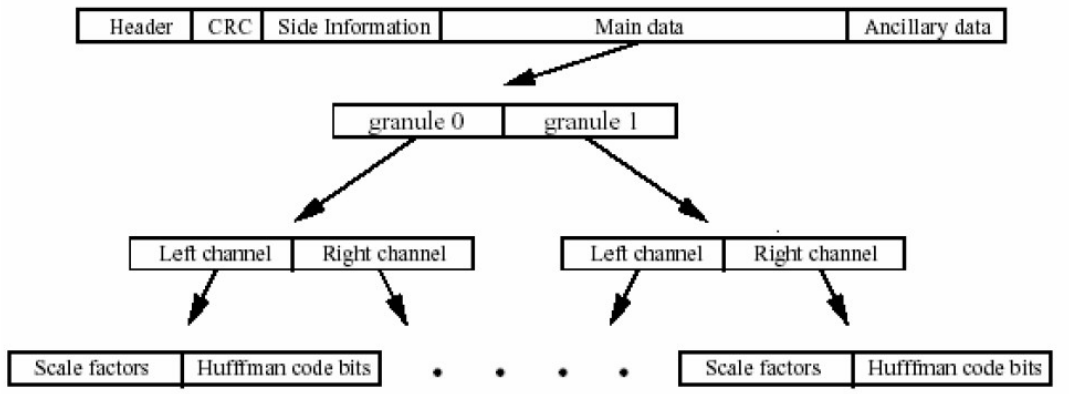


Fig.4 MP3 Frame Data Structure

Header is 32 bits and it confirms the frame. Information such as MPEG version, MPEG layer, bitrate and sampling rate are all contained in the header. Header doesn't have to be in the beginning of each frame therefore each header has a sync-word to mark its position in the file. And usually the information in headers of every frame in one file is consensus.

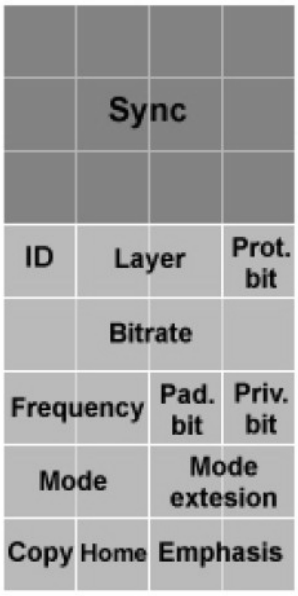


Fig.5 MP3 Frame Header Layout

Side Information can be either 17 bytes or 32 bytes depending on whether it is single-channel or dual-channel. It is usually follow the header and contains important information of the frame that will used for further decoding steps including scalefactor decoding, Huffman decoding, requantization and etc.

In the main data of MPEG-1 Layer III file, there are two granules for each channel while in MPEG-2 Layer III file there is only one for each channel. Our program is just for decoding the MPEG-1 Layer III files. (It means in MPEG-1 if the frame is single-channel it has two granules while if the frame is dual-channel it has four granules) Each granule is made up of two parts: scalefactor data and Huffman data. The Huffman data is actually the encoded PCM waveform samples and scalefactor is used for requantization. Huffman data of each granule contains data of 576 values which is evenly divided as 32 subbands.

What also worth mentioned is that a granule can be classified as a long block granule, a short block granule or a mixed block granule (a mixed block contains both long block parts and short block parts). The information of block type is stored in the side information. In long blocks, 576 values stands for 576 frequency lines while in short blocks, they stands for 192 frequency lines. Different block types also divide these 576 values into different number of scalefactor bands. Scalefactor band is a different concept from subband and it is used in Huffman decode and requantization. And unlike subband, scalefactor bands of one granule may have different number of values. Moreover, there are a lot of differences in the future decoding step with different type of granules.

Ancillary data can be in or not in a frame, which can be defined by the user and it follows the Huffman data.

3.2 Read Header and Side Information

To decode a frame of MP3 file, the first step is to read in and translate the header and side information. The algorithm locates a frame by searching the sync-word. Usually more than one sync-word is checked to make sure it is really a MP3 file. After locating a frame header, the program then can read in the header and side information.

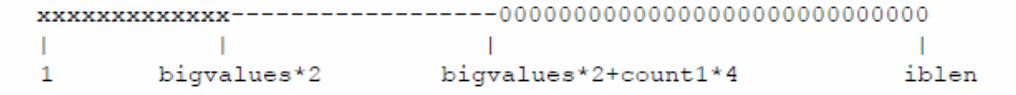
We also need to calculate the frame length since frame length is the distance between the headers of current frame and consecutive frame.

3.2 Scalefactor Decoding

As mentioned in the above, each granule contains scalefactor data and Huffman data. Also there are three block types for granule and different block type define different division of scalefactor bands of each granule. The bit lengths of scalefactor of different scalefactor bands and different block type granules are different. The program use *scalefactor_compress*, which is in the side information to check a predefined table, the data of which are the bit lengths of scalefactor of each scalefactor band. Then according to the lengths in the table, the program can read in the scalefactor from the bit stream of main data for each subband of the granule. We also have to calculate the length of whole length of scalefactor data of a granule.

3.3 Huffman Decoding

The Huffman decoding process is complicated in MP3 decoding. There are many tables generated by encoding for the reason of the encoding efficiency. So when decoding transition happens from one scalefactor band to another, the table may be changed. The 576 values are divided into three parts. What's more, the big value region can be divided into three different regions. (Big value of each frame is stored in the side information) Different parts use different tables to decode. When decoding the big-values, every entry returns two values. When in the small value decoding, every entry return four values. All the rest values are zeros.



The values 000 are all zero.
The values --- are -1,0 or +1. Their number is a multiple of 4.
The values xxx are not bound.
Iblen is 576.

3.4 Requantization and Reordering

Quantization is the process of converting a real number with almost infinite precision into an integer number and it is applied in the MP3 encoding. Therefore in decoding we have to do the reverse. The Huffman data of 576 values should be requantized. The formulas of requantization of long blocks and short blocks are slightly different, which are in the below :

$$xr_i = sign(is_i) * |is_i|^{\frac{4}{3}} * 2^{\frac{1}{4}(global\_gain[gr] - 210 - 8 * subblock\_gain[window][gr])} * 2^{-(scalefac\_multiplier * scalefacs[gr][ch][sfb][window])}$$

Fig.6 Requantization Formula for Short Block

$$xr_i = sign(is_i) * |is_i|^{\frac{4}{3}} * 2^{\frac{1}{4}(global\_gain[gr] - 210)} * 2^{-(scalefac\_multiplier * (scalefac\_l[sfb][ch][gr] + preflag[gr] * pretab[sfb]))}$$

Fig.7 Requantization Formula for long Block

The parameters in the formula are in the side information or can be check in a predefined table. And notice that scalefactor is used in requantization.

Also if the data are in short block type granule or in the short block part of the mixed block type granule, the frequency line should be reordered.

3.5 Stereo Processing

The two channels of typical stereo signal are not independent and joint stereo tries to exploit the existing similarities. Joint stereo processing is complicated because short blocks are handled differently than long blocks. Also, granules can contain a mixture of long and short blocks and the bands in the granule can be combined with different stereo modes.

3.6 Alias reduction

Alias reduction is for long block types. It is to reduce the aliasing that is introduced by the use of ideal non-band pass filter. Alias reduction is done by eight butterfly calculation for each subband. The coefficients for the butterfly calculation are calculated using the values in a predefined table.

3.7 IMDCT

IMDCT is short for inversed modified discrete cosine transform. There are four types of discrete cosine transform (DCT). Modified discrete cosine transform (MDCT) is a Fourier-related transform based on the type-IV DCT, with the additional property of being lapped: it is designed to be performed on consecutive blocks of a larger dataset, where subsequent blocks are overlapped so that the last half of one block coincides with the first half of the next block. MDCT is widely used in signal processing including MP3 encoding. The IMDCT is the inversed computing of MDCT. IMDCT takes in 18 or 6 frequency lines (long or short block) of each subband and generate 36 outputs. The formula is in the below:

$$x(i) = \sum_{k=0}^{(n/2)-1} X(k)\cos(\frac{\pi}{2n}(2i+1+\frac{n}{2})(2k+1)).$$

Fig.8 Formula of IMDCT in MP3 Decoding

These output samples are multiplied with a 36-point window before they can be passed on to the next step in the decoding process. Windowing contains four different types of windows namely, normal, short, start and stop. Information on what type to use is found in the side information part of each frame.

Then the samples also have to be divided into two halves and the first half of which will be overlapped with the second half of the previous block since MDCT is overlapping the data.

3.8 Subband Synthesis

This is the last part of MP3 decoding. It is in charge of turning the IMDCT outputs into final PCM samples. Subband synthesis has five sub-steps: First do matrix computing, taking one of the 18 values of each subband, which in total 32 values as input and receive 64 outputs; then putting these outputs into a FIFO buffer; take half of them and form a 512-element vector; window the vector by 512 coefficients; finally overlap these results and generate PCM output. The flowchart of subband synthesis is below:

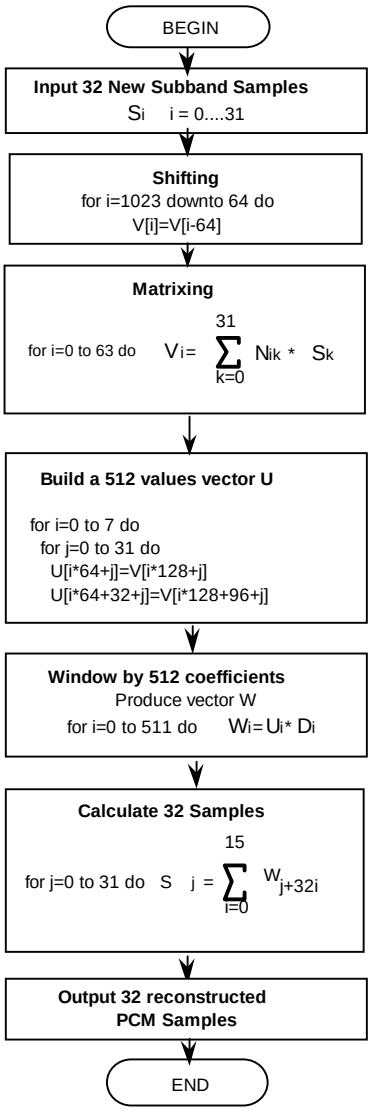


Fig.9 Flow chart of Subband Synthesis

IV. Software Analysis and Optimization

4.1 Observation and Analysis

When we first implement our MP3 decoder on the DE2 board, we found the decoding speed is extremely slow. It is more than 20 minutes per frame decoding. We then set the Nios II to higher version Nios II/f. Then the decoding is about 4 times faster than before but still too slow. We use timer to monitor which functions consumes most time.

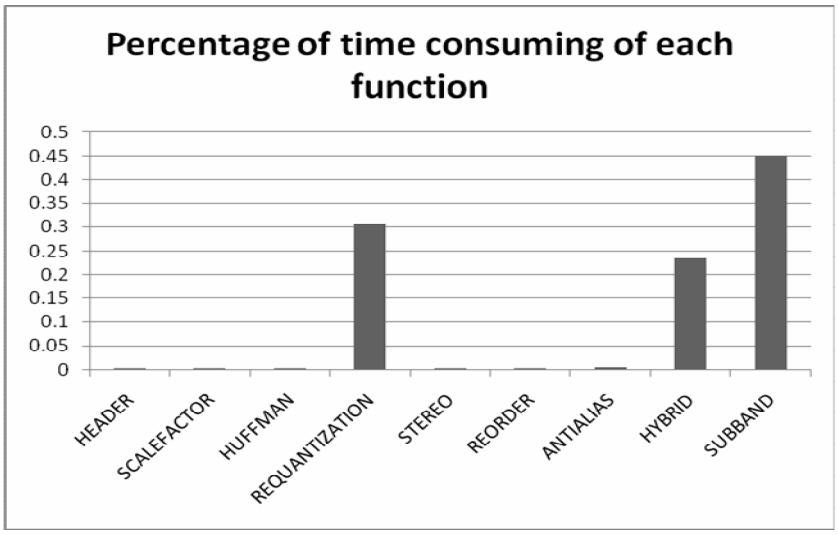


Fig.10 The percentage of time consuming of each function

We found the function " requantization" , "hybrid" and " subband" takes most of the decoding time. We analyzed that the speed is limited by the floating point operation and mathematic operation (such as power, sin, cos). Since up to 100 thousand times of floating point multiplication and, up to 1 thousand times of mathematic operations (power, sin, cos) are involved in decoding process in each frame, manual optimization of these operations instead of using the standard c library will be expected to gain a huge boost in decoding speed.

4.2 Optimization

4.2.1 Look-up tables

The operation of power as well as sin and cos consume a lot of time when decoding MP3. Since the variable may be in a certain range, we can calculate the possible values first and put them in tables.

For example:

```
xr[sb][ss] = pow( 2.0 , (0.25 * (gr_info->global_gain - 210.0)));
```

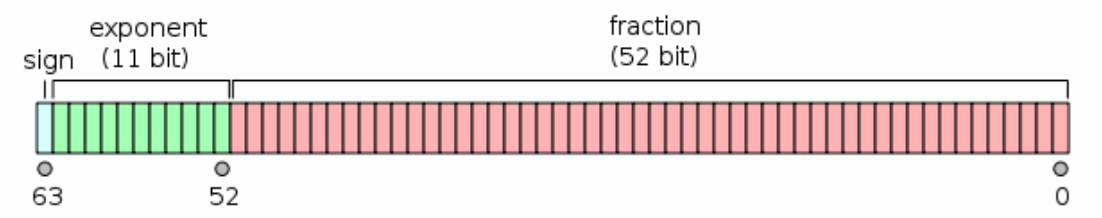
gr_info->global_gain is 8 bits. So we can calculate all the 256 possible values first and generate the table for this power operation.

After that, we can use the variable to loop up the table. In this way, we significantly accelerate the operation of power, sin, cos . The data is in the following section.

4.2.2 Custom floating point operation function

Since the double -precision floating point operations on the DE2 board are extremely slow, we design function using bit operation to do the multiplication, addition/subtraction of double -precision floating point.

As a fundamental, we introduce IEEE 754 double -precision binary format first:



$$(-1)^{signbit} \times 2^{exponentbits-1023} \times 1.fractions$$

The above is the format of double –precision floating point stored in the memory. Our idea is to extract all the components (sign, exponent, and fraction) and then use long integers and bit operations to do the floating point operation.

For example, we can design the double -precision floating point multiplication in this ways.

$$A=(-1)^{signbit\_A} \times 2^{exps\_A-1023} \times 1.fractions$$

$$=(-1)^{signbit\_A} \times 2^{exps\_A-1078} \times (0x1FFFFFFFFFFFFF||fractions\_A)$$

$$B=(-1)^{signbit} \times 2^{exp\_B-1023} \times 1.fractions$$

$$=(-1)^{signbit\_B} \times 2^{exps\_B-1078} \times (0x1FFFFFFFFFFFFF||fractions\_B)$$

We use the long integer multiplication to calculate the fraction part of the result. And the exponent of the result need to contain a new_exp offset corresponding to the new_fractions, because of the fraction occupying 52 bits in the memory.

$$A * B = (-1)^{signbit\_A + signbit\_B} \times 2^{exp\_A + exp\_B - 2046 + new\_exp} \times 1.new\_fractions$$

We can use integer to calculate the sign bit and exponents of the result. After that , we use bit operation to combine them into IEEE 754 format and make coercion transformation into floating point to get returned.

Pseudo codes for floating point addition and multiplication are shown in Appendix.

4.2.3 Custom floating point operation Instruction

A more efficient way to alleviate the heavy computation task for floating point can be realized by introducing custom floating point instruction under the environment of SOPC. After NIOS-II is configured with this floating point computation instruction, the complier will automatically generates appropriate assemblely codes with this instruction, thus avoiding using the standard C library and hence yield great enhancement in the decoding speed.

4.3 Optimization Result

Table 2 and fig.11 shows the optimization result using the technique discussed above.

Table 2 The time used per frame's decoding

	Decoding time (s) per frame (timed by timer)	Decoding time (s) per frame (timed by watch)
Initial implement	-	1560
Un-optimized	46.7	300
Software Optz.	12.8	30
Hardware Optz.	0.58	1

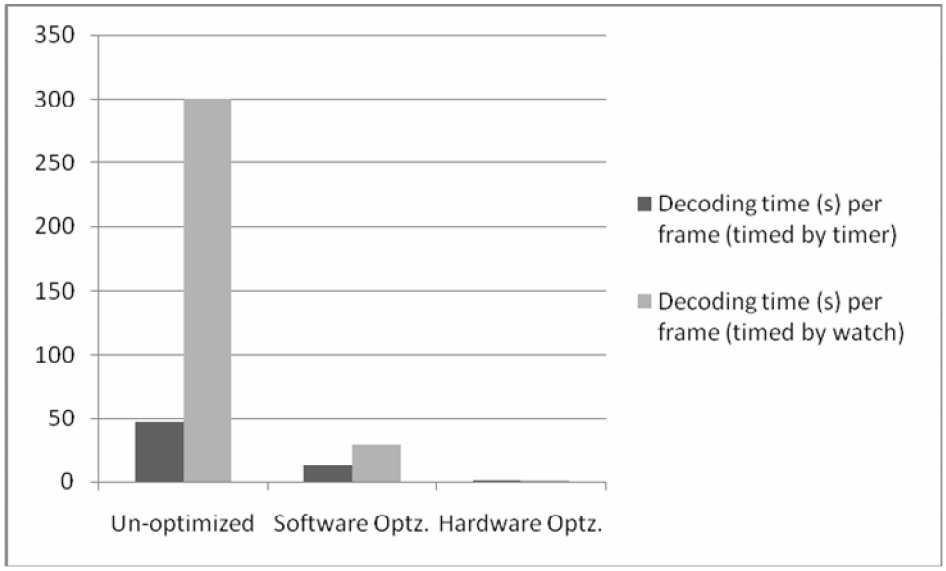


Fig.11 Speed up of optimization

The above graph shows the speed up of the optimization. The '"un-optimized" column is the original implementation with setting to Nios II/f. The "software optz." column is the optimization including look-up tables and custom floating point optimization. The last column is the optimization of the custom instruction for Nios II Processor.

V. Conclusion

After understanding the principle of mp3 decoding, we realize the bottleneck of the decoding process lies in heavy computation task of floating point. A few methods are attempted to alleviate the problems, including switching to faster processor, using look-up table, and improving the algorithms of floating point multiplication. At the end of the design, we are able to execute the MP3 decoding process at the speed of 0.6 second per frame, which is roughly 1500 times faster than the original version, while still producing the same PCM outputs compared with the results generated on Dell server.

However, the weakness of our algorithms is the overemphasis of accuracy. Each optimization we attempted is based on the motivation of lossless decoding. In reality the audio quality and hence the accuracy may not be that important under portable environment. Thus future efforts can be put into using fixed point computation instead of floating point.

VI. Appendix

6.1 hello_world.c

```
#define TESTFRAME 800

#include "basic_io.h"
#include "basic.h"
#include "header_reader.h"
#include "get_scalefactor.h"
#include "requantize.h"
#include "huffman_decode.h"
#include "reorder.h"
#include "joint_stereo.h"
#include "antialias.h"
#include "hybrid.h"
#include "subbandsynthesis.h"
#include "math.h"
#include "alt_types.h"
#include "io.h"
#include "system.h"
#include "stdio.h"

#define IOWR_LED_DATA(base, offset, data) \
   IOWR_16DIRECT(base, (offset) * 2, data)
#define IORD_LED_DATA(base, offset) \
   IORD_16DIRECT(base, (offset) * 2)
#define IOWR_LED_SPEED(base, data) \
   IOWR_16DIRECT(base + 32, 0, data)



unsigned long F32ToHex(float valueF)
{

      unsigned long* valueL = (unsigned long *) &valueF;
      return *valueL;


}

inline int gen_wave32 (float flt)
      {
            unsigned long    hex=0;
            unsigned int exp,outcome;
            unsigned long    fraction;
            unsigned char sign_whole;
            unsigned char sign_exp;
            unsigned long result;
            unsigned long result2;
            hex=F32ToHex(flt);
            fraction=hex&0x007FFFFF;
            exp=127-((hex&0x7F800000)>>23);
```

```
        //sign_exp=((exp&0x800)>>11);
        //exp=exp&0x7FF;
        sign_whole=(hex&0x80000000)>>31;
        fraction=fraction|0x00800000;

              result=(fraction<<7)>>(exp-1);

        result&=0x00FF0000;
        result>>=16;
       // outcome=result;
        return result;
    }


void main()
{
    unsigned char *address_of_first_frame;
    unsigned char *index;
    unsigned char valid[2]={1,1};
    unsigned long int sycnword_count=0;
    unsigned char ii=0; //This variable is used to denote the number of frame that are being analyzed.
    int clip=0;
    int vga,vga_temp;

    printf("OK start.\n");//test;
    find_the_first_frame(&address_of_first_frame,&sycnword_count);
    frame_stat=0;//test;
    sycnword_count=sycnword_count-2;
    block_type_0_count=0;//test;
    //printf("address_of_first_frame is %d\n",address_of_first_frame - data_header);//test;
    index=address_of_first_frame;
    si[0].main_data_end=0;si[1].main_data_end=0;


    initialize_huffman();

    for(frameNum=1;frameNum<TESTFRAME+1;frameNum++)
    {

        ii++;
        if(ii>1) ii=0;

        header_reader(&index,&sycnword_count,&header_info[ii],&si[ii],&si[bit_add(ii)],&main_data_beg_bs[ii],&valid[ii]);
        /*if((valid[1]==1)&&(valid[0]==1))
        {
        //printf("I have found the %dst frame starting at this address:%d\n",frameNum,index-header_info[ii].frame_length-data_header);
        }
        else
        {
              //printf("bad frame detected.I quit.\n");exit(0);

        }*/
```

```
int ch,gr;
for(gr=0;gr< grNum;gr++)
{gr_global=gr;
for(ch=0;ch< stereo;ch++)
{
	get_scale_factors(&si[ii],&scalefac,gr,ch,main_data_beg_bs[ii]);
	//printf("start of huffman\n");
	huffman_decode(is, &si[ii], ch, gr,&header_info[ii],&main_data_beg_bs[ii]);

	main_data_beg_bs[ii].byte_idx_ptr+=(si[ii].ch[ch].gr[gr].part2_3_length/8);
	if(main_data_beg_bs[ii].bit_idx>=(si[ii].ch[ch].gr[gr].part2_3_length%8))
	{
		main_data_beg_bs[ii].bit_idx-=(si[ii].ch[ch].gr[gr].part2_3_length%8);
	}
	else
	{
		main_data_beg_bs[ii].byte_idx_ptr++;
		main_data_beg_bs[ii].bit_idx=main_data_beg_bs[ii].bit_idx+8-si[ii].ch[ch].gr[gr].part2_3_length%8;
	}
	//printf("start of requantize\n");

	requantize(&header_info[ii],&scalefac,&si[ii],is,ro[ch],gr,ch);
	//printf("end of requantize\n");
	//print_scalefac(frameNum, gr, ch, si[ii]);//test;
	//print_requantize(frameNum,gr,ch,ro[ch]);//test;

}
//printf("start of stereo\n");
join_stereo(ro,lr,&scalefac,&(si[ii].ch[ch].gr[gr]),&header_info[ii]);

for(ch=0;ch< stereo;ch++)
{
	ch_global=ch;
	reorder (lr[ch],re,&(si[ii].ch[ch].gr[gr]),&header_info[ii]);
	antialias(re, hybridIn, &(si[ii].ch[ch].gr[gr]), &header_info[ii]);

	for(vga=0;vga<7;vga++){
	vga_temp=(vga<<3)+vga;
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, vga_temp,350-gen_wave32(hybridIn[vga][1]));
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, vga_temp+1,350-gen_wave32(hybridIn[vga][3]));
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, vga_temp+2,350-gen_wave32(hybridIn[vga][5]));
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, vga_temp+3,350-gen_wave32(hybridIn[vga][7]));
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, vga_temp+4,350-gen_wave32(hybridIn[vga][9]));
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, vga_temp+5,350-gen_wave32(hybridIn[vga][11]));
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, vga_temp+6,350-gen_wave32(hybridIn[vga][13]));
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, vga_temp+7,350-gen_wave32(hybridIn[vga][15]));
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, vga_temp+8,350-gen_wave32(hybridIn[vga][17]));
	}
	IOWR_LED_DATA(DE2_VGA_RASTER_INST_BASE, 63,350-gen_wave32(hybridIn[7][7]));

	unsigned char sb=0;
	unsigned char ss=0;
	for (sb=0; sb<SBLIMIT; sb++)
```

```
                {
                        hybrid(hybridIn[sb], hybridOut[sb], sb, ch, &(si[ii].ch[ch].gr[gr]),&header_info[ii]);
                }
//printf("done.\n");
                for (ss=0;ss<18;ss++)
                        for (sb=0; sb<SBLIMIT; sb++)
                                if ((ss%2) && (sb%2))
                                        hybridOut[sb][ss] = -hybridOut[sb][ss];

                //print_hybrid_output(frameNum,gr,ch,hybridOut);//test;
                //printf("synthesis start\n");
                for (ss=0;ss<18;ss++)
                {
                        ss_global=ss;
                        for (sb=0; sb<SBLIMIT; sb++)
                                polyPhaseIn[sb] = hybridOut[sb][ss];
                        SubBandSynthesis (polyPhaseIn, ch,&(pcm_output[frameNum-1][gr_global][ch_global][ss_global][0]));
                }
                //printf("synthesis done\n");
        }

        //print_pcm_sample(frameNum,gr,pcm_sample);//test;

        //file_output(pcm_sample,fp,&sample_frames);
        }
                printf("%d ",frameNum);
    }
printf("done\n");

       for(frameNum=0;frameNum<TESTFRAME;frameNum++)
        {//printf("\nframe%d\n",frameNum);
         for(gr_global=0;gr_global<2;gr_global++)
        {//printf("\ngr%d\n",gr_global);
        for(ch_global=0;ch_global< stereo;ch_global++)
                {//printf("\nchannel%d\n",ch_global);
                for (ss_global=0;ss_global<18;ss_global++)
                        {//printf("\nss%d\n",ss_global);
                          for(pcm_cnt_global=0;pcm_cnt_global<SBLIMIT;)
                             {
                        if(!IORD_16DIRECT(0x00681104,0))
                        {
                                usleep(15);
//                                short monitor;
//                                monitor=pcm_output[frameNum][gr_global][ch_global][ss_global][pcm_cnt_global];
//                                printf(" %d ",pcm_output[frameNum][gr_global][ch_global][ss_global][pcm_cnt_global]);
                                IOWR_16DIRECT(0x00681104,0,(unsigned
short)pcm_output[frameNum][gr_global][ch_global][ss_global][pcm_cnt_global]);
                                pcm_cnt_global+=2;
                        }
                                }
                }
              }
        }
```

```
	}
//----------------------test information print out----------------------------;
/*    printf("Frame_stat is %d\n",frame_stat);printf("bad_block_count is %d\n",bad_block_count);
	printf("big_value_0_count is %d\n",big_value_0_count);
	printf("big_value_1_count is %d\n",big_value_1_count);
	printf("block_type_0_count is %d\n",block_type_0_count);*/
	//printf("index taken by next circulation is %d\n",index);
	//printf("ptr in main_data_bs structure, bit_idx is %d, byte_idx(offset) is %d.\n", main_data_bs.bit_idx, (unsigned long
int)(main_data_bs.byte_idx_ptr-data_header));printf("\n");}
	//byte_idx tells you the location of maindata in forms of    offset from the starting point of data_header
	/*printf("ht[33] info :----------\n");
		{
			printf("table name is %s"	,ht[33].tablename);
			printf("xlen is:%d	",ht[33].xlen);
			printf("ylen is:%d	",ht[33].ylen);
			printf("lintbis is:%d	",ht[33].linbits);
			printf("treelen is:%d	",ht[33].treelen);
			printf("the first    2 values are:0x%x 0x%x	",ht[33].val[0][0],ht[33].val[0][1]);
			printf("the second 2 values are:0x%x 0x%x	",ht[33].val[1][0],ht[33].val[1][1]);
			printf("the third    2 values are:0x%x 0x%x	\n",ht[33].val[2][0],ht[33].val[2][1]);

		}*/
}
```

6.2 basic.h

```
#include <stdio.h>
#include <stdlib.h>

#include "dataheader.h"
#include "math.h"

#define get1bit(bs) getbits(bs,1)
#define SBLIMIT 32
#define SSLIMIT 18
#define PI 3.14159265358979
unsigned int frameNum=0;//use to count the frame
	unsigned int pcm_cnt_global;
	unsigned char ss_global=0;
	unsigned char ch_global=0;
	unsigned char gr_global=0;
	short pcm_output[TESTFRAME][2][2][SSLIMIT][SBLIMIT];
//-------------------Header_Info Definition-----------------;
typedef struct
{
unsigned char ID;//1 bit
unsigned char layer;//2 bit
unsigned char protection_bit;//1 bit. If it is 1, CRC check exits. CRC is 16-bit long and follows frame header.
unsigned int bitrate_index;//4 bit
unsigned int sampling_frequency_index;//2 bit
unsigned char pading_bit;//1 bit
unsigned char private_bit;//1 bit
unsigned char mode;//2 bit.00:Stero,11:Joint Stero,10 Dual Channel,01 Single Channel
unsigned char mode_extension;//2 bit
```

```
unsigned char copy_right;//1 bit
unsigned char original_home;//1 bit
unsigned char emphasis;//2 bit
unsigned int frame_length;

}header_info_type;

//--------------------Frame_Info Definition-----------------;
unsigned char stereo=0;
unsigned char grNum=2;

//-----------------Side_Info Definition--------------------;
typedef struct
{
   unsigned main_data_end;//9bits
   unsigned private_bits;//5bits
   struct {
         unsigned scfsi[4];//4bits
          struct gr_info_type {
                unsigned part2_3_length;//12bits
                unsigned big_values;//9bits
                unsigned global_gain;//8bits
                unsigned scalefac_compress;//4bits
                unsigned window_switching_flag;//1bits
                unsigned block_type;//2bits
                unsigned mixed_block_flag;//1bit
                unsigned table_select[3];//5 bits each
                unsigned subblock_gain[3];//3 bits each
                unsigned region0_count;//4bits
                unsigned region1_count;//3bits
                unsigned preflag;//1bits
                unsigned scalefac_scale;//1bits
                unsigned count1table_select;//1bits
                } gr[2];
          } ch[2];
} side_info_type;

//-------------------Scalefactor Definition-------------------------;
typedef struct
{
   struct {
          int l[22];               // [cb]
          int s[3][13];          // [window][cb]
   }ch[2];                                 // [ch]
} scalefac_type;

//---------------Bit Stream Definitions & Functions------------;

typedef struct    bit_stream_struc {
       unsigned char *byte_idx_ptr;      /* pointer to current byte in the arrray */
       char    bit_idx;        /* pointer to leftmost bit of current byte */
                                  /* the bit_idx for leftmost bit =7*/
} bit_stream_struc;
```

```
unsigned char mask[8]={0x1, 0x2, 0x4, 0x8, 0x10, 0x20, 0x40, 0x80};

void bs_stucture_init(bit_stream_struc *bs)
{
        bs->byte_idx_ptr=data_header;
        bs->bit_idx=7;
}

unsigned long int getbits(bit_stream_struc *bs,unsigned long n) /*read n bits*/
{
      unsigned long int val=0;
      while(n!=0)
      {
         val=((((*(bs->byte_idx_ptr))&mask[bs->bit_idx])>>bs->bit_idx)<<(n-1))|val;
         bs->bit_idx=bs->bit_idx-1;
         if(bs->bit_idx<0) {bs->bit_idx=7;bs->byte_idx_ptr++;}
         n--;
       }
      return val;
}

unsigned char bit_add(unsigned char i)
{
   i=i+1;
   if(i>1) i=0;
   return i;
}

//---------------------File Output Functions----------------------;
void SwapBytesInWords( short *loc, int words )
{
      int i;
      short thisval;
      char *dst, *src;
      src = (char *) &thisval;
      for ( i = 0; i < words; i++ )
      {
            thisval = *loc;
            dst = (char *) loc++;
            dst[0] = src[1];
            dst[1] = src[0];
      }
}

void file_output(pcm_sample,outFile,psampFrames)
short pcm_sample[2][SSLIMIT][SBLIMIT];
FILE *outFile;
unsigned long *psampFrames;
{
      int i,j,l;
      static short int outsamp[1600];
      static long k = 0;
```

```
      for (i=0;i<SSLIMIT;i++)
         for (j=0;j<SBLIMIT;j++)
         {
               (*psampFrames)++;
                  for (l=0;l<stereo;l++)
                  {
                        if (!(k%1600) && k)
                        {
                            //Samples are big-endian. If this is a little-endian machine we must swap
                            SwapBytesInWords( outsamp, 1600 );
                              fwrite(outsamp,2,1600,outFile);
                              k = 0;
                        }
                        outsamp[k++] = pcm_sample[l][i][j];
                  }
            }
}

//--------------------Tables&Preset Data-----------------;

int slen[2][16] = {{0, 0, 0, 0, 3, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4},
                           {0, 1, 2, 3, 0, 1, 2, 3, 1, 2, 3, 1, 2, 3, 2, 3}};

struct    {
     int l[23];
     int s[14];
} sfBandIndex[6] =
      {{{0,6,12,18,24,30,36,44,54,66,80,96,116,140,168,200,238,284,336,396,464,522,576},
        {0,4,8,12,18,24,32,42,56,74,100,132,174,192}},
      {{0,6,12,18,24,30,36,44,54,66,80,96,114,136,162,194,232,278,330,394,464,540,576},
        {0,4,8,12,18,26,36,48,62,80,104,136,180,192}},
      {{0,6,12,18,24,30,36,44,54,66,80,96,116,140,168,200,238,284,336,396,464,522,576},
        {0,4,8,12,18,26,36,48,62,80,104,134,174,192}},

      {{0,4,8,12,16,20,24,30,36,44,52,62,74,90,110,134,162,196,238,288,342,418,576},
        {0,4,8,12,16,22,30,40,52,66,84,106,136,192}},
      {{0,4,8,12,16,20,24,30,36,42,50,60,72,88,106,128,156,190,230,276,330,384,576},
        {0,4,8,12,16,22,28,38,50,64,80,100,126,192}},
      {{0,4,8,12,16,20,24,30,36,44,54,66,82,102,126,156,194,240,296,364,448,550,576},
        {0,4,8,12,16,22,30,42,58,78,104,138,180,192}}};

//-------------------Global Data---------------------------;

bit_stream_struc main_data_beg_bs[2];
header_info_type header_info[2];//header_info[0] store the side information for previous frame.
                                                //header_info[1] store the side information for current frame.
side_info_type si[2];//si[0] store the side information for previous frame.
                              //si[1] store the side information for current frame.
scalefac_type scalefac;
long int is[SBLIMIT][SSLIMIT];
float xr[SBLIMIT][SSLIMIT];
float lr[2][SBLIMIT][SSLIMIT];
```

```
float ro[2][SBLIMIT][SSLIMIT];
float re[SBLIMIT][SSLIMIT];
float hybridIn[SBLIMIT][SSLIMIT];/* Hybrid filter input */
float hybridOut[SBLIMIT][SSLIMIT];/* Hybrid filter out */
float polyPhaseIn[SBLIMIT];          /* PolyPhase Input. */
short pcm_sample[2][SSLIMIT][SBLIMIT];

//typedef short PCM[2][SSLIMIT][SBLIMIT];
//PCM *pcm_sample;
```

6.3 header_reader.h

```
//------------------------test variables-------------------------;
unsigned int frame_stat=0;
unsigned int bad_block_count=0;
unsigned char bad_block_type_flag=0;
int big_value_0_count=0;
int big_value_1_count=0;
long int block_type_0_count=0;
//char window_switching_flag_test=0;

//------------------------Bitrate_table-------------------------;
unsigned int bitrate_table[2][4][16]=
{
{
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},//blank for 1,0,0
{0,32,48,56,64,80,96,112,128,144,160,176,192,224,256,0},//table for 1,3,0
{0,8,16,24,32,40,48,56,64,80,96,112,128,144,160,0},//table for 1,2,0
{0,8,16,24,32,40,48,56,64,80,96,112,128,144,160,0}//table for 1,1,0
},

{
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},//blank for 0,0,0
{0,32,64,96,128,160,192,224,256,288,320,352,384,416,448,0},//table for 0,1,0
{0,32,48,56,64,80,96,112,128,160,192,224,256,320,384,0},//table for 0,2,0
{0,32,40,48,56,64,80,96,112,128,160,192,224,256,320,0}//table for 0,3,0
}
};
//Note:
//first index refers to MPEG version. 1 -> MPEG1, 0 -> MPEG2 .Agree with definition in frame header
//Second index refers to layer number. Agree with definition in frame header.01->layer3,10->layer2,11->layer1.
//Third refers to birate index.
/*birate[0][1][]={0,32,64,96,128,160,192,224,256,288,320,352,384,416,448,0}//layer 3, MPEG 1
birate[0][2][]={0,32,48,56,64,80,96,112,128,160,192,224,256,320,384,0}//layer 2, MPEG 1
birate[0][3][]={0,32,40,48,56,64,80,96,112,128,160,192,224,256,320,0}//layer1, MPEG 1
birate[1][1,2][]={0,8,16,24,32,40,48,56,64,80,96,112,128,144,160,0}//layer 3 & layer 2, MPEG2
birate[1][3][]={0,32,48,56,64,80,96,112,128,144,160,176,192,224,256,0}//Mpeg 2 layer 1
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}<-This is for blank table*/


//------------------------Sample_rate_table-------------------------;
unsigned int sample_rate_table[3][4]=
{
```

```
{22050,24000,16000,0},
{44100,48000,32000,0},
{11025,12000,8000,0}
};
//Note:1.first index refers to MPEG version. 1 -> MPEG1, 0 -> MPEG2, 3 -> MPEG2.5
/*sample_rate_table[0]={44100,48000,32000,0};//MPEG1
sample_rate_table[1]={22050,24000,16000,0};//MPEG2
sample_rate_table[1]={11025,12000,8000,0};//MPEG2.5*/


/**********************Function:Header_Reader Definition Starts*************************/
/**Function Parameter Distription:"Index" is a pointer refering the position from which the function starts to search for
syscword."FrameCount" is used to document the number of frames in the fils."*header_info" and "*si" is a pointer used by the function to
documnent the header infomation and side infomation.Finally, when the function is finished, the location    of the starting point of main data
is stored in "*main_data_bs"**/
/**Function Return Distription: This function return a parameter that tells the location of next frame.This location is caculated based on the
infomarion found in the header**/

inline header_reader(unsigned char **index,
                     unsigned long int *sycnword_count,
                     header_info_type *header_info,
                     side_info_type *si_current,
                     side_info_type *si_previous,
                     bit_stream_struc *main_data_beg_bs,
                     unsigned char *valid)
{
    int ch, gr, i;
    unsigned char *starting_index;//this variable is used to locate the postition of syncword in the current frame
    unsigned char *side_info_start_ptr;
    unsigned char position_flag=0;//if 0, indicating the sync_word is in forms of "0x*F 0xFF".Else in the form of"0xFF 0xF*"
                                        // ^-starting index                    ^-start idx
    unsigned char flag=0;
    bit_stream_struc bs;

/************************************Start anaylzing header**********************************/
      while(!flag)
      {
         if((**index)==0xff)
         {
               if(((*(*index-1))&0x0f)==0x0f)
               {
                     flag=1;
                     starting_index=*index-1;
                     position_flag=0;
               }
               if(((*(*index+1))&0xf0)==0xf0)
               {
                     flag=1;
                     starting_index=*index;
                     position_flag=1;
               }
         }
         (*index)++;
```

```
        }
    (*index)--;
    (*sycnword_count)++;
//  printf("\nI find the %dst syncword offset is at %d, position_flag is %d.\n",
//        *sycnword_count,(unsigned int)(starting_index-data_header),position_flag);
    if(position_flag==0)
    {
          bs.byte_idx_ptr=starting_index+2;
          bs.bit_idx=7;
          //printf("bs.byte_idx_ptr now is %d",bs.byte_idx_ptr-data_header);
    }
    else
    {
          bs.byte_idx_ptr=starting_index+1;
          //printf("bs.byte_idx_ptr now is %d",bs.byte_idx_ptr-data_header);
          bs.bit_idx=3;
    }
    header_info->ID = getbits(&bs,1);
    header_info->layer = 4-getbits(&bs,2);
    header_info->protection_bit = !get1bit(&bs); /* error protect. TRUE/FALSE */
    header_info->bitrate_index = getbits(&bs,4);
    header_info->sampling_frequency_index = getbits(&bs,2);
    header_info->pading_bit = get1bit(&bs);
    header_info->private_bit = get1bit(&bs);
    header_info->mode = getbits(&bs,2);
    header_info->mode_extension = getbits(&bs,2);
    header_info->copy_right = get1bit(&bs);
    header_info->original_home = get1bit(&bs);
    header_info->emphasis = getbits(&bs,2);

    if(header_info->protection_bit==0)
          side_info_start_ptr=starting_index+4;
    else
          side_info_start_ptr=starting_index+6;
    //printf("side_info_start_ptr is now %d",side_info_start_ptr-data_header);

    if(sample_rate_table[header_info->ID][header_info->sampling_frequency_index])
    {

header_info->frame_length=144*1000*bitrate_table[header_info->ID][header_info->layer][header_info->bitrate_index]/sample_rate_table[
header_info->ID][header_info->sampling_frequency_index]+header_info->pading_bit;
    }
    else header_info->frame_length=0;
    if((sample_rate_table[header_info->ID][header_info->sampling_frequency_index]==44100)
                &&(bitrate_table[header_info->ID][header_info->layer][header_info->bitrate_index]==320)
                &&(header_info->ID)
                &&(header_info->layer==3))
    {*valid=1;}
    else
    {*valid=0;goto LP;}
/*************************************end of anaylzing header**********************************/

//************************************Start Analyzing Side Info*************************//
```

```
  bs.byte_idx_ptr=side_info_start_ptr;
  stereo = (header_info->mode==0x3)?1:2;
  if(position_flag==0)
{bs.bit_idx=3;}
  else{bs.bit_idx=7;}
  if(header_info->ID == 1)
  {
     si_current->main_data_end = getbits(&bs, 9);
       if (stereo == 1)
          si_current->private_bits = getbits(&bs,5);
            else si_current->private_bits = getbits(&bs,3);

        for (ch=0; ch<stereo; ch++)
              for (i=0; i<4; i++)
                 si_current->ch[ch].scfsi[i] = getbits(&bs,1);


          for (gr=0; gr< grNum ; gr++) {
               for (ch=0; ch<stereo; ch++) {
                  si_current->ch[ch].gr[gr].part2_3_length = getbits(&bs, 12);
                  si_current->ch[ch].gr[gr].big_values = getbits(&bs, 9);
                  si_current->ch[ch].gr[gr].global_gain = getbits(&bs, 8);
                  si_current->ch[ch].gr[gr].scalefac_compress = getbits(&bs, 4);
                  si_current->ch[ch].gr[gr].window_switching_flag = getbits(&bs,1);
                   if (si_current->ch[ch].gr[gr].window_switching_flag) {//window_switching_flag_test=1;//test;
                       si_current->ch[ch].gr[gr].block_type = getbits(&bs, 2);
                       si_current->ch[ch].gr[gr].mixed_block_flag = getbits(&bs,1);
                        for (i=0; i<2; i++)
                             si_current->ch[ch].gr[gr].table_select[i] = getbits(&bs, 5);
                        for (i=0; i<3; i++)
                             si_current->ch[ch].gr[gr].subblock_gain[i] = getbits(&bs, 3);

              /* Set region_count parameters since they are implicit in this case. */

                        if (si_current->ch[ch].gr[gr].block_type == 0) {
                           printf("Side info bad: block_type == 0 in split block.\n");
            bad_block_count++;//test;
            exit(0);
            goto LP;
                        }
                        else if (si_current->ch[ch].gr[gr].block_type == 2
                                 && si_current->ch[ch].gr[gr].mixed_block_flag == 0)
                             si_current->ch[ch].gr[gr].region0_count = 8; /* MI 9; */
                        else si_current->ch[ch].gr[gr].region0_count = 7; /* MI 8; */
                        si_current->ch[ch].gr[gr].region1_count = 20 -
                                    si_current->ch[ch].gr[gr].region0_count;
                   }
                  else {
                       for (i=0; i<3; i++)
                            si_current->ch[ch].gr[gr].table_select[i] = getbits(&bs, 5);
                       si_current->ch[ch].gr[gr].region0_count = getbits(&bs, 4);
                       si_current->ch[ch].gr[gr].region1_count = getbits(&bs, 3);
                       si_current->ch[ch].gr[gr].block_type = 0;
```

```
                }
                    si_current->ch[ch].gr[gr].preflag = getbits(&bs,1);
                    si_current->ch[ch].gr[gr].scalefac_scale = getbits(&bs,1);
                    si_current->ch[ch].gr[gr].count1table_select = getbits(&bs,1);
            }
        }
    }
    else /* Layer 3 LSF */
    {
          exit(0);
  }

if(si_previous->main_data_end==0)
{main_data_beg_bs->bit_idx=bs.bit_idx;
main_data_beg_bs->byte_idx_ptr=bs.byte_idx_ptr;}
else
{main_data_beg_bs->bit_idx=7;
main_data_beg_bs->byte_idx_ptr=starting_index-(si_previous->main_data_end)+1;}
//***********************************End Analyzing Side Info*****************************//
/*if((si->ch[ch].gr[gr].block_type != 0)
   (sample_rate_table[header_info->ID][header_info->sampling_frequency_index]==44100)
   &&(bitrate_table[header_info->ID][header_info->layer][header_info->bitrate_index]==192)
   &&(!si->main_data_begin)
   &&(frame_length==522)
   &&(!header_info->ID)
   &&(header_info->layer==3)
   &&(si->main_data_begin==0))*/
   {frame_stat++;//test;
//***********************************Start printing the header information*********************//
/*if(sample_rate_table[header_info->ID][header_info->sampling_frequency_index])
{printf("frame_length is %d\n",header_info->frame_length);}
else
{printf("Sampling freq is 0.\n");goto LP;}
printf("Header information as follows:\n");
printf("ID is 0x%x ",header_info->ID);
if(header_info->layer) printf("layeris 0x%x ",header_info->layer); else printf("Wrong layer number! ");
printf("protection_bit is 0x%x    ",header_info->protection_bit);
printf("bitrate_index is 0x%x,bitrate is %d ", header_info->bitrate_index,
bitrate_table[header_info->ID][header_info->layer][header_info->bitrate_index]);
printf("sampling_freq_index is 0x%x,sampling_frequency is %d ",header_info->sampling_frequency_index,
sample_rate_table[header_info->ID][header_info->sampling_frequency_index]);
printf("pading_bit is 0x%x ",header_info->pading_bit);
printf("private_bit is 0x%x ",header_info->private_bit);
printf("mode index is 0x%x, ",header_info->mode);
if(header_info->mode==3) printf("Single Channel.SideInfo is 17 bits. ");
else if(header_info->mode==2) printf("Dual Channel. SideInfo is 32 bits. ");
else if(header_info->mode==1) printf("Joint Stero Mode. ");
else if(header_info->mode==0) printf("Stero Mode. ");
printf("mode_extension is 0x%x ",header_info->mode_extension);
printf("copy_right is 0x%x ",header_info->copy_right);
printf("original_home is 0x%x ",header_info->original_home);
printf("emphasis is 0x%x \n",header_info->emphasis);*/
//***********************************End of printing the header information*********************//
```

```
//************************************Start of printing Side Info*****************************//
/*
printf("Side Information as follows:\n");
printf("Main_data_End:%d    ",si_current->main_data_end);
printf("Main_data_begin:%d    ",main_data_beg_bs->byte_idx_ptr-data_header);
printf("Private_bits:%d    ",si_current->private_bits);
for (ch=0; ch<stereo; ch++)
                    for (i=0; i<4; i++)
                    printf("channel[%d],scfsi[%d] is:%d    ",ch,i,si_current->ch[ch].scfsi[i]);printf("\n");
for (gr=0; gr< grNum ; gr++) {
                    for (ch=0; ch<stereo; ch++) {
               printf("Info of Granul %d,channel %d is as following:    ",gr,ch);
                        printf("part2_3_length:%d      ",si_current->ch[ch].gr[gr].part2_3_length);
                        printf("big_values:%d    ",si_current->ch[ch].gr[gr].big_values);
            if(si_current->ch[ch].gr[gr].big_values!=0) big_value_1_count++; else big_value_0_count++;//test;
               printf("global_gain:%d    ",si_current->ch[ch].gr[gr].global_gain);
               printf("scalefac_compress:%d    ",si_current->ch[ch].gr[gr].scalefac_compress);
               printf("window_switching_flag:%d    ",si_current->ch[ch].gr[gr].window_switching_flag);
                        if (si_current->ch[ch].gr[gr].window_switching_flag) {
                            printf("block_type:0x%x    ",si_current->ch[ch].gr[gr].block_type);
                            if((si_current->ch[ch].gr[gr].block_type)==0) {block_type_0_count++;}//test;
                            printf("mixed_block_flag:0x%x    ",si_current->ch[ch].gr[gr].mixed_block_flag);
                            for (i=0; i<2; i++)
                                printf("table_select [%d]:%d    ",i,si_current->ch[ch].gr[gr].table_select[i]);
                            for (i=0; i<3; i++)
                   printf("subblock_gain [%d]:%d    ",i,si_current->ch[ch].gr[gr].subblock_gain[i]);
                      printf("region0_count:%d    ",si_current->ch[ch].gr[gr].region0_count);
                     printf("region1_count:%d    ",si_current->ch[ch].gr[gr].region1_count);
                        }
                        else {
                           for (i=0; i<3; i++)
                          printf("table_select [%d]:%d    ",i,si_current->ch[ch].gr[gr].table_select[i]);
                          printf("region0_count:%d    ",si_current->ch[ch].gr[gr].region0_count);
                        printf("region1_count:%d    ",si_current->ch[ch].gr[gr].region1_count);
                          printf("block_type:0x%x    ",si_current->ch[ch].gr[gr].block_type);
                          if((si_current->ch[ch].gr[gr].block_type)==0) block_type_0_count++;//test;
                        }
    printf("scalefac_scale:%d    ",si_current->ch[ch].gr[gr].scalefac_scale);
    printf("count1table_select:%d\n",si_current->ch[ch].gr[gr].count1table_select );
                }
        }*/
        //printf("\n");
//************************************End of printing Side Info*****************************//
}
LP:;
if(header_info->frame_length)
*index=*index+header_info->frame_length;
else *index=*index+1;
//printf("\nindex now is %d\n",*index-data_header);
//printf("I do return!!\n");
//returning the position of next frame
```

```
}
/*****************************Header_Reader Definition Ends*************************/

/*****************************Function:Find_the_first_frame*************************/
/**input:address of data_header*************************/
/**output:address of the first frame, the syncword offset of this frame,*********************************/
inline void find_the_first_frame
        (unsigned char **address_of_first_frame,
        unsigned long int *sycnword_count)
{
   unsigned char *index=&data_header;
   unsigned char *prvious_frame_index;
   unsigned char *first_frame_index=0;
   unsigned char valid[2]={0,0};
   unsigned char first_time_valid=0;
   unsigned char i=0;
   bit_stream_struc main_data_bs[2];
   header_info_type header_info[2];
   side_info_type si[2];
   header_info[0].frame_length=0;
   header_info[1].frame_length=0;
   for(;;)
   {
        prvious_frame_index=index-header_info[i].frame_length;
        i++;
        if(i>1) i=0;
        header_reader(&index,sycnword_count,&header_info[i],&si[1],&si[0],&main_data_bs[i],&valid[i]);
        if((!first_time_valid)&&(valid[1])&&(valid[0]))
        {
             first_frame_index=prvious_frame_index;
             (*address_of_first_frame)=first_frame_index;
             first_time_valid=1;
//           printf("First frame found.\n");
//           printf("address_of_first_frame is %d\n",*address_of_first_frame-data_header);
             return;
        }
   }
}
```

6.4 huffman_decode.h

```
#define HUFFBITS unsigned long int
#define HTN 34
#define MXOFF     250
#include "huffarray.h"

HUFFBITS dmask = 1 << (sizeof(HUFFBITS)*8-1);

struct huffcodetab {
   char tablename[3];       /*string, containing table_description    */
   unsigned int xlen;       /*max. x-index+                           */
   unsigned int ylen;       /*max. y-index+                           */
   unsigned int linbits; /*number of linbits                 */
   unsigned int linmax;    /*max number to be stored in linbits       */
   int ref;           /*a positive value indicates a reference*/
```

```
    HUFFBITS *table;    /*pointer to array[xlen][ylen]            */
    unsigned char *hlen;    /*pointer to array[xlen][ylen]          */
    unsigned char (*val)[2];/*decoder tree                         */
    unsigned int treelen; /*length of decoder tree                */
} ;
struct huffcodetab ht[HTN];

/***********huffman definition ends*******/
/* read the huffman decoder table */
inline int read_decoder_table()
{
    unsigned char tab_num_cnt=0;
    for(tab_num_cnt=0;tab_num_cnt<HTN;tab_num_cnt++)
    {
            ht[tab_num_cnt].tablename[2]='\0';
            ht[tab_num_cnt].tablename[1]=48+tab_header[tab_num_cnt][0]%10;
            ht[tab_num_cnt].tablename[0]=48+tab_header[tab_num_cnt][0]/10;
            ht[tab_num_cnt].treelen=tab_header[tab_num_cnt][1];
            ht[tab_num_cnt].xlen=tab_header[tab_num_cnt][2];
            ht[tab_num_cnt].ylen=tab_header[tab_num_cnt][3];
            ht[tab_num_cnt].linbits=tab_header[tab_num_cnt][4];
    }
    ht[0].val=v_tab_0;
    ht[1].val=v_tab_1;
    ht[2].val=v_tab_2;
    ht[3].val=v_tab_3;
    ht[4].val=v_tab_4;
    ht[5].val=v_tab_5;
    ht[6].val=v_tab_6;
    ht[7].val=v_tab_7;
    ht[8].val=v_tab_8;
    ht[9].val=v_tab_9;
    ht[10].val=v_tab_10;
    ht[11].val=v_tab_11;
    ht[12].val=v_tab_12;
    ht[13].val=v_tab_13;
    ht[14].val=v_tab_14;
    ht[15].val=v_tab_15;
    ht[16].val=v_tab_16;
    ht[17].val=v_tab_17;
    ht[18].val=v_tab_18;
    ht[19].val=v_tab_19;
    ht[20].val=v_tab_20;
    ht[21].val=v_tab_21;
    ht[22].val=v_tab_22;
    ht[23].val=v_tab_23;
    ht[24].val=v_tab_24;
    ht[25].val=v_tab_25;
    ht[26].val=v_tab_26;
    ht[27].val=v_tab_27;
    ht[28].val=v_tab_28;
    ht[29].val=v_tab_29;
    ht[30].val=v_tab_30;
```

```
   ht[31].val=v_tab_31;
   ht[32].val=v_tab_32;
   ht[33].val=v_tab_33;
    return 34;
}

/****************** huffman initial function*******************/
int huffman_initialized = 0;
void initialize_huffman() {
   read_decoder_table();
}




/****************** huffman decoder function*******************/


inline int huffman_decoder
   (bit_stream_struc *main_data_beg_bs,
     struct huffcodetab *h,
     int *x, int *y, int *v, int *w)
{

    HUFFBITS level;
    int point = 0;
    int error = 1;
    int bit_used_count=0;
    level          = dmask;
    //printf("1111111\n");
    if (h->val == NULL) return 2;
    //printf("tree legnth read by huffman_decoder is %d \n",h->treelen);
    /* table 0 needs no bits */
    if ( h->treelen == 0)
    {    *x = *y = 0;
         return bit_used_count;
    }


    /* Lookup in Huffman table. */

    do {
       if (h->val[point][0]==0) {      /*end of tree*/
          *x = h->val[point][1] >> 4;
          *y = h->val[point][1] & 0xf;
                    //return bit_used_count;
          error = 0;

          break;
       }
       if (get1bit(main_data_beg_bs)) {
          while (h->val[point][1] >= MXOFF) point += h->val[point][1];
          point += h->val[point][1];
```

```
            bit_used_count++;
            //printf("\nDecoding Boss:main_data_beg_bs.byte_idx_ptr is %d\n",main_data_beg_bs->byte_idx_ptr-data_header);
            //printf("\nDecoding Boss:main_data_beg_bs.bit_idx %d\n",main_data_beg_bs->bit_idx);
         }
         else {
            while (h->val[point][0] >= MXOFF) point += h->val[point][0];
            point += h->val[point][0];
            bit_used_count++;
         }
         level >>= 1;
      } while (level     || (point < ht->treelen) );
      /* Check for error. */
      if (error) { /* set x and y to a medium value as a simple concealment */
         printf("Illegal Huffman code in data.\n");
         *x = (h->xlen-1 << 1);
         *y = (h->ylen-1 << 1);
      }
      /* Process sign encodings for quadruples tables. */

      if (h->tablename[0] == '3'
            && (h->tablename[1] == '2' || h->tablename[1] == '3')) {
         *v = (*y>>3) & 1;
         *w = (*y>>2) & 1;
         *x = (*y>>1) & 1;
         *y = *y & 1;

         /* v, w, x and y are reversed in the bitstream.
               switch them around to make test bistream work. */

/*       {int i=*v; *v=*y; *y=i; i=*w; *w=*x; *x=i;}    MI */

         if (*v)
         {
           bit_used_count++;
           if (get1bit(main_data_beg_bs) == 1) *v = -*v;
         }

         if (*w)
         {
           bit_used_count++;
           if (get1bit(main_data_beg_bs) == 1) *w = -*w;
         }
         if (*x)
         {
           bit_used_count++;
           if (get1bit(main_data_beg_bs) == 1) *x = -*x;
         }
         if (*y)
         {
           bit_used_count++;
           if (get1bit(main_data_beg_bs) == 1) *y = -*y;
         }
```

```
        return bit_used_count;
        }

    /* Process sign and escape encodings for dual tables. */

    else {

          /* x and y are reversed in the test bitstream.
             Reverse x and y here to make test bitstream work. */

/*       removed 11/11/92 -ag
            {int i=*x; *x=*y; *y=i;}
*///    printf("22221\n");
        if (h->linbits)
           if ((h->xlen-1) == *x)
              {*x += getbits(main_data_beg_bs,h->linbits); bit_used_count+=h->linbits;}
        if (*x)
        {
          bit_used_count++;
          if (get1bit(main_data_beg_bs) == 1) *x = -*x;
        }
        if (h->linbits)
           if ((h->ylen-1) == *y)
              {*y += getbits(main_data_beg_bs,h->linbits);    bit_used_count+=h->linbits;}
        if (*y)
        {
          bit_used_count++;
          if (get1bit(main_data_beg_bs) == 1) *y = -*y;
        }
        return bit_used_count;
        }
     //main_data_beg_bs->byte_idx_ptr=byte_idx_ptr_temp_2;
        //main_data_beg_bs->bit_idx=bit_idx_temp_2;
    return error;
}




/********************** mp3_Huffman decoder***************/
inline void huffman_decode
   (long int is[SBLIMIT][SSLIMIT],
     side_info_type *si,
     int ch, int gr,
     header_info_type *header_info,
     bit_stream_struc *main_data_beg_bs)
{
      unsigned char *byte_idx_ptr_temp;
      unsigned char bit_idx_temp;

      byte_idx_ptr_temp = main_data_beg_bs->byte_idx_ptr;
      bit_idx_temp=main_data_beg_bs->bit_idx;
```

```
    int i, x=1, y=1;
    int v, w;
    int bit_used=0;
    struct huffcodetab *h;
    h=malloc(sizeof(struct huffcodetab));
    h=(struct huffcodetab *)h;

    int region1Start;
    int region2Start;
    int samplefreq;
    int currentBit, grBits;
    int part2_length;
    struct gr_info_type *gi;
    //int bt = (*si).ch[ch].gr[gr].window_switching_flag && ((*si).ch[ch].gr[gr].block_type == 2);

    gi = (struct gr_info_type *) &(*si).ch[ch].gr[gr];
    samplefreq = header_info->sampling_frequency_index + (header_info->ID * 3);
/*calculate part2_length*/

      if (/*gi->window_switching_flag &&*/ (gi->block_type == 2)) {
          if (gi->mixed_block_flag)
            {
              part2_length=17*slen[0][gi->scalefac_compress]
                           +18*slen[1][gi->scalefac_compress];

             }
             else {
               part2_length=18*slen[0][gi->scalefac_compress]
                           +18*slen[1][gi->scalefac_compress];
            }
          }
          else {      /* LONG types 0,1,3 */
                part2_length=11*slen[0][gi->scalefac_compress]
                             +10*slen[1][gi->scalefac_compress];

          }

/*
printf("The following is is serving as within one granual:\n");
    printf("part2_length is %d\n",part2_length);*/
    /*printf("main_data_beg_bs.byte_idx_ptr is %d\n",main_data_beg_bs->byte_idx_ptr-data_header);
    printf("main_data_beg_bs.bit_idx %d\n",main_data_beg_bs->bit_idx);*/

    /*if((ch==1)&&(gr==1)) main_data_beg_bs->byte_idx_ptr=main_data_beg_bs->byte_idx_ptr+part2_length/8-1;
    else*/ main_data_beg_bs->byte_idx_ptr=main_data_beg_bs->byte_idx_ptr+part2_length/8;
    if(( main_data_beg_bs->bit_idx <    (part2_length)%8))
    {

      main_data_beg_bs->bit_idx=8-(part2_length)%8+main_data_beg_bs->bit_idx;
      main_data_beg_bs->byte_idx_ptr++;
    }
    else
    main_data_beg_bs->bit_idx=main_data_beg_bs->bit_idx-((part2_length)%8);
```

```
/*printf("---------huffman coded data address for gr[%d],ch[%d] is as follows--------\n",gr,ch);
printf("row is %d,      ",(main_data_beg_bs->byte_idx_ptr-data_header)/39);
printf("column is %d.\n ",(main_data_beg_bs->byte_idx_ptr-data_header)%39);
   printf("main_data_beg_bs.byte_idx_ptr is %d        ",main_data_beg_bs->byte_idx_ptr-data_header);
   printf("main_data_beg_bs.bit_idx %d\n",main_data_beg_bs->bit_idx);
   printf("---------huffman coded data address for gr[%d],ch[%d] Ends--------\n",gr,ch);*/

 /* Find region boundary for short block case. */

 if ( ((*si).ch[ch].gr[gr].window_switching_flag) &&
        ((*si).ch[ch].gr[gr].block_type == 2) ) {

      /* Region2. */
        region1Start = 36;    /* sfb[9/3]*3=36 */
        region2Start = 576; /* No Region2 for short block case. */
 }
 else {                  /* Find region boundary for long block case. */

      region1Start = sfBandIndex[samplefreq]
                                  .l[(*si).ch[ch].gr[gr].region0_count + 1]; /* MI */
      region2Start = sfBandIndex[samplefreq]
                                    .l[(*si).ch[ch].gr[gr].region0_count +
                                    (*si).ch[ch].gr[gr].region1_count + 2]; /* MI */
      }

 /* Read bigvalues area. */
//       printf("big value              is %d",(*si).ch[ch].gr[gr].big_values*2);


 for (i=0; i<(*si).ch[ch].gr[gr].big_values*2; i+=2) {

      if          (i<region1Start) h = &ht[(*si).ch[ch].gr[gr].table_select[0]];
      else if (i<region2Start) h = &ht[(*si).ch[ch].gr[gr].table_select[1]];
           else                          h = &ht[(*si).ch[ch].gr[gr].table_select[2]];
      //printf("tree legnth inside the fucntion III_huffman_decode is %d \n\n",h->treelen);
      bit_used+=huffman_decoder(main_data_beg_bs,h, &x, &y, &v, &w);
      is[i/SSLIMIT][i%SSLIMIT] = x;
      is[(i+1)/SSLIMIT][(i+1)%SSLIMIT] = y;
      }
 /* Read count1 area. */

 h = &ht[((*si).ch[ch].gr[gr].count1table_select)+32];
 /*printf("count1table_select is %d\n",((*si).ch[ch].gr[gr].count1table_select)+32);*/
 while ((bit_used <    ((*si).ch[ch].gr[gr].part2_3_length - part2_length)) &&
    ( i < SSLIMIT*SBLIMIT )) {

      bit_used+=huffman_decoder(main_data_beg_bs,h, &x, &y, &v, &w);
      is[i/SSLIMIT][i%SSLIMIT] = v;
      is[(i+1)/SSLIMIT][(i+1)%SSLIMIT] = w;
      is[(i+2)/SSLIMIT][(i+2)%SSLIMIT] = x;
      is[(i+3)/SSLIMIT][(i+3)%SSLIMIT] = y;
      i += 4;
      }
```

```
    /* Zero out rest. */
    for (; i<SSLIMIT*SBLIMIT; i++)
        is[i/SSLIMIT][i%SSLIMIT] = 0;
    /*printf("the 576 sampling value is the following:\n");*/
/*     for(x=0;x<SBLIMIT;x++)
    {    printf("subband %d :",x);
            for(y=0;y<SSLIMIT;y++)
            printf("%d ",is[x][y]);

            printf("\n");


     }
*/

    main_data_beg_bs->byte_idx_ptr=byte_idx_ptr_temp;
    main_data_beg_bs->bit_idx=bit_idx_temp;
}




/***********huffman definition ends*******/



6.5get_scalfefactor.h
inline void get_scale_factors
   (side_info_type *si,
   scalefac_type *scalefac,
   int gr,
   int ch,
   bit_stream_struc bitstream)
{
   int sfb, i, window;
   struct gr_info_type *gr_info = &(si->ch[ch].gr[gr]);
   bit_stream_struc *bs=&bitstream;

   if (gr_info->window_switching_flag && (gr_info->block_type == 2))
   {
        if (gr_info->mixed_block_flag)
        { // mixed blocks;
            for (sfb = 0; sfb < 8; sfb++)
                scalefac->ch[ch].l[sfb] = getbits(bs,slen[0][gr_info->scalefac_compress]);
            for (sfb = 3; sfb < 6; sfb++)
                for (window=0; window<3; window++)
                    scalefac->ch[ch].s[window][sfb] = getbits(bs,slen[0][gr_info->scalefac_compress]);
            for (sfb = 6; sfb < 12; sfb++)
                for (window=0; window<3; window++)
                    scalefac->ch[ch].s[window][sfb] = getbits(bs,slen[1][gr_info->scalefac_compress]);
            for (sfb=12,window=0; window<3; window++)
```

```
                scalefac->ch[ch].s[window][sfb] = 0;
        }
        else
        {   // short blocks;
            for (sfb=0; sfb < 6; sfb++)
                for (window=0; window<3; window++)
                    scalefac->ch[ch].s[window][sfb] = getbits(bs,slen[0][gr_info->scalefac_compress]);
            for (sfb=6; sfb < 12; sfb++)
                for (window=0; window<3; window++)
                    scalefac->ch[ch].s[window][sfb] = getbits(bs,slen[1][gr_info->scalefac_compress]);
            for (sfb=12,window=0; window<3; window++)
                scalefac->ch[ch].s[window][sfb] = 0;
        }
    }
    else
    {     // long blocks;
        for (sfb=0; sfb < 11; sfb++)
            scalefac->ch[ch].l[sfb] = getbits(bs,slen[0][gr_info->scalefac_compress]);
        for (sfb=11; sfb < 21; sfb++)
            scalefac->ch[ch].l[sfb] = getbits(bs,slen[1][gr_info->scalefac_compress]);
        scalefac->ch[ch].l[21] = 0;
    }
}
```

6.7 requantize.h

```
int pretab[22] = {0,0,0,0,0,0,0,0,0,0,1,1,1,1,2,2,3,3,3,2,0};

float powershort1[8]={
        1.0000000000000000,0.2500000000000000,0.0625000000000000,0.0156250000000000,
        0.0039062500000000,0.0009765625000000,0.0002441406250000,0.0000610351562500,

};

float powershort2[32]={
        1.0000000000000000,0.7071067811865476,0.5000000000000000,0.3535533905932738,
        0.2500000000000000,0.1767766952966369,0.1250000000000000,0.0883883476483184,
        0.0625000000000000,0.0441941738241592,0.0312500000000000,0.0220970869120796,
        0.0156250000000000,0.0110485434560398,0.0078125000000000,0.0055242717280199,
        0.0039062500000000,0.0027621358640100,0.0019531250000000,0.0013810679320050,
        0.0009765625000000,0.0006905339660025,0.0004882812500000,0.0003452669830012,
        0.0002441406250000,0.0001726334915006,0.0001220703125000,0.0000863167457503,
        0.0000610351562500,0.0000431583728752,0.0000305175781250,0.0000215791864376,

};

float powertable1[256]={
        0.0000000000000002,0.0000000000000002,0.0000000000000002,0.0000000000000003,
        0.0000000000000003,0.0000000000000004,0.0000000000000004,0.0000000000000005,
        0.0000000000000006,0.0000000000000007,0.0000000000000009,0.0000000000000011,
        0.0000000000000013,0.0000000000000015,0.0000000000000018,0.0000000000000021,
        0.0000000000000025,0.0000000000000030,0.0000000000000036,0.0000000000000042,
        0.0000000000000050,0.0000000000000060,0.0000000000000071,0.0000000000000084,
        0.0000000000000100,0.0000000000000119,0.0000000000000142,0.0000000000000169,
```

0.0000000000000201,0.0000000000000239,0.0000000000000284,0.0000000000000338,
0.0000000000000402,0.0000000000000478,0.0000000000000568,0.0000000000000676,
0.0000000000000804,0.0000000000000956,0.0000000000001137,0.0000000000001352,
0.0000000000001608,0.0000000000001912,0.0000000000002274,0.0000000000002704,
0.0000000000003216,0.0000000000003824,0.0000000000004547,0.0000000000005408,
0.0000000000006431,0.0000000000007648,0.0000000000009095,0.0000000000010816,
0.0000000000012862,0.0000000000015296,0.0000000000018190,0.0000000000021632,
0.0000000000025724,0.0000000000030592,0.0000000000036380,0.0000000000043263,
0.0000000000051449,0.0000000000061183,0.0000000000072760,0.0000000000086526,
0.0000000000102898,0.0000000000122367,0.0000000000145519,0.0000000000173052,
0.0000000000205795,0.0000000000244733,0.0000000000291038,0.0000000000346105,
0.0000000000411590,0.0000000000489466,0.0000000000582077,0.0000000000692210,
0.0000000000823181,0.0000000000978932,0.0000000001164153,0.0000000001384419,
0.0000000001646361,0.0000000001957865,0.0000000002328306,0.0000000002768839,
0.0000000003292723,0.0000000003915729,0.0000000004656613,0.0000000005537677,
0.0000000006585445,0.0000000007831458,0.0000000009313226,0.0000000011075354,
0.0000000013170890,0.0000000015662916,0.0000000018626451,0.0000000022150709,
0.0000000026341780,0.0000000031325833,0.0000000037252903,0.0000000044301417,
0.0000000052683561,0.0000000062651665,0.0000000074505806,0.0000000088602835,
0.0000000105367121,0.0000000125303330,0.0000000149011612,0.0000000177205669,
0.0000000210734243,0.0000000250606661,0.0000000298023224,0.0000000354411338,
0.0000000421468485,0.0000000501213321,0.0000000596046448,0.0000000708822677,
0.0000000842936970,0.0000001002426642,0.0000001192092896,0.0000001417645353,
0.0000001685873940,0.0000002004853285,0.0000002384185791,0.0000002835290706,
0.0000003371747881,0.0000004009706570,0.0000004768371582,0.0000005670581412,
0.0000006743495762,0.0000008019413140,0.0000009536743164,0.0000011341162825,
0.0000013486991523,0.0000016038826280,0.0000019073486328,0.0000022682325649,
0.0000026973983047,0.0000032077652559,0.0000038146972656,0.0000045364651299,
0.0000053947966094,0.0000064155305119,0.0000076293945312,0.0000090729302597,
0.0000107895932188,0.0000128310610238,0.0000152587890625,0.0000181458605195,
0.0000215791864376,0.0000256621220475,0.0000305175781250,0.0000362917210389,
0.0000431583728752,0.0000513242440951,0.0000610351562500,0.0000725834420778,
0.0000863167457503,0.0001026484881902,0.0001220703125000,0.0001451668841556,
0.0001726334915006,0.0002052969763803,0.0002441406250000,0.0002903337683112,
0.0003452669830012,0.0004105939527606,0.0004882812500000,0.0005806675366224,
0.0006905339660025,0.0008211879055212,0.0009765625000000,0.0011613350732448,
0.0013810679320050,0.0016423758110424,0.0019531250000000,0.0023226701464897,
0.0027621358640100,0.0032847516220848,0.0039062500000000,0.0046453402929794,
0.0055242717280199,0.0065695032441696,0.0078125000000000,0.0092906805859588,
0.0110485434560398,0.0131390064883393,0.0156250000000000,0.0185813611719175,
0.0220970869120796,0.0262780129766786,0.0312500000000000,0.0371627223438350,
0.0441941738241592,0.0525560259533572,0.0625000000000000,0.0743254446876701,
0.0883883476483184,0.1051120519067143,0.1250000000000000,0.1486508893753401,
0.1767766952966369,0.2102241038134286,0.2500000000000000,0.2973017787506803,
0.3535533905932738,0.4204482076268573,0.5000000000000000,0.5946035575013605,
0.7071067811865476,0.8408964152537145,1.0000000000000000,1.1892071150027210,
1.4142135623730951,1.6817928305074290,2.0000000000000000,2.3784142300054421,
2.8284271247461903,3.3635856610148580,4.0000000000000000,4.7568284600108841,
5.6568542494923806,6.7271713220297160,8.0000000000000000,9.5136569200217682,
11.313708498984761,13.454342644059432,16.000000000000000,19.027313840043536,
22.627416997969522,26.908685288118864,32.000000000000000,38.054627680087073,
45.254833995939045,53.817370576237728,64.000000000000000,76.109255360174146,
90.509667991878089,107.634741152475456,128.000000000000000,152.218510720348291,

```
        181.0193359837561786,215.2694823049509125,256.0000000000000000,304.4370214406965829,
        362.0386719675123572,430.5389646099018250,512.0000000000000000,608.8740428813931658,
        724.0773439350247145,861.0779292198036501,1024.0000000000000000,1217.7480857627863315,
        1448.1546878700494290,1722.1558584396073002,2048.0000000000000000,2435.4961715255726631,
};

float powertable2[16]={
   1.0000000000000000,0.7071067811865476,0.5000000000000000,0.3535533905932738,
   0.2500000000000000,0.1767766952966369,0.1250000000000000,0.0883883476483184,
   0.0625000000000000,0.0441941738241592,0.0312500000000000,0.0220970869120796,
   0.0156250000000000,0.0110485434560398,0.0078125000000000,0.0055242717280199,
};

float powertable3[8200]={
        0.0000000000000000,1.0000000000000000,2.5198420997897464,4.3267487109222245,
        6.3496042078727974,8.5498797333834844,10.9027235569928358,13.3905182794067219,
        15.9999999999999982,18.7207544074671333,21.5443469003188319,24.4637809962624679,
        27.4731418212799596,30.5673509403698418,33.7419916984532122,36.9931811149570464,
        40.3174735966359350,43.7117870411899929,47.1733450957601264,50.6996313257169433,
        54.2883523318981176,57.9374077040035189,61.6448652744184997,65.4089405365859875,
        69.2279793747555914,73.1004434553216385,77.0248977785916082,80.9999999999999858,
        85.0244912125185266,89.0971879448895550,93.2169751786157406,97.3828002241331632,
        101.5936673259647449,105.8486328898622446,110.1468012434344104,114.4873208566006042,
        118.8693809602065272,123.2922085109002381,127.7550654583605763,132.2572462775524684,
        136.7980757341357219,141.3769068556919137,145.9931190852308589,150.6461165966290991,
        155.3353267543467382,160.0601987020527872,164.8202020667334864,169.6148257665186065,
        174.4435769118853443,179.3059797911255657,184.2015749320192697,189.1299182325756192,
        194.0905801544968483,199.0831449737167702,204.1072100829694023,209.1623853418764725,
        214.2482924705075220,219.3645644827778369,224.5108451564121310,229.6867885365223003,
        234.8920584701317580,240.1263281692324938,245.3892798001850508,250.6806040974726102,
        255.9999999999999147,261.3471743082886860,266.7218413610644916,272.1237227298604466,
        277.5525469303796058,283.0080491494618968,288.4899709865989053,293.9980602090224693,
        299.5320705194740754,305.0917613358298581,310.6768975818220611,316.2872494881558509,
        321.9225924033717661,327.5827066138553505,333.2673771724374205,338.9763937350702463,
        344.7095504051013108,350.4666455847001316,356.2474818330260860,362.0518657307513877,
        367.8796077505825792,373.7305221334451062,379.6044267700207797,385.5011430873460654,
        391.4204959401994302,397.3623135070237140,403.3264271901446705,409.3126715200626222,
        415.3208840636079913,421.3509053357647076,427.4025787149761868,433.4757503617616976,
        439.5702691404792972,445.6859865440827093,451.8227566217275921,457.9804359090912840,
        464.1588833612777307,470.3579602881872574,476.5775302922363039,482.8174592083204288,
        489.0776150459174119,495.3578679332358092,501.6580900633168767,507.9781556420036850,
        514.3179408376964830,520.6773237328167170,527.0561842769060377,533.4544042412917406,
        539.8718671752512819,546.3084583636150455,552.7640647857460863,559.2385750758419363,
        565.7318794845041339,572.2438698415234057,578.7744395198337770,585.3234834005884295,
        591.8908978393126290,598.4765806330925670,605.0804309887604404,611.7023494920364328,
        618.3422380775919009,624.9999999999997726,631.6755398055374826,638.3687633048116368,
        645.0795775461748462,651.8078907899041496,658.5536124831149891,665.3166532353835692,
        672.0969247950522458,678.8943400261942998,685.7088128862142185,692.5402584040620013,
        699.3885926590397730,706.2537327601805828,713.1355968261797216,720.0341039658603677,
        726.9491742591543471,733.8807287385820928,740.8286893712154324,747.7929790411053546,
        754.7735215321619080,761.7702415114704309,768.7830645130295579,775.8119169218989555,
        782.8567259587424587,789.9174196647544477,796.9939268869579791,804.0861772638627372,
        811.1941012114709793,818.3176299096222692,825.4566952886656281,832.6112300164486442,
```

839.7811674856160380,846.9664418012055194,854.1669877685351366,861.3827408813714328,
868.6136373103697679,875.8596138917820326,883.1206081164195894,890.3965581188675742,
897.6874026669418072,904.9930811513817162,912.3135335757718849,919.6487005466875644,
926.9985232640561890,934.3629435117289859,941.7419036482585852,949.1353465978742179,
956.5432158416521133,963.9654554088734812,971.4020098685654148,978.8528243212217603,
986.3178443906958819,993.7970162162635006,1001.2902864448500395,1008.7976022234180391,
1016.3189111915103240,1023.8541614739464194,1031.4033016736652826,1038.9662808647137808,
1046.5430485853758000,1054.1335548314366406,1061.7377500495838376,1069.3555851309356513,
1076.9870114046977960,1084.6319806319440886,1092.2904449995173763,1099.9623571140482454,
1107.6476699960892347,1115.3463370743606902,1123.0583121801059860,1130.7835495415540663,
1138.5220037784856686,1146.2736298969009567,1154.0383832837878799,1161.8162197019860287,
1169.6070952851459879,1177.4109665327807761,1185.2277903054077797,1193.0575238197798171,
1200.9001246442001047,1208.7555506939247607,1216.6237602266442082,1224.5047118380477968,
1232.3983644574657319,1240.3046773435874002,1248.2236100802567762,1256.1551225723394509,
1264.0991750416619652,1272.0557280230227661,1280.0247423602691015,1288.0061792024444003,
1295.9999999999995453,1304.0061665010680372,1312.0246407478061883,1320.0553850727928875,
1328.0983620954903017,1336.1535347187650586,1344.2208661254646813,1352.3003197750522304,
1360.3918594002961981,1368.4954490040145174,1376.6110528558708666,1384.7386354892244071,
1392.8781616980295439,1401.0295965337854796,1409.1929053025353369,1417.3680535619118928,
1425.5550071182326519,1433.7537320236374399,1441.9641945732744261,1450.1863613025282120,
1458.4201989842913463,1466.6656746262797242,1474.9227554683875496,1483.1914089800841339,
1491.4716028578516216,1499.7633050226595515,1508.0664836174794345,1516.3811070048375313,
1524.7071437644028720,1533.0445626906127927,1541.3933327903341706,1549.7534232805580814,
1558.1248035861303833,1566.5074433375150420,1574.9013123685908795,1583.3063807144794737,
1591.7226186094069362,1600.1499964845941122,1608.5884849661799763,1617.0380548731736781,
1625.4986772154356913,1633.9703231916887489,1642.4529641875576544,1650.9465717736345596,
1659.4511177035751643,1667.9665739122185641,1676.4929125137352912,1685.0301057998010492,
1693.5781262377956864,1702.1369464690269524,1710.7065393069794936,1719.2868777355877228,
1727.8779349075323353,1736.4796841425595630,1745.0920989258249847,1753.7151529062582540,
1762.3488198949503385,1770.9930738635630405,1779.6478889427596641,1788.3132394206563731,
1796.9890997412946945,1805.6754445031333489,1814.3722484575621365,1823.0794865074321933,
1831.7971337056094399,1840.5251652535437188,1849.2635564998579412,1858.0122829389563321,
1866.7713202096492751,1875.5406440937965726,1884.3202305149686708,1893.1100555371240262,
1901.9100953633042081,1910.7203263343453727,1919.5407249276056518,1928.3712677557098232,
1937.2119315653083049,1946.0626932358525210,1954.9235297783859551,1963.7944183343499844,
1972.6753361744035828,1981.5662606972593949,1990.4671694285329977,1999.3780400196069422,
2008.2988502465077545,2017.2295780087981711,2026.1702013284818804,2035.1206983489212234,
2044.0810473337687654,2053.0512266659125089,2062.0312148464308848,2071.0209904935645682,
2080.0205323416957981,2089.0298192403442954,2098.0488301531713660,2107.0775441569994655,
2116.1159404408390401,2125.1639983049317379,2134.2216971597995325,2143.2890165253097621,
2152.3659360297483545,2161.4524354089030567,2170.5484945051616705,2179.6540932666143817,
2188.7692117461710950,2197.8938301006887741,2207.0279285901042385,2216.1714875765837860,
2225.3244875236759981,2234.4869089954781884,2243.6587326558101267,2252.8399392673982220,
2262.0305096910701650,2271.2304248849536634,2280.4396659036897290,2289.6582138976523311,
2298.8860501121762354,2308.1231558867925742,2317.3695126544766936,2326.6251019409005494,
2335.8899053636932877,2345.1639046317131942,2354.4470815443232823,2363.7394179906791578,
2373.0408959490205234,2382.3514974859731410,2391.6712047558557970,2400.9999999999990905,
2410.3378655460651316,2419.6847838073813364,2429.0407372822746765,2438.4057085534191174,
2447.7796802871857835,2457.1626352330008558,2466.5545562227111986,2475.9554261699563540,
2485.3652280695473564,2494.7839449968487315,2504.2115601071736819,2513.6480566351788184,
2523.0934178942675317,2532.5476272760024585,2542.0106682495188579,2551.4825243609479912,
2560.9631792328441406,2570.4526165636184487,2579.9508201269791243,2589.4577737713743772,
2598.9734614194458118,2608.4978670674822752,2618.0309747848837105,2627.5727687136259192,

2637.1232330677353275,2646.6823521327646631,2656.2501102652768168,2665.8264918923327969,
2675.4114815109842311,2685.0050636877722354,2694.6072230582294651,2704.2179443263894427,
2713.8372122642972499,2723.4650117115279500,2733.1013275747095577,2742.7461448270482833,
2752.3994485078601429,2762.0612237221084797,2771.7314556399419416,2781.4101294962406428,
2791.0972305901655091,2800.7927442847094426,2810.4966560062589451,2820.2089512441521038,
2829.9296155502465808,2839.6586345384894230,2849.3959938844918724,2859.1416793251064519,
2868.8956766580085969,2878.6579717412846549,2888.4285504930212483,2898.2073988908973661,
2907.9945029717837315,2917.7898488313439884,2927.5934226236377071,2937.4052105607311205,
2947.2251989123078602,2957.0533740052865141,2966.8897222234368201,2976.7342300070049532,
2986.5868838523397244,2996.4476703115196869,3006.3165759919888842,3016.1935875561907778,
3026.0786917212094522,3035.9718752584108188,3045.8731249930901868,3055.7824278041207435,
3065.6997706236038539,3075.6251404365279996,3085.5585242804245354,3095.4999092450298122,
3105.4492824719491182,3115.4066311543256234,3125.3719425365088682,3135.3452039137287102,
3145.3264026317715434,3155.3155260866592471,3165.3125617243294982,3175.3174970403229054,
3185.3303195794674139,3195.3510169355699873,3205.3795767511078338,3215.4159867169250901,
3225.4602345719290497,3235.5123081027927583,3245.5721951436557902,3255.6398835758300265,
3265.7153613275099815,3275.7986163734794900,3285.8896367348288550,3295.9884104786665375,
3306.0949257178394873,3316.2091706106516540,3326.3311333605879554,3336.4608022160382461,
3346.5981654700231047,3356.7432114599264423,3366.8959285672249280,3377.0563052172210519,
3387.2243298787821004,3397.3999910640764028,3407.5832773283127608,3417.7741772694862448,
3427.9726795281198974,3438.1787727870123490,3448.3924457709872513,3458.6136872466445311,
3468.8424860221107338,3479.0788309467975523,3489.3227109111553546,3499.5741148464344406,
3509.8330317244444814,3520.0994505573185052,3530.3733603972750643,3540.6547503363885880,
3550.9436095063533685,3561.2399270782580061,3571.5436922623534883,3581.8548943078308184,
3592.1735225025936415,3602.4995661730372376,3612.8330146838275141,3623.1738574376813631,
3633.5220838751502015,3643.8776834744030566,3654.2406457510141990,3664.6109602577494115,
3674.9886165843563504,3685.3736043573544521,3695.7659132398293877,3706.1655329312247886,
3716.5724531671398836,3726.9866637191262271,3737.4081543944876103,3747.8369150360781532,
3758.2729355221072183,3768.7162057659411403,3779.1667157159076851,3789.6244553551055105,
3800.0894147012081703,3810.5615838062767580,3821.0409527565693679,3831.5275116723532847,
3842.0212507077194459,3852.5221600503959962,3863.0302299215672974,3873.5454505756893013,
3884.0678123003108340,3894.5973054158921514,3905.1339202756284976,3915.6776472652732082,
3926.2284768029603583,3936.7863993390337782,3947.3514053558706109,3957.9234853677135106,
3968.5026299204969291,3979.0888295916797688,3989.6820749900775809,4000.2823567556947637,
4010.8896655595613083,4021.5039921035654515,4032.1253271202945143,4042.7536613728693737,
4053.3889856547857562,4064.0312907897550758,4074.6805676315448181,4085.3368070638221070,
4095.9999999999981810,4106.6701373830710509,4117.3472101854749781,4128.0312094089258608,
4138.7221260842679840,4149.4199512713266813,4160.1246760587582685,4170.8362915638981576,
4181.5547889326180666,4192.2801593391768620,4203.0123939860741302,4213.7514841039101157,
4224.4974209512383823,4235.2501958144257515,4246.0098000075095115,4256.7762248720573552,
4267.5494617770309560,4278.3295021186422673,4289.1163373202198272,4299.9099588320714247,
4310.7103581313494942,4321.5175267219137822,4332.3314561342003799,4343.1521379250880273,
4353.9795636777671461,4364.8137250016052349,4375.6546135320222675,4386.5022209303588170,
4397.3565388837469072,4408.2175591049826835,4419.0852733324018118,4429.9596733297530591,
4440.8407508860727830,4451.7284978155603312,4462.6229059574570783,4473.5239671759227349,
4484.4316733599125655,4495.3460164230582450,4506.2669883035496241,4517.1945809640119478,
4528.1287863913894398,4539.0695965968279779,4550.0170036155586786,4560.9709995067805721,
4571.9315763535460064,4582.8987262626469601,4593.8724413645004461,4604.8527138130348249,
4615.8395357855815746,4626.8328994827570568,4637.8327971283588340,4648.8392209692510733,
4659.8521632752563164,4670.8716163390472502,4681.8975724760393859,4692.9300240242837390,
4703.9689633443595085,4715.0143828192667570,4726.0662748543254565,4737.1246318770681683,
4748.1894463371372694,4759.2607107061803617,4770.3384174777493172,4781.4225591671993243,
4792.5131283115852057,4803.6101174695613736,4814.7135192212854236,4825.8233261683153614,

4836.9395309335095590,4848.0621261609348949,4859.1911045157630724,4870.3264586841778510,
4881.4681813732768205,4892.6162653109768144,4903.7707032459193215,4914.9314879473749897,
4926.0986122051508573,4937.2720688294966749,4948.4518506510112275,4959.6379505205550231,
4970.8303613091520674,4982.0290759079043710,4993.2340872278973620,5004.4453882001153033,
5015.6629717753467048,5026.8868309241006500,5038.1169586365131181,5049.3533479222660389,
5060.5959918104927056,5071.8448833496995576,5083.1000156076734129,5094.3613816713996130,
5105.6289746469747115,5116.9027876595246198,5128.1828138531200239,5139.4690463906918012,
5150.7614784539473476,5162.0601032432932698,5173.3649139777471646,5184.6759038948594025,
5195.9930662506321823,5207.3163943194385865,5218.6458813939434549,5229.9815207850233492,
5241.3233058216846985,5252.6712298509919492,5264.0252862379829821,5275.3854683655954432,
5286.7517696345876175,5298.1241834634638508,5309.5027032883954234,5320.8873225631459718,
5332.2780347589978192,5343.6748333646755782,5355.0777118862715724,5366.4866638471721672,
5377.9016827879850098,5389.3227622664635419,5400.7498958574369681,5412.1830771527356774,
5423.6222997611230312,5435.0675573082189658,5446.5188434364317800,5457.9761518048871949,
5469.4394760893592320,5480.9088099821974538,5492.3841471922605706,5503.8654814448454999,
5515.3528064816200640,5526.8461160605520490,5538.3454039558464501,5549.8506639578736213,
5561.3618898731028821,5572.8790755240361250,5584.4022147491450596,5595.9313014027975441,
5607.4663293552011964,5619.0072924923297251,5630.5541847158656310,5642.1069999431283577,
5653.6657321070169928,5665.2303751559429656,5676.8009230537654730,5688.3773697797332716,
5699.9597093284164657,5711.5479357096473905,5723.1420429484587657,5734.7420250850209413,
5746.3478761745809607,5757.9595902874016247,5769.5771615087005557,5781.2005839385910804,
5792.8298516920212933,5804.4649588987149400,5816.1058997031113904,5827.7526682643065215,
5839.4052587559972380,5851.0636653664196274,5862.7278822982907514,5874.3979037687540767,
5886.0737240093203582,5897.7553372658094304,5909.4427377982956386,5921.1359198810505404,
5932.8348778024874264,5944.5396058651031126,5956.2500983854260994,5967.9663496939574543,
5979.6883541351207896,5991.4161060672022359,6003.1495998623004198,6014.8888299062700753,
6026.6337905986674741,6038.3844763527031319,6050.1408815951781435,6061.9030007664414370,
6073.6708283203315659,6085.4443587241266869,6097.2235864584890805,6109.0085060174196769,
6120.7991119081998477,6132.5953986513450218,6144.3973607805519350,6156.2049928426458791,
6168.0182893975361367,6179.8372450181577733,6191.6618542904307105,6203.4921118132024276,
6215.3280121982015771,6227.1695500699925105,6239.0167200659188893,6250.8695168360627576,
6262.7279350431890634,6274.5919693627056404,6286.4616144826068194,6298.3368651034325012,
6310.2177159382172249,6322.1041617124456025,6333.9961971640032061,6345.8938170431310937,
6357.7970161123785147,6369.7057891465583452,6381.6201309327016133,6393.5400362700074766,
6405.4654999698032043,6417.3965168554977936,6429.3330817625328564,6441.2751895383453302,
6453.2228350423138181,6465.1760131457240277,6477.1347187317160206,6489.0989466952469229,
6501.0686919430445414,6513.0439493935627979,6525.0247139769417117,6537.0109806349610153,
6549.0027443210010460,6560.9999999999963620,6573.0027426483984527,6585.0109672541284453,
6597.0246688165370870,6609.0438423463656363,6621.0684828657003891,6633.0985854079353885,
6645.1341450177269508,6657.1751567509572851,6669.2216156746899287,6681.2735168671342763,
6693.3308554176001053,6705.3936264264593774,6717.4618250051080395,6729.5354462759260059,
6741.6144853722335029,6753.6989374382601454,6765.7887976290967345,6777.8840611106634242,
6789.9847230596660665,6802.0907786635625598,6814.2022231205201024,6826.3190516393797225,
6838.4412594396180793,6850.5688417513074455,6862.7017938150829650,6874.8401108820989975,
6886.9837882139991052,6899.1328210828723968,6911.2872047712207859,6923.4469345719198827,
6935.6120057881862522,6947.7824137335364867,6959.9581537317535549,6972.1392211168531503,
6984.3256112330409451,6996.5173194346862147,7008.7143410862772726,7020.9166715623941855,
7033.1243062476678460,7045.3372405367481406,7057.5554698342684787,7069.7789895548103232,
7082.0077951228713573,7094.2418819728272865,7106.4812455489018248,7118.7258813051284960,
7130.9757847053224395,7143.2309512230394830,7155.4913763415515859,7167.7570555538031840,
7180.0279843623893612,7192.3041582795131035,7204.5855728269571046,7216.8722235360519335,
7229.1641059476405644,7241.4612156120492727,7253.7635480890503459,7266.0710989478375268,
7278.3838637669869058,7290.7018381344296358,7303.0250176474173713,7315.3533979124931648,

7327.6869745454596341,7340.0257431713462211,7352.3696994243800873,7364.7188389479542820,
7377.0731573945968194,7389.4326504259415742,7401.7973137126937218,7414.1671429346060904,
7426.5421337804427822,7438.9222819479509781,7451.3075831438345631,7463.6980330837177462,
7476.0936274921214135,7488.4943621024303866,7500.9002326568652279,7513.3112349064522277,
7525.7273646109943002,7538.1486175390446078,7550.5749894678729106,7563.0064761834419187,
7575.4430734803736414,7587.8847771619248306,7600.3315830399596962,7612.7834869349153450,
7625.2404846757799532,7637.7025721000636622,7650.1697450537676559,7662.6419993913596045,
7675.1193309757445604,7687.6017356782404022,7700.0892093785441830,7712.5817479647112123,
7725.0793473331250425,7737.5820033884729128,7750.0897120437139165,7762.6024692200580830,
7775.1202708469354548,7787.6431128619733499,7800.1709912109645302,7812.7039018478481012,
7825.2418407346767708,7837.7848038415968404,7850.3327871468154626,7862.8857866365806331,
7875.4437983051539049,7888.0068181547840140,7900.5748421956805032,7913.1478664459900756,
7925.7258869317720382,7938.3088996869719267,7950.8969007533951299,7963.4898861806850618,
7976.0878520262958773,7988.6907943554688245,8001.2987092412085985,8013.9115927642569659,
8026.5294410130691176,8039.1522500837891130,8051.7800160802271421,8064.4127351138349695,
8077.0504033036795590,8089.6930167764221551,8102.3405716662946361,8114.9930641150731390,
8127.6504902720571408,8140.3128462940449026,8152.9801283453098222,8165.6523325975786065,
8178.3294552300048963,8191.0114924291528951,8203.6984403889655368,8216.3902953107462963,
8229.0870534031419083,8241.7887108821068978,8254.4952639708935749,8267.2067089000211126,
8279.9230419072573568,8292.6442592375951790,8305.3703571432306489,8318.1013318835430255,
8330.8371797250656527,8343.5778969414750463,8356.3234798135581514,8369.0739246291977906,
8381.8292276833508367,8394.5893852780209272,8407.3543937222420936,8420.1242493320587528,
8432.8989484304947837,8445.6784873475498898,8458.4628624201577622,8471.2520699921806226,
8484.0461064143837575,8496.8449680444082333,8509.6486512467636203,8522.4571523927952512,
8535.2704678606660309,8548.0885940353437036,8560.9115273085662920,8573.7392640788402787,
8586.5718007514005876,8599.4091337382069469,8612.2512594579147844,8625.0981743358552194,
8637.9498748040205101,8650.8063573010385880,8663.6676182721566875,8676.5336541692249739,
8689.4044614506638027,8702.2800365814600809,8715.1603760331418016,8728.0454762837507587,
8740.9353338178389095,8753.8299451264356321,8766.7293067070331745,8779.6334150635721016,
8792.5422667064158304,8805.4558581523324392,8818.3741859244819352,8831.2972465523907886,
8844.2250365719355614,8857.1575525253265369,8870.0947909610840725,8883.0367484340295050,
8895.9834215052524087,8908.9348067421069572,8921.8909007181846391,8934.8517000132997055,
8947.8172012134709803,8960.7874009109000326,8973.7622957039602625,8986.7418821971732541,
8999.7261570011924050,9012.7151167327883741,9025.7087580148236157,9038.7070774762469227,
9051.7100717520643229,9064.7177374833281647,9077.7307713171152893,9090.7470699065179360,
9103.7687299106146384,9116.7950479944647668,9129.8260208290812443,9142.8616450914232701,
9155.9019174643726728,9168.9468346367157210,9181.9963933031358465,9195.0505901641845412,
9208.1094219262740808,9221.1728853016556968,9234.2409770084050251,9247.3136937704075535,
9260.3910323173386132,9273.4729893846470077,9286.5595617135422799,9299.6507460509747034,
9312.7465391496207303,9325.8469377678684396,9338.9519386698011658,9352.0615386251756718,
9365.1757344094130531,9378.2945228035841865,9391.4179005943842640,9404.5458645741273358,
9417.6784115407263016,9430.8155382976747205,9443.9572416540358972,9457.1035184244265110,
9470.2543654290002451,9483.4097794934295962,9496.5697574488931423,9509.7342961320664472,
9522.9033923850911378,9536.0770430555803614,9549.2552449965824053,9562.4379950665825163,
9575.6252901294792537,9588.8171270545735752,9602.0135027165488282,9615.2144139954634738,
9628.4198577767256211,9641.6298309510930267,9654.8443304146439914,9668.0633530687719031,
9681.2868958201670466,9694.5149555808002333,9707.7475292679191625,9720.9846138040156802,
9734.2262061168275977,9747.4723031393186830,9760.7229018096641084,9773.9779990712322615,
9787.2375918725811061,9800.5016771674327174,9813.7702519146696432,9827.0433130783094384,
9840.3208576275028463,9853.6028825365119701,9866.8893847846993594,9880.1803613565116393,
9893.4758092414685962,9906.7757254341522639,9920.0801069341850962,9933.3889507462245092,
9946.7022538799428730,9960.0200133500220545,9973.3422261761297705,9986.6688893829159497,
9999.9999999999945430,10013.3355550619289716,10026.6755516082212125,10040.0199866833008855,

10053.3688573365088814,10066.7221606220809917,10080.0798935991442704,10093.4420533316970250,
10106.8086368885979027,10120.1796413435495197,10133.5550637750948226,10146.9349012665952614,
10160.3191509062198747,10173.7078097869361955,10187.1008750064956985,10200.4983436674174300,
10213.9002128769843694,10227.3064797472216014,10240.7171413948890404,10254.1321949414668779,
10267.5516375131464883,10280.9754662408140575,10294.4036782600396691,10307.8362707110663905,
10321.2732407387957210,10334.7145854927803157,10348.1603021272039769,10361.6103878008780157,
10375.0648396772212436,10388.5236549242581532,10401.9868307145934523,10415.4543642254120641,
10428.9262526384645753,10442.4024931400490459,10455.8830829210073716,10469.3680191767089127,
10482.8572991070395801,10496.3509199163927406,10509.8488788136528456,10523.3511730121881556,
10536.8577997298380069,10550.3687561889000790,10563.8840396161212993,10577.4036472426851105,
10590.9275763041969185,10604.4558240406786354,10617.9883876965559466,10631.5252645206419402,
10645.0664517661352875,10658.6119466905984154,10672.1617465559556877,10685.7158486284752144,
10699.2742501787615765,10712.8369484817467310,10726.4039408166754583,10739.9752244670908112,
10753.5507967208341142,10767.1306548700267740,10780.7147962110575463,10794.3032180445788981,
10807.8959176754869986,10821.4928924129217194,10835.0941395702484442,10848.6996564650471555,
10862.3094404191087960,10875.9234887584152602,10889.5417988131375751,10903.1643679176195292,
10916.7911934103722160,10930.4222726340558438,10944.0576029354797356,10957.6971816655823204,
10971.3410061794274952,10984.9890738361900731,10998.6413819991485070,11012.2979280356757954,
11025.9587093172231107,11039.6237232193161617,11053.2929671215406415,11066.9664384075385897,
11080.6441344649902021,11094.3260526856083743,11108.0121904651277873,11121.7025452032958128,
11135.3971143038634182,11149.0958951745706145,11162.7988852271428186,11176.5060818772781204,
11190.2174825446345494,11203.9330846528282564,11217.6528856294153229,11231.3768829058863048,
11245.1050739176589559,11258.8374561040618573,11272.5740269083325984,11286.3147837776014057,
11300.0597241628875054,11313.8088455190827517,11327.5621453049516276,11341.3196209831112355,
11355.0812700200331165,11368.8470898860232410,11382.6170780552183714,11396.3912320055787859,
11410.1695492188737262,11423.9520271806759411,11437.7386633803489531,11451.5294553110416018,
11465.3244004696789489,11479.1234963569513639,11492.9267404773036105,11506.7341303389312088,
11520.5456634537640639,11534.3613373374664661,11548.1811495094225393,11562.0050974927235075,
11575.8331788141695142,11589.6653910042532516,11603.5017315971490461,11617.3421981307146780,
11631.1867881464677339,11645.0354991895892454,11658.8883288089109556,11672.7452745569044055,
11686.6063339896754769,11700.4715046669552976,11714.3407841520856891,11728.2141700120209862,
11742.0916598173116654,11755.9732511421007075,11769.8589415641108644,11783.7487286646355642,
11797.6426100285389111,11811.5405832442374958,11825.4426459036967572,11839.3487956024200685,
11853.2590299394450994,11867.1733465173329023,11881.0917429421551788,11895.0142168234924611,
11908.9407657744268363,11922.8713874115255749,11936.8060793548393121,11950.7448392278965912,
11964.6876646576838539,11978.6345532746527169,11992.5855027126999630,12006.5405106091675407,
12020.4995746048280125,12034.4626923438790982,12048.4298614739382174,12062.4010796460315760,
12076.3763445145887090,12090.3556537374315667,12104.3390049757690576,12118.3263958941879537,
12132.3178241606437950,12146.3132874464572524,12160.3127834263032128,12174.3163097782053228,
12188.3238641835250746,12202.3354443269545300,12216.3510478965108632,12230.3706725835309044,
12244.3943160826565872,12258.4219760918313114,12272.4536503122963040,12286.4893364485742495,
12300.5290322084711079,12314.5727353030579252,12328.6204434466781095,12342.6721543569219648,
12356.7278657546376053,12370.7875753639091272,12384.8512809120547900,12398.9189801296233782,
12412.9906707503814687,12427.0663505113061547,12441.1460171525814076,12455.2296684175889823,
12469.3173020529011410,12483.4089158082697395,12497.5045074366298650,12511.6040746940780082,
12525.7076153398775205,12539.8151271364440618,12553.9266078493419627,12568.0420552472751297,
12582.1614671020815877,12596.2848411887262046,12610.4121752852897771,12624.5434671729708498,
12638.6787146360693441,12652.8179154619847395,12666.9610674412087974,12681.1081683673164662,
12695.2592160369622434,12709.4142082498692616,12723.5731428088274697,12737.7360175196808996,
12751.9028301913258474,12766.0735786357035977,12780.2482606677876902,12794.4268741055875580,
12808.6094167701321567,12822.7958864854681451,12836.9862810786526097,12851.1805983797439694,
12865.3788362218019756,12879.5809924408713414,12893.7870648759835603,12907.9970513691441738,
12922.2109497653345898,12936.4287579124957119,12950.6504736615243019,12964.8760948662729788,

12979.1056193835338490,12993.3390450730385055,13007.5763697974543902,13021.8175914223684231,
13036.0627078162851831,13050.3117168506287271,13064.5646163997225813,13078.8214043407915597,
13093.0820785539544886,13107.3466369222169305,13121.6150773314657272,13135.8873976704580855,
13150.1635958308270347,13164.4436697070595983,13178.7276171965022513,13193.0154361993518251,
13207.3071246186482313,13221.6026803602653672,13235.9021013329111156,13250.2053854481182498,
13264.5125306202389766,13278.8235347664340225,13293.1383958066762716,13307.4571116637343948,
13321.7796802631764876,13336.1060995333555184,13350.4363674054093281,13364.7704818132497167,
13379.1084406935624429,13393.4502419857963105,13407.7958836321577110,13422.1453635776069859,
13436.4986797698547889,13450.8558301593457145,13465.2168126992655743,13479.5816253455286642,
13493.9502660567723069,13508.3227327943495766,13522.6990235223292984,13537.0791362074833160,
13551.4630688192864909,13565.8508193299057893,13580.2423857142002817,13594.6377659497102286,
13609.0369580166570813,13623.4399598979271104,13637.8467695790805010,13652.2573850483331626,
13666.6718042965603672,13681.0900253172840166,13695.5120461066690041,13709.9378646635213954,
13724.3674789892775152,13738.8008870880039467,13753.2380869663847989,13767.6790766337271634,
13782.1238541019392869,13796.5724173855451227,13811.0247645016588649,13825.4808934699976817,
13839.9408023128598870,13854.4044890551340359,13868.8719517242825532,13883.3431883503417339,
13897.8181969659144670,13912.2969756061684166,13926.7795223088251078,13941.2658351141599269,
13955.7559120649912074,13970.2497512066820491,13984.7473505871257657,13999.2487082567513426,
14013.7538222685107030,14028.2626906778732518,14042.7753115428276942,14057.2916829238674836,
14071.8118028839944600,14086.3356694887042977,14100.8632808059937815,14115.3946349063407979,
14129.9297298627097916,14144.4685637505481282,14159.0111346477697225,14173.5574406347604963,
14188.1074797943692829,14202.6612502119005512,14217.2187499751180439,14231.7799771742265875,
14246.3449299018793681,14260.9136062531633797,14275.4860043256012432,14290.0621222191457491,
14304.6419580361707631,14319.2255098814657686,14333.8127758622358670,14348.4037540880981396,
14362.9984426710670959,14377.5968397255601303,14392.1989433683884272,14406.8047517187478661,
14421.4142628982226597,14436.0274750307744398,14450.6443862427404383,14465.2649946628280304,
14479.8892984221056395,14494.5172956540045561,14509.1489844943134813,14523.7843630811657931,
14538.4234295550486422,14553.0661820587811235,14567.7126187375270092,14582.3627377387765591,
14597.0165372123483394,14611.6740153103819466,14626.3351701873398270,14640.9999999999927240,
14655.6685029074178601,14670.3406770710025739,14685.0165206544261309,14699.6960318236706371,
14714.3792087469992111,14729.0660495949668984,14743.7565525404079381,14758.4507157584303059,
14773.1485374264175334,14787.8500157240177941,14802.5551488331420842,14817.2639349379605846,
14831.9763722248972044,14846.6924588826241234,14861.4121931020599732,14876.1355730763625616,
14890.8625970009234152,14905.5932630733714177,14920.3275694935582578,14935.0655144635566103,
14949.8070961876619549,14964.5523128723816626,14979.3011627264313574,14994.0536439607349166,
15008.8097547884135565,15023.5694934247876517,15038.3328580873694591,15053.0998469958576607,
15067.8704583721337258,15082.6446904402637301,15097.4225414264838037,15112.2040095592019497,
15126.9890930689944071,15141.7777901885965548,15156.5700991529047315,15171.3660181989671401,
15186.1655455659856671,15200.9686794953013305,15215.7754182304015558,15230.5857600169092620,
15245.3997031025792239,15260.2172457372980716,15275.0383861730733770,15289.8631226640354726,
15304.6914534664319945,15319.5233768386206066,15334.3588910410690005,15349.1979943363458005,
15364.0406849891278398,15378.8869612661765132,15393.7368214363559673,15408.5902637706094538,
15423.4472865419720620,15438.3078880255543481,15453.1720664985423355,15468.0398202401956951,
15482.9111475318404700,15497.7860466568690754,15512.6645159007330221,15527.5465535509392794,
15542.4321578970448172,15557.3213272306602448,15572.2140598454352585,15587.1103540370640985,
15602.0102081032728165,15616.9136203438229131,15631.8205890605058812,15646.7311125571359298,
15661.6451891395463463,15676.5628171155931341,15691.4839947951386421,15706.4087204900624783,
15721.3369925142415013,15736.2688091835607338,15751.2041688159006299,15766.1430697311352560,
15781.0855102511322912,15796.0314886997402937,15810.9810034027977963,15825.9340526881187543,
15840.8906348854889075,15855.8507483266730560,15870.8143913454005087,15885.7815622773614450,
15900.7522594602141908,15915.7264812335652095,15930.7042259389836545,15945.6854919199777214,
15960.6702775220092008,15975.6585810924807447,15990.6504009807304101,16005.6457355380352965,
16020.6445831175988133,16035.6469420745561365,16050.6528107659669331,16065.6621875508062658,

16080.6750707899736881,16095.6914588462732354,16110.7113500844243390,16125.7347428710527311,
16140.7616355746849877,16155.7920265657467098,16170.8259142165607045,16185.8632969013378897,
16200.9041729961827514,16215.9485408790787915,16230.9963989298994420,16246.0477455303862371,
16261.1025790641633648,16276.1608979167212965,16291.2227004754204245,16306.2879851294837863,
16321.3567502699952456,16336.4289942898958543,16351.5047155839820334,16366.5839125489001162,
16381.6665835831408913,16396.7527270870414213,16411.8423414627759485,16426.9354251143631700,
16442.0319764476444107,16457.1319938702981744,16472.2354757918292307,16487.3424206235613383,
16502.4528267786408833,16517.5666926720332413,16532.6840167205155012,16547.8047973426764656,
16562.9290329589020985,16578.0567219913937151,16593.1878628641497926,16608.3224540029623313,
16623.4604938354168553,16638.6019807908960502,16653.7469133005579351,16668.8952897973540530,
16684.0471087160149182,16699.2023684930463787,16714.3610675667259784,16729.5232043771065946,
16744.6887773660091625,16759.8577849770117609,16775.0302256554641644,16790.2060978484660154,
16805.3854000048741000,16820.5681305753023480,16835.7542880121036433,16850.9438707693807373,
16866.1368773029826116,16881.3333060704935633,16896.5331555312295677,16911.7364241462491918,
16926.9431103783317667,16942.1532126919919392,16957.3667295534542063,16972.5836594306820189,
16987.8040007933377638,17003.0277521128155058,17018.2549118622046080,17033.4854785163115594,
17048.7194505516454228,17063.9568264464214735,17079.1976046805466467,17094.4417837356304517,
17109.6893620949667820,17124.9403382435521053,17140.1947106680636352,17155.4524778568520560,
17170.7136382999669877,17185.9781904891278828,17201.2461329177240259,17216.5174640808254480,
17231.7921824751647364,17247.0702865991406725,17262.3517749528255081,17277.6366460379358614,
17292.9248983578545449,17308.2165304176232894,17323.5115407239209162,17338.8099277850888029,
17354.1116901111054176,17369.4168262135935947,17384.7253346058205352,17400.0372138026832545,
17415.3524623207158584,17430.6710786780895432,17445.9930613945871301,17461.3184089916358062,
17476.6471199922743835,17491.9791929211678507,17507.3146263045855449,17522.6534186704229796,
17537.9955685481872933,17553.3410744689863350,17568.6899349655359401,17584.0421485721562931,
17599.3977138247682888,17614.7566292608898948,17630.1188934196252376,17645.4845048416827922,
17660.8534620693535544,17676.2257636465110409,17691.6014081186185649,17706.9803940327183227,
17722.3627199374241172,17737.7483843829359103,17753.1373859210143564,17768.5297231049989932,
17783.9253944897900510,17799.3243986318557290,17814.7267340892249194,17830.1323994214799313,
17845.5413931897674047,17860.9537139567801205,17876.3693602867715526,17891.7883307455304021,
17907.2106239003951487,17922.6362383202540514,17938.0651725755269581,17953.4974252381762199,
17968.9329948816921387,17984.3718800811038818,17999.8140794129722053,18015.2595914553712646,
18030.7084147879140801,18046.1605479917307093,18061.6159896494646091,18077.0747383452835493,
18092.5367926648614230,18108.0021511953927984,18123.4708125255710911,18138.9427752455994778,
18154.4180379471908964,18169.8965992235462181,18185.3784576693797135,18200.8636118808863102,
18216.3520604557670595,18231.8438019932036696,18247.3388350938730582,18262.8371583599364385,
18278.3387703950320429,18293.8436698042896751,18309.3518551943088823,18324.8633251731662313,
18340.3780783504116698,18355.8961133370685275,18371.4174287456226011,18386.9420231900330691,
18402.4698952857179393,18418.0010436495504109,18433.5354668998697889,18449.0731636564742075,
18464.6141325406024407,18480.1583721749557299,18495.7058811836759560,18511.2566581923565536,
18526.8107018280352349,18542.3680107191830757,18557.9285834957154293,18573.4924187889846507,
18589.0595152317728207,18604.6298714583026594,18620.2034861042120610,18635.7803578065795591,
18651.3604852038988611,18666.9438669360861240,18682.5305016444799548,18698.1203879718414100,
18713.7135245623321680,18729.3099100615399948,18744.9095431164569163,18760.5124223754792183,
18776.1185464884183602,18791.7279141064791474,18807.3405238822742831,18822.9563744698098162,
18838.5754645244887797,18854.1977927031111903,18869.8233576638631348,18885.4521580663276836,
18901.0841925714703393,18916.7194598416390363,18932.3579585405641410,18947.9996873333620897,
18963.6446448865208367,18979.2928298679071304,18994.9442409467592370,19010.5988767936869408,
19026.2567360806679062,19041.9178174810476776,19057.5821196695324033,19073.2496413221997500,
19088.9203811164734361,19104.5943377311450604,19120.2715098463559116,19135.9518961436042446,
19151.6354953057380044,19167.3223060169475502,19183.0123269627838454,19198.7055568301220774,
19214.4019943071980379,19230.1016380835790187,19245.8044868501674500,19261.5105392992081761,
19277.2197941242739034,19292.9322500202652009,19308.6479056834214134,19324.3667598113024724,

19340.0888111027925333,19355.8140582580999762,19371.5424999787537672,19387.2741349675998208,
19403.0089619287973619,19418.7469795678225637,19434.4881865914685477,19450.2325817078271939,
19465.9801636263036926,19481.7309310576129064,19497.4848827137611806,19513.2420173080681707,
19529.0023335551413766,19544.7658301708979707,19560.5325058725393319,19576.3023593785655976,
19592.0753894087611116,19607.8515946842089761,19623.6309739272692241,19639.4135258615897328,
19655.1992492121025862,19670.9881427050167986,19686.7802050678255910,19702.5754350292882009,
19718.3738313194480725,19734.1753926696146664,19749.9801178123707359,19765.7880054815686890,
19781.5990544123233121,19797.4132633410081326,19813.2306310052736080,19829.0511561440143851,
19844.8748374973947648,19860.7016738068268751,19876.5316638149852224,19892.3648062657885021,
19908.2010999044068740,19924.0405434772583249,19939.8831357320123061,19955.7288754175788199,
19971.5777612841047812,19987.4297920829849318,20003.2849665668472880,20019.1432834895604174,
20035.0047416062188859,20050.8693396731614484,20066.7370764479419449,20082.6079506893620419,
20098.4819611574275768,20114.3591066133849381,20130.2393858196992369,20146.1227975400543073,
20162.0093405393527064,20177.8990135837157140,20193.7918154404760571,20209.6877448781815474,
20225.5868006665914436,20241.4889815766691754,20257.3942863805968955,20273.3027138517536514,
20289.2142627647153859,20305.1289318952767644,20321.0467200204147957,20336.9676259183179354,
20352.8916483683606202,20368.8187861511141818,20384.7490380483468471,20400.6824028430091857,
20416.6188793192486628,20432.5584662623914483,20448.5011624589533312,20464.4469666966288059,
20480.3958777643019857,20496.3478944520247751,20512.3030155510314216,20528.2612398537348781,
20544.2225661537195265,20560.1869932457411778,20576.1545199257197964,20592.1251449907576898,
20608.0988672391067666,20624.0756854701976408,20640.0555984846178035,20656.0386050841152610,
20672.0247040715948970,20688.0138942511257483,20704.0061744279264531,20720.0015434083725268,
20735.9999999999890861,20752.0015430114544870,20768.0061712525966868,20784.0138835343823303,
20800.0246786689313012,20816.0385554695058090,20832.0555127505067503,20848.0755493274737091,
20864.0986640170849569,20880.1248556371610903,20896.1541230066468415,20912.1864649456256302,
20928.2218802753122873,20944.2603678180494171,20960.3019263973110355,20976.3465548376843799,
20992.3942519648953748,21008.4450166057868046,21024.4988475883183128,21040.5557437415736786,
21056.6157038957535406,21072.6787268821681209,21088.7448115332517773,21104.8139566825375368,
21120.8861611646825622,21136.9614238154390478,21153.0397434716833232,21169.1211189713794738,
21185.2055491536048066,21201.2930328585352981,21217.3835689274528704,21233.4771562027308391,
21249.5737935278411896,21265.6734797473582148,21281.7762137069366872,21297.8819942533336871,
21313.9908202343976882,21330.1026904990540061,21346.2176038973302639,21362.3355592803272884,
21378.4565555002409383,21394.5805914103329997,21410.7076658649639285,21426.8377777195564704,
21442.9709258306284028,21459.1071090557561547,21475.2463262536039110,21491.3885762838945084,
21507.5338580074312631,21523.6821702860870573,21539.8335119827970630,21555.9878819615660177,
21572.1452790874609491,21588.3057022266148124,21604.4691502462155768,21620.6356220145207772,
21636.8051164008320484,21652.9776322755205911,21669.1531685100089817,21685.3317239767638966,
21701.5132975493179401,21717.6978881022441783,21733.8854945111670531,21750.0761156527587445,
21766.2697504047355324,21782.4663976458614343,21798.6660562559336540,21814.8687251158007712,
21831.0744031073445512,21847.2830891134835838,21863.4947820181769202,21879.7094807064167981,
21895.9271840642286406,21912.1478909786674194,21928.3716003378176538,21944.5983110307970492,
21960.8280219477455830,21977.0607319798291428,21993.2964400192431640,22009.5351449591980781,
22025.7768456939302268,22042.0215411186909478,22058.2692301297574886,22074.5199116244111792,
22090.7735845009592595,22107.0302476587166893,22123.2898999980134249,22139.5525404201871424,
22155.8181678275868762,22172.0867811235693807,22188.3583792124954925,22204.6329609997301304,
22220.9105253916386573,22237.1910712956014322,22253.4745976199810684,22269.7611032741478994,
22286.0505871684690646,22302.3430482143121480,22318.6384853240269877,22334.9368974109675037,
22351.2382833894698706,22367.5426421748707071,22383.8499726834816101,22400.1602738326182589,
22416.4735445405676728,22432.7897837266027636,22449.1089903109859733,22465.4311632149583602,
22481.7563013607395987,22498.0844036715279799,22514.4154690714967728,22530.7494964858015010,
22547.0864848405617522,22563.4264330628793687,22579.7693400808238948,22596.1152048234362155,
22612.4640262207212800,22628.8158032036553777,22645.1705347041788627,22661.5282196551997913,
22677.8888569905866461,22694.2524456451683363,22710.6189845547341974,22726.9884726560339914,

22743.3609088867779064,22759.7362921856220055,22776.1146214921864157,22792.4958957470444147,
22808.8801138917187927,22825.2672748686782143,22841.6573776213481324,22858.0504210940962366,
22874.4464042322433670,22890.8453259820526000,22907.2471852907219727,22923.6519811064063106,
22940.0597123781953997,22956.4703780561139865,22972.8839770911290543,22989.3005084351498226,
23005.7199710410168336,23022.1423638624983141,23038.5676858543047274,23054.9959359720778593,
23071.4271131723871804,23087.8612164127298456,23104.2982446515306947,23120.7381968481458898,
23137.1810719628483639,23153.6268689568460104,23170.0755867922634934,23186.5272244321422477,
23202.9817808404477546,23219.4392549820659042,23235.8996458227957191,23252.3629523293566308,
23268.8291734693775652,23285.2983082114078570,23301.7703555248990597,23318.2453143802231352,
23334.7231837486579025,23351.2039626023870369,23367.6876499145037087,23384.1742446590069449,
23400.6637458107979910,23417.1561523456803116,23433.6514632403668656,23450.1496774724619172,
23466.6507940204719489,23483.1548118638056621,23499.6617299827630632,23516.1715473585391010,
23532.6842629732309433,23549.1998758098234248,23565.7183848521854088,23582.2397890850916156,
23598.7640874941971560,23615.2912790660411702,23631.8213627880577405,23648.3543376485649787,
23664.8902026367613871,23681.4289567427331349,23697.9705989574431442,23714.5151282727383659,
23731.0625436813425040,23747.6128441768632911,23764.1660287537779368,23780.7220964074404037,
23797.2810461340850452,23813.8428769308156916,23830.4075877956056502,23846.9751777273013431,
23863.5456457256223075,23880.1189907911502814,23896.6952119253364799,23913.2743081304979569,
23929.8562784098212433,23946.4411217673477950,23963.0288372079885448,23979.6194237375129887,
23996.2128803625491855,24012.8092060905837570,24029.4083999299655261,24046.0104608898982406,
24062.6153879804332973,24079.2231802124915703,24095.8338365978270303,24112.4473561490631255,
24129.0637378796673147,24145.6829808039510681,24162.3050839370807807,24178.9300462950668589,
24195.5578668947673577,24212.1885447538843437,24228.8220788909602561,24245.4584683253888215,
24262.0977120773968636,24278.7398091680479411,24295.3847586192605377,24312.0325594537680445,
24328.6832106951624155,24345.3367113678577880,24361.9930604971086723,24378.6522571089953999,
24395.3143002304423135,24411.9791888891923008,24428.6469221138249850,24445.3174989337458101,
24461.9909183791933174,24478.6671794812245935,24495.3462812717261841,24512.0282227834068181,
24528.7130030498010456,24545.4006211052655999,24562.0910759849757596,24578.7843667249253485,
24595.4804923619267356,24612.1794519336144731,24628.8812444784380205,24645.5858690356544685,
24662.2933246453430911,24679.0036103483980696,24695.7167251865139406,24712.4326682022110617,
24729.1514384388065082,24745.8730349404359004,24762.5974567520315759,24779.3247029193444178,
24796.0547724889256642,24812.7876645081232709,24829.5233780251001008,24846.2619120888193720,
24863.0032657490337442,24879.7474380563071463,24896.4944280620038626,24913.2442348182776186,
24929.9968573780824954,24946.7522947951656533,24963.5105461240782461,24980.2716104201572307,
24997.0354867395253677,25013.8021741391130490,25030.5716716766291938,25047.3439784105721628,
25064.1190934002370341,25080.8970157056974131,25097.6777443878127087,25114.4612785082390474,
25131.2476171294001688,25148.0367593145165301,25164.8287041275834781,25181.6234506333748868,
25198.4209978974504338,25215.2213449861446861,25232.0244909665743762,25248.8304349066274881,
25265.6391758749741712,25282.4507129410485504,25299.2650451750705543,25316.0821716480240866,
25332.9020914316679409,25349.7248035985321621,25366.5503072219144087,25383.3786013758835907,
25400.2096851352689555,25417.0435575756782782,25433.8802177734723955,25450.7196648057833954,
25467.5618977505073417,25484.4069156862969976,25501.2547176925727399,25518.1053028495116450,
25534.9586702380511269,25551.8148189398889372,25568.6737480374795268,25585.5354566140267707,
25602.3999437535021571,25619.2672085406193219,25636.1372500608522387,25653.0100674004315806,
25669.8856596463265305,25686.7640258862702467,25703.6451652087343973,25720.5290767029473500,
25737.4157594588759821,25754.3052125672438706,25771.1974351195167401,25788.0924262078988249,
25804.9901849253437831,25821.8907103655474202,25838.7940016229440516,25855.7000577927174163,
25872.6088779707752110,25889.5204612537781941,25906.4348067391183577,25923.3519135249225656,
25940.2717807100634673,25957.1944073941376701,25974.1197926774766529,25991.0479356611540425,
26007.9788354469637852,26024.9124911374419753,26041.8489018358413887,26058.7880666461605870,
26075.7299846731075377,26092.6746550221323560,26109.6220767994091148,26126.5722491118285689,
26143.5251710670163448,26160.4808417733147508,26177.4392603397900530,26194.4004258762288373,
26211.3643374931452854,26228.3309943017666228,26245.3003954140403948,26262.2725399426271906,

26279.2474270009188331,26296.2250557030056370,26313.2054251637018751,26330.1885344985385018,
26347.1743828237558773,26364.1629692563074059,26381.1542929138522595,26398.1483529147735680,
26415.1451483781493152,26432.1446784237778047,26449.1469421721558319,26466.1519387444932363,
26483.1596672627019871,26500.1701268494034593,26517.1833166279211582,26534.1992357222770806,
26551.2178832571989915,26568.2392583581204235,26585.2633601511734014,26602.2901877631811658,
26619.3197403216763632,26636.3520169548828562,26653.3870167917266372,26670.4247389618249144,
26687.4651825954933884,26704.5083468237389752,26721.5542307782670832,26738.6028335914670606,
26755.6541543964303855,26772.7081923269288382,26789.7649465174326906,26806.8244161030961550,
26823.8866002197610214,26840.9514980039602960,26858.0191085929145629,26875.0894311245174322,
26892.1624647373646440,26909.2382085707213264,26926.3166617645474616,26943.3978234594724199,
26960.4816927968131495,26977.5682689185741765,26994.6575509674221394,27011.7495380867221684,
27028.8442294204978680,27045.9416241134640586,27063.0417213110049488,27080.1445201591814111,
27097.2500198047273443,27114.3582193950496730,27131.4691180782356241,27148.5827150030308985,
27165.6990093188578612,27182.8180001758155413,27199.9396867246650800,27217.0640681168370065,
27234.1911435044276004,27251.3209120402025292,27268.4533728775932104,27285.5885251706931740,
27302.7263680742689758,27319.8669007437347318,27337.0101223351812223,27354.1560320053577016,
27371.3046289116682601,27388.4559122121827386,27405.6098810656258138,27422.7665346313879127,
27439.9258720695070224,27457.0878925406832423,27474.2525952062751458,27491.4199792282925046,
27508.5900437693999265,27525.7627879929168557,27542.9382110628102964,27560.1163121437057271,
27577.2970904008761863,27594.4805450002422731,27611.6666751083830604,27628.8554798925179057,
27646.0469585205137264,27663.2411101608886383,27680.4379339828010416,27697.6374291560678103,
27714.8395948511315510,27732.0444302390897064,27749.2519344916872797,27766.4621067812986439,
27783.6749462809493707,27800.8904521643016778,27818.1086236056544294,27835.3294597799540497,
27852.5529598627799714,27869.7791230303446355,27887.0079484595044050,27904.2394353277450136,
27921.4735828131961171,27938.7103900946131034,27955.9498563513916451,27973.1919807635495090,
27990.4367625117447460,28007.6842007772720535,28024.9342947420409473,28042.1870435886012274,
28059.4424465001284261,28076.7005026604274462,28093.9612112539289228,28111.2245714656928612,
28128.4905824814013613,28145.7592434873622551,28163.0305536705091072,28180.3045122183939384,
28197.5811183191981399,28214.8603711617251975,28232.1422699353897769,28249.4268138302395528,
28266.7140020369297417,28284.0038337467449310,28301.2963081515845261,28318.5914244439591130,
28335.8891818170013721,28353.1895794644624402,28370.4926165807046345,28387.7982923607014527,
28405.1066060000484867,28422.4175566949452332,28439.7311436422060069,28457.0473660392635793,
28474.3662230841473502,28491.6877139755124517,28509.0118379126106447,28526.3385940953048703,
28543.6679817240692500,28560.9999999999854481,28578.3346481247317570,28595.6719253006049257,
28613.0118307304983318,28630.3543636179092573,28647.6995231669425266,28665.0473085822995927,
28682.3977190692894510,28699.7507538338177255,28717.1064120823903068,28734.4646930221206276,
28751.8255958607114735,28769.1891198064622586,28786.5552640682799392,28803.9240278556644625,
28821.2954103787014901,28838.6694108480878640,28856.0460284751025029,28873.4252624716282298,
28890.8071120501299447,28908.1915764236728137,28925.5786548059149936,28942.9683464110967179,
28960.3606504540548485,28977.7555661502155999,28995.1530927155909012,29012.5532293667856720,
29029.9559753209869086,29047.3613297959745978,29064.7692920101071650,29082.1798611823360261,
29099.5930365321873978,29117.0088172797804873,29134.4272026458129403,29151.8481918515717553,
29169.2717841189114552,29186.6979786702831916,29204.1267747287056409,29221.5581715177904698,
29238.9921682617168699,29256.4287641852497472,29273.8679585137251706,29291.3097504730576475,
29308.7541392897473997,29326.2011241908548982,29343.6507044040299661,29361.1028791574826755,
29378.5576476800124510,29396.0150092009753280,29413.4749629503094184,29430.9375081585239968,
29448.4026440566922247,29465.8703698764693399,29483.3406848500708293,29500.8135882102797041,
29518.2890791904537764,29535.7671570245111070,29553.2478209469445574,29570.7310701928072376,
29588.2169039977234206,29605.7053215978703520,29623.1963222300000780,29640.6899051314285316,
29658.1860695400282566,29675.6848146942356834,29693.1861398330474913,29710.6900441960278840,
29728.1965270232976764,29745.7055875555270177,29763.2172250339644961,29780.7314387003971206,
29798.2482277971830626,29815.7675915672298288,29833.2895292540051742,29850.8140401015298266,
29868.3411233543811250,29885.8707782576930185,29903.4030040571451536,29920.9377999989737873,

29938.4751653299754253,29956.0150992974813562,29973.5576011493940314,29991.1026701341470471,
30008.6503055007415242,30026.2005064987097285,30043.7532723781441746,30061.3086023896830739,
30078.8664957845066965,30096.4269518143519235,30113.9899697314940568,30131.5555487887504569,
30149.1236882394914574,30166.6943873376294505,30184.2676453376079735,30201.8434614944344503,
30219.4218350636401738,30237.0027653013094096,30254.5862514640575682,30272.1722928090457572,
30289.7608885939771426,30307.3520380770896736,30324.9457405171597202,30342.5419951735020732,
30360.1408013059663062,30377.7421581749440520,30395.3460650413580879,30412.9525211666659743,
30430.5615258128636924,30448.1730782424747304,30465.7871777185609972,30483.4038235047191847,
30501.0230148650698538,30518.6447510642719863,30536.2690313675157086,30553.8958550405150163,
30571.5252213495186879,30589.1571295613066468,30606.7915789431754092,30624.4285687629635504,
30642.0680982890298765,30659.7101667902607005,30677.3547735360698425,30695.0019177963913535,
30712.6515988416867913,30730.3038159429452207,30747.9585683716759377,30765.6158553999121068,
30783.2756763002107618,30800.9380303456455295,30818.6029168098139053,30836.2703349668372539,
30853.9402840913535329,30871.6127634585209307,30889.2877723440105910,30906.9653100240248023,
30924.6453757752715319,30942.3279688749826164,30960.0130886009028472,30977.7007342312936089,
30995.3909050449292408,31013.0836003211006755,31030.7788193396190763,31048.4765613807976479,
31066.1768257254698256,31083.8796116549783619,31101.5849184511789645,31119.2927453964402957,
31137.0030917736366973,31154.7159568661554658,31172.4313399578932149,31190.1492403332595131,
31207.8696572771623323,31225.5925900750225992,31243.3180380127705575,31261.0460003768384922,
31278.7764764541716431,31296.5094655322100152,31314.2449668989102065,31331.9829798427199421,
31349.7235036525999021,31367.4665376180128078,31385.2120810289234214,31402.9601331757949083,
31420.7106933495961130,31438.4637608417906449,31456.2193349443514307,31473.9774149497425242,
31491.7380001509336580,31509.5010898413893301,31527.2666833150688035,31545.0347798664370202,
31562.8053787904500496,31580.5784793825623638,31598.3540809387195623,31616.1321827553692856,
31633.9127841294503014,31651.6958843583961425,31669.4814827401314687,31687.2695785730757052,
31705.0601711561430420,31722.8532597887351585,31740.6488437707484991,31758.4469224025669973,
31776.2474949850657140,31794.0505608196144749,31811.8561192080596811,31829.6641694527534128,
31847.4747108565206872,31865.2877427226849250,31883.1032643550461216,31900.9212750578990381,
31918.7417741360186483,31936.5647608946710534,31954.3902346395989298,31972.2181946770397190,
31990.0486403137037996,32007.8815708567926777,32025.7169856139844342,32043.5548838934446394,
32061.3952650038154388,32079.2381282542228291,32097.0834729542693822,32114.9312984140487970,
32132.7816039441167959,32150.6343888555238664,32168.4896524597897951,32186.3473940689145820,
32204.2076129953711643,32222.0703085521199682,32239.9354800525834435,32257.8031268106715288,
32275.6732481407671003,32293.5458433577186952,32311.4209117768623400,32329.2984527139960846,
32347.1784654853945540,32365.0609494078125863,32382.9459037984634051,32400.8333279750404472,
32418.7232212557064486,32436.6155829590934445,32454.5104124043064076,32472.4077089109159715,
32490.3074717989657074,32508.2097003889612097,32526.1143940018773719,32544.0215519591656630,
32561.9311735827322991,32579.8432581949564337,32597.7578051186792436,32615.6748136772112048,
32633.5942831943284546,32651.5162129942582396,32669.4406024017116579,32687.3674507418472786,
32705.2967573402966082,32723.2285215231459006,32741.1627426169434329,32759.0994199487031437,
32777.0385528459009947,32794.9801406364640570,32812.9241826487923390,32830.8706782117296825,
32848.8196266545928665,32866.7710273071497795,32884.7248794996194192,32902.6811825626864447,
32920.6399358274939004,32938.6011386256432161,32956.5647902891796548,32974.5308901506068651,
32992.4994375428941566,33010.4704317994473968,33028.4438722541453899,33046.4197582413107739,
33064.3980890957100200,33082.3788641525825369,33100.3620827475897386,33118.3477442168805283,
33136.3358478970258147,33154.3263931250621681,33172.3193792384699918,33190.3148055751735228,
33208.3126714735553833,33226.3129762724420289,33244.3157193111110246,33262.3208999292837689,
33280.3285174671254936,33298.3385712652598158,33316.3510606647469103,33334.3659850070907851,
33352.3833436342392815,33370.4031358885913505,33388.4253611129897763,33406.4500186507211765,
33424.4771078455014504,33442.5066280415121582,33460.5385785833495902,33478.5729588160829735,
33496.6097680851889891,33514.6490057366172550,33532.6906711167393951,33550.7347635723563144,
33568.7812824507345795,33586.8302270995627623,33604.8815968669732683,33622.9353911015277845,
33640.9916091522391071,33659.0502503685420379,33677.1113141003224882,33695.1747996978810988,

33713.2407065119841718,33731.3090338938054629,33749.3797811949698371,33767.4529477675314411,
33785.5285329639737029,33803.6065361372093321,33821.6869566406021477,33839.7697938279379741,
33857.8550470534246415,33875.9427156717065373,33894.0327990378718823,33912.1252965074309031,
33930.2202074363158317,33948.3175311808881816,33966.417267097960575 5,33984.5194145447458141,
34002.6239728789005312,34020.7309414585106424,34038.8403196420767927,34056.9521067885361845,
34075.0663022572553018,34093.1829054080153583,34111.3019156010268489,34129.4233321969295503,
34147.5471545567852445,34165.6733820420777192,34183.8020140147200436,34201.9330498370327405,
34220.0664888717801659,34238.2023304821414058,34256.3405740317029995,34274.4812188844953198,
34292.6242644049489172,34310.7697099579381756,34328.9175549087303807,34347.0677986230293754,
34365.2204404669537325,34383.3754798070513061,34401.5329160102628521,34419.6927484439729596,
34437.8549764759663958,34456.0195994744499330,34474.1866168080596253,34492.3560278458171524,
34510.5278319571880274,34528.7020285120524932,34546.8786168806764181,34565.0575964337695041,
34583.2389665424489067,34601.4227265782319591,34619.6088759130652761,34637.7974139192956500,
34655.9883399696918787,34674.1816534374229377,34692.3773536960798083,34710.5754401196682011,
34728.7759120825794525,34746.9787689596487326,34765.1840101260822848,34783.3916349575374625,
34801.6016428300499683,34819.8140331200629589,34838.0288052044561482,34856.2459584604803240,
34874.4654922658228315,34892.6874059985566419,34910.9116990371767315,34929.1383707605637028,
34947.3674205480274395,34965.5988477792707272,34983.8326518343892531,35002.0688320939079858,
35020.3073879387375200,35038.5483187501886277,35056.7916239099795348,35075.0373028002504725,
35093.2853548035127460,35111.5357793026851141,35129.7885756811156170,35148.0437433225160930,
35166.3012816110131098,35184.5611899311406887,35202.8234676678257529,35221.0881142063881271,
35239.3551289325550897,35257.6245112324468209,35275.8962604925836786,35294.1703760998861981,
35312.4468574416678166,35330.7257039056275971,35349.0069148798866081,35367.2904897529442678,
35385.5764279136856203,35403.8647287514177151,35422.1553916558113997,35440.4484160169668030,
35458.7438012253405759,35477.0415466718040989,35495.3416517476216541,35513.6441158444358734,
35531.9489383543041185,35550.2561186696548248,35568.5656561833093292,35586.8775502884964226,
35605.1918003788159695,35623.5084058482680121,35641.8273660912382184,35660.1486805025051581,
35678.4723484772330266,35696.7983694109789212,35715.1267426996782888,35733.4574677396594780,
35751.7905439276437392,35770.1259706607379485,35788.4637473364200559,35806.8038733525681891,
35825.1463481074533775,35843.4911709997104481,35861.8383414283671300,35880.1878587928513298,
35898.5397224929547519,35916.8939319288620027,35935.2504865011287620,35953.6093856107181637,
35971.9706286589571391,35990.3342150475582457,36008.7001441786123905,36027.0684154545961064,
36045.4390282783715520,36063.8119820531646837,36082.1872761826089118,36100.5649100706941681,
36118.9448831217887346,36137.3271947406537947,36155.7118443324288819,36174.0988313026173273,
36192.4881550571153639,36210.8798150021902984,36229.2738105444732355,36247.6701410910027334,
36266.0688060491665965,36284.4698048267382546,36302.8731368318622117,36321.2788014730685973,
36339.6867981592513388,36358.0971262996827136,36376.5097853040133487,36394.9247745822576690,
36413.3420935448157252,36431.7617416024440899,36450.1837181662922376,36468.6080226478588884,
36487.0346544590283884,36505.4636130120634334,36523.8948977195832413,36542.3285079945781035,
36560.7644432504093857,36579.2027029008313548,36597.6432863599256962,36616.0861930421815487,
36634.5314223624372971,36652.9789737358951243,36671.4288465781428386,36689.8810403051247704,
36708.3355543331490480,36726.7923880789021496,36745.2515409594270750,36763.7130123921378981,
36782.1768017948124907,36800.6429085855925223,36819.1113321829907363,36837.5820720058691222,
36856.0551274734825711,36874.5304980054206681,36893.0081830216513481,36911.4881819425063441,
36929.9704941886739107,36948.4551191812061006,36966.9420563415187644,36985.4313050913915504,
37003.9228648529606289,37022.4167350487332442,37040.9129151015586103,37059.4114044346570154,
37077.9122024716198212,37096.4153086363876355,37114.9207223532430362,37133.4284430468615028,
37151.9384701422532089,37170.4508030647848500,37188.9654412402087473,37207.4823840945973643,
37226.0016310544015141,37244.5231815464285319,37263.0470349978422746,37281.5731908361485694,
37300.1016484892243170,37318.6324073852956644,37337.1654669529452804,37355.7008266211123555,
37374.2384858190853265,37392.7784439765091520,37411.3207005233853124,37429.8652548900572583,
37448.4121065072322381,37466.9612548059667461,37485.5126992176810745,37504.0664391741156578,
37522.6224741074038320,37541.1808034499917994,37559.7414266346968361,37578.3043430946927401,

37596.8695522634880035,37615.4370535749403643,37634.0068464632713585,37652.5789303630444920,
37671.1533047091652406,37689.7299689368956024,37708.3089224818468210,37726.8901647799648345,
37745.4736952675593784,37764.0595133812748827,37782.6476185581122991,37801.2380102354145492,
37819.8306878508592490,37838.4256508424950880,37857.0228986486908980,37875.6224307081720326,
37894.2242464600130916,37912.8283453436160926,37931.4347267987468513,37950.0433902655058773,
37968.6543351843283745,37987.2675609959987924,38005.8830671416653786,38024.5008530627746950,
38043.1209182011589292,38061.7432619989631348,38080.3678838986816118,38098.9947833431579056,
38117.6239597755629802,38136.2554126394170453,38154.8891413785750046,38173.5251454372337321,
38192.1634242599393474,38210.8039772915508365,38229.4468039772837074,38248.0919037627027137,
38266.7392760936854756,38285.3889204164661351,38304.0408361776062520,38322.6950228240020806,
38341.3514798028991208,38360.0102065618630149,38378.6712025488159270,38397.3344672119928873,
38415.9999999999781721,38434.6678003616834758,38453.3378677463697386,38472.0102016036107671,
38490.6848013833368896,38509.3616665357840247,38528.0407965115518891,38546.7221907615530654,
38565.4058487370348303,38584.0917698895937065,38602.7799536711318069,38621.4703995339077665,
38640.1631069304858102,38658.8580753137939610,38677.5553041370585561,38696.2547928538624546,
38714.9565409180941060,38733.6605477839912055,38752.3668129061115906,38771.0753357393477927,
38789.7861157389197615,38808.4991523603675887,38827.2144450595733360,38845.9319932927392074,
38864.6517965163875488,38883.3738541873826762,38902.0981657629163237,38920.8247307004858158,
38939.5535484579377226,38958.2846184934314806,38977.0179402654612204,38995.7535132328339387,
39014.4913368546986021,39033.2314105905170436,39051.9737339000785141,39070.7183062434851308,
39089.4651270811882569,39108.2141958739448455,39126.9655120828319923,39145.7190751692614867,
39164.4748845949652605,39183.2329398219881114,39201.9932403127095313,39220.7557855298146023,
39239.5205749363303767,39258.2876079955894966,39277.0568841712447465,39295.8284029272908811,
39314.6021637280064169,39333.3781660380191170,39352.1564093222696101,39370.9368930460041156,
39389.7196166748108226,39408.5045796745835105,39427.2917815115215490,39446.0812216521735536,
39464.8728995633719023,39483.6668147122909431,39502.4629665664106142,39521.2613545935382717,
39540.0619782617795863,39558.8648370395676466,39577.6699303956556832,39596.4772577991097933,
39615.2868187193016638,39634.0986126259231241,39652.9126389889934217,39671.7288972788228421,
39690.5473869660636410,39709.3681075216518366,39728.1910584168581408,39747.0162391232588561,
39765.8436491127504269,39784.6732878575276118,39803.5051548301053117,39822.3392495033185696,
39841.1755713502934668,39860.0141198444980546,39878.8548944596768706,39897.6978946699091466,
39916.5431199495797046,39935.3905697733716806,39954.2402436163029051,39973.0921409536749707,
39991.9462612611168879,40010.8026040145487059,40029.6611686902251677,40048.5219547646775027,
40067.3849617147789104,40086.2501890176790766,40105.1176361508551054,40123.9873025920896907,
40142.8591878194711171,40161.7332913113787072,40180.6096125465264777,40199.4881510039122077,
40218.3689061628538184,40237.2518775029602693,40256.1370645041533862,40275.0244666466678609,
40293.9140834110294236,40312.8059142780839466,40331.6999587289610645,40350.5962162451032782,
40369.4946863082732307,40388.3953684005100513,40407.2982620041730115,40426.2033666019196971,
40445.1106816767060081,40464.0202067117934348,40482.9319411907563335,40501.8458845974455471,
40520.7620364160320605,40539.6803961309851729,40558.6009632270724978,40577.5237371893672389,
40596.4487175032336381,40615.3759036543415277,40634.3052951286590542,40653.2368914124526782,
40672.1706919922944508,40691.1066963550474611,40710.0449039878731128,40728.9853143782383995,
40747.9279270139013533,40766.8727413829183206,40785.8197569736512378,40804.7689732747458038,
40823.7203897751605837,40842.6740059641306289,40861.6298213312111329,40880.5878353662264999,
40899.5480475593285519,40918.5104574009310454,40937.4750643817606033,40956.4418679928494385,
40975.4108677254989743,40994.3820630713235005,41013.3554535222356208,41032.3310385704171495,
41051.3088177083627670,41070.2887904288581922,41089.2709562249874580,41108.2553145901110838,
41127.2418650178879034,41146.2306070022896165,41165.2215400375425816,41184.2146636181932990,
41203.2099772390793078,41222.2074803953073570,41241.2071725822970620,41260.2090532957518008,
41279.2131220316587132,41298.2193782863032538,41317.2278215562546393,41336.2384513383804006,
41355.2512671298318310,41374.2662684280367102,41393.2834547307429602,41412.3028255359531613,
41431.3243803419827600,41450.3481186474164133,41469.3740399511443684,41488.4021437523260829,
41507.4324295504266047,41526.4648968451874680,41545.4995451366266934,41564.5363739250751678,

41583.5753827111257124,41602.6165709956621868,41621.6599382798740407,41640.7054840652053826,
41659.7532078534059110,41678.8031091464945348,41697.8551874468030292,41716.9094422569105518,
41735.9658730797091266,41755.0244794183599879,41774.0852607763154083,41793.1482166572968708,
41812.2133465653314488,41831.2806500047081499,41850.3501264800142962,41869.4217754961064202,
41888.4955965581320925,41907.5715891715153703,41926.6497528419567971,41945.7300870754625066,
41964.8125913782860152,41983.8972652569791535,42002.9841082183775143,42022.0731197695931769,
42041.1642994180074311,42060.2576466713071568,42079.3531610374193406,42098.4508420245911111,
42117.5506891413242556,42136.6527018964043236,42155.7568797988933511,42174.8632223581371363,
42193.9717290837579640,42213.0823994856546051,42232.1952330740023172,42251.3102293592455680,
42270.4273878521271399,42289.5467080636444734,42308.6681895050787716,42327.7918316879949998,
42346.9176341242273338,42366.0455963258864358,42385.1757178053521784,42404.3079980752954725,
42423.4424366486418876,42442.5790330386080313,42461.7177867586724460,42480.8586973225974361,
42500.0017642444217927,42519.1469870384462411,42538.2943652192479931,42557.4438983016880229,
42576.5955858008819632,42595.7494272322364850,42614.9054221114201937,42634.0635699543709052,
42653.2238702773174737,42672.3863225967288599,42691.5509264293796150,42710.7176812922916724,
42729.8865867027634522,42749.0576421783625847,42768.2308472369404626,42787.4062013966104132,
42806.5837041757404222,42825.7633550929895137,42844.9451536672859220,42864.1290994178052642,
42883.3151918640141957,42902.5034305256485823,42921.6938149226916721,42940.8863445754104760,
42960.0810190043484909,42979.2778377302965964,42998.4768002743221587,43017.6779061577690300,
43036.8811549022284453,43056.0865460295826779,43075.2940790619613836,43094.5037535217634286,
43113.7155689316641656,43132.9295248146008817,43152.1456206937655224,43171.3638560926192440,
43190.5842305349069647,43209.8067435446209856,43229.0313946460155421,43248.2581833636213560,
43267.4871092222238076,43286.7181717468847637,43305.9513704629061976,43325.1867048958811210,
43344.4241745716499281,43363.6637790163222235,43382.9055177562622703,43402.1493903181035421,
43421.3953962287487229,43440.6435350153478794,43459.8938062053202884,43479.1462093263471616,
43498.4007439063789207,43517.6574094736060943,43536.9162055564956972,43556.1771316837839549,
43575.4401873844399233,43594.7053721877236967,43613.9726856231354759,43633.2421272204373963,
43652.5136965096680797,43671.7873930210989784,43691.0632162852707552,43710.3411658330005594,
43729.6212411953456467,43748.9034419036252075,43768.1877674894130905,43787.4742174845523550,
43806.7627914211261668,43826.0534888315014541,43845.3463092482779757,43864.6412522043247009,
43883.9383172327652574,43903.2375038669779315,43922.5388116405956680,43941.8422400875133462,
43961.1477887418805039,43980.4554571381013375,43999.7652448108347016,44019.0771512950013857,
44038.3911761257550097,44057.7073188385402318,44077.0255789690199890,44096.3459560531409807,
44115.6684496270827367,44134.9930592272867216,44154.3197843904563342,44173.6486246535350801,
44192.9795795537283993,44212.3126486284891143,44231.6478314155319822,44250.9851274528045906,
44270.3245362785382895,44289.6660574311827077,44309.0096904494639602,44328.3554348723555449,
44347.7032902390637901,44367.0532560890787863,44386.4053319621089031,44405.7595173981389962,
44425.1158119373867521,44444.4742151203317917,44463.8347264876938425,44483.1973455804618425,
44502.5620719398430083,44521.9289051073283190,44541.2978446246343083,44560.6688900337321684,
44580.0420408768477500,44599.4172966964542866,44618.7946570352723938,44638.1741214362555183,
44657.5556894426408689,44676.9393605978693813,44696.3251344456730294,44715.7130105300020659,
44735.1029883950541262,44754.4950675852960558,44773.8892476454202551,44793.2855281203737832,
44812.6839085553438053,44832.0843884957794216,44851.4869674873552867,44870.8916450760152657,
44890.2984208079142263,44909.7072942294907989,44929.1182648874091683,44948.5313323285663500,
44967.9464961001358461,44987.3637557495021611,45006.7831108243262861,45026.2045608724729391,
45045.6281054420978762,45065.0537440815605805,45084.4814763394897454,45103.9113017647396191,
45123.3432199064263841,45142.7772303138845018,45162.2133325367103680,45181.6515261247332091,
45201.0918106280369102,45220.5341855969236349,45239.9786505819647573,45259.4252051339572063,
45278.8738488039380172,45298.3245811431916081,45317.7774017032352276,45337.2323100358480588,
45356.6893056930202874,45376.1483882269967580,45395.6095571902696975,45415.0728121355568874,
45434.5381526158234919,45454.0055781842820579,45473.4750883943561348,45492.9466827997457585,
45512.4203609543619677,45531.8961224123631837,45551.3739667281552101,45570.8538934563621297,
45590.3359021518699592,45609.8199923697757185,45629.3061636654383619,45648.7944155944423983,

45668.2847477126124431,45687.7771595760059427,45707.2716507409204496,45726.7682207638936234,
45746.2668692016959540,45765.7675956113234861,45785.2703995500341989,45804.7752805752970744,
45824.2822382448212011,45843.7912721165703260,45863.3023817487191991,45882.8155666996826767,
45902.3308265281302738,45921.8481607929352322,45941.3675690532254521,45960.8890508683543885,
45980.4126057979301549,45999.9382334017573157,46019.4659332399023697,46038.9957048726573703,
46058.5275478605472017,46078.0614617643295787,46097.5974461450023227,46117.1355005637742579,
46136.6756245821088669,46156.2178177617024630,46175.7620796644623624,46195.3084098525432637,
46214.8568078883326962,46234.4072733344437438,46253.9598057537150453,46273.5144047092398978,
46293.0710697643153253,46312.6298004824784584,46332.1905964274992584,46351.7534571633805172,
46371.3183822543505812,46390.8853712648633518,46410.4544237596201128,46430.0255393035258749,
46449.5987174617330311,46469.1739577996195294,46488.7512598827815964,46508.3306232770701172,
46527.9120475485324278,46547.4955322634705226,46567.0810769883973990,46586.6686812900588848,
46606.2583447354336386,46625.8500668917185976,46645.4438473263508058,46665.0396856069855858,
46684.6375813014965388,46704.2375339780046488,46723.8395432048419025,46743.4436085505731171,
46763.0497295839886647,46782.6579058741044719,46802.2681369901620201,46821.8804225016283453,
46841.4947619781960384,46861.1111549897759687,46880.7296011065263883,46900.3500998987947241,
46919.9726509371903376,46939.5972537925263168,46959.2239080358413048,46978.8526132383994991,
46998.4833689716906520,47018.1161748074300704,47037.7510303175513400,47057.3879350742136012,
47077.0268886498088250,47096.6678906169399852,47116.3109405484283343,47135.9560380173279555,
47155.6031825969184865,47175.2523738606978441,47194.9036113823749474,47214.5568947358988225,
47234.2122234954222222,47253.8695972353380057,47273.5290155302500352,47293.1904779549804516,
47312.8539840845769504,47332.5195334943055059,47352.1871257596576470,47371.8567604563431814,
47391.5284371602974716,47411.2021554476523306,47430.8779148947869544,47450.5557150782988174,
47470.2355555749818450,47489.9174359618627932,47509.6013558162012487,47529.2873147154532489,
47548.9753122373076621,47568.6653479596716352,47588.3574214606560417,47608.0515323186045862,
47627.7476801120719756,47647.4458644198457478,47667.1460848209098913,47686.8483408944739494,
47706.5526322199730203,47726.2589583770459285,47745.9673189455570537,47765.6777135055817780,
47785.3901416374283144,47805.1046029216013267,47824.8210969388237572,47844.5396232700441033,
47864.2601814964291407,47883.9827711993493722,47903.7073919603935792,47923.4340433613760979,
47943.1627249843077152,47962.8934364114393247,47982.6261772252182709,48002.3609470083101769,
48022.0977453435989446,48041.8365718141722027,48061.5774260033504106,48081.3203074946504785,
48101.0652158718148712,48120.8121507187897805,48140.5611116197396768,48160.3120981590473093,
48180.0651099213064299,48199.8201464913072414,48219.5772074540727772,48239.3362923948443495,
48259.0974008990451694,48278.8605325523385545,48298.6256869405915495,48318.3928636498749256,
48338.1620622664850089,48357.9332823769145762,48377.7065235678892350,48397.4817854263164918,
48417.2590675393439597,48437.0383694943084265,48456.8196908787649591,48476.6030312804869027,
48496.3883902874513296,48516.1757674878390389,48535.9651624700418324,48555.7565748226843425,
48575.5500041345658246,48595.3454499947183649,48615.1429119923777762,48634.9423897169908741,
48654.7438827582009253,48674.5473907058767509,48694.3529131500908989,48714.1604496811123681,
48733.9699998894357122,48753.7815633657592116,48773.5951397009848733,48793.4107284862111555,
48813.2283293127693469,48833.0479417721871869,48852.8695654561888659,48872.6931999567168532,
48892.5188448659246205,48912.3464997761548148,48932.1761642799756373,48952.0078379701517406,
48971.8415204396587797,48991.6772112816761364,49011.5149100895869196,49031.3546164569779648,
49051.1963299776543863,49071.0400502456104732,49090.8857768550587934,49110.7335094004083658,
49130.5832474762792117,49150.4349906774878036,49170.2887385990616167,49190.1444908362391288,
49210.0022469844407169,49229.8620066393268644,49249.7237693967181258,49269.5875348526751623,
49289.4533026034478098,49309.3210722454823554,49329.1908433754506405,49349.0626155901918537,
49368.9363884867852903,49388.8121616624921444,49408.6899347147846129,49428.5697072413240676,
49448.4514788399901590,49468.3352491088662646,49488.2210176462103846,49508.1087840505206259,
49527.9985479204697185,49547.8903088549341192,49567.7840664530085633,49587.6798203139769612,
49607.5775700373123982,49627.4773152227207902,49647.3790554700754001,49667.2827903794604936,
49687.1885195511786151,49707.0962425857069320,49727.0059590837408905,49746.9176686461651116,
49766.8313708740679431,49786.7470653687341837,49806.6647517316596350,49826.5844295645147213,

49846.5060984692026977,49866.4297580477941665,49886.3554079025780084,49906.2830476360322791,
49926.2126768508460373,49946.1442951498829643,49966.0779021362250205,49986.0134974131506169,
50005.9510805841346155,50025.8906512528337771,50045.8322090231231414,50065.7757534990669228,
50085.7212842849330627,50105.6688009851641254,50125.6183032044282299,50145.5697905475753942,
50165.5232626196520869,50185.4787190259012277,50205.4361593717694632,50225.3955832628926146,
50245.3569903051029542,50265.3203801044292049,50285.2857522671038168,50305.2531063995338627,
50325.2224421083374182,50345.1937590003290097,50365.1670566825196147,50385.1423347621021094,
50405.1195928464730969,50425.0988305432183552,50445.0800474601273891,50465.0632432051788783,
50485.0484173865406774,50505.0355696125770919,50525.0246994918561541,50545.0158066331277951,
50565.0088906453383970,50585.0039511376307928,50605.0009877193297143,50624.9999999999708962,
50645.0009875892646960,50665.0039500971324742,50685.0088871336774901,50705.0157983091921778,
50725.0246832341654226,50745.0355415192825603,50765.0483727754108259,50785.0631766136211809,
50805.0799526451592101,50825.0987004814887769,50845.1194197342410916,50865.1421100152438157,
50885.1667709365210612,50905.1934021102788392,50925.2220031489341636,50945.2525736650713952,
50965.2851132714713458,50985.3196215811185539,51005.3560982071721810,51025.3945427629805636,
51045.4349548620957648,51065.4773341182444710,51085.5216801453570952,51105.5679925575459492,
51125.6162709691125201,51145.6665149945401936,51165.7187242485233583,51185.7728983459164738,
51205.8290369017777266,51225.8871395313617541,51245.9472058501050924,51266.0092354736189009,
51286.0732280177180655,51306.1391830983993714,51326.2071003318560543,51346.2769793344486970,
51366.3488197227561614,51386.4226211135101039,51406.4983831236531842,51426.5761053703099606,
51446.6557874707868905,51466.7374290425868821,51486.8210297033801908,51506.9065890710480744,
51526.9941067636318621,51547.0835823993911617,51567.1750155967383762,51587.2684059742969112,
51607.3637531508575194,51627.4610567454146803,51647.5603163771302206,51667.6615316653624177,
51687.7647022296514479,51707.8698276897266624,51727.9769076654993114,51748.0859417770552682,
51768.1969296446768567,51788.3098708888355759,51808.4247651301702717,51828.5416119895235170,
51848.6604110879052314,51868.7811620465145097,51888.9038644867396215,51909.0285180301434593,
51929.1551222984853666,51949.2836769136847579,51969.4141814978720504,51989.5466356733450084,
52009.6810390625905711,52029.8173912882630248,52049.9556919732131064,52070.0959407404807280,
52090.2381372132731485,52110.3822810149868019,52130.5283717692000209,52150.6764090996657615,
52170.8263926303334301,52190.9783219853197807,52211.1321967889307416,52231.2880166656541405,
52251.4457812401451520,52271.6054901372699533,52291.7671429820402409,52311.9307393996641622,
52332.0962790155463153,52352.2637614552440937,52372.4331863445186173,52392.6045533092838014,
52412.7778619756645639,52432.9531119699458941,52453.1303029185946798,52473.3094344482742599,
52493.4905061857934925,52513.6735177581795142,52533.8584687926049810,52554.0453589164462755,
52574.2341877572544036,52594.4249549427404418,52614.6176601008119178,52634.8123028595582582,
52655.0088828472289606,52675.2073996922699735,52695.4078530232945923,52715.6102424690980115,
52735.8145676586573245,52756.0208282211096957,52776.2290237858032924,52796.4391539822245250,
52816.6512184400562546,52836.8652167891705176,52857.0811486595994211,52877.2990136815496953,
52897.5188114854245214,52917.7405417017725995,52937.9642039613536326,52958.1897978950801189,
52978.4173231340464554,52998.6467793095289380,53018.8781660529784858,53039.1114829960060888,
53059.3467297704191878,53079.5839060081925709,53099.8230113414829248,53120.0640454025997315,
53140.3070078240634757,53160.5518982385328854,53180.7987162788704154,53201.0474615780913155,
53221.2981337694000104,53241.5507324861755478,53261.8052573619643226,53282.0617080304873525,
53302.3200841256402782,53322.5803852814933634,53342.8426111322987708,53363.1067613124687341,
53383.3728354565973859,53403.6408831994534819,53423.9107541759731248,53444.1825980212597642,
53464.4563643706133007,53484.7320528594791540,53505.0096631234991946,53525.2891947984680883,
53545.5706475203623995,53565.8540209253333160,53586.1393146496993722,53606.4265283299537259,
53626.7156616027641576,53647.0067141049585189,53667.2996854735465604,53687.5945753457199316,
53707.8913833588158013,53728.1901091503605130,53748.4907523580550333,53768.7933126197531237,
53789.0977895734977210,53809.4041828574845567,53829.7124921101058135,53850.0227169698991929,
53870.3348570755842957,53890.6489120660553453,53910.9648815803666366,53931.2827652577398112,
53951.6025627375856857,53971.9242736594605958,53992.2478976631100522,54012.5734343884396367,
54032.9008834755295538,54053.2302445646200795,54073.5615172961333883,54093.8947013106444501,

54114.2297962489101337,54134.5668017518546549,54154.9057174605695764,54175.2465430163138080,
54195.5892780605063308,54215.9339222347553005,54236.2804751808143919,54256.6289365406264551,
54276.9793059562798589,54297.3315830700448714,54317.6857675243591075,54338.0418589618275291,
54358.3998570252151694,54378.7597613574544084,54399.1215716016668011,54419.4852874011048698,
54439.8509083992248634,54460.2184342396139982,54480.5878645660559414,54500.9591990224798792,
54521.3324372529968969,54541.7075789018781506,54562.0846236135548679,54582.4635710326401750,
54602.8444208038854413,54623.2271725722457631,54643.6118259828072041,54663.9983806808377267,
54684.3868363117726403,54704.7771925212073256,54725.1694489548972342,54745.5636052587724407,
54765.9596610789230908,54786.3576160616139532,54806.7574698532553157,54827.1592221004393650,
54847.5628724499038071,54867.9684205485827988,54888.3758660435341881,54908.7852085820122738,
54929.1964478114168742,54949.6095833793224301,54970.0246149334634538,54990.4415421217272524,
55010.8603645921903080,55031.2810819930600701,55051.7036939727331628,55072.1282001797590056,
55092.5546002628398128,55112.9828938708742498,55133.4130806528773974,55153.8451602580607869,
55174.2791323357887450,55194.7149965355856693,55215.1527525071433047,55235.5923999002989149,
55256.0339383650789387,55276.4773675516553340,55296.9226871103601297,55317.3698966916854260,
55337.8189959463052219,55358.2699845250244834,55378.7228620788300759,55399.1776282588689355,
55419.6342827164407936,55440.0928251030127285,55460.5532550702046137,55481.0155722698036698,
55501.4797763537644641,55521.9458669741870835,55542.4138437833389617,55562.8837064336548792,
55583.3554545777151361,55603.8290878682601033,55624.3046059582193266,55644.7820085006387671,
55665.2612951487535611,55685.7424655559516395,55706.2255193757737288,55726.7104562619279022,
55747.1972758682750282,55767.6859778488433221,55788.1765618578137946,55808.6690275495275273,
55829.1633745784783969,55849.6596025993276271,55870.1577112668892369,55890.6577002361445921,
55911.1595691622205777,55931.6633177004114259,55952.1689455061641638,55972.6764522350858897,
55993.1858375429437729,56013.6971010856505018,56034.2102425193006638,56054.7252615001198137,
56075.2421576845081290,56095.7609307290113065,56116.2815802903423901,56136.8041060253672185,
56157.3285075911044260,56177.8547846447399934,56198.3829368435981451,56218.9129638451850042,
56239.4448653071376611,56259.9786408872678294,56280.5142902435254655,56301.0518130340424250,
56321.5912089170815307,56342.1324775510802283,56362.6756185946142068,56383.2206317064192262,
56403.7675165453983936,56424.3162727706076112,56444.8669000412410242,56465.4193980166674010,
56485.9737663563937531,56506.5300047201017151,56527.0881127676111646,56547.6480901589020505,
56568.2099365541071165,56588.7736516135191778,56609.3392349975838442,56629.9066863668995211,
56650.4760053822101327,56671.0471917044196744,56691.6202449945994886,56712.1951649139591609,
56732.7719511238683481,56753.3506032858349499,56773.9311210615414893,56794.5135041128232842,
56815.0977521016466198,56835.6838646901524044,56856.2718415406270651,56876.8616823155098245,
56897.4533866773927002,56918.0469542890277808,56938.6423848132981220,56959.2396779132614029,
56979.8388332521208213,57000.4398504932250944,57021.0427293000902864,57041.6474693363707047,
57062.2540702658734517,57082.8625317525584251,57103.4728534605528694,57124.0850350541077205,
57144.6990761976485373,57165.3149765557391220,57185.9327357931033475,57206.5523535746106063,
57227.1738295652758097,57247.7971634302812163,57268.4223548349400517,57289.0494034447328886,
57309.6783089252858190,57330.3090709423704538,57350.9416891619111993,57371.5761632499852567,
57392.2124928728153463,57412.8506776967842598,57433.4907173884057556,57454.1326116143682157,
57474.7763600414909888,57495.4219623367462191,57516.0694181672661216,57536.7187272003138787,
57557.3698891033200198,57578.0229035438605933,57598.6777701896426152,57619.3344887085477239,
57639.9930587685885257,57660.6534800379376975,57681.3157521849061595,57701.9798748779649031,
57722.6458477857304388,57743.3136705769502441,57763.9833429205464199,57784.6548644855720340,
57805.3282349412329495,57826.0034539568805485,57846.6805212020262843,57867.3594363463125774,
57888.0401990595273674,57908.7228090116332169,57929.4072658727091039,57950.0935693130013533,
57970.7817190028945333,57991.4717146129114553,58012.1635558137495536,58032.8572422762226779,
58053.5527736713120248,58074.2501496701297583,58094.9493699439481134,58115.6504341641848441,
58136.3533420023886720,58157.0580931302756653,58177.7646872196928598,58198.4731239426400862,
58219.1834029712554184,58239.8955239778370014,58260.6094866348212236,58281.3252906147754402,
58302.0429355904343538,58322.7624212346781860,58343.4837472205108497,58364.2069132210963289,
58384.9319189097514027,58405.6587639599238173,58426.3874480451995623,58447.1179708393392502,

58467.8503320162126329,58488.5845312498640851,58509.3205682144616731,58530.0584425843335339,
58550.7981540339314961,58571.5397022378747351,58592.2830868709061178,58613.0283076079285820,
58633.7753641239833087,58654.5242560942497221,58675.2749831940527656,58696.0275450988774537,
58716.7819414843252162,58737.5381720261575538,58758.2962364002742106,58779.0561342827277258,
58799.8178653496943298,58820.5814292775030481,58841.3468257426429773,58862.1140544217123534,
58882.8831149914840353,58903.6540071288472973,58924.4267305108514847,58945.2012848146841861,
58965.9776697176639573,58986.7558848972694250,59007.5359300311174593,59028.3178047969486215,
59049.1015088726635440,59069.8870419363011024,59090.6744036660456914,59111.4635937402126729,
59132.2546118372629280,59153.0474576358028571,59173.8421308145698276,59194.6386310524685541,
59215.4369580285056145,59236.2371114218549337,59257.0390909118286800,59277.8428961778772646,
59298.6485268995893421,59319.4559827566845343,59340.2652634290498099,59361.0763685966958292,
59381.8892979397569434,59402.7040511385421269,59423.5206278734767693,59444.3390278251390555,
59465.1592506742308615,59485.9812961015995825,59506.8051637882526848,59527.6308534153140499,
59548.4583646640458028,59569.2876972158628632,59590.1188507523183944,59610.9518249550892506,
59631.7866195060123573,59652.6232340870483313,59673.4616683803105843,59694.3019220680289436,
59715.1439948325933074,59735.9878863565245410,59756.8335963224817533,59777.6811244132550200,
59798.5304703117944882,59819.3816337011594442,59840.2346142645692453,59861.0894116853814921,
59881.9460256470702006,59902.8044558332694578,59923.6647019277443178,59944.5267636143835261,
59965.3906405772431754,59986.2563325004884973,60007.1238390684375190,60027.9931599655392347,
60048.8642948763808818,60069.7372434856879408,60090.6120054783241358,60111.4885805392841576,
60132.3669683537082165,60153.2471686068674899,60174.1291809841641225,60195.0130051711530541,
60215.8986408535129158,60236.7860877170605818,60257.6753454477511696,60278.5664137316707638,
60299.4592922550436924,60320.3539807042470784,60341.2504787657599081,60362.1487861262285151,
60383.0489024724156479,60403.9508274912368506,60424.8545608697168063,60445.7601022950402694,
60466.6674514545156853,60487.5766080355897429,60508.4875717258473742,60529.4003422129972023,
60550.3149191848933697,60571.2313023295209859,60592.1494913350034039,60613.0694858895876678,
60633.9912856816736166,60654.9148903997847810,60675.8402997325683828,60696.7675133688317146,
60717.6965309974839329,60738.6273523076015408,60759.5599769883701811,60780.4944047291282914,
60801.4306352193234488,60822.3686681485560257,60843.3085032065646374,60864.2501400832043146,
60885.1935784684683313,60906.1388180524954805,60927.0858585255409707,60948.0346995780055295,
60968.9853409004208515,60989.9377821834423230,61010.8920231178635731,61031.8480633946164744,
61052.8059027047638665,61073.7655407394922804,61094.7269771901337663,61115.6902117481367895,
61136.6552441051026108,61157.6220739527416299,61178.5907009829170420,61199.5611248876157333,
61220.5333453589482815,61241.5073620891707833,61262.4831747706630267,61283.4607830959430430,
61304.4401867576452787,61325.4213854485569755,61346.4043788615817903,61367.3891666897616233,
61388.3757486262620660,61409.3641243643869529,61430.3542935975710861,61451.3462560193729587,
61472.3400113234965829,61493.3355592037623865,61514.3328993541217642,61535.3320314686716301,
61556.3329552416180377,61577.3356703673125594,61598.3401765402377350,61619.3464734549925197,
61640.3545608063286636,61661.3644382890997804,61682.3761055983122787,61703.3895624290889828,
61724.4048084766909597,61745.4218434365102439,61766.4406670040625613,61787.4612788749873289,
61808.4836787450694828,61829.5078663102030987,61850.5338412664350471,61871.5616033099286142,
61892.5911521369707771,61913.6224874439867563,61934.6556089275254635,61955.6905162842667778,
61976.7272092110215453,61997.7656874047243036,62018.8059505624478334,62039.8479983813813305,
62060.8918305588376825,62081.9374467922898475,62102.9848467792980955,62124.0340302175754914,
62145.0849968049660674,62166.1377462394157192,62187.1922782190304133,62208.2485924420252559,
62229.3066886067390442,62250.3665664116560947,62271.4282255553771392,62292.4916657366266008,
62313.5568866542671458,62334.6238880072705797,62355.6926694947615033,62376.7632308159736567,
62397.8355716702717473,62418.9096917571441736,62439.9855907762103016,62461.0632684272204642,
62482.1427244100486860,62503.2239584246854065,62524.3069701712665847,62545.3917593500300427,
62566.4783256613663980,62587.5666688057681313,62608.6567884838805185,62629.7486843964506988,
62650.8423562443567789,62671.9378037286223844,62693.0350265503657283,62714.1340244108578190,
62735.2347970114788041,62756.3373440537325223,62777.4416652392756077,62798.5477602698520059,
62819.6556288473584573,62840.7652706738008419,62861.8766854513232829,62882.9898728821863187,

62904.1048326687741792,62925.2215645136020612,62946.3400681193088531,62967.4603431886571343,
62988.5823894245258998,63009.7062065299396636,63030.8317942080175271,63051.9591521620386629,
63073.0882800953695551,63094.2191777115294826,63115.3518447141541401,63136.4862808069883613,
63157.6224856939224992,63178.7604590789560461,63199.9002006662194617,63221.0417101599668968,
63242.1849872645689175,63263.3300316845343332,63284.4768431244738167,63305.6254212891435600,
63326.7757658834088943,63347.9278766122588422,63369.0817531808133936,63390.2373952943162294,
63411.3948026581201702,63432.5539749777162797,63453.7149119587120367,63474.8776133068386116,
63496.0420787279435899,63517.2083079279982485,63538.3763006131193833,63559.5460564895038260,
63580.7175752635157551,63601.8908566416066606,63623.0659003303735517,63644.2427060365153011,
63665.4212734668690246,63686.6016023283809773,63707.7836923281356576,63728.9675431733339792,
63750.1531545712787192,63771.3405262294181739,63792.5296578553170548,63813.7205491566492128,
63834.9131998412267421,63856.1076096169781522,63877.3037781919410918,63898.5017052742841770,
63919.7013905722997151,63940.9028337944037048,63962.1060346491140081,63983.3109928450940060,
64004.5177080911089433,64025.7261800960477558,64046.9364085689376225,64068.1483932189003099,
64089.3621337551958277,64110.5776298871933250,64131.7948813243929180,64153.0138877764038625,
64174.2346489529663813,64195.4571645639371127,64216.6814343192891101,64237.9074579291118425,
64259.1352351036257460,64280.3647655531603959,64301.5960489881690592,64322.8290851192359696,
64344.0638736570399487,64365.3004143123980612,64386.5387067962510628,64407.7787508196342969,
64429.0205460937213502,64450.2640923298095004,64471.5093892392906127,64492.7564365337093477,
64514.0052339247049531,64535.2557811240330921,64556.5080778435803950,64577.7621237953571836,
64599.0179186914683669,64620.2754622441716492,64641.5347541658047703,64662.7957941688437131,
64684.0585819658954279,64705.3231172696614522,64726.5893997929742909,64747.8574292487755883,
64769.1272053501379560,64790.3987278102358687,64811.6719963423747686,64832.9470106599692372,
64854.2237704765575472,64875.5022755057943868,64896.7825254614508594,64918.0645200574144837,
64939.3482590076819179,64960.6337420263880631,64981.9209688277624082,65003.2099391261654091,
65024.5006526360666612,65045.7931090720667271,65067.0873081488607568,65088.3832495812821435,
65109.6809330842588679,65130.9803583728644298,65152.2815251622596406,65173.5844331677362788,
65194.8890821047025383,65216.1954716886830283,65237.5036016353187733,65258.8134716603526613,
65280.1250814796658233,65301.4384308092412539,65322.7535193651783629,65344.0703468637075275,
65365.3889130211464362,65386.7092175539582968,65408.0312601787009044,65429.3550406120557454,
65450.6805585708207218,65472.0078137719101505,65493.3368059323547641,65514.6675347692798823,
65535.9999999999563443,65557.3342013417568523,65578.6701385121705243,65600.0078112287883414,
65621.3472192093322519,65642.6883621716260677,65664.0312398336391198,65685.3758519134134986,
65706.7221981291368138,65728.0702781990839867,65749.4200918416609056,65770.7716387754044263,
65792.1249187189387158,65813.4799313910043566,65834.8366765104583465,65856.1951537963032024,
65877.5553629675996490,65898.9173037435539300,65920.2809758434887044,65941.6463789868430467,
65963.0135128931578947,65984.3823772820760496,66005.7529718733858317,66027.1252963869628729,
66048.4993505427992204,66069.8751340610178886,66091.2526466618437553,66112.6318880656181136,
66134.0128579927695682,66155.3955561638867948,66176.7799822996312287,66198.1661361207952723,
66219.5540173482731916,66240.9436257031047717,66262.3349609063880052,66283.7280226793955080,
66305.1228107434435515,66326.5193248200230300,66347.9175646306975977,66369.3175298971618759,
66390.7192203412269009,66412.1226356847910210,66433.5277756498835515,66454.9346399586356711,
66476.3432283333240775,66497.7535404962836765,66519.1655761699948926,66540.5793350770400139,
66561.9948169401177438,66583.4120214820432011,66604.8309484257333679,66626.2515974942216417,
66647.6739684106287314,66669.0980608982354170,66690.5238746803806862,66711.9514094805635978,
66733.3806650223705219,66754.8116410294751404,66776.2443372257112060,66797.6787533349852310,
66819.1148890813201433,66840.5527441888843896,66861.9923183819046244,66883.4336113847384695,
66904.8766229218890658,66926.3213527179032099,66947.7678004975023214,66969.2159659854660276,
66990.6658489067340270,67012.1174489863042254,67033.5707659493346000,67055.0257995210558875,
67076.4825494268152397,67097.9410153920762241,67119.4011971424333751,67140.8630944035539869,
67162.3267069012217689,67183.7920343613513978,67205.2590765099594137,67226.7278330731496681,
67248.1983037771715317,67269.6704883483471349,67291.1443865131441271,67312.6199979980883654,
67334.0973225298803300,67355.5763598352932604,67377.0571096411877079,67398.5395716745697428,

67420.0237456625472987,67441.5096313323301729,67462.9972284112300258,67484.4865366266894853,
67505.9775557062239386,67527.4702853774942923,67548.9647253682633163,67570.4608754063665401,
67591.9587352197995642,67613.4583045366307488,67634.9595830850303173,67656.4625705933285644,
67677.9672667898994405,67699.4736714032478631,67720.9817841620242689,67742.4916047949227504,
67764.0031330307974713,67785.5163685985753546,67807.0313112273142906,67828.5479606461740332,
67850.0663165844016476,67871.5863787713897182,67893.1081469365890371,67914.6316208096104674,
67936.1568001201376319,67957.6836845979705686,67979.2122739730111789,68000.7425679752632277,
68022.2745663348759990,68043.8082687820569845,68065.3436750471446430,68086.8807848605792969,
68108.4195979529176839,68129.9601140547893010,68151.5023328969691647,68173.0462542103196029,
68194.5918777258339105,68216.1392031745635904,68237.6882302877056645,68259.2389587965444662,
68280.7913884324807441,68302.3455189270316623,68323.9013500117871445,68345.4588814184826333,
68367.0181128789117793,68388.5790441250283038,68410.1416748888441361,68431.7060049025021726,
68453.2720338982617250,68474.8397616084548645,68496.4091877655446297,68517.9803121020813705,
68539.5531343507318525,68561.1276542442792561,68582.7038715155795217,68604.2817858976341086,
68625.8613971235026838,68647.4427049263904337,68669.0257090396044077,68690.6104091965244152,
68712.1968051306612324,68733.7848965756274993,68755.3746832651231671,68776.9661649329937063,
68798.5593413131282432,68820.1542121395905269,68841.7507771464734105,68863.3490360680443700,
68884.9489886386290891,68906.5506345926842187,68928.1539736647391692,68949.7590055894397665,
68971.3657301015773555,68992.9741469359869370,69014.5842558276344789,69036.1960565115878126,
69057.8095487230166327,69079.4247321972070495,69101.0416066695324844,69122.6601718754682224,
69144.2804275506059639,69165.9023734306247206,69187.5260092513344716,69209.1513347486179555,
69230.7783496584743261,69252.4070537169900490,69274.0374466604116606,69295.6695282250002492,
69317.3032981471915264,69338.9387561634939630,69360.5759020105324453,69382.2147354250046192,
69403.8552561437536497,69425.4974639036809094,69447.1413584418332903,69468.7869394953304436,
69490.4342068013938842,69512.0831600973906461,69533.7337991207168670,69555.3861236089287559,
69577.0401332996698329,69598.6958279306854820,69620.3532072397938464,69642.0122709649731405,
69663.6730188442597864,69685.3354506157920696,69706.9995660178392427,69728.6653647887433181,
69750.3328466669627232,69772.0020113910577493,69793.6728586996905506,69815.3453883316105930,
69837.0196000256692059,69858.6954935208486859,69880.3730685562040890,69902.0523248709068866,
69923.7332622042158619,69945.4158802954916609,69967.1001788842113456,69988.7861577099392889,
70010.4738165123562794,70032.1631550312158652,70053.8541730064025614,70075.5468701778736431,
70097.2412462857173523,70118.9373010701092426,70140.6350342712976271,70162.3344456296908902,
70184.0355348857410718,70205.7383017800166272,70227.4427460532169789,70249.1488674460997572,
70270.8566656995390076,70292.5661405545106390,70314.2772917521069758,70335.9901190334931016,
70357.7046221399359638,70379.4208008128189249,70401.1386547936126590,70422.8581838238897035,
70444.5793876453390112,70466.3022659997222945,70488.0268186289176811,70509.7530452748760581,
70531.4809456797083840,70553.2105195855547208,70574.9417667347006500,70596.6746868695045123,
70618.4092797324556159,70640.1455450661014766,70661.8834826131060254,70683.6230921162641607,
70705.3643733184144367,70727.1073259625263745,70748.8519497916713590,70770.5982445490080863,
70792.3462099777825642,70814.0958458213717677,70835.8471518232254311,70857.6001277268951526,
70879.3547732760343933,70901.1110882144130301,70922.8690722858591471,70944.6287252343317959,
70966.3900468038773397,70988.1530367386294529,71009.9176947828527773,71031.6840206808847142,
71053.4520141771499766,71075.2216750162042445,71096.9930029426614055,71118.7659977012663148,
71140.5406590368511388,71162.3169866943353554,71184.0949804187403060,71205.8746399552182993,
71227.6559650489507476,71249.4389554452936864,71271.2236108896322548,71293.0099311274825595,
71314.7979159044771222,71336.5875649663066724,71358.3788780587638030,71380.1718549277720740,
71401.9664953193132533,71423.7627989794855239,71445.5607656544889323,71467.3603950905962847,
71489.1616870342113543,71510.9646412318106741,71532.7692574299726402,71554.5755353753629606,
71576.3834748147492064,71598.1930754950299161,71620.0043371631327318,71641.8172595661453670,
71663.6318424512137426,71685.4480855656001950,71707.2659886566398200,71729.0855514717841288,
71750.9067737585864961,71772.7296552646730561,71794.5541957377718063,71816.3803949257126078,
71838.2082525764417369,71860.0377684379636776,71881.8689422583847772,71903.7017737859423505,
71925.5362627689319197,71947.3724089557508705,71969.2102120948984521,71991.0496719349757768,

72012.8907882246858208,72034.7335607127897674,72056.5779891481652157,72078.4240732798207318,
72100.2718128567939857,72122.1212076282536145,72143.9722573434701189,72165.8249617518013110,
72187.6793206026923144,72209.5353336456901161,72231.3930006304290146,72253.2523213066451717,
72275.1132954241766129,72296.9759227329486748,72318.8402029829594539,72340.7061359243380139,
72362.5737213072716258,72384.4429588820930803,72406.3138483991788235,72428.1863896090362687,
72450.0605822622164851,72471.9364261094306130,72493.8139209014334483,72515.6930663890962023,
72537.5738623233919498,72559.4563084553519730,72581.3404045361385215,72603.2261503169866046,
72625.1135455492476467,72647.0025899843312800,72668.8932833737635519,72690.7856254691723734,
72712.6796160222729668,72734.5752547848533140,72756.4725415088178124,72778.3714759461436188,
72800.2720578489388572,72822.1742869693553075,72844.0781630596902687,72865.9836858722846955,
72887.8908551595959580,72909.7996706741832895,72931.7101321686932351,72953.6222393958450994,
72975.5359921084891539,72997.4513900595193263,73019.3684330019605113,73041.2871206889249152,
73063.2074528736120556,73085.1294293092942098,73107.0530497493891744,73128.9783139473438496,
73150.9052216567361029,73172.8337726312165614,73194.7639666245668195,73216.6958033906121273,
73238.6292826832795981,73260.5644042566273129,73282.5011678647570079,73304.4395732619013870,
73326.3796202023368096,73348.3213084404851543,73370.2646377308410592,73392.2096078279573703,
73414.1562184865324525,73436.1044694613228785,73458.0543605071725324,73480.0058913790562656,
73501.9590618319925852,73523.9138716211164137,73545.8703205016645370,73567.8284082289319485,
73589.7881345583300572,73611.7494992453575833,73633.7125020456151105,73655.6771427147468785,
73677.6434210085571976,73699.6113366828794824,73721.5808894936926663,73743.5520791970193386,
73765.5249055489985039,73787.4993683058564784,73809.4754672239068896,73831.4532020595506765,
73853.4325725692906417,73875.4135785097168991,73897.3962196375068743,73919.3804957094107522,
73941.3664064823096851,73963.3539517131430330,73985.3431311589520192,74007.3339445768651785,
74029.3263917240983574,74051.3204723579692654,74073.3161862358829239,74095.3135331153025618,
74117.3125127538369270,74139.3131249091384234,74161.3153693389758701,74183.3192458011908457,
74205.3247540537267923,74227.3318938546290156,74249.3406649619864766,74271.3510671340336557,
74293.3631001290486893,74315.3767637054406805,74337.3920576216623886,74359.4089816362975398,
74381.4275355080026202,74403.4477189955068752,74425.4695318576705176,74447.4929738533828640,
74469.5180447416933021,74491.5447442816803232,74513.5730722325388342,74535.6030283535510534,
74557.6346124040865107,74579.6678241436020471,74601.7026633316418156,74623.7391297278372804,
74645.7772230919363210,74667.8169431837159209,74689.8582897631131345,74711.9012625900941202,
74733.9458614247414516,74755.9920860272250138,74778.0399361578020034,74800.0894115768169286,
74822.1405120447016088,74844.1932373219606234,74866.2475871692295186,74888.3035613471874967,
74910.3611596166301752,74932.4203817384113790,74954.4812274735013489,74976.5436965829721885,
74998.6077888279251056,75020.6735039696068270,75042.7408417693222873,75064.8098019884637324,
75086.8803843885398237,75108.9525887311028782,75131.0264147778361803,75153.1018622904666699,
75175.1789310308522545,75197.2576207609236008,75219.3379312426695833,75241.4198622382100439,
75263.5034135097375838,75285.5885848195030121,75307.6753759298735531,75329.7637866033182945,
75351.8538166023645317,75373.9454656896123197,75396.0387336278072326,75418.1336201797239482,
75440.2301251082535600,75462.3282481763599208,75484.4279891471087467,75506.5293477836530656,
75528.6323238491895609,75550.7369171070749871,75572.8431273206952028,75594.9509542535379296,
75617.0603976691927528,75639.1714573313074652,75661.2841330036462750,75683.3984244500315981,
75705.5143314344022656,75727.6318537207407644,75749.7509910731751006,75771.8717432558623841,
75793.9941100330761401,75816.1180911691772053,75838.2436864285846241,75860.3708955758484080,
75882.4997183755622245,75904.6301545924216043,75926.7622039912239416,75948.8958663368248381,
75971.0311413941817591,75993.1680289283249294,76015.3065287044009892,76037.4466404876002343,
76059.5883640432148241,76081.7316991366533330,76103.8766455333534395,76126.0232029988837894,
76148.1713712988712359,76170.3211501990444958,76192.4725394652050454,76214.6255388632562244,
76236.7801481591741322,76258.9363671190076275,76281.0941955089219846,76303.2536330951406853,
76325.4146796439745231,76347.5773349218507065,76369.7415986952255480,76391.9074707306863274,
76414.0749507948785322,76436.2440386545640649,76458.4147340765484842,76480.5870368277537636,
76502.7609466751746368,76524.9364633858931484,76547.1135867270495510,76569.2923164659150643,
76591.4726523698191158,76613.6545942061638925,76635.8381417424679967,76658.0232947463082382,

76680.2100529853487387,76702.3984162273409311,76724.5883842401381116,76746.7799567916372325,
76768.9731336498662131,76791.1679145828966284,76813.3642993589019170,76835.5622877461573808,
76857.7618795129674254,76879.9630744277965277,76902.1658722591091646,76924.3702727755298838,
76946.5762757457268890,76968.7838809384411434,76990.9930881225154735,77013.2038970668945694,
77035.4163075405667769,77057.6303193126223050,77079.8459321522386745,77102.0631458286952693,
77124.2819601113005774,77146.5023747694795020,77168.7243895727588097,77190.9480042907234747,
77213.1732186930312309,77235.4000325494416757,77257.6284456298017176,77279.8584577040310251,
77302.0900685421220260,77324.3232779141690116,77346.5580855903390329,77368.7944913408864522,
77391.0324949361383915,77413.2720961465238361,77435.5132947425299790,77457.7560904947313247,
77480.0004831738042412,77502.2464725504978560,77524.4940583956340561,77546.7432404801074881,
77568.9940185749437660,77591.2463924511976074,77613.5003618800255936,77635.7559266326570651,
77658.0130864804377779,77680.2718411947571440,77702.5321905470918864,77724.7941343090205919,
77747.0576722521946067,77769.3228041483234847,77791.5895297692477470,77813.8578488868370187,
77836.1277612730627880,77858.3992666999984067,77880.6723649397899862,77902.9470557646272937,
77925.2233389468310634,77947.5012142587802373,77969.7806814729265170,77992.0617403618380195,
78014.3443906981265172,78036.6286322544910945,78058.9144648037472507,78081.2018881187250372,
78103.4909019724145764,78125.7815061378205428,78148.0737003880640259,78170.3674844963388750,
78192.6628582359262509,78214.9598213801655220,78237.2583737024979200,78259.5585149764519883,
78281.8602449756144779,78304.1635634736594511,78326.4684702443628339,78348.7749650615296559,
78371.0830476991250180,78393.3927179310994688,78415.7039755315781804,78438.0168202747008763,
78460.3312519346945919,78482.6472702859027777,78504.9648751027270919,78527.2840661596274003,
78549.6048432311945362,78571.9272060920484364,78594.2511545169109013,78616.5766882806055946,
78638.9038071579852840,78661.2325109240337042,78683.5627993537782459,78705.8946722223481629,
78728.2281293049454689,78750.5631703768594889,78772.8997952134232037,78795.2380035901005613,
78817.5777952823991654,78839.9191700659284834,78862.2621277163561899,78884.6066680094518233,
78906.9527907210431295,78929.3004956270451657,78951.6497825034603011,78974.0006511263782158,
78996.3531012719322462,79018.7071327163575916,79041.0627452359767631,79063.4199386071850313,
79085.7787126064358745,79108.1390670102846343,79130.5010015953885159,79152.8645161384192761,
79175.2296104161796393,79197.5962842055305373,79219.9645372834202135,79242.3343694268696709,
79264.7057804129872238,79287.0787700189539464,79309.4533380220091203,79331.8294841995084425,
79354.2072083288658177,79376.5865101875824621,79398.9673895532177994,79421.3498462034331169,
79443.7338799159479095,79466.1194904685835354,79488.5066776392195607,79510.8954412058228627,
79533.2857809464330785,79555.6776966391626047,79578.0711880622257013,79600.4662549938948359,
79622.8628972125152359,79645.2611144965485437,79667.6609066244709538,79690.0622733748750761,
79712.4652145264553837,79734.8697298579354538,79757.2758191481261747,79779.6834821759548504,
79802.0927187203778885,79824.5035285604535602,79846.9159114753274480,79869.3298672441887902,
79891.7453956463432405,79914.1624964611546602,79936.5811694680451183,79959.0014144465530990,
79981.4232311762607424,80003.8466194368520519,80026.2715790080837905,80048.6981096697709290,
80071.1262112018303014,80093.5558833842369495,80115.9871259970532265,80138.4199388204142451,
80160.8543216345278779,80183.2902742196893087,80205.7277963562810328,80228.1668878247146495,
80250.6075484055327252,80273.0497778793360339,80295.4935760267981095,80317.9389426286506932,
80340.3858774657273898,80362.8343803189491155,80385.2844509692804422,80407.7360891977878055,
80430.1892947855958482,80452.6440675139165251,80475.1004071640345501,80497.5583135173219489,
80520.0177863552089548,80542.4788254592131125,80564.9414306109247264,80587.4056015920068603,
80609.8713381841953378,80632.3386401693278458,80654.8075073293002788,80677.2779394460667390,
80699.7499363016831921,80722.2234976782783633,80744.6986233580391854,80767.1753131232399028,
80789.6535667562420713,80812.1333840394654544,80834.6147647554025752,80857.0977086866478203,
80879.5822156158537837,80902.0682853257312672,80924.5559175990929361,80947.0451122188242152,
80969.5358689678832889,80992.0281876292719971,81014.5220679861231474,81037.0175098216132028,
81059.5145129190059379,81082.0130770616087830,81104.5132020328310318,81127.0148876161838416,
81149.5181335951929213,81172.0229397535003955,81194.5293058748065960,81217.0372317428991664,
81239.5467171416385099,81262.0577618549577892,81284.5703656668483745,81307.0845283614034997,
81329.6002497227746062,81352.1175295351858949,81374.6363675829488784,81397.1567636504478287,

81419.6787175221252255,81442.2022289825108601,81464.7272978162218351,81487.2539238079334609,
81509.7821067423792556,81532.3118464043946005,81554.8431425789021887,81577.3759950508392649,
81599.9104036052740412,81622.4463680273329373,81644.9838881022151327,81667.5229636151780142,
81690.0635943515808322,81712.6057800968410447,81735.1495206364488695,81757.6948157559672836,
81780.2416652410465758,81802.7900688774097944,81825.3400264508236432,81847.8915377471712418,
81870.4446025523793651,81892.9992206524766516,81915.5553918335062917,81938.1131158816715470,
81960.6723925831756787,81983.2332217243383639,82005.7956030915374868,82028.3595364712236915,
82050.9250216499058297,82073.4920584142091684,82096.0606465507880785,82118.6307858463987941,
82141.2024760878412053,82163.7757170620316174,82186.3505085559299914,82208.9268503565690480,
82231.5047422510542674,82254.0841840265784413,82276.6651754703925690,82299.2477163698495133,
82321.8318065123166889,82344.4174456853070296,82367.0046336763480213,82389.5933702730544610,
82412.1836552631430095,82434.7754884343739832,82457.3688695745950099,82479.9637984716973733,
82502.5602749136887724,82525.1582986886060098,82547.7578695846023038,82570.3589873898599762,
82592.9616518926777644,82615.5658628813980613,82638.1716201444214676,82660.7789234702649992,
82683.3877726474747760,82705.9981674647133332,82728.6101077106577577,82751.2235931741161039,
82773.8386236439400818,82796.4551989090396091,82819.0733187584410189,82841.6929829811851960,
82864.3141913664294407,82886.9369437033747090,82909.5612397813238204,82932.1870793896378018,
82954.8144623177358881,82977.4433883551246254,83000.0738572913687676,83022.7058689161203802,
83045.3394230191042880,83067.9745193900889717,83090.6111578189593274,83113.2493380956293549,
83135.8890600101003656,83158.5303233524609823,83181.1731279128580354,83203.8174734814965632,
83226.4633598486689152,83249.1107868047402008,83271.7597541401337367,83294.4102616453747032,
83317.0623091110028327,83339.7158963277033763,83362.3710230861615855,83385.0276891771645751,
83407.6858943915867712,83430.3456385203608079,83453.0069213544775266,83475.6697426850005286,
83498.3341023030952783,83520.9999999999417923,83543.6674355668656062,83566.3364087951922556,
83589.0069194763491396,83611.6789674018509686,83634.3525523632415570,83657.0276741521665826,
83679.7043325603590347,83702.3825273795519024,83725.0622584016382461,83747.7435254185111262,
83770.4263282221800182,83793.1106666046835016,83815.7965403581620194,83838.4839492748287739,
83861.1728931469406234,83883.8633717668417376,83906.5553849269635975,83929.2489324197522365,
83951.9440140377992066,83974.6406295736960601,83997.3387788201507647,84020.0384615699294955,
84042.7396776158566354,84065.4424267508293269,84088.1467087678465759,84110.8525234599219402,
84133.5598706201708410,84156.2687500417960109,84178.9791615180292865,84201.6911048422043677,
84224.4045798077131622,84247.1195862080057850,84269.8361238366196631,84292.5541924871504307,
84315.2737919532810338,84337.9949220287380740,84360.7175825073354645,84383.4417731829453260,
84406.1674938495125389,84428.8947443010692950,84451.6235243316914421,84474.3538337355421390,
84497.0856723068282008,84519.8190398398728576,84542.5539361289993394,84565.2903609686763957,
84588.0283141534018796,84610.7677954777172999,84633.5088047362951329,84656.2513417238224065,
84678.9954062350734603,84701.7409980649099452,84724.4881170082517201,84747.2367628600622993,
84769.9869354154070606,84792.7386344694095897,84815.4918598172516795,84838.2466112541878829,
84861.0028885755600641,84883.7606915767682949,84906.5200200532563031,84929.2808738005696796,
84952.0432526143122232,84974.8071562901459401,84997.5725846238055965,85020.3395374111132696,
85043.1080144479492446,85065.8780155302374624,85088.6495404539891751,85111.4225890153029468,
85134.1971610103209969,85156.9732562352437526,85179.7508744863735046,85202.5300155600707512,
85225.3106792527396465,85248.0928653608571040,85270.8765736810164526,85293.6618040098110214,
85316.4485561439505545,85339.2368298801884521,85362.0266250153508736,85384.8179413463512901,
85407.6107786701322766,85430.4051367837237194,85453.2010154842573684,85475.9984145688649733,
85498.7973338347946992,85521.5977730793529190,85544.3997320999042131,85567.2032106938859215,
85590.0082086588081438,85612.8147257922391873,85635.6227618918201188,85658.4323167552647647,
85681.2433901803306071,85704.0559819648769917,85726.8700919068069197,85749.6857198040816002,
85772.5028654547641054,85795.3215286569611635,85818.1417092088522622,85840.9634069086750969,
85863.7866215547401225,85886.6113529454451054,85909.4376008792169159,85932.2653651545697358,
85955.0946455700905062,85977.9254419244098244,86000.7577540162747027,86023.5915816444321536,
86046.4269246077456046,86069.2637827051221393,86092.1021557355561526,86114.9420434980711434,
86137.7834457918070257,86160.6263624159182655,86183.4707931696757441,86206.3167378523794468,

86229.1641962634021183,86252.0131682022038149,86274.8636534683028003,86297.7156518612609943,
86320.5691631807276281,86343.4241872264246922,86366.2807237981178332,86389.1387726956745610,
86411.9983337189769372,86434.8594066680088872,86457.7219913428270957,86480.5860875435319031,
86503.4516950702964095,86526.3188137233519228,86549.1874433030316141,86572.0575836096832063,
86594.9292344437708380,86617.8023956057731993,86640.6770668962708442,86663.5532481159025338,
86686.4309390653797891,86709.3101395454432350,86732.1908493569644634,86755.0730683008005144,
86777.9567961779539473,86800.8420327894418733,86823.7287779363541631,86846.6170314198534470,
86869.5067930411751149,86892.3980626016127644,86915.2908399025182007,86938.1851247453159885,
86961.0809169314889004,86983.9782162626070203,87006.8770225402695360,87029.7773355661774985,
87052.6791551420901669,87075.5824810697959037,87098.4873131511849351,87121.3936511882202467,
87144.3014949828939280,87167.2108443372853799,87190.1216990535322111,87213.0340589338447899,
87235.9479237805062439,87258.8632933958288049,87281.7801675822411198,87304.6985461421718355,
87327.6184288781805662,87350.5398155928560300,87373.4627060888451524,87396.3871001688967226,
87419.3129976357740816,87442.2403982923569856,87465.1693019415397430,87488.0997083863185253,
87511.0316174297331600,87533.9650288748962339,87556.8999425250076456,87579.8363581832818454,
87602.7742756530205952,87625.7136947376129683,87648.6546152404916938,87671.5970369651477085,
87694.5409597151447088,87717.4863832941045985,87740.4333075057365932,87763.3817321537790122,
87786.3316570420574863,87809.2830819744558539,87832.2360067549161613,87855.1904311874532141,
87878.1463550761545775,87901.1037782251514727,87924.0627004386333283,87947.0231215208914364,
87969.9850412762461929,87992.9484595091053052,88015.9133760239055846,88038.8797906251711538,
88061.8477031175134471,88084.8171133055730024,88107.7880209940485656,88130.7604259877261939,
88153.7343280914647039,88176.7097271101374645,88199.6866228487488115,88222.6650151123030810,
88245.6449037059064722,88268.6262884347088402,88291.6091691039473517,88314.5935455189028289,
88337.5794174849143019,88360.5667848074081121,88383.5556472918542568,88406.5460047437954927,
88429.5378569688182324,88452.5312037726107519,88475.5260449608904310,88498.5223803394474089,
88521.5202097141300328,88544.5195328908739612,88567.5203496756585082,88590.5226598745066440,
88613.5264632935432019,88636.5317597389221191,88659.5385490168991964,88682.5468309337447863,
88705.5566052958456567,88728.5678719095885754,88751.5806305814912776,88774.5948811180714983,
88797.6106233259633882,88820.6278570118302014,88843.6465819823933998,88866.6667980444617569,
88889.6885050048877019,88912.7117026706109755,88935.7363908486004220,88958.7625693458976457,
88981.7902379696315620,89004.8193965269601904,89027.8500448251143098,89050.8821826714120107,
89073.9158098732004873,89096.9509262378851417,89119.9875315729732392,89143.0256256860011490,
89166.0652083845634479,89189.1062794763565762,89212.1488387691060780,89235.1928860705811530,
89258.2384211886674166,89281.2854439312650356,89304.3339541063760407,89327.3839515220170142,
89350.4354359863064019,89373.4884073074063053,89396.5428652935370337,89419.5988097530062078,
89442.6562404941651039,89465.7151573253941024,89488.7755600552191027,89511.8374484921369003,
89534.9008224447461544,89557.9656817217328353,89581.0320261318120174,89604.0998554837424308,
89627.1691695863992209,89650.2399682486720849,89673.3122512795380317,89696.3860184880177258,
89719.4612696832045913,89742.5380046742502600,89765.6162232703645714,89788.6959252808301244,
89811.7771105149877258,89834.8597787822072860,89857.9439298919751309,89881.0295636538066901,
89904.1166798772610491,89927.2052783720137086,89950.2953589477401692,89973.3869214142177952,
89996.4799655812676065,90019.5744912587833824,90042.6704982566880062,90065.7679863850207767,
90088.8669554538355442,90111.9674052732589189,90135.0693356534757186,90158.1727464047580725,
90181.2776373374072136,90204.3840082617971348,90227.4918589883600362,90250.6011893275863258,
90273.7119990900391713,90296.8242880863253959,90319.9380561271391343,90343.0533030231890734,
90366.1700285852857633,90389.2882326242979616,90412.4079149511380820,90435.5290753767767455,
90458.6517137122573331,90481.7758297686814331,90504.9014233572088415,90528.0284942890575621,
90551.1570423755038064,90574.2870674279110972,90597.4185692576429574,90620.5515476761938771,
90643.6860024950728985,90666.8219335258472711,90689.9593405801861081,90713.0982234697730746,
90736.2385820063645951,90759.3804160018044058,90782.5237252679507947,90805.6685096167639131,
90828.8147688602330163,90851.9625028104346711,90875.1117112794599961,90898.2623940795310773,
90921.4145510228554485,90944.5681819217425073,90967.7232865885598585,90990.8798648357187631,
91014.0379164757177932,91037.1974413210700732,91060.3584391843905905,91083.5209098783379886,

91106.6848532156291185,91129.8502690090390388,91153.0171570714010159,91176.1855172156210756,
91199.3553492546488997,91222.5266530014923774,91245.6994282692467095,91268.8736748710362008,
91292.0493926200579153,91315.2265813295525732,91338.4052408128336538,91361.5853708832873963,
91384.7669713543436956,91407.9500420394761022,91431.1345827522454783,91454.3205933062563417,
91477.5080735151714180,91500.6970231927261921,91523.8874421526852529,91547.0793302089296049,
91570.2726871753257001,91593.4675128658564063,91616.6638070945336949,91639.8615696754422970,
91663.0608004227251513,91686.2614991505543003,91709.4636656732182018,91732.6672998050198657,
91755.8724013603205094,91779.0789701535686618,91802.2870059992565075,91825.4965087119198870,
91848.7074781061674003,91871.9199139966658549,91895.1338161981693702,91918.3491845254175132,
91941.5660187932808185,91964.7843188166589243,91988.0040844104951248,92011.2253153898200253,
92034.4480115697078872,92057.6721727652911795,92080.8977987917460268,92104.1248894643649692,
92127.3534445984114427,92150.5834640092798509,92173.8149475123791490,92197.0478949232201558,
92220.2823060573136900,92243.5181807302724337,92266.7555187577527249,92289.9943199554691091,
92313.2345841391943395,92336.4763111247739289,92359.7195007280824939,92382.9641527650674107,
92406.2102670517488150,92429.4578434041613946,92452.7068816384708043,92475.9573815708135953,
92499.2093430174427340,92522.4627657946693944,92545.7176497188193025,92568.9739946063054958,
92592.2318002736137714,92615.4910665372590302,92638.7517932138289325,92662.0139801199402427,
92685.2776270723261405,92708.5427338877198054,92731.8093003829562804,92755.0773263748706086,
92778.3468116804142483,92801.6177561165823136,92824.8901595003844704,92848.1640216489468003,
92871.4393423794244882,92894.7161215090163751,92917.9943588550231652,92941.2740542347455630,
92964.5552074655715842,92987.8378183649620041,93011.1218867504067020,93034.4074124394683167,
93057.6943952497676946,93080.9828349989547860,93104.2727315047668526,93127.5640845849993639,
93150.8568940574914450,93174.1511597401404288,93197.4468814509164076,93220.7440590078040259,
93244.0426922288897913,93267.3427809323038673,93290.6443249362346251,93313.9473240589140914,
93337.2517781186325010,93360.5576869337673998,93383.8650503226963338,93407.1738681039278163,
93430.4841400959412567,93453.7958661173615837,93477.1090459867991740,93500.4236795229517156,
93523.7397665445605526,93547.0573068704543402,93570.3763003194908379,93593.6967467105714604,
93617.0186458626994863,93640.3419975948927458,93663.6668017262272770,93686.9930580758809811,
93710.3207664630317595,93733.6499267069302732,93756.9805386269144947,93780.3126020423369482,
93803.6461167726374697,93826.9810826372849988,93850.3174994558357866,93873.6553670478606364,
93896.9946852330322145,93920.3354538310377393,93943.6776726616662927,93967.0213415447069565,
93990.3664603000506759,94013.7130287476320518,94037.0610467074293410,94060.4105139994935598,
94083.7614304439193802,94107.1137958608451299,94130.4676100705110002,94153.8228728931571823,
94177.1795841491111787,94200.5377436587586999,94223.8973512425291119,94247.2584067209099885,
94270.6209099144325592,94293.9848606437299168,94317.3502587294206023,94340.7171039922395721,
94364.0853962529363343,94387.4551353323477088,94410.8263210513396189,94434.1989532308507478,
94457.5730316918779863,94480.9485562554473290,94504.3255267426575301,94527.7039429746801034,
94551.0838047727156663,94574.4651119580230443,94597.8478643519338220,94621.2320617758232402,
94644.6177040510956431,94668.0047909992717905,94691.3933224418724421,94714.7832982005056692,
94738.1747180967940949,94761.5675819524767576,94784.9618895893072477,94808.3576408290973632,
94831.7548354937025579,94855.1534734050655970,94878.5535543851729017,94901.9550782560545485,
94925.3580448397842702,94948.7624539585231105,94972.1683054344757693,94995.5755990898906020,
95018.9843347470741719,95042.3945122283912497,95065.8061313562648138,95089.2191919531760504,
95112.6336938416352496,95136.0496368442400126,95159.4670207836170448,95182.8858454824658111,
95206.3061107635294320,95229.7278164496092359,95253.1509623635647586,95276.5755483283137437,
95300.0015741668030387,95323.4290397020522505,95346.8579447571537457,95370.2882891552144429,
95393.7200727194285719,95417.1532952730194665,95440.5879566392977722,95464.0240566415886860,
95487.4615951033047168,95510.9005718479020288,95534.3409866988658905,95557.7828394797834335,
95581.2261300142563414,95604.6708581259590574,95628.1170236385951284,95651.5646263759845169,
95675.0136661619180813,95698.4641428203030955,95721.9160561750759371,95745.3694060502311913,
95768.8241922698070994,95792.2804146579146618,95815.7380730387085350,95839.1971672363870312,
95862.6576970752212219,95886.1196623795403866,95909.5830629736883566,95933.0478986821108265,
95956.5141693292680429,95979.9818747397075640,96003.4510147380060516,96026.9215891487983754,

96050.3935977967921644,96073.8670405067241518,96097.3419171033747261,96120.8182274116261397,
96144.2959712563751964,96167.7751484625769081,96191.2557588552444940,96214.7378022594493814,
96238.2212785003066529,96261.7061874029895989,96285.1925287927151658,96308.6803024947876111,
96332.1695083345257444,96355.6601461373211350,96379.1522157286090078,96402.6457169338682434,
96426.1406495786650339,96449.6370134886092274,96473.1348084893106716,96496.6340344065101817,
96520.1346910659631249,96543.6367782934685238,96567.1402959148836089,96590.6452437561529223,
96614.1516216432210058,96637.6594294021342648,96661.1686668589536566,96684.6793338397983462,
96708.1914301708748098,96731.7049556783895241,96755.2199101886653807,96778.7362935280107195,
96802.2541055228357436,96825.7733459995797602,96849.2940147847402841,96872.8161117048730375,
96896.3396365865773987,96919.8645892565109534,96943.3909695413894951,96966.9187772679579211,
96990.4480122630484402,97013.9786743535223650,97037.5107633662846638,97061.0442791283276165,
97084.5792214666726068,97108.1155902083846740,97131.6533851805870654,97155.1926062104903394,
97178.7332531252905028,97202.2753257522999775,97225.8188239188602893,97249.3637474523420678,
97272.9100961801887024,97296.4578699299163418,97320.0070685290411348,97343.5576918051956454,
97367.1097395860124379,97390.6632116991968360,97414.2181079724978190,97437.7744282337371260,
97461.3321723107655998,97484.8913400315068429,97508.4519312238990096,97532.0139457159821177,
97555.5773833358107368,97579.1422439115121961,97602.7085272712574806,97626.2762332432612311,
97649.8453616558108479,97673.4159123372228350,97696.9878851158864563,97720.5612798202055274,
97744.1360962787002791,97767.7123343198763905,97791.2899937723414041,97814.8690744647028623,
97838.4495762256847229,97862.0314988839963917,97885.6148422684491379,97909.1996062078833347,
97932.7857905311830109,97956.3733950673195068,97979.9624196452641627,98003.5528640940756304,
98027.1447282428562175,98050.7380119207664393,98074.3327149569959147,98097.9288371808070224,
98121.5263784215057967,98145.1253385084564798,98168.7257172710669693,98192.3275145387742668,
98215.9307301411317894,98239.5353639076638501,98263.1414156680111773,98286.7488852518144995,
98310.3577724888164084,98333.9680772087594960,98357.5797992414882174,98381.1929384168470278,
98404.8074945647822460,98428.4234675152401906,98452.0408570982690435,98475.6596631439169869,
98499.2798854823195143,98522.9015239436557749,98546.5245783581631258,98570.1490485560934758,
98593.7749343677860452,98617.4022356236237101,98641.0309521540475544,98664.6610837895132136,
98688.2926303605636349,98711.9255916977708694,98735.5599676317942794,98759.1957579932932276,
98782.8329626130143879,98806.4715813217335381,98830.1116139502846636,98853.7530603295745095,
98877.3959202905098209,98901.0401936640992062,98924.6858802813803777,98948.3329799734201515,
98971.9814925713872071,98995.6314179064502241,99019.2827558098506415,99042.9355061128735542,
99066.5896686468768166,99090.2452432432328351,99113.9022297334013274,99137.5606279488565633,
99161.2204377211455721,99184.8816588818590390,99208.5442912626313046,99232.2083346951694693,
99255.8737890112097375,99279.5406540425465209,99303.2089296210324392,99326.8786155785201117,
99350.5497117469931254,99374.2222179584350670,99397.8961340448877309,99421.5714598384220153,
99445.2481951712106820,99468.9263398754410446,99492.6058937833440723,99516.2868567272089422,
99539.9692285393975908,99563.6530090522865066,99587.3381980983249377,99611.0247955100057879,
99634.7128011198656168,99658.4022147604991915,99682.0930362645449350,99705.7852654646994779,
99729.4789021936885547,99753.1739462843252113,99776.8703975694370456,99800.5682558819098631,
99824.2675210546876770,99847.9681929207727080,99871.6702713131817291,99895.3737560650042724,
99919.0786470093880780,99942.7849439795245416,99966.4926468086341629,99990.2017553300102009,
100013.9122693769895704,100037.6241887829528423,100061.3375133813387947,100085.0522430056298617,
100108.7683774893521331,100132.4859166661044583,100156.2048603695002384,100179.9252084332256345,
100203.6469606910104631,100227.3701169766427483,100251.0946771239396185,100274.8206409667764092,
100298.5480083390866639,100322.2767790748330299,100346.0069530080654658,100369.7385299728339305,
100393.4715098032756941,100417.2058923335425789,100440.9416773978882702,100464.6788648305519018,
100488.4174544658599189,100512.1574461381969741,100535.8988396819622722,100559.6416349316132255,
100583.3858317216945579,100607.1314298867364414,100630.8784292613709113,100654.6268296802445548,
100678.3766309780912707,100702.1278329896449577,100725.8804355497122742,100749.6344384931726381,
100773.3898416549054673,100797.1466448698629392,100820.9048479730699910,100844.6644507995661115,
100868.4254531844344456,100892.1878549628454493,100915.9516559699841309,100939.7168560410937062,
100963.4834550114610465,100987.2514527164457832,101011.0208489914220991,101034.7916436718223849,

101058.5638365931663429,101082.3374275909445714,101106.1124165007786360,101129.8888031582755502,
101153.6665873991150875,101177.4457690590497805,101201.2263479738321621,101225.0083239792875247,
101248.7916969112993684,101272.5764666057948489,101296.3626328987302259,101320.1501956261199666,
101343.9391546240367461,101367.7295097285968950,101391.5212607759603998,101415.3144076023309026,
101439.1089500439702533,101462.9048879371985095,101486.7022211183357285,101510.5009494238183834,
101534.3010726900829468,101558.1025907536240993,101581.9055034509801771,101605.7098106187622761,
101629.5155120935960440,101653.3226077121798880,101677.1310973112558713,101700.9409807275951607,
101724.7522577980416827,101748.5649283594684675,101772.3789922488067532,101796.1944493030314334,
101820.0112993591610575,101843.8295422542723827,101867.6491778254858218,101891.4702059099654434,
101915.2926263449189719,101939.1164389676268911,101962.9416436153696850,101986.7682401255297009,
102010.5962283354892861,102034.4256080827180995,102058.2563792047149036,102082.0885415390075650,
102105.9220949231967097,102129.7570391949411714,102153.5933741919143358,102177.4310997518477961,
102201.2702157125313533,102225.1107219117984641,102248.9526181875262409,102272.7959043776354520,
102296.6405803200905211,102320.4866458529286319,102344.3341008142160717,102368.1829450420482317,
102392.0331783746078145,102415.8848006500775227,102439.7378117067273706,102463.5922113828710280,
102487.4479995168367168,102511.3051759470399702,102535.1637405118963216,102559.0236930499231676,
102582.8850333996524569,102606.7477613996743457,102630.6118768885935424,102654.4773797051166184,
102678.3442696879501455,102702.2125466758734547,102726.0822105077095330,102749.9532610223104712,
102773.8256980586011196,102797.6995214555354323,102821.5747310521255713,102845.4513266874273540,
102869.3293082005111501,102893.2086754305637442,102917.0894282167428173,102940.9715663983079139,
102964.8550898145476822,102988.7399983047798742,103012.6262917083950015,103036.5139698648126796,
103060.4030326135107316,103084.2934797939960845,103108.1853112458484247,103132.0785268086619908,
103155.9731263221183326,103179.8691096258990001,103203.7664765597728547,103227.6652269635233097,
103251.5653606770065380,103275.4668775401078165,103299.3697773927560775,103323.2740600749530131,
103347.1797254267003154,103371.0867732881015399,103394.9952034992456902,103418.9050159003381850,
103442.8162103315698914,103466.7287866331898840,103490.6427446455345489,103514.5580842089402722,
103538.4748051638016477,103562.3929073505714769,103586.3123906097316649,103610.2332547818368766,
103634.1554997074417770,103658.0791252271883423,103682.0041311817622045,103705.9305174118780997,
103729.8582837582944194,103753.7874300618277630,103777.7179561633383855,103801.6498619037156459,
103825.5831471239362145,103849.5178116649767617,103873.4538553679012693,103897.3912780737591675,
103921.3300796237163013,103945.2702598589385161,103969.2118186206644168,103993.1547557501326082,
104017.0990710886981105,104041.0447644777013920,104064.9918357585411286,104088.9402847726887558,
104112.8901113616302609,104136.8413153669243911,104160.7938966301444452,104184.7478549929510336,
104208.7031902969902148,104232.6599023840099107,104256.6179910957725951,104280.5774562741135014,
104304.5382977608678630,104328.5005153979727766,104352.4641090273653390,104376.4290784910408547,
104400.3954236310528358,104424.3631442894838983,104448.3322403084603138,104472.3027115301956655,
104496.2745577968889847,104520.2477789507975103,104544.2223748342803447,104568.1983452896529343,
104592.1756901593616931,104616.1544092858239310,104640.1345025115588214,104664.1159696791000897,
104688.0988106310251169,104712.0830252099840436,104736.0686132586415624,104760.0555746197205735,
104784.0439091360021848,104808.0336166502675042,104832.0246970054140547,104856.0171500443102559,
104880.0109756099118385,104904.0061735452181892,104928.0027436932577984,104952.0006858971319161,
104975.9999999999272404,105000.0006858448614366,105024.0027432751085144,105048.0061721339588985,
105072.0109722647030139,105096.0171435107040452,105120.0246857153397286,105144.0335987220751122,
105168.0438823743606918,105192.0555365157488268,105216.0685609898209805,105240.0829556401731679,
105264.0987203104741639,105288.1158548444363987,105312.1343590858159587,105336.1542328783980338,
105360.1754760660405736,105384.1980884926160797,105408.2220700020552613,105432.2474204383324832,
105456.2741396454657661,105480.3022274675313383,105504.3316837486345321,105528.3625083329097833,
105552.3947010645788396,105576.4282617878634483,105600.4631903470581165,105624.4994865864864551,
105648.5371503505302826,105672.5761814836005215,105696.6165798301663017,105720.6583452347258572,
105744.7014775418356294,105768.7459765960957156,105792.7918422421353171,105816.8390743246418424,
105840.8876726883463562,105864.9376371780090267,105888.9889676384627819,105913.0416639145551017,
105937.0957258511916734,105961.1511532933218405,105985.2079460859531537,106009.2661040740931639,
106033.3256271028367337,106057.3865150172932772,106081.4487676626595203,106105.5123848841176368,

106129.5773665269516641,106153.6437124364310876,106177.7114224579127040,106201.7804964367969660,
106225.8509342184843263,106249.9227356484771008,106273.9959005722776055,106298.0704288354609162,
106322.1463202836166602,106346.2235747624072246,106370.3021921175095486,106394.3821721946878824,
106418.4635148396919249,106442.5462198983732378,106466.6302872165833833,106490.7157166402466828,
106514.8025080152874580,106538.8906611877464456,106562.9801760036352789,106587.0710523090529023,
106611.1632899501128122,106635.2568887730158167,106659.3518486239481717,106683.4481693491834449,
106707.5458507950243074,106731.6448928078170866,106755.7452952339517651,106779.8470579198619816,
106803.9501807120104786,106828.0546634569327580,106852.1605060011788737,106876.2677081913716393,
106900.3762698741338681,106924.4861908961902373,106948.5974711042508716,106972.7101103451132076,
106996.8241084655892337,107020.9394653125345940,107045.0561807328776922,107069.1742545735614840,
107093.2936866815871326,107117.4144769039703533,107141.5366250878141727,107165.6601310802361695,
107189.7849947283975780,107213.9112158795178402,107238.0387943808455020,107262.1677300796727650,
107286.2980228233354865,107310.4296724592277315,107334.5626788347581169,107358.6970417974080192,
107382.8327611946733668,107406.9698368741228478,107431.1082686833542539,107455.2480564699944807,
107479.3892000817140797,107503.5316993662709137,107527.6755541713937419,107551.8207643449277384,
107575.9673297346889740,107600.1152501886099344,107624.2645255546085536,107648.4151556806609733,
107672.5671404148015426,107696.7204796050791629,107720.8751730996300466,107745.0312207465758547,
107769.1886223941401113,107793.3473778905317886,107817.5074870840326184,107841.6689498229825404,
107865.8317659557214938,107889.9959353306767298,107914.1614577962900512,107938.3283332010469167,
107962.4965613934764406,107986.6661422221659450,108010.8370755357173039,108035.0093611828197027,
108059.1829990121477749,108083.3579888724489138,108107.5343306125287199,108131.7120240812073462,
108155.8910691273486009,108180.0714655998745002,108204.2532133477507159,108228.4363122199574718,
108252.6207620655331993,108276.8065627335745376,108300.9937140732072294,108325.1822159335861215,
108349.3720681639388204,108373.5632706134929322,108397.7558231315633748,108421.9497255674650660,
108446.1449777706002351,108470.3415795903565595,108494.5395308762235800,108518.7388314776908373,
108542.9394812443206320,108567.1414800256752642,108591.3448276713897940,108615.5495240311429370,
108639.7555689546425128,108663.9629622916545486,108688.1717038919596234,108712.3817936054110760,
108736.5932312818767969,108760.8060167712828843,108785.0201499235845404,108809.2356305888097268,
108833.4524586169864051,108857.6706338582152966,108881.8901561626116745,108906.1110253803635715,
108930.3332413616735721,108954.5568039568170207,108978.7817130160692614,109003.0079683897638461,
109027.2355699283070862,109051.4645174821052933,109075.6948109016229864,109099.9264500373683404,
109124.1594347398931859,109148.3937648597639054,109172.6294402476341929,109196.8664607541577425,
109221.1048262300610077,109245.3445365260995459,109269.5855914930434665,109293.8279909817501903,
109318.0717348431062419,109342.3168229280126980,109366.5632550874288427,109390.8110311723721679,
109415.0601510338747175,109439.3106145230121911,109463.5624214909330476,109487.8155717888002982,
109512.0700652678060578,109536.3259017792006489,109560.5830811742926016,109584.8416033044049982,
109609.1014680209045764,109633.3626751752162818,109657.6252246187796118,109681.8891162030922715,
109706.1543497797101736,109730.4209252002037829,109754.6888423161872197,109778.9581009793182602,
109803.2287010412983363,109827.5006423538870877,109851.7739247688441537,109876.0485481380019337,
109900.3245123132364824,109924.6018171464384068,109948.8804624895710731,109973.1604481945978478,
109997.4417741135694087,110021.7244400985509856,110046.0084460016369121,110070.2937916750088334,
110094.5804769708192907,110118.8685017413372407,110143.1578658388170879,110167.4485691155714449,
110191.7406114239711314,110216.0339926163869677,110240.3287125452770852,110264.6247710631141672,
110288.9221680224000011,110313.2209032757091336,110337.5209766756452154,110361.8223880748264492,
110386.1251373259437969,110410.4292242817173246,110434.7346487949107541,110459.0414107183169108,
110483.3495099047868280,110507.6589462071860908,110531.9697194784675958,110556.2818295715696877,
110580.5952763395034708,110604.9100596353237052,110629.2261793120851507,110653.5436352229444310,
110677.8624272210581694,110702.1825551596266450,110726.5040188918792410,110750.8268182711326517,
110775.1509531507035717,110799.4764233839523513,110823.8032288242829964,110848.1313693251431687,
110872.4608447400387377,110896.7916549224755727,110921.1237997260323027,110945.4572790043021087,
110969.7920926109654829,110994.1282403996738140,111018.4657222241658019,111042.8045379382238025,
111067.1446873956592754,111091.4861704502836801,111115.8289869560248917,111140.1731367667962331,
111164.5186197365692351,111188.8654357193299802,111213.2135845691664144,111237.5630661401373800,

111261.9138802863890305,111286.2660268620675197,111310.6195057214063127,111334.9743167186388746,
111359.3304597080423264,111383.6879345439665485,111408.0467410807759734,111432.4068791728641372,
111456.7683486746827839,111481.1311494407273130,111505.4952813255076762,111529.8607441836065846,
111554.2275378696358530,111578.5956622382072965,111602.9651171440491453,111627.3359024418459740,
111651.7080179863987723,111676.0814636324794265,111700.4562392349616857,111724.8323446487047477,
111749.2097797286405694,111773.5885443297447637,111797.9686383069929434,111822.3500615154480329,
111846.7328138101875084,111871.1168950463325018,111895.5023050790332491,111919.8890437634981936,
111944.2771109549503308,111968.6665065086999675,111993.0572302800428588,112017.4492821243475191,
112041.8426618969970150,112066.2373694534326205,112090.6334046491247136,112115.0307673396164319,
112139.4294573804218089,112163.8294746271567419,112188.2308189354516799,112212.6334901609661756,
112237.0374881594325416,112261.4428127865830902,112285.8494638982228935,112310.2574413501715753,
112334.6667449982924154,112359.0773746984923491,112383.4893303067219676,112407.9026116789755179,
112432.3172186712617986,112456.7331511396478163,112481.1504089402442332,112505.5689919291908154,
112529.9888999626564328,112554.4101328968827147,112578.8326905881112907,112603.2565728926274460,
112627.6817796667892253,112652.1083107669837773,112676.5361660495836986,112700.9653453710780013,
112725.3958485879411455,112749.8276755567203509,112774.2608261339773890,112798.6953001763031352,
112823.1310975403757766,112847.5682180828589480,112872.0066616604890442,112896.4464281300315633,
112920.8875173482665559,112945.3299291720759356,112969.7736634583125124,112994.2187200638873037,
113018.6650988457840867,113043.1127996609866386,113067.5618223665223923,113092.0121668194769882,
113116.4638328769506188,113140.9168203961016843,113165.3711292341322405,113189.8267592482443433,
113214.2837102957273601,113238.7419822338706581,113263.2015749200218124,113287.6624882115720538,
113312.1247219659271650,113336.5882760405511362,113361.0531502929516137,113385.5193445806653472,
113409.9868587612436386,113434.4556926923251012,113458.9258462315483484,113483.3973192366102012,
113507.8701115652220324,113532.3442230751679745,113556.8196536242467118,113581.2964030703005847,
113605.7744712712155888,113630.2538580849068239,113654.7345633693330456,113679.2165869824966649,
113703.6999287824146450,113728.1845886271767085,113752.6705663748725783,113777.1578618836792884,
113801.6464750117738731,113826.1364056173624704,113850.6276535587385297,113875.1202186941809487,
113899.6141008820413845,113924.1092999807005981,113948.6058158485539025,113973.1036483440693701,
113997.6027973257441772,114022.1032626521046041,114046.6050441817060346,114071.1081417731766123,
114095.6125552851444809,114120.1182845762814395,114144.6253295053320471,114169.1336899310408626,
114193.6433657121961005,114218.1543567076441832,114242.6666627762606367,114267.1802837769355392,
114291.6952195686171763,114316.2114700103120413,114340.7290349610266276,114365.2479142798256362,
114389.7681078258028720,114414.2896154580957955,114438.8124370358855231,114463.3365724183677230,
114487.8620214647962712,114512.3887840344832512,114536.9168599867116427,114561.4462491808662890,
114585.9769514763611369,114610.5089667325955816,114635.0422948090854334,114659.5769355653173989,
114684.1128888608363923,114708.6501545552600874,114733.1887325081770541,114757.7286225792777259,
114782.2698246282525361,114806.8123385148355737,114831.3561640988191357,114855.9013012399809668,
114880.4477497982152272,114904.9955096333724214,114929.5445806054049172,114954.0949625742505305,
114978.6466553999198368,115003.1996589424670674,115027.7539730619464535,115052.3095976184704341,
115076.8665324721805518,115101.4247774832911091,115125.9843325120018562,115150.5451974185853032,
115175.1073720633430639,115199.6708563065913040,115224.2356500087335007,115248.8017530301440274,
115273.3691652312991209,115297.9378864726604661,115322.5079166147625074,115347.0792555181687931,
115371.6519030434719753,115396.2258590512938099,115420.8011234023142606,115445.3776959572423948,
115469.9555765768163837,115494.5347651218180545,115519.1152614530583378,115543.6970654314063722,
115568.2801769177604001,115592.8645957730332157,115617.4503218581958208,115642.0373550342483213,
115666.6256951622344786,115691.2153421032271581,115715.8062957183574326,115740.3985558687563753,
115764.9921224156278186,115789.5869952201755950,115814.1831741436762968,115838.7806590474356199,
115863.3794497927592602,115887.9795462410547771,115912.5809482537006261,115937.1836556921625743,
115961.7876684179209406,115986.3929862924851477,116010.9996091774228262,116035.6075369343161583,
116060.2167694247909822,116084.8273065105313435,116109.4391480532212881,116134.0522939146030694,
116158.6667439564625965,116183.2824980406003306,116207.8995560288603883,116232.5179177831596462,
116257.1375831653858768,116281.7585520375141641,116306.3808242615341442,116331.0043996994936606,
116355.6292782134260051,116380.2554596654663328,116404.8829439177497989,116429.5117308324552141,

116454.1418202717759414,116478.7732120979926549,116503.4059061733714771,116528.0399023602512898,
116552.6752005209709750,116577.3118005179421743,116601.9497022135910811,116626.5889054704020964,
116651.2294101508596214,116675.8712161175062647,116700.5143232329428429,116725.1587313597556204,
116749.8044403606036212,116774.4514500981749734,116799.0997604352014605,116823.7493712344294181,
116848.4002823586633895,116873.0524936707224697,116897.7060050334839616,116922.3608163098397199,
116947.0169273627543589,116971.6743380551779410,116996.3330482501332881,117020.9930578106723260,
117045.6543665998760844,117070.3169744808546966,117094.9808813167765038,117119.6460869708389509,
117144.3125913062394829,117168.9803941862919601,117193.6494954742520349,117218.3198950334772235,
117242.9915927273250418,117267.6645884192257654,117292.3388819726242218,117317.0144732509652385,
117341.6913621177955065,117366.3695484366617166,117391.0490320711542154,117415.7298128848779015,
117440.4118907415249851,117465.0952655047440203,117489.7799370383145288,117514.4659052059723763,
117539.1531698715261882,117563.8417308988282457,117588.5315881517308298,117613.2227414941589814,
117637.9151907900668448,117662.6089359034085646,117687.3039766982110450,117712.0003130385302939,
117736.6979447884659749,117761.3968718121177517,117786.0970939736725995,117810.7986111373029416,
117835.5014231672539609,117860.2055299277708400,117884.9109312831860734,117909.6176270978030516,
117934.3256172360270284,117959.0349015622487059,117983.7454799409169937,118008.4573522365099052,
118033.1705183135345578,118057.8849780365562765,118082.6007312701403862,118107.3177778789249714,
118132.0361177275772206,118156.7557506807643222,118181.4766766032116720,118206.1988953597174259,
118230.9224068150360836,118255.6472108340385603,118280.3733072815666674,118305.1006960225349758,
118329.8293769218726084,118354.5593498445814475,118379.2906146556488238,118404.0231712201202754,
118428.7570194030849962,118453.4921590696467320,118478.2285900849819882,118502.9663123142381664,
118527.7053256226790836,118552.4456298755249009,118577.1872249380830908,118601.9301106756756781,
118626.6742869536828948,118651.4197536374704214,118676.1665105925058015,118700.9145576842129230,
118725.6638947781320894,118750.4145217397890519,118775.1664384347532177,118799.9196447286230978,
118824.6741404870554106,118849.4299255757359788,118874.1869998603506247,118898.9453632066579303,
118923.7050154804310296,118948.4659565475158161,118973.2281862737436313,118997.9917045249894727,
119022.7565111672010971,119047.5226060663262615,119072.2899890883418266,119097.0586600992974127,
119121.8286189652280882,119146.5998655522562331,119171.3723997265042271,119196.1462213541235542,
119220.9213303013384575,119245.6977264343586285,119270.4754096194665181,119295.2543797229736811,
119320.0346366112062242,119344.8161801505484618,119369.5990102073992603,119394.3831266482156934,
119419.1685293394693872,119443.9552181476610713,119468.7431929393496830,119493.5324535811232636,
119518.3229999395844061,119543.1148318813939113,119567.9079492732416838,119592.7023519818321802,
119617.4980398739280645,119642.2950128163211048,119667.0932706758321729,119691.8928133193112444,
119716.6936406136665028,119741.4957524258352350,119766.2991486227401765,119791.1038290714059258,
119815.9097936388716334,119840.7170421921764500,119865.5255745984322857,119890.3353907247656025,
119915.1464904383610701,119939.9588736064179102,119964.7725400961498963,119989.5874897748581134,
120014.4037225098290946,120039.2212381684075808,120064.0400366179674165,120088.8601177259115502,
120113.6814813597011380,120138.5041273867827840,120163.3280556747049559,120188.1532660909870174,
120212.9797585032065399,120237.8075327789993025,120262.6365887860010844,120287.4669263918767683,
120312.2985454643639969,120337.1314458712149644,120361.9656274801964173,120386.8010901591333095,
120411.6378337758942507,120436.4758581983478507,120461.3151632944209268,120486.1557489320694003,
120510.9976149792782962,120535.8407613040617434,120560.6851877745066304,120585.5308942586707417,
120610.3778806246991735,120635.2261467407370219,120660.0756924749875907,120684.9265176956832875,
120709.7786222710565198,120734.6320060694415588,120759.4866689591290196,120784.3426108084968291,
120809.1998314859520178,120834.0583308599161683,120858.9181087988399668,120883.7791651712468592,
120908.6414998456457397,120933.5051126906037098,120958.3700035747315269,120983.2361723666545004,
121008.1036189350415953,121032.9723431486054324,121057.8423448760586325,121082.7136239861720242,
121107.5861803477455396,121132.4600138296373188,121157.3351243006909499,121182.2115116298227804,
121207.0891756859491579,121231.9681163380737416,121256.8483334551710868,121281.7298269062885083,
121306.6125965604878729,121331.4966422868892550,121356.3819639546127291,121381.2685614328511292,
121406.1564345907827374,121431.0455832976731472,121455.9360074227588484,121480.8277068353927461,
121505.7206814048695378,121530.6149310005712323,121555.5104554919234943,121580.4072547483519884,
121605.3053286393260350,121630.2046770343586104,121655.1052998029917944,121680.0071968147822190,

121704.9103679393447237,121729.8148130463232519,121754.7205320053908508,121779.6275246862351196,
121804.5357909586164169,121829.4453306923096534,121854.3561437571042916,121879.2682300228480017,
121904.1815893594030058,121929.0962216366897337,121954.0121267246431671,121978.9293044932273915,
122003.8477548124647001,122028.7674775523773860,122053.6884725830313982,122078.6107397745508933,
122103.5342789970745798,122128.4590901207557181,122153.3851730158057762,122178.3125275524653262,
122203.2411536009894917,122228.1710510317207081,122253.1022197149432031,122278.0346595210721716,
122302.9683703204937046,122327.9033519836229971,122352.8396043809625553,122377.7771273830003338,
122402.7159208602533909,122427.6559846833260963,122452.5973187227791641,122477.5399228492751718,
122502.4837969334766967,122527.4289408460608684,122552.3753544577921275,122577.3230376394203631,
122602.2719902617245680,122627.2222121955710463,122652.1737033118115505,122677.1264634813269367,
122702.0804925750562688,122727.0357904639677145,122751.9923570190585451,122776.9501921113551361,
122801.9092956118984148,122826.8696673918020679,122851.8313073221943341,122876.7942152742180042,
122901.7583911190886283,122926.7238347279926529,122951.6905459722329397,122976.6585247230686946,
123001.6277708518173313,123026.5982842298690230,123051.5700647285702871,123076.5431122193695046,
123101.5174265737005044,123126.4930076630698750,123151.4698553589842049,123176.4479695329937385,
123201.4273500566778239,123226.4079968016594648,123251.3899096395907691,123276.3730884421383962,
123301.3575330810272135,123326.3432434279966401,123351.3302193548297510,123376.3184607333387248,
123401.3079674353502924,123426.2987393327639438,123451.2907762974791694,123476.2840782014391152,
123501.2786449166014791,123526.2744763149821665,123551.2715722686261870,123576.2699326495931018,
123601.2695573299861280,123626.2704461819521384,123651.2725990776380058,123676.2760158892633626,
123701.2806964890332893,123726.2866407492401777,123751.2938485421764199,123776.3023197401635116,
123801.3120542155520525,123826.3230518407508498,123851.3353124881687108,123876.3488360302872024,
123901.3636223395733396,123926.3796712885523448,123951.3969827497930964,123976.4155565958790248,
124001.4353926994081121,124026.4564909330511000,124051.4788511694787303,124076.5024732814199524,
124101.5273571416037157,124126.5535026228171773,124151.5809095978765981,124176.6095779396127909,
124201.6395075209147763,124226.6706982146715745,124251.7031498938449658,124276.7368624313821783,
124301.7718357003031997,124326.8080695736425696,124351.8455639244639315,124376.8843186258454807,
124401.9243335509527242,124426.9656085729220649,124452.0081435649772175,124477.0519384003127925,
124502.0969929521961603,124527.1433070939237950,124552.1908806988212746,124577.2397136402287288,
124602.2898057915444952,124627.3411570261814632,124652.3937672175961779,124677.4476362392597366,
124702.5027639646868920,124727.5591502674215008,124752.6167950210365234,124777.6756980991631281,
124802.7358593754179310,124827.7972787234757561,124852.8599560170405312,124877.9238911298598396,
124902.9890839356812648,124928.0555343083105981,124953.1232421215972863,124978.1922072493762244,
125003.2624295655405149,125028.3339089440414682,125053.4066452588158427,125078.4806383838440524,
125103.5558881931792712,125128.6323945608310169,125153.7101573609106708,125178.7891764675150625,
125203.8694517548137810,125228.9509830969618633,125254.0337703681725543,125279.1178134427027544,
125304.2031121947948122,125329.2896664987783879,125354.3774762289831415,125379.4665412597678369,
125404.5568614655203419,125429.6484367207012838,125454.7412668997421861,125479.8353518771473318,
125504.9306915274355561,125530.0272857251693495,125555.1251343449257547,125580.2242372613254702,
125605.3245943490182981,125630.4262054826976964,125655.5290705370425712,125680.6331893868191401,
125705.7385619067936204,125730.8451879717758857,125755.9530674566049129,125781.0622002361342311,
125806.1725861852755770,125831.2842251789552392,125856.3971170921286102,125881.5112617998092901,
125906.6266591769817751,125931.7433090987469768,125956.8612114401621511,125981.9803660763427615,
126007.1007728824479273,126032.2224317336513195,126057.3453425051702652,126082.4695050722511951,
126107.5949193101405399,126132.7215850941720419,126157.8495022996648913,126182.9786708019819343,
126208.1090904765296727,126233.2407611987146083,126258.3736828440305544,126283.5078552879422205,
126308.6432784059870755,126333.7799520736880368,126358.9178761666698847,126384.0570505604991922,
126409.1974751308589475,126434.3391497534030350,126459.4820743038580986,126484.6262486579362303,
126509.7716726914222818,126534.9183462801011046,126560.0662692998157581,126585.2154416264238534,
126610.3658631358121056,126635.5175337039254373,126660.6704532066796673,126685.8246215200779261,
126710.9800385201378958,126736.1367040829063626,126761.2946180844446644,126786.4537804008723469,
126811.6141909083380597,126836.7758494829759002,126861.9387560010218294,126887.1029103386827046,
126912.2683123722381424,126937.4349619779532077,126962.6028590321657248,126987.7720034112280700,

127012.9423949915217236,127038.1140336494718213,127063.2869192615034990,127088.4610517040855484,
127113.6364308537595207,127138.8130565870233113,127163.9909287804766791,127189.1700473106902791,
127214.3504120542929741,127239.5320228879718343,127264.7148796883848263,127289.8989823322626762,
127315.0843306963506620,127340.2709246574377175,127365.4587640923418803,127390.6478488778811879,
127415.8381788909318857,127441.0297540084138745,127466.2225741072470555,127491.4166390643949853,
127516.6119487568503246,127541.8085030616493896,127567.0063018558284966,127592.2053450164676178,
127617.4056324207049329,127642.6071639456786215,127667.8099394685559673,127693.0139588665333577,
127718.2192220168799395,127743.4257287968212040,127768.6334790836845059,127793.8424727547826478,
127819.0527096874866402,127844.2641897591674933,127869.4769128472398734,127894.6908788291766541,
127919.9060875824216055,127945.1225389845203608,127970.3402329129748978,127995.5591692453745054,
128020.7793478593230247,128046.0007686324388487,128071.2234314423694741,128096.4473361668351572,
128121.6724826835270505,128146.8988708702090662,128172.1265006046451163,128197.3553717646718724,
128222.5854842280969024,128247.8168378728150856,128273.0494325767067494,128298.2832682177104289,
128323.5183446737937629,128348.7546618229389424,128373.9922195431572618,128399.2310177125182236,
128424.4710562090913299,128449.7123349109751871,128474.9548536963120569,128500.1986124432878569,
128525.4436110300885048,128550.6898493349435739,128575.9373272361262934,128601.1860446119098924,
128626.4360013406112557,128651.6871973005909240,128676.9396323702094378,128702.1933064279000973,
128727.4482193520816509,128752.7043710212456062,128777.9617613138543675,128803.2203901084576501,
128828.4802572836197214,128853.7413627179194009,128879.0037062899646116,128904.2672878784069326,
128929.5321073619270464,128954.7981646192347398,128980.0654595290543512,129005.3339919701538747,
129030.6037618213449605,129055.8747689614247065,129081.1470132692629704,129106.4204946237587137,
129131.6952129037963459,129156.9711679883475881,129182.2483597563550575,129207.5267880868486827,
129232.8064528588438407,129258.0873539514141157,129283.3694912436476443,129308.6528646146616666,
129333.9374739436025266,129359.2233191096602241,129384.5103999920247588,129409.7987164699734421,
129435.0882684227399295,129460.3790557296306361,129485.6710782699810807,129510.9643359231413342,
129536.2588285685051233,129561.5545560854807263,129586.8515183535200777,129612.1497152520896634,
129637.4491466606996255,129662.7498124588746578,129688.0517125261976616,129713.3548467422369868,
129738.6592149866337422,129763.9648171390290372,129789.2716530790930847,129814.5797226865543053,
129839.8890258411556715,129865.1995624226401560,129890.5113323108234908,129915.8243353855359601,
129941.1385715266223997,129966.4540406139713014,129991.7707425275002606,130017.0886771471559769,
130042.4078443528997013,130067.7282440247363411,130093.0498760427144589,130118.3727402868680656,
130143.6968366373184836,130169.0221649741579313,130194.3487251775513869,130219.6765171276638284,
130245.0055407047038898,130270.3357957889093086,130295.6672822605469264,130320.9999999999126885,
130346.3339488873316441,130371.6691288031433942,130397.0055396277311957,130422.3431812415219611,
130447.6820535249426030,130473.0221563584491378,130498.3634896225557895,130523.7060531977913342,
130549.0498469646990998,130574.3948708038806217,130599.7411245959228836,130625.0886082214856287,
130650.4373215612286003,130675.7872644958697492,130701.1384369061270263,130726.4908386727474863,
130751.8444696765363915,130777.1993297983135562,130802.5554189188987948,130827.9127369191846810,
130853.2712836800637888,130878.6310590824723477,130903.9920630073756911,130929.3542953357537044,
130954.7177559486153768,130980.0824447270279052,131005.4483615520584863,131030.8155063048179727,
131056.1838788664172171,131081.5534791180398315,131106.9243069408694282,131132.2963622161187232,
131157.6696448250440881,131183.0441546489018947,131208.4198915690358263,131233.7968554667604621,
131259.1750462234194856,131284.5544637204147875,131309.9351078391773626,131335.3169784611673094,
131360.7000754678447265,131386.0843987406988163,131411.4699481612769887,131436.8567236111557577,
131462.2447249719407409,131487.6339521252084523,131513.0244049526227172,131538.4160833358764648,
131563.8089871566335205,131589.2031162967032287,131614.5984706377785187,131639.9950500616978388,
131665.3928544502414297,131690.7918836853059474,131716.1921376487298403,131741.5936162224097643,
131766.9963192883005831,131792.4002467283862643,131817.8053984246216714,131843.2117742590489797,
131868.6193741137103643,131894.0281978706480004,131919.4382454120204784,131944.8495166199281812,
131970.2620113765588030,131995.6757295640709344,132021.0906710646813735,132046.5068357606651261,
132071.9242235342971981,132097.3428342678234912,132122.7626678436645307,132148.1837241440953221,
132173.6060030515654944,132199.0295044484664686,132224.4542282172187697,132249.8801742403593380,
132275.3073424003086984,132300.7357325796619989,132326.1653446609561797,132351.5961785267572850,

132377.0282340596895665,132402.4615111424354836,132427.8960096575901844,132453.3317294878943358,
132478.7686705160886049,132504.2068326249136589,132529.6462156971392687,132555.0868196155934129,
132580.5286442631040700,132605.9716895225283224,132631.4159552767814603,132656.8614414088078775,
132682.3081478014937602,132707.7560743378708139,132733.2052209009416401,132758.6555873737379443,
132784.1071736392914318,132809.5599795807502232,132835.0140050811751280,132860.4692500237724744,
132885.9257142916612793,132911.3833977681060787,132936.8423003362840973,132962.3024218794598710,
132987.7637622809561435,133013.2263214240665548,133038.6900991921429522,133064.1550954685371835,
133089.6213101366593037,133115.0887430799484719,133140.5573941818438470,133166.0272633258136921,
133191.4983503954135813,133216.9706552741408814,133242.4441778456093743,133267.9189179933455307,
133293.3948756010213401,133318.8720505522796884,133344.3504427307925653,133369.8300520202319603,
133395.3108783043862786,133420.7929214670148212,133446.2761813918477856,133471.7606579627608880,
133497.2463510635716375,133522.7332605781557504,133548.2213863903889433,133573.7107283842633478,
133599.2012864436546806,133624.6930604526132811,133650.1860502951021772,133675.6802558551717084,
133701.1756770169013180,133726.6723136643704493,133752.1701656817167532,133777.6692329530487768,
133803.1695153625914827,133828.6710127945407294,133854.1737251330923755,133879.6776522625295911,
133905.1827940671355464,133930.6891504312516190,133956.1967212391609792,133981.7055063752632122,
134007.2155057239870075,134032.7267191697028466,134058.2391465968976263,134083.7527878900000360,
134109.2676429335842840,134134.7837116121663712,134160.3009938102622982,134185.8194894125044812,
134211.3391983035253361,134236.8601203678990714,134262.3822554903745186,134287.9056035555840936,
134313.4301644483057316,134338.9559380532591604,134364.4829242552514188,134390.0111229390895460,
134415.5405339895514771,134441.0711572915897705,134466.6029927300114650,134492.1360401897982229,
134517.6702995558443945,134543.2057707131607458,134568.7424535467580426,134594.2803479415888432,
134619.8194537827803288,134645.3597709553723689,134670.9012993445212487,134696.4440388352959417,
134721.9879893129109405,134747.5331506625225302,134773.0795227693743072,134798.6271055187098682,
134824.1758987957728095,134849.7259024858940393,134875.2771164743753616,134900.8295406466058921,
134926.3831748879165389,134951.9380190837255213,134977.4940731195092667,135003.0513368806859944,
135028.6098102527612355,135054.1694931212696247,135079.7303853717166930,135105.2924868896661792,
135130.8557975607691333,135156.4203172706183977,135181.9860459048359189,135207.5529833491600584,
135233.1211294892418664,135258.6904842108779121,135284.2610473997483496,135309.8328189416788518,
135335.4057987224950921,135360.9799866280227434,135386.5553825441165827,135412.1319863566895947,
135437.7097979516838677,135463.2888172149832826,135488.8690440326172393,135514.4504782905569300,
135540.0331198748608585,135565.6169686715584248,135591.2020245667663403,135616.7882874465431087,
135642.3757571970636491,135667.9644337044737767,135693.5543168549775146,135719.1454065347497817,
135744.7377026301110163,135770.3312050272652414,135795.9259136125328951,135821.5218282722344156,
135847.1189488927193452,135872.7172753603954334,135898.3168075616122223,135923.9175453828356694,
135949.5194887105317321,135975.1226374311372638,136000.7269914312346373,136026.3325505972898100,
136051.9393148159142584,136077.5472839736903552,136103.1564579572295770,136128.7668366531725042,
136154.3784199482179247,136179.9912077290064190,136205.6051998823240865,136231.2203962948988192,
136256.8367968534876127,136282.4544014449347742,136308.0732099560264032,136333.6932222736650147,
136359.3144382846949156,136384.9368578760768287,136410.5604809346841648,136436.1853073475358542,
136461.8113370015635155,136487.4385697838442866,136513.0670055814262014,136538.6966442812990863,
136564.3274857706564944,136589.9595299365464598,136615.5927766661625355,136641.2272258466691710,
136666.8628773652308155,136692.4997311091574375,136718.1377869656425901,136743.7770448219671380,
136769.4175045654701535,136795.0591660834616050,136820.7020292633096687,136846.3460939924407285,
136871.9913601581938565,136897.6378276480536442,136923.2854963494755793,136948.9343661499733571,
136974.5844369370606728,137000.2357085982512217,137025.8881810211460106,137051.5418540933169425,
137077.1967277024232317,137102.8528017360949889,137128.5100760820205323,137154.1685506278590765,
137179.8282252614153549,137205.4890998704067897,137231.1511743425799068,137256.8144485657976475,
137282.4789224278938491,137308.1445958166732453,137333.8114686200860888,137359.4795407260244247,
137385.1488120224094018,137410.8192823971912730,137436.4909517384367064,137462.1638199340668507,
137487.8378868722065818,137513.5131524408934638,137539.1896165282232687,137564.8672790222917683,
137590.5461398113111500,137616.2261987833771855,137641.9074558267602697,137667.5899108296725899,
137693.2735636803263333,137718.9584142670209985,137744.6444624780851882,137770.3317082018184010,

137796.0201513266074471,137821.7097917408100329,137847.4006293328420725,137873.0926639911485836,
137898.7858956041745842,137924.4803240604232997,137950.1759492483688518,137975.8727710566017777,
138001.5707893736544065,138027.2700040881463792,138052.9704150886391290,138078.6720222638396081,
138104.3748255023965612,138130.0788246929587331,138155.7840197243203875,138181.4904104851593729,
138207.1979968642990571,138232.9067787505045999,138258.6167560326284729,138284.3279285994940437,
138310.0402963399537839,138335.7538591429765802,138361.4686168974440079,138387.1845694923249539,
138412.9017168165883049,138438.6200587592320517,138464.3395952093123924,138490.0603260558564216,
138515.7822511879785452,138541.5053704947349615,138567.2296838652982842,138592.9551911888411269,
138618.6818923545069993,138644.4097872515267227,138670.1388757691311184,138695.8691577965801116,
138721.6006332231627312,138747.3333019382262137,138773.0671638310595881,138798.8022187910391949,
138824.5384667075704783,138850.2759074700588826,138876.0145409679389559,138901.7543670907034539,
138927.4953857278160285,138953.2375967687985394,138978.9810001032019500,139004.7255956206063274,
139030.4713832105917390,139056.2183627627964597,139081.9665341668296605,139107.7158973123878241,
139133.4664520891674329,139159.2181983868940733,139184.9711360953224357,139210.7252651042072102,
139236.4805853033612948,139262.2370965825684834,139287.9947988317580894,139313.7536919407139067,
139339.5137757994234562,139365.2750502977578435,139391.0375153256754857,139416.8011707731639035,
139442.5660165302397218,139468.3320524868904613,139494.0992785332200583,139519.8676945592742413,
139545.6373004551569466,139571.4080961110012140,139597.1800814169691876,139622.9532562632230110,
139648.7276205400121398,139674.5031741374987178,139700.2799169460195117,139726.0578488557948731,
139751.8369697571324650,139777.6172795404272620,139803.3987780959578231,139829.1814653141482268,
139854.9653410853934474,139880.7504053001466673,139906.5366578488610685,139932.3240986219898332,
139958.1127275100734551,139983.9025444036524277,140009.6935491932672448,140035.4857417694875039,
140061.2791220229410101,140087.0736898442846723,140112.8694451241462957,140138.6663877532118931,
140164.4645176221965812,140190.2638346218445804,140216.0643386429292150,140241.8660295762238093,
140267.6689073125307914,140293.4729717426816933,140319.2782227575371508,140345.0846602480160072,
140370.8922841049788985,140396.7010942194319796,140422.5110904822649900,140448.3222727844840847,
140474.1346410171245225,140499.9481950712215621,140525.7629348378104623,140551.5788602080137935,
140577.3959710728959180,140603.2142673236376140,140629.0337488513614517,140654.8544155473064166,
140680.6762673026241828,140706.4993040085828397,140732.3235255564504769,140758.1489318374951836,
140783.9755227430432569,140809.8032981644209940,140835.6322579929837957,140861.4624021201452706,
140887.2937304372899234,140913.1262428358604666,140938.9599392073287163,140964.7948194431664888,
140990.6308834348747041,141016.4681310740124900,141042.3065622521389741,141068.1461768608132843,
141093.9869747916818596,141119.8289559363620356,141145.6721201865002513,141171.5164674337720498,
141197.3619975699402858,141223.2087104866805021,141249.0566060757846572,141274.9056842290447094,
141300.7559448382235132,141326.6073877952003386,141352.4600129917962477,141378.3138203199196141,
141404.1688096714788117,141430.0249809383822139,141455.8823340125964023,141481.7408687861170620,
141507.6005851509398781,141533.4614829990896396,141559.3235622226493433,141585.1868227136437781,
141611.0512643642141484,141636.9168870665016584,141662.7836907126184087,141688.6516751947929151,
141714.5208404051954858,141740.3911862360546365,141766.2627125796279870,141792.1354193282022607,
141818.0093063740641810,141843.8843736095586792,141869.7606209270015825,141895.6380482187960297,
141921.5166553773451596,141947.3964422950521111,141973.2774088643782306,141999.1595549777848646,
142025.0428805277624633,142050.9273854068887886,142076.8130695076542906,142102.6999327226367313,
142128.5879749444429763,142154.4771960657089949,142180.3675959790416528,142206.2591745771351270,
142232.1519317526544910,142258.0458673983812332,142283.9409814069804270,142309.8372736712626647,
142335.7347440840094350,142361.6333925380604342,142387.5332189261971507,142413.4342231413174886,
142439.3364050763484556,142465.2397646241297480,142491.1443016776465811,142517.0500161298259627,
142542.9569078736822121,142568.8649768022296485,142594.7742228084825911,142620.6846457855135668,
142646.5962456263659988,142672.5090222242288291,142698.4229754721454810,142724.3381052633340005,
142750.2544114909251221,142776.1718940481659956,142802.0905528282746673,142828.0103877244691830,
142853.9313986300840043,142879.8535854383662809,142905.7769480426795781,142931.7014863363583572,
142957.6272002127661835,142983.5540895653248299,143009.4821542874269653,143035.4113942725525703,
143061.3418094141525216,143087.2733996057067998,143113.2061647407535929,143139.1401047128310893,
143165.0752194155065808,143191.0115087423764635,143216.9489725870371331,143242.8876108431431931,

143268.8274234043492470,143294.7684101643681061,143320.7105710168834776,143346.6539058556372765,
143372.5984145743714180,143398.5440970669151284,143424.4909532270103227,143450.4389829485735390,
143476.3881861253757961,143502.3385626513336319,143528.2901124203635845,143554.2428353263821919,
143580.1967312633350957,143606.1518001251970418,143632.1080418059718795,143658.0654561996925622,
143684.0240432003920432,143709.9838027021323796,143735.9447345990047324,143761.9068387851875741,
143787.8701151547429617,143813.8345636018784717,143839.8001840208016802,143865.7669763056910597,
143891.7349403508123942,143917.7040760504023638,143943.6743832987558562,143969.6458619901968632,
143995.6185120190493762,144021.5923332796664909,144047.5673256664595101,144073.5434890737815294,
144099.5208233960729558,144125.4993285278324038,144151.4790043635002803,144177.4598507975752000,
144203.4418677245848812,144229.4250550390861463,144255.4094126356358174,144281.3949404088489246,
144307.3816382533113938,144333.3695060637255665,144359.3585437346773688,144385.3487511609273497,
144411.3401282371778507,144437.3326748581312131,144463.3263909185479861,144489.3212763132469263,
144515.3173309370467905,144541.3145546847372316,144567.3129474511952139,144593.3125091312977020,
144619.3132396199507639,144645.3151388120604679,144671.3182066026201937,144697.3224428865651134,
144723.3278475588886067,144749.3344205146713648,144775.3421616488776635,144801.3510708566464018,
144827.3611480330291670,144853.3723930731357541,144879.3848058721341658,144905.3983863251633011,
144931.4131343274202663,144957.4290497741021682,144983.4461325604643207,145009.4643825817329343,
145035.4837997332215309,145061.5043839102145284,145087.5261350080545526,145113.5490529220842291,
145139.5731375476461835,145165.5983887801703531,145191.6248065150866751,145217.6523906478250865,
145243.6811410738446284,145269.7110576886334457,145295.7421403877378907,145321.7743890666752122,
145347.8078036209917627,145373.8423839462921023,145399.8781299381807912,145425.9150414922914933,
145451.9531185042578727,145477.9923608698009048,145504.0327684845833573,145530.0743412443553098,
145556.1170790448377375,145582.1609817818098236,145608.2060493510798551,145634.2522816484852228,
145660.2996785698051099,145686.3482400109642185,145712.3979658678290434,145738.4488560362951830,
145764.5009104123164434,145790.5541288918466307,145816.6085113708686549,145842.6640577453654259,
145868.7207679114071652,145894.7786417650058866,145920.8376792022609152,145946.8978801192424726,
145972.9592444120789878,145999.0217719769279938,146025.0854627099470235,146051.1503165073227137,
146077.2163332652708050,146103.2835128800361417,146129.3518552478926722,146155.4213602650561370,
146181.4920278279168997,146207.5638578327198047,146233.6368501758843195,146259.7110047537717037,
146285.7863214627432171,146311.8628001992765348,146337.9404408597620204,146364.0192433407064527,
146390.0992075385584030,146416.1803333498537540,146442.2626206711574923,146468.3460693989763968,
146494.4306794299336616,146520.5164506606233772,146546.6033829876396339,146572.6914763076929376,
146598.7807305174064822,146624.8711455135198776,146650.9627211927436292,146677.0554574517882429,
146703.1493541874515358,146729.2444112965022214,146755.3406286757672206,146781.4380062220734544,
146807.5365438322769478,146833.6362414032919332,146859.7370988319744356,146885.8391160152677912,
146911.9422928501444403,146938.0466292335477192,146964.1521250624791719,146990.2587802339694463,
147016.3665946450491901,147042.4755681928072590,147068.5857007742743008,147094.6969922865973786,
147120.8094426269235555,147146.9230516923707910,147173.0378193801443558,147199.1537455874495208,
147225.2708302114915568,147251.3890731495339423,147277.5084742988110520,147303.6290335566445719,
147329.7507508203561883,147355.8736259872675873,147381.9976589547295589,147408.1228496201511007,
147434.2491978809121065,147460.3767036344797816,147486.5053667782631237,147512.6351872097584419,
147538.7661648264620453,147564.8982995258702431,147591.0315912055666558,147617.1660397630766965,
147643.3016450960130896,147669.4384071019885596,147695.5763256785867270,147721.7154007235367317,
147747.8556321344512980,147773.9970198090886697,147800.1395636451197788,147826.2832635403319728,
147852.4281193924834952,147878.5741310993325897,147904.7212985587248113,147930.8696216685057152,
147957.0191003265208565,147983.1697344306157902,148009.3215238787524868,148035.4744685688347090,
148061.6285683988244273,148087.7838232666545082,148113.9402330703451298,148140.0977977079164702,
148166.2565170773887075,148192.4163910768693313,148218.5774196043785196,148244.7396025580819696,
148270.9029398360580672,148297.0674313365016133,148323.2330769575492013,148349.3998765974247362,
148375.5678301543230191,148401.7369375265261624,148427.9071986122580711,148454.0786133098299615,
148480.2511815175239462,148506.4249031336803455,148532.5997780566685833,148558.7758061848580837,
148584.9529874166473746,148611.1313216504349839,148637.3108087847067509,148663.4914487178903073,
148689.6732413485005964,148715.8561865750234574,148742.0402842960029375,148768.2255344099830836,

148794.4119368155661505,148820.5994914113252889,148846.7881980958918575,148872.9780567679263186,
148899.1690673260600306,148925.3612296690116636,148951.5545436954707839,148977.7490093041851651,
149003.9446263939025812,149030.1413948633999098,149056.3393146115122363,149082.5383855369873345,
149108.7386075387476012,149134.9399805155990180,149161.1425043664639816,149187.3461789902648889,
149213.5510042858950328,149239.7569801523350179,149265.9641064885363448,149292.1723831935378257,
149318.3818101663491689,149344.5923873059800826,149370.8041145115566906,149397.0169916821178049,
149423.2310187167895492,149449.4461955147271510,149475.6625219750276301,149501.8799979969335254,
149528.0986234796000645,149554.3183983222697861,149580.5393224241852295,149606.7613956845889334,
149632.9846180027816445,149659.2089892780932132,149685.4345094098243862,149711.6611782973341178,
149737.8889958400104661,149764.1179619372414891,149790.3480764884443488,149816.5793393930944148,
149842.8117505506088492,149869.0453098604630213,149895.2800172222196124,149921.5158725353830960,
149947.7528756995161530,149973.9910266141523607,150000.2303251789126080,150026.4707712934177835,
150052.7123648573178798,150078.9551057702628896,150105.1989939319028053,150131.4440292420040350,
150157.6902116002456751,150183.9375409063941333,150210.1860170602449216,150236.4356399615353439,
150262.6864095101191197,150288.9383256058208644,150315.1913881485234015,150341.4455970380513463,
150367.7009521743748337,150393.9574534573475830,150420.2151007869688328,150446.4738940632087179,
150472.7338331860082690,150498.9949180554249324,150525.2571485714579467,150551.5205246341938619,
150577.7850461436901242,150604.0507130000332836,150630.3175251033680979,150656.5854823538393248,
150682.8545846515917219,150709.1248318967991509,150735.3962239897227846,150761.6687608305073809,
150787.9424423195014242,150814.2172683569078799,150840.4932388430170249,150866.7703536782064475,
150893.0486127627664246,150919.3280159970454406,150945.6085632814792916,150971.8902545164164621,
150998.1730896022927482,151024.4570684395730495,151050.7421909287222661,151077.0284569702052977,
151103.3158664645452518,151129.6044193122943398,151155.8941154140047729,151182.1849546701996587,
151208.4769369815476239,151234.7700622486299835,151261.0643303720862605,151287.3597412525850814,
151313.6562947908241767,151339.9539908874721732,151366.2528294432850089,151392.5528103590186220,
151418.8539335354289506,151445.1561988733010367,151471.4596062734490260,151497.7641556367452722,
151524.0698468639748171,151550.3766798560682219,151576.6846545139269438,151602.9937707384524401,
151629.3040284305752721,151655.6154274912842084,151681.9279678215680178,151708.2416493224154692,
151734.5564718948444352,151760.8724354399309959,151787.1895398587221280,151813.5077850523521192,
151839.8271709218679462,151866.1476973684621044,151892.4693642932688817,151918.7921715974807739,
151945.1161191822902765,151971.4412069488898851,151997.7674347985594068,152024.0948026325495448,
152050.4233103521401063,152076.7529578586400021,152103.0837450533872470,152129.4156718377198558,
152155.7487381130340509,152182.0829437806678470,152208.4182887420756742,152234.7547728987119626,
152261.0923961519729346,152287.4311584033712279,152313.7710595543903764,152340.1120995065721218,
152366.4542781614582054,152392.7975954205612652,152419.1420511855394579,152445.4876453579345252,
152471.8343778394046240,152498.1822485316079110,152524.5312573361734394,152550.8814041548175737,
152577.2326888892566785,152603.5851114412071183,152629.9386717124143615,152656.2933696046820842,
152682.6492050197848585,152709.0061778595554642,152735.3642880258266814,152761.7235354204312898,
152788.0839199452893808,152814.4454415022919420,152840.8080999933299609,152867.1718953203817364,
152893.5368273854255676,152919.9028960904106498,152946.2701013373734895,152972.6384430283214897,
152999.0079210652911570,153025.3785353504063096,153051.7502857857034542,153078.1231722733355127,
153104.4971947153971996,153130.8723530140996445,153157.2486470715666655,153183.6260767900093924,
153210.0046420716680586,153236.3843428187537938,153262.7651789335359354,153289.1471503182838205,
153315.5302568753540982,153341.9144985070161056,153368.2998751155973878,153394.6863866035419051,
153421.0740328731480986,153447.4628138268890325,153473.8527293671795633,153500.2437793964345474,
153526.6359638171561528,153553.0292825318174437,153579.4237354429496918,153605.8193224530841690,
153632.2160434647812508,153658.6138983805722091,153685.0128871031047311,153711.4130095349682961,
153737.8142655788105913,153764.2166551372793037,153790.6201781130512245,153817.0248344088613521,
153843.4306239273864776,153869.8375465713907033,153896.2456022436672356,153922.6547908469219692,
153949.0651122840354219,153975.4765664578008000,154001.8891532710695174,154028.3028726266929880,
154054.7177244276099373,154081.1337085766717792,154107.5508249768172391,154133.9690735310141463,
154160.3884541422303300,154186.8089667134627234,154213.2306111477373634,154239.6533873480511829,
154266.0772952174593229,154292.5023346590751316,154318.9285055759828538,154345.3558078712667339,

154371.7842414480983280,154398.2138062096491922,154424.6445020590326749,154451.0763288995076437,
154477.5092866342747584,154503.9433751665928867,154530.3785943996917922,154556.8149442368885502,
154583.2524245814420283,154609.6910353367275093,154636.1307764060329646,154662.5716476927918848,
154689.0136491003213450,154715.4567805320839398,154741.9010418914549518,154768.3464330819260795,
154794.7929540069599170,154821.2406045700190589,154847.6893846746243071,154874.1392942243255675,
154900.5903331226436421,154927.0425012731575407,154953.4957985794753768,154979.9502249452052638,
155006.4057802739553154,155032.8624644694209564,155059.3202774352394044,155085.7792190751351882,
155112.2392892928037327,155138.7004879919695668,155165.1628150764154270,155191.6262704498949461,
155218.0908540162490681,155244.5565656792314257,155271.0234053427411709,155297.4913729105901439,
155323.9604682866774965,155350.4306913749023806,155376.9020420791930519,155403.3745203034777660,
155429.8481259517138824,155456.3228589278878644,155482.7987191360152792,155509.2757064801116940,
155535.7538208642217796,155562.2330621923902072,155588.7134303687198553,155615.1949252973136026,
155641.6775468822743278,155668.1612950277922209,155694.6461696379701607,155721.1321706170565449,
155747.6192978692124598,155774.1075512986863032,155800.5969308097264729,155827.0874363066104706,
155853.5790676935866941,155880.0718248749617487,155906.5657077551295515,155933.0607162383676041,
155959.5568502290989272,155986.0541096316592302,156012.5524943505006377,156039.0520042900170665,
156065.5526393547188491,156092.0543994489999022,156118.5572844773996621,156145.0612943444284610,
156171.5664289545966312,156198.0726882124727126,156224.5800720225961413,156251.0885802895936649,
156277.5982129180920310,156304.1089698126597796,156330.6208508780109696,156357.1338560188014526,
156383.6479851396870799,156410.1632381454401184,156436.6796149407455232,156463.1971154304046649,
156489.7157395191316027,156516.2354871117568109,156542.7563581131107640,156569.2783524279948324,
156595.8014699612685945,156622.3257106178207323,156648.8510743025399279,156675.3775609203148633,
156701.9051703761215322,156728.4339025749068242,156754.9637574216176290,156781.4947348212881479,
156808.0268346788943745,156834.5600568995287176,156861.0944013881671708,156887.6298680499312468,
156914.1664567899424583,156940.7041675132641103,156967.2430001250468194,156993.7829545304703061,
157020.3240306346851867,157046.8662283429002855,157073.4095475603244267,157099.9539881921955384,
157126.4995501437806524,157153.0462333203176968,157179.5940376271610148,157206.1429629695776384,
157232.6930092529219110,157259.2441763825772796,157285.7964642638689838,157312.3498728022095747,
157338.9044019030407071,157365.4600514717458282,157392.0168214138539042,157418.5747116347774863,
157445.1337220400455408,157471.6938525351288263,157498.2551030256436206,157524.8174734170606826,
157551.3809636150253937,157577.9455735250958242,157604.5113030528591480,157631.0781521040189546,
157657.6461205841624178,157684.2152083990222309,157710.7854154542728793,157737.3567416555888485,
157763.9291869087610394,157790.5027511195221450,157817.0774341936339624,157843.6532360369164962,
157870.2301565551606473,157896.8081956542155240,157923.3873532399011310,157949.9676292181247845,
157976.5490234947938006,158003.1315359757572878,158029.7151665670098737,158056.2999151744879782,
158082.8857817041571252,158109.4727660619828384,158136.0608681540179532,158162.6500878862862010,
158189.2404251648404170,158215.8318798957334366,158242.4244519850471988,158269.0181413389218505,
158295.6129478634684347,158322.2088714648562018,158348.8059120492252987,158375.4040695228031836,
158402.0033437917591073,158428.6037347623496316,158455.2052423408022150,158481.8078664333734196,
158508.4116069464071188,158535.0164637861598749,158561.6224368589755613,158588.2295260711980518,
158614.8377313292003237,158641.4470525393553544,158668.0574896080652252,158694.6690424417902250,
158721.2817109469324350,158747.8954950299812481,158774.5103945973969530,158801.1264095557271503,
158827.7435398114321288,158854.3617852711176965,158880.9811458413023502,158907.6016214285627939,
158934.2232119395630434,158960.8459172808506992,158987.4697373590897769,159014.0946720809733961,
159040.7207213531364687,159067.3478850823012181,159093.9761631751898676,159120.6055553385246407,
159147.2360620790568646,159173.8676827035960741,159200.5004173188935965,159227.1342658318171743,
159253.7692281491763424,159280.4053041778097395,159307.0424938246142119,159333.6807969964866061,
159360.3202136003237683,159386.9607435430516489,159413.6023867316544056,159440.2451430730870925,
159466.8890124743338674,159493.5339948424370959,159520.1800900843809359,159546.8272981072368566,
159573.4756188180472236,159600.1250521239708178,159626.7755979320209008,159653.4272561494144611,
159680.0800266832520720,159706.7339094406925142,159733.3889043289236724,159760.0450112551625352,
159786.7022301266551949,159813.3605608505895361,159840.0200033342698589,159866.6805574849713594,
159893.3422232099692337,159920.0050004166259896,159946.6688890122459270,159973.3338889042206574,

159999.9999999998835847,160026.6672222066845279,160053.3355554320150986,160080.0049995832960121,
160106.6755545680061914,160133.3472202935954556,160160.0199966676009353,160186.6938835975015536,
160213.3688809908344410,160240.0449887551658321,160266.7222067980619613,160293.4005350270890631,
160320.0799733498715796,160346.7605216740630567,160373.4421799072879367,160400.1249479571997654,
160426.8088257315102965,160453.4938131379312836,160480.1799100841744803,160506.8671164779516403,
160533.5554322270909324,160560.2448572393332142,160586.9353914224775508,160613.6270346843521111,
160640.3197869328141678,160667.0136480756918900,160693.7086180208716542,160720.4046966762689408,
160747.1018839497992303,160773.8001797493780032,160800.4995839829789475,160827.2000965585466474,
160853.9017173841129988,160880.6044463676516898,160907.3082834172237199,160934.0132284408609848,
160960.7192813466535881,160987.4264420426916331,161014.1347104370361194,161040.8440864378353581,
161067.5545699532667641,161094.2661608914786484,161120.9788591606193222,161147.6926646689244080,
161174.4075773246295284,161201.1235970359412022,161227.8407237111241557,161254.5589572584722191,
161281.2782975862792227,161307.9987446028681006,161334.7202982165617868,161361.4429583357123192,
161388.1667248687008396,161414.8915977239084896,161441.6175768097746186,161468.3446620346803684,
161495.0728533071232960,161521.8021505355718546,161548.5325536284653936,161575.2640624943596777,
161601.9966770417522639,161628.7303971791989170,161655.4652228152554017,161682.2011538584774826,
161708.9381902175373398,161735.6763318009907380,161762.4155785175098572,161789.1559302757086698,
161815.8973869843175635,161842.6399485520087183,161869.3836148874834180,161896.1283858995011542,
161922.8742614967923146,161949.6212415881163906,161976.3693260822910815,162003.1185148881049827,
162029.8688079144048970,162056.6202050700085238,162083.3727062638208736,162110.1263114046596456,
162136.8810204015171621,162163.6368331632402260,162190.3937495987920556,162217.1517696171067655,
162243.9108931272348855,162270.6711200380814262,162297.4324502587260213,162324.1948836981900968,
162350.9584202654950786,162377.7230598697497044,162404.4888024200336076,162431.2556478254264221,
162458.0235959950659890,162484.7926468381483573,162511.5628002637822647,162538.3340561811637599,
162565.1064144994888920,162591.8798751280119177,162618.6544379759288859,162645.4301029525231570,
162672.2068699670780916,162698.9847389288770501,162725.7637097472324967,162752.5437823314859997,
162779.3249565909500234,162806.1072324350534473,162832.8906097731669433,162859.6750885146611836,
162886.4606685690232553,162913.2473498456238303,162940.0351322540082037,162966.8240157035761513,
162993.6140001039020717,163020.4050853644439485,163047.1972713948052842,163073.9905581044731662,
163100.7849454030510969,163127.5804332001425792,163154.3770214053511154,163181.1747099283093121,
163207.9734986786497757,163234.7733875660633203,163261.5743765002407599,163288.3764653908729088,
163315.1796541476505809,163341.9839426803810056,163368.7893308987549972,163395.5958187126088887,
163422.4034060317208059,163449.2120927658979781,163476.0218788249767385,163502.8327641188225243,
163529.6447485573007725,163556.4578320502769202,163583.2720145077037159,163610.0872958394465968,
163636.9036759555165190,163663.7211547658371273,163690.5397321804193780,163717.3594081092160195,
163744.1801824622962158,163771.0020551496418193,163797.8250260813802015,163824.6490951675223187,
163851.4742623181955423,163878.3005274434981402,163905.1278904535574839,163931.9563512585300487,
163958.7859097685723100,163985.6165658938698471,164012.4483195446373429,164039.2811706310894806,
164066.1151190634409431,164092.9501647519937251,164119.7863076069916133,164146.6235475387366023,
164173.4618844575597905,164200.3013182737631723,164227.1418488977069501,164253.9834762397804298,
164280.8262002103147097,164307.6700207197864074,164334.5149376785557251,164361.3609509970992804,
164388.2080605858645868,164415.0562663553282619,164441.9055682159960270,164468.7559660783736035,
164495.6074598529958166,164522.4600494503974915,164549.3137347811716609,164576.1685157559113577,
164603.0243922851805110,164629.8813642796594650,164656.7394316499121487,164683.5985943066771142,
164710.4588521606056020,164737.3202051223779563,164764.1826531027036253,164791.0461960123502649,
164817.9108337620564271,164844.7765662625606637,164871.6433934246888384,164898.5113151592377108,
164925.3803313770331442,164952.2504419888718985,164979.1216469056962524,165005.9939460383029655,
165032.8673392976343166,165059.7418265946034808,165086.6174078401527368,165113.4940829451952595,
165140.3718518207024317,165167.2507143777038436,165194.1306705271708779,165221.0117201801331248,
165247.8938632476201747,165274.7770996407198254,165301.6614292704907712,165328.5468520480208099,
};

```
inline void requantize
   (header_info_type *header_info,
   scalefac_type *scalefac,
   side_info_type *si,
   long int is[SBLIMIT][SSLIMIT],
   float xr[SBLIMIT][SSLIMIT],
   int gr,
   int ch)
{
    int ss,sb,cb=0,sfreq;
    int next_cb, cb_begin, cb_width, sign;
    float temp;
    struct gr_info_type *gr_info = &(si->ch[ch].gr[gr]);

    sfreq=header_info->sampling_frequency_index + (header_info->ID * 3);

    //-------------------assign scalefactor band index----------------------;

    if (gr_info->window_switching_flag && (gr_info->block_type == 2) )
        if (gr_info->mixed_block_flag)
            next_cb = sfBandIndex[sfreq].l[1];    // mixed blocks;
        else
        {
            next_cb = sfBandIndex[sfreq].s[1]*3; // short blocks;
            cb_width = sfBandIndex[sfreq].s[1];
            cb_begin = 0;
        }
    else
        next_cb=sfBandIndex[sfreq].l[1];    // long blocks;



    for (sb=0 ; sb < SBLIMIT ; sb++)
        for (ss=0 ; ss < SSLIMIT ; ss++)
        {
     //----------------scalefactor band transition--------------------------;
            if ( (sb*SSLIMIT)+ss == next_cb)
            {
                if (gr_info->window_switching_flag && (gr_info->block_type == 2))
                {
                    if (gr_info->mixed_block_flag)    // mixed blocks;
                    {
                       if (((sb*SSLIMIT)+ss) == sfBandIndex[sfreq].l[8])
                         {
                            next_cb = sfBandIndex[sfreq].s[4]*3;
                            cb = 3;
                            cb_width = sfBandIndex[sfreq].s[cb+1] -
                                        sfBandIndex[sfreq].s[cb];
                            cb_begin = sfBandIndex[sfreq].s[cb]*3;
                         }
                         else if (((sb*SSLIMIT)+ss) < sfBandIndex[sfreq].l[8])
```

```
                    next_cb = sfBandIndex[sfreq].l[(++cb)+1];
                else
                {
                    next_cb = sfBandIndex[sfreq].s[(++cb)+1]*3;
                    cb_width = sfBandIndex[sfreq].s[cb+1] -
                                        sfBandIndex[sfreq].s[cb];
                    cb_begin = sfBandIndex[sfreq].s[cb]*3;
                }
            }
            else    // short blocks;
            {
                next_cb = sfBandIndex[sfreq].s[(++cb)+1]*3;
                cb_width = sfBandIndex[sfreq].s[cb+1] -
                                    sfBandIndex[sfreq].s[cb];
              cb_begin = sfBandIndex[sfreq].s[cb]*3;
            }
        }
        else // long blocks;
            next_cb = sfBandIndex[sfreq].l[(++cb)+1];
    }

//------------------compute scaling for each block----------------------------;

    xr[sb][ss] = powertable1[gr_info->global_gain];
    //temp = pow( 2.0 , (0.25 * (gr_info->global_gain - 210)));
    //printf(" power for initialization is    %f \n", gr_info->global_gain- 210.0);

    if (gr_info->window_switching_flag && (
        ((gr_info->block_type == 2) && (gr_info->mixed_block_flag == 0)) | |
        ((gr_info->block_type == 2) && gr_info->mixed_block_flag && (sb >= 2)) ))
    { //short blocks;
        //printf("herehere hre \n");

        xr[sb][ss] *=powershort1[gr_info->subblock_gain[(((sb*SSLIMIT)+ss) - cb_begin)/cb_width]];

        // xr[sb][ss] =floatmul(xr[sb][ss], pow(2.0,    -floatmul(2.0,
                //    gr_info->subblock_gain[(((sb*SSLIMIT)+ss) - cb_begin)/cb_width])));//-0.25*8

        // printf("gain is %.16f\n",pow(2.0,    -floatmul(2.0,
                // gr_info->subblock_gain[(((sb*SSLIMIT)+ss) - cb_begin)/cb_width])));
         //printf(" power for short block 1st part is    %f \n", gr_info->subblock_gain[(((sb*SSLIMIT)+ss) - cb_begin)/cb_width]);


         xr[sb][ss] *=powershort2[(1+gr_info->scalefac_scale)*(scalefac->ch[ch].s[(((sb*SSLIMIT)+ss) -
cb_begin)/cb_width][cb])];//-0.25*2
         //printf("gain is %.16f\n",(1.0+gr_info->scalefac_scale)
                //*(scalefac->ch[ch].s[(((sb*SSLIMIT)+ss) - cb_begin)/cb_width][cb]));
         //printf(" power for short block 2nd part is    %f \n", (1.0+gr_info->scalefac_scale)
                //*(scalefac->ch[ch].s[(((sb*SSLIMIT)+ss) - cb_begin)/cb_width][cb]));
    }

    else {      //long blocks;
```

```
                    xr[sb][ss] *=powertable2[(1+gr_info->scalefac_scale)*(scalefac->ch[ch].l[cb] + gr_info->preflag * pretab[cb])];
                    //xr[sb][ss] =floatmul(xr[sb][ss], powertable2[(1+gr_info->scalefac_scale)
                            //*(scalefac->ch[ch].l[cb] + gr_info->preflag * pretab[cb])]);

                //printf(" power for long block is %f \n", (1.0+gr_info->scalefac_scale)
                        //    *(scalefac->ch[ch].l[cb] + gr_info->preflag * pretab[cb]));
            }

            sign = (is[sb][ss]<0) ? 1 : 0;

            if(abs(is[sb][ss])==0)xr[sb][ss] =0.0;
            else if(abs(is[sb][ss])==1){;}
            else
                xr[sb][ss] *=powertable3[abs(is[sb][ss])];
                //xr[sb][ss] =floatmul(    xr[sb][ss],powertable3[abs(is[sb][ss])]);
            //printf("abs is %d\n",abs(is[sb][ss]));
            //temp=pow( (float) abs(is[sb][ss]), ((float)4.0/3.0) );

            //printf("test power is %.16f\n",temp);
            if (sign) xr[sb][ss] = -xr[sb][ss];
    }
}
```

6.6 joint_stereo.h

```
#define MPG_MD_JOINT_STEREO 1

static void i_stereo_k_values(float io,int is_pos,int i,float k[2][576])
{
    if(is_pos == 0)
    {
        k[0][i] = 1;
        k[1][i] = 1;
    }
    else if ((is_pos % 2) == 1)
    {
        k[0][i] = pow(io,(float)((is_pos + 1)/2));
        k[1][i] = 1;
    }
    else
    {
        k[0][i] = 1;
        k[1][i] = pow(io,(float)(is_pos/2));
    }
}


inline void join_stereo(
        float xr[2][SBLIMIT][SSLIMIT],
        float lr[2][SBLIMIT][SSLIMIT],
        scalefac_type *scalefac,
        struct gr_info_type *gr_info,
        header_info_type *header_info
        )
```

```
{
    int sfreq;

    int ms_stereo = (header_info->mode == MPG_MD_JOINT_STEREO) &&
                            (header_info->mode_extension & 0x2);
    int i_stereo = (header_info->mode == MPG_MD_JOINT_STEREO) &&
                           (header_info->mode_extension & 0x1);
    int js_bound;    /* frequency line that marks the beggining of the zero part */
    int sfb,next_sfb_boundary;
    int i,j,sb,ss,ch;
    short is_pos[SBLIMIT*SSLIMIT];
    float is_ratio[SBLIMIT*SSLIMIT];
    float io;
    float k[2][SBLIMIT*SSLIMIT];

    int lsf     = (header_info->ID == 0);

     if(    (gr_info->scalefac_compress % 2) == 1)
      {
         io = (float)0.707106781188;
      }
      else
      {
         io = (float)0.840896415256;
      }


    sfreq=header_info->sampling_frequency_index + (header_info->ID * 3);


    /* intialization */
    for ( i=0; i<SBLIMIT*SSLIMIT; i++ )
         is_pos[i] = 7;

    if ((stereo == 2) && i_stereo )
    {    if (gr_info->window_switching_flag && (gr_info->block_type == 2)) // short block and mixed block
         {    if( gr_info->mixed_block_flag )// mixed block
              {    int max_sfb = 0;

                   for ( j=0; j<3; j++ )
                   {    int sfbcnt;
                        sfbcnt = 2;
                        for( sfb=12; sfb >=3; sfb-- )
                        {    int lines;
                             lines = sfBandIndex[sfreq].s[sfb+1]-sfBandIndex[sfreq].s[sfb];
                             i = 3*sfBandIndex[sfreq].s[sfb] + (j+1) * lines - 1;
                             while ( lines > 0 )
                             {    if ( xr[1][i/SSLIMIT][i%SSLIMIT] != 0.0 )
                                  {    sfbcnt = sfb;
                                       sfb = -10;
                                       lines = -10;
```

```
            }
            lines--;
            i--;
        }
    }
    sfb = sfbcnt + 1;

    if ( sfb > max_sfb )
        max_sfb = sfb;

    while( sfb<12 )
    {   sb = sfBandIndex[sfreq].s[sfb+1]-sfBandIndex[sfreq].s[sfb];
        i = 3*sfBandIndex[sfreq].s[sfb] + j * sb;
        for ( ; sb > 0; sb--)
        {   is_pos[i] = scalefac->ch[1].s[j][sfb];
            if ( is_pos[i] != 7 )
                if( lsf )
                {
                    i_stereo_k_values(is_pos[i],io,i,k);
                }
                else
                {
                    is_ratio[i] = tan((float)is_pos[i] * (PI / 12));
                }
            i++;
        }
        sfb++;
    }

    sb = sfBandIndex[sfreq].s[12]-sfBandIndex[sfreq].s[11];
    sfb = 3*sfBandIndex[sfreq].s[11] + j * sb;
    sb = sfBandIndex[sfreq].s[13]-sfBandIndex[sfreq].s[12];

    i = 3*sfBandIndex[sfreq].s[11] + j * sb;
    for ( ; sb > 0; sb-- )
    {   is_pos[i] = is_pos[sfb];
        is_ratio[i] = is_ratio[sfb];
        k[0][i] = k[0][sfb];
        k[1][i] = k[1][sfb];
        i++;
    }
}
if ( max_sfb <= 3 )
{   i = 2;
    ss = 17;
    sb = -1;
    while ( i >= 0 )
    {   if ( xr[1][i][ss] != 0.0 )
        {   sb = i*18+ss;
            i = -1;
        } else
        {   ss--;
            if ( ss < 0 )
```

```
                {   i--;
                    ss = 17;
                }
            }
        }
        i = 0;
        while ( sfBandIndex[sfreq].l[i] <= sb )
            i++;
        sfb = i;
        i = sfBandIndex[sfreq].l[i];
        for ( ; sfb<8; sfb++ )
        {   sb = sfBandIndex[sfreq].l[sfb+1]-sfBandIndex[sfreq].l[sfb];
            for ( ; sb > 0; sb--)
            {   is_pos[i] = scalefac->ch[1].l[sfb];
                if ( is_pos[i] != 7 )
                    if ( lsf )
                    {
                            i_stereo_k_values(is_pos[i],io,i,k);
                    }
                    else
                    {
                        is_ratio[i] = tan((float)is_pos[i] * (PI / 12));
                    }
                i++;
            }
        }
    }
} //end mixed block
else// short block
{   for ( j=0; j<3; j++ )
    {   int sfbcnt;
        sfbcnt = -1;
        for( sfb=12; sfb >=0; sfb-- )
        {   int lines;
            lines = sfBandIndex[sfreq].s[sfb+1]-sfBandIndex[sfreq].s[sfb];
            i = 3*sfBandIndex[sfreq].s[sfb] + (j+1) * lines - 1;
            while ( lines > 0 )
            {   if ( xr[1][i/SSLIMIT][i%SSLIMIT] != 0.0 )
                {   sfbcnt = sfb;
                    sfb = -10;
                    lines = -10;
                }
                lines--;
                i--;
            }
        }
        sfb = sfbcnt + 1;
        while( sfb<12 )
        {   sb = sfBandIndex[sfreq].s[sfb+1]-sfBandIndex[sfreq].s[sfb];
            i = 3*sfBandIndex[sfreq].s[sfb] + j * sb;
            for ( ; sb > 0; sb--)
            {   is_pos[i] = scalefac->ch[1].s[j][sfb];
                if ( is_pos[i] != 7 )
```

```
                        if( lsf )
                        {
                                i_stereo_k_values(is_pos[i],io,i,k);
                        }
                        else
                        {
                               is_ratio[i] = tan( (float)is_pos[i] * (PI / 12));
                        }
                    i++;
                }
                sfb++;
            }

            sb = sfBandIndex[sfreq].s[12]-sfBandIndex[sfreq].s[11];
            sfb = 3*sfBandIndex[sfreq].s[11] + j * sb;
            sb = sfBandIndex[sfreq].s[13]-sfBandIndex[sfreq].s[12];

            i = 3*sfBandIndex[sfreq].s[11] + j * sb;
            for ( ; sb > 0; sb-- )
            {   is_pos[i] = is_pos[sfb];
                is_ratio[i] = is_ratio[sfb];
                k[0][i] = k[0][sfb];
                k[1][i] = k[1][sfb];
                i++;
            }
        }
    } // end short block
} // end short block and mixed block

else// long block
{   i = 31;
    ss = 17;
    sb = 0;
    while ( i >= 0 )
    {   if ( xr[1][i][ss] != 0.0 )
        {   sb = i*18+ss;
            i = -1;
        } else
        {   ss--;
            if ( ss < 0 )
            {   i--;
                ss = 17;
            }
        }
    }
    i = 0;
    while ( sfBandIndex[sfreq].l[i] <= sb )
        i++;
    sfb = i;
    i = sfBandIndex[sfreq].l[i];
    for ( ; sfb<21; sfb++ )
    {   sb = sfBandIndex[sfreq].l[sfb+1] - sfBandIndex[sfreq].l[sfb];
        for ( ; sb > 0; sb--)
```

```
            {    is_pos[i] = scalefac->ch[1].l[sfb];
                 if ( is_pos[i] != 7 )
                          if( lsf )
                          {
                                   i_stereo_k_values(is_pos[i],io,i,k);
                          }
                          else
                          {
                                   is_ratio[i] = tan((float)is_pos[i] * (PI / 12));
                          }
                 i++;
            }
        }
        sfb = sfBandIndex[sfreq].l[20];
        for ( sb = 576 - sfBandIndex[sfreq].l[21]; sb > 0; sb-- )
        {    is_pos[i] = is_pos[sfb];
             is_ratio[i] = is_ratio[sfb];
             k[0][i] = k[0][sfb];
             k[1][i] = k[1][sfb];
             i++;
        }
    }
}

//end    if ((stereo == 2) && i_stereo )



for(ch=0;ch<2;ch++)
    for(sb=0;sb<SBLIMIT;sb++)
        for(ss=0;ss<SSLIMIT;ss++)
            lr[ch][sb][ss] = 0;



if (stereo==2) // stereo
    for(sb=0;sb<SBLIMIT;sb++)
        for(ss=0;ss<SSLIMIT;ss++) {
            i = (sb*18)+ss;
            if ( is_pos[i] == 7 ) {
                if ( ms_stereo ) {
                    lr[0][sb][ss] = (xr[0][sb][ss]+xr[1][sb][ss])/(float)1.41421356;
                    lr[1][sb][ss] = (xr[0][sb][ss]-xr[1][sb][ss])/(float)1.41421356;
                }
                else {
                    lr[0][sb][ss] = xr[0][sb][ss];
                    lr[1][sb][ss] = xr[1][sb][ss];
                }
            }
            else if (i_stereo ) {
                  if ( lsf )
```

```
                        {
                            lr[0][sb][ss] = xr[0][sb][ss] * k[0][i];
                            lr[1][sb][ss] = xr[0][sb][ss] * k[1][i];
                        }
                        else
                        {
                            lr[0][sb][ss] = xr[0][sb][ss] * (is_ratio[i]/(1+is_ratio[i]));
                            lr[1][sb][ss] = xr[0][sb][ss] * (1/(1+is_ratio[i]));
                        }
                    }
                    else {
                        printf("Error in streo processing\n");
                    }
                }
    else    /* mono , bypass xr[0][][] to lr[0][][]*/
        for(sb=0;sb<SBLIMIT;sb++)
            for(ss=0;ss<SSLIMIT;ss++)
                lr[0][sb][ss] = xr[0][sb][ss];

}
```

6.7 antialias.h

```
float Ci[8]={-0.6,-0.535,-0.33,-0.185,-0.095,-0.041,-0.0142,-0.0037};
//#include "float.h"



inline void antialias(xr, hybridIn, gr_info, header_info)
float xr[SBLIMIT][SSLIMIT];
float hybridIn[SBLIMIT][SSLIMIT];
struct gr_info_type *gr_info;
header_info_type *header_info;
{
    static int      init = 1;
    static float ca[8],cs[8];
    float           bu,bd;    /* upper and lower butterfly inputs */
    int             ss,sb,sblim;

    if (init) {
        int i;
        float       sq;
        for (i=0;i<8;i++) {
            sq=sqrt(1.0+Ci[i]*Ci[i]);
            cs[i] = 1.0/sq;
            ca[i] = Ci[i]/sq;
        }
        init = 0;
    }

    /* clear all inputs */

      for(sb=0;sb<SBLIMIT;sb++)
          for(ss=0;ss<SSLIMIT;ss++)
              hybridIn[sb][ss] = xr[sb][ss];
```

```
    if    ( gr_info->window_switching_flag && (gr_info->block_type == 2) &&
            !gr_info->mixed_block_flag ) return;

    if ( gr_info->window_switching_flag && gr_info->mixed_block_flag &&
       (gr_info->block_type == 2))
          sblim = 1;
    else
          sblim = SBLIMIT-1;

    /* 31 alias-reduction operations between each pair of sub-bands */
    /* with 8 butterflies between each pair                         */

    for(sb=0;sb<sblim;sb++)
          for(ss=0;ss<8;ss++) {
                bu = xr[sb][17-ss];
                bd = xr[sb+1][ss];
                //float temp1=cs[ss];
                //float temp2=ca[ss];
               // hybridIn[sb][17-ss] =floatadd (floatmul(bu,temp1), -floatmul(bd, temp2));
                //hybridIn[sb+1][ss] = floatadd (floatmul(bd,temp1), floatmul(bu, temp2));
                hybridIn[sb][17-ss] = (bu * cs[ss]) - (bd * ca[ss]);
                hybridIn[sb+1][ss] = (bd * cs[ss]) + (bu * ca[ss]);
                }
}
```

6.8 reorder.h

```
inline void reorder
   (float xr[SBLIMIT][SSLIMIT],
    float ro[SBLIMIT][SSLIMIT],
    struct gr_info_type *gr_info,
    header_info_type *header_info)
{
    int sfreq;
    int sfb, sfb_start, sfb_lines;
    int sb, ss, window, freq, src_line, des_line;

    sfreq=header_info->sampling_frequency_index + (header_info->ID * 3);

    for(sb=0;sb<SBLIMIT;sb++)
          for(ss=0;ss<SSLIMIT;ss++)
                ro[sb][ss] = 0;

    if (gr_info->window_switching_flag && (gr_info->block_type == 2)) {
          if (gr_info->mixed_block_flag) {
                /* NO REORDER FOR LOW 2 SUBBANDS */
                for (sb=0 ; sb < 2 ; sb++)
                      for (ss=0 ; ss < SSLIMIT ; ss++) {
                            ro[sb][ss] = xr[sb][ss];
                      }
                /* REORDERING FOR REST SWITCHED SHORT */
                for(sfb=3,sfb_start=sfBandIndex[sfreq].s[3],
                      sfb_lines=sfBandIndex[sfreq].s[4] - sfb_start;
```

```
                sfb < 13; sfb++,sfb_start=sfBandIndex[sfreq].s[sfb],
                (sfb_lines=sfBandIndex[sfreq].s[sfb+1] - sfb_start))
                    for(window=0; window<3; window++)
                        for(freq=0;freq<sfb_lines;freq++) {
                            src_line = sfb_start*3 + window*sfb_lines + freq;
                            des_line = (sfb_start*3) + window + (freq*3);
                            ro[des_line/SSLIMIT][des_line%SSLIMIT] =
                                        xr[src_line/SSLIMIT][src_line%SSLIMIT];
                        }
        }
        else {    /* pure short */
            for(sfb=0,sfb_start=0,sfb_lines=sfBandIndex[sfreq].s[1];
                sfb < 13; sfb++,sfb_start=sfBandIndex[sfreq].s[sfb],
                (sfb_lines=sfBandIndex[sfreq].s[sfb+1] - sfb_start))
                    for(window=0; window<3; window++)
                        for(freq=0;freq<sfb_lines;freq++) {
                            src_line = sfb_start*3 + window*sfb_lines + freq;
                            des_line = (sfb_start*3) + window + (freq*3);
                            ro[des_line/SSLIMIT][des_line%SSLIMIT] =
                                        xr[src_line/SSLIMIT][src_line%SSLIMIT];
                        }
        }
    }
    else {      /*long blocks */
        for (sb=0 ; sb < SBLIMIT ; sb++)
            for (ss=0 ; ss < SSLIMIT ; ss++)
                ro[sb][ss] = xr[sb][ss];
    }
}
```

6.9 hybird.h

```
//type0: win[0][0]=sin( PI/36 *(0+0.5)
float    win[4][36]=
{
{0.0436193873653360,0.1305261922200515,0.2164396139381027,0.3007057995042728,0.3826834323650894,0.4617486132350335,0.537
2996083468233,0.6087614290087201,0.6755902076156597,
0.7372773368101234,0.7933533402912346,0.8433914458128852,0.8870108331782213,0.9238795325112863,0.9537169507482265,0.9762
960071199330,0.9914448613738103,0.9990482215818577,
0.9990482215818578,0.9914448613738106,0.9762960071199338,0.9537169507482275,0.9238795325112875,0.8870108331782226,0.8433
914458128868,0.7933533402912365,0.7372773368101255,
0.6755902076156620,0.6087614290087225,0.5372996083468260,0.4617486132350363,0.3826834323650923,0.3007057995042758,0.2164
396139381058,0.1305261922200547,0.0436193873653392},

//type1:win[1][i]
{0.0436193873653360,0.1305261922200515,0.2164396139381027,0.3007057995042728,0.3826834323650894,0.4617486132350335,0.537
2996083468233,0.6087614290087201,0.6755902076156597,
0.7372773368101234,0.7933533402912346,0.8433914458128852,0.8870108331782213,0.9238795325112863,0.9537169507482265,0.9762
960071199330,0.9914448613738103,0.9990482215818577,
1.0000000000000000,1.0000000000000000,1.0000000000000000,1.0000000000000000,1.0000000000000000,1.0000000000000000,0.9914
448613738106,0.9238795325112875,0.7933533402912365,
0.6087614290087225,0.3826834323650923,0.1305261922200547,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000
000000000000,0.0000000000000000,0.0000000000000000},
```

```
//type2:win[2][i]
{0.1305261922200515,0.3826834323650894,0.6087614290087201,0.7933533402912346,0.9238795325112863,0.9914448613738103,0.9914448613738106,0.9238795325112875,0.7933533402912365,
0.6087614290087225,0.3826834323650923,0.1305261922200547,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,
0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,
0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000},

//type3: win[3][i]
{0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.0000000000000000,0.1305261922200515,0.3826834323650894,0.6087614290087201,
0.7933533402912346,0.9238795325112863,0.9914448613738103,1.0000000000000000,1.0000000000000000,1.0000000000000000,1.0000000000000000,1.0000000000000000,1.0000000000000000,
0.9990482215818578,0.9914448613738106,0.9762960071199338,0.9537169507482275,0.9238795325112875,0.8870108331782226,0.8433914458128868,0.7933533402912365,0.7372773368101255,
0.6755902076156620,0.6087614290087225,0.5372996083468260,0.4617486132350363,0.3826834323650923,0.3007057995042758,0.2164396139381058,0.1305261922200547,0.0436193873653392}
};
//COS[i]=cos(PI/(2*36) * i):
float COS[4*36]=
{1.0000000000000000,
0.9990482215818578,0.9961946980917455,0.9914448613738104,0.9848077530122081,0.9762960071199334,0.9659258262890683,0.9537169507482270,0.9396926207859085,
0.9238795325112870,0.9063077870366502,0.8870108331782219,0.8660254037844389,0.8433914458128861,0.8191520442889921,0.7933533402912356,0.7660444431189785,
0.7372773368101245,0.7071067811865481,0.6755902076156609,0.6427876096865400,0.6087614290087214,0.5735764363510469,0.5372996083468247,0.5000000000000009,
0.4617486132350349,0.4226182617407004,0.3826834323650908,0.3420201433256699,0.3007057995042743,0.2588190451025220,0.2164396139381042,0.1736481776669317,
0.1305261922200530,0.0871557427476597,0.0436193873653376,0.0000000000000016,-0.0436193873653343,-0.0871557427476565,-0.1305261922200498,-0.1736481776669286,
-0.2164396139381010,-0.2588190451025189,-0.3007057995042712,-0.3420201433256668,-0.3826834323650878,-0.4226182617406977,-0.4617486132350322,-0.4999999999999983,
-0.5372996083468221,-0.5735764363510444,-0.6087614290087189,-0.6427876096865376,-0.6755902076156586,-0.7071067811865459,-0.7372773368101224,-0.7660444431189765,
-0.7933533402912336,-0.8191520442889904,-0.8433914458128843,-0.8660254037844374,-0.8870108331782205,-0.9063077870366488,-0.9238795325112857,-0.9396926207859074,
-0.9537169507482260,-0.9659258262890675,-0.9762960071199327,-0.9848077530122076,-0.9914448613738100,-0.9961946980917452,-0.9990482215818576,-1.0000000000000000,
-0.9990482215818579,-0.9961946980917459,-0.9914448613738108,-0.9848077530122087,-0.9762960071199341,-0.9659258262890692,-0.9537169507482280,-0.9396926207859096,
-0.9238795325112882,-0.9063077870366515,-0.8870108331782235,-0.8660254037844406,-0.8433914458128878,-0.8191520442889940,-0.7933533402912376,-0.7660444431189807,
-0.7372773368101269,-0.7071067811865505,-0.6755902076156634,-0.6427876096865422,-0.6087614290087237,-0.5735764363510493,-0.5372996083468272,-0.5000000000000034,
-0.4617486132350375,-0.4226182617407032,-0.3826834323650936,-0.3420201433256727,-0.3007057995042772,-0.2588190451025250,-0.2164396139381072,-0.1736481776669347,
-0.1305261922200560,-0.0871557427476627,-0.0436193873653406,-0.0000000000000046,0.0436193873653313,0.0871557427476535,0.1305261922200469,0.1736481776669256,
0.2164396139380981,0.2588190451025160,0.3007057995042683,0.3420201433256640,0.3826834323650851,0.4226182617406948,0.4617486132350293,0.4999999999999954,
```

```
0.5372996083468193,0.5735764363510417,0.6087614290087163,0.6427876096865351,0.6755902076156562,0.7071067811865436,0.7372
773368101203,0.7660444431189745,
0.7933533402912317,0.8191520442889885,0.8433914458128826,0.8660254037844357,0.8870108331782189,0.9063077870366475,0.9238
795325112844,0.9396926207859063,
0.9537169507482250,0.9659258262890666,0.9762960071199320,0.9848077530122070,0.9914448613738096,0.9961946980917450,0.9990
482215818575};
//Done COS

//block type=2 , cos( PI/(2*N)*(2*p(variable)+1+N/2)*(2*m(variable)+1)): cos_tab_1[p][m]
float cos_tab_1[12][6]=
{
{0.6087614290087214,-0.9238795325112857,-0.1305261922200560,0.9914448613738113,-0.3826834323650817,-0.7933533402912418},
{0.3826834323650908,-0.9238795325112882,0.9238795325112844,-0.3826834323650817,-0.3826834323651004,0.9238795325112915},
{0.1305261922200530,-0.3826834323650936,0.6087614290087267,-0.7933533402912418,0.9238795325112915,-0.9914448613738125},
{-0.1305261922200498,0.3826834323650851,-0.6087614290087134,0.7933533402912272,-0.9238795325112810,0.9914448613738079},
{-0.3826834323650878,0.9238795325112844,-0.9238795325112908,0.3826834323651022,0.3826834323650731,-0.9238795325112787},
{-0.6087614290087189,0.9238795325112894,0.1305261922200396,-0.9914448613738084,0.3826834323651073,0.7933533402912194},
{-0.7933533402912336,0.3826834323650970,0.9914448613738086,0.1305261922200323,-0.9238795325112957,-0.6087614290086988},
{-0.9238795325112857,-0.3826834323650817,0.3826834323651022,0.9238795325112944,0.9238795325112774,0.3826834323650595},
{-0.9914448613738100,-0.9238795325112831,-0.7933533402912261,-0.6087614290087046,-0.3826834323650630,-0.1305261922200211},
{-0.9914448613738108,-0.9238795325112908,-0.7933533402912463,-0.6087614290087398,-0.3826834323651175,-0.1305261922200872},
{-0.9238795325112882,-0.3826834323651004,0.3826834323650731,0.9238795325112774,0.9238795325112986,0.3826834323651277},
{-0.7933533402912376,0.3826834323650783,0.9914448613738127,0.1305261922200798,-0.9238795325112745,-0.6087614290087573}
};
//Blocktype2Done!

void inv_mdct
      (float in[18],
       float out[36],
       int block_type)
{
int        i,m,N,p;
float    tmp[12],sum;
      for(i=0;i<36;i++)
          out[i]=0;

      if(block_type == 2){
          N=12;
          for(i=0;i<3;i++){
              for(p= 0;p<N;p++){
                  sum = 0.0;
                  for(m=0;m<N/2;m++)
                      {
                        int k = (m) + (m<<1);
                        sum += in[i+k] * (cos_tab_1[p][m]);
                        /*if(((cos( PI/(2*N)*(2*p+1+N/2)*(2*m+1) ))-(cos_tab_1[p][m]))>=0.0000001)
                              {printf("idmv error!\n");
                              exit(10);}*/
                      }
                  tmp[p] = sum * win[block_type][p] ;
              }
              for(p=0;p<N;p++){
                  int k = (i<<1) + (i<<2);
```

```
                    out[k+p+6] += tmp[p];
                }
            }
        }
        else{
            N=36;
            for(p= 0;p<N;p++){
                sum = 0.0;
                for(m=0;m<N/2;m++){
                    int k = ((p*m)<<2) + (p<<1) + (m<<5)+(m<<2)+(m<<1)+19;
                    while(k>=144){
                        k = k - 144;
                    }
                    sum += in[m] * COS[k];
                }

                out[p] = sum * win[block_type][p];
            }
        }
}

void hybrid
        (float fsIn[SSLIMIT],      /* freq samples per subband in */
         float tsOut[SSLIMIT],     /* time samples per subband out */
         int sb,
         int ch,
         struct gr_info_type *gr_info,
         header_info_type *header_info)
{
    int ss;
    float rawout[36];
    static float prevblck[2][SBLIMIT][SSLIMIT];
    static int init = 1;
    int bt;

    if (init) {
        int i,j,k;

        for(i=0;i<2;i++)
            for(j=0;j<SBLIMIT;j++)
                for(k=0;k<SSLIMIT;k++)
                    prevblck[i][j][k]=0.0;
        init = 0;
    }

    bt = (gr_info->window_switching_flag && gr_info->mixed_block_flag &&
                (sb < 2)) ? 0 : gr_info->block_type;

    inv_mdct( fsIn, rawout, bt);

    /* overlap addition */
    for(ss=0; ss<SSLIMIT; ss++) {
        tsOut[ss] = rawout[ss] + prevblck[ch][sb][ss];
```

```
            prevblck[ch][sb][ss] = rawout[ss+18];
        }
    }
```

6.10 subbandsynthesis.h

```
#include "dewindowarray.h"
#include "syn_costable.h"
#define SCALE 32768

inline void create_syn_filter(double doublefilter[64][SBLIMIT]) {
    register int i, k;

    for (i = 0; i < 64; i++)
            for (k = 0; k < 32; k++) {
                    //if ((filter[i][k] = 1e9*cos((float)((PI/64*i+PI/4)*(2*k+1)))) >= 0)
                    if ((doublefilter[i][k] = syn_costable[(i << 5) + k]) >= 0)
                            modf(doublefilter[i][k] + 0.5, &doublefilter[i][k]);
                    else
                            modf(doublefilter[i][k] - 0.5, &doublefilter[i][k]);

                    //printf("filter    before is %.16f\n",filter[i][k]);
                    doublefilter[i][k] *= 1e-9;
                    //printf("syn_costable[(i<<5)+k] is %.16f\n",syn_costable[(i<<5)+k]);
                    //printf("filter after is %.16f\n", doublefilter[i][k]);
            }

}

inline void SubBandSynthesis(float *bandPtr, int ch, short *samples) {
    register int i, j, k;
    int iii, jjj;
    register float *bufOffsetPtr, sum;
    register long foo;
    static int init = 1;
    typedef float NN[64][32];
    typedef double MM[64][32];
    static NN *filter;
    static MM *doublefilter;
    typedef float BB[2][1024];
    static BB *buf;
    static int bufOffset[2] = { 64, 64 };
    if (init) {
            buf = (BB *) malloc(sizeof(BB));
            doublefilter = (MM *) malloc(sizeof(MM));
            filter = (NN *) malloc(sizeof(NN));
            create_syn_filter(*doublefilter);
            for (iii = 0; iii < 64; iii++) {
                    for (jjj = 0; jjj < 32; jjj++)
                            (*filter)[iii][jjj] = (float) (*doublefilter)[iii][jjj];
                    ;
            }

            init = 0;
```

```
    }

    bufOffset[ch] = (bufOffset[ch] - 64) & 0x3ff;
    bufOffsetPtr = &((*buf)[ch][bufOffset[ch]]);

    for (i = 0; i < 64; i++) {
            sum = 0;
            for (k = 0; k < 32; k++)
                    sum += bandPtr[k] * (*filter)[i][k];
            bufOffsetPtr[i] = sum;
    }

    for (j = 0; j < 32; j++) {
            sum = 0;
            for (i = 0; i < 16; i++) {
                    k = j + (i << 5);
                    sum += dewindow[k] * (*buf)[ch][((k + (((i + 1) >> 1) << 6))
                                    + bufOffset[ch]) & 0x3ff];
            }

            if (sum > 0) {
                    foo = (long) (sum * (float) SCALE + (float) 0.5);
            } else {
                    foo = (long) (sum * (float) SCALE - (float) 0.5);
            }

            if (foo >= (long) SCALE) {
                    samples[j] = (short) (SCALE - 1);
            } else if (foo < (long) -SCALE) {
                    samples[j] = (short) (-SCALE);
            } else
                    samples[j] = (short) foo;

             //pcm_output[frameNum-1][gr_global][ch_global][ss_global][j]=samples[j];
    }
}
```

6.11 de2_vag_raster.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity de2_vga_raster is

    port (
        reset_n : in std_logic;
        clk     : in std_logic;                         -- Should be 25.125 MHz

        write : in std_logic;----------------
        chipselect : in std_logic;----------------
        address : in unsigned(6 downto 0);---------------
        writedata : in unsigned(15 downto 0);-------------

        VGA_CLK,                                        -- Clock
```

```
        VGA_HS,                                         -- H_SYNC
        VGA_VS,                                         -- V_SYNC
        VGA_BLANK,                                      -- BLANK
        VGA_SYNC : out std_logic;               -- SYNC
        VGA_R,                                          -- Red[9:0]
        VGA_G,                                          -- Green[9:0]
        VGA_B : out unsigned(9 downto 0) -- Blue[9:0]
        );

end de2_vga_raster;

architecture rtl of de2_vga_raster is

    -- Video parameters
signal clk25: std_logic;
    constant HTOTAL             : integer := 800;
    constant HSYNC              : integer := 96;
    constant HBACK_PORCH    : integer := 48;
    constant HACTIVE            : integer := 640;
    constant HFRONT_PORCH : integer := 16;

    constant VTOTAL             : integer := 525;
    constant VSYNC              : integer := 2;
    constant VBACK_PORCH    : integer := 33;
    constant VACTIVE            : integer := 480;
    constant VFRONT_PORCH : integer := 10;



--type matrix is array (0 to 63) of integer;
--constant RECTANGLE_HSTART : matrix;
--    RECTANGLE_HSTART[0]<=0;
--
--      FOR    i IN 1 to 64    LOOP
--RECTANGLE_HSTART[i]<=(RECTANGLE[i]+10);
--i=i+1;
--end loop
--
--
    constant RECTANGLE_HSTART1 : integer := 1;
    constant RECTANGLE_HEND1      : integer := 8;
    constant RECTANGLE_HSTART2 : integer := 10;
    constant RECTANGLE_HEND2      : integer := 18;
    constant RECTANGLE_HSTART3 : integer := 20;
    constant RECTANGLE_HEND3      : integer := 28;
    constant RECTANGLE_HSTART4 : integer := 30;
    constant RECTANGLE_HEND4      : integer := 38;
    constant RECTANGLE_HSTART5 : integer := 40;
    constant RECTANGLE_HEND5      : integer := 48;
    constant RECTANGLE_HSTART6 : integer := 50;
    constant RECTANGLE_HEND6      : integer := 58;
    constant RECTANGLE_HSTART7 : integer := 60;
    constant RECTANGLE_HEND7      : integer := 68;
```

```
constant RECTANGLE_HSTART8 : integer := 70;
constant RECTANGLE_HEND8   : integer := 78;
constant RECTANGLE_HSTART9 : integer := 80;
constant RECTANGLE_HEND9   : integer := 88;
constant RECTANGLE_HSTART10 : integer := 90;
constant RECTANGLE_HEND10   : integer := 98;

constant RECTANGLE_HSTART11 : integer := 100;
constant RECTANGLE_HEND11   : integer := 108;
constant RECTANGLE_HSTART12 : integer := 110;
constant RECTANGLE_HEND12   : integer := 118;
constant RECTANGLE_HSTART13 : integer := 120;
constant RECTANGLE_HEND13   : integer := 128;
constant RECTANGLE_HSTART14 : integer := 130;
constant RECTANGLE_HEND14   : integer := 138;
constant RECTANGLE_HSTART15 : integer := 140;
constant RECTANGLE_HEND15   : integer := 148;
constant RECTANGLE_HSTART16 : integer := 150;
constant RECTANGLE_HEND16   : integer := 158;
constant RECTANGLE_HSTART17 : integer := 160;
constant RECTANGLE_HEND17   : integer := 168;
constant RECTANGLE_HSTART18 : integer := 170;
constant RECTANGLE_HEND18   : integer := 178;
constant RECTANGLE_HSTART19 : integer := 180;
constant RECTANGLE_HEND19   : integer := 188;
constant RECTANGLE_HSTART20 : integer := 190;
constant RECTANGLE_HEND20   : integer := 198;



constant RECTANGLE_HSTART21 : integer := 200;
constant RECTANGLE_HEND21   : integer := 208;
constant RECTANGLE_HSTART22 : integer := 210;
constant RECTANGLE_HEND22   : integer := 218;
constant RECTANGLE_HSTART23 : integer := 220;
constant RECTANGLE_HEND23   : integer := 228;
constant RECTANGLE_HSTART24 : integer := 230;
constant RECTANGLE_HEND24   : integer := 238;
constant RECTANGLE_HSTART25 : integer := 240;
constant RECTANGLE_HEND25   : integer := 248;
constant RECTANGLE_HSTART26 : integer := 250;
constant RECTANGLE_HEND26   : integer := 258;
constant RECTANGLE_HSTART27 : integer := 260;
constant RECTANGLE_HEND27   : integer := 268;
constant RECTANGLE_HSTART28 : integer := 270;
constant RECTANGLE_HEND28   : integer := 278;
constant RECTANGLE_HSTART29 : integer := 280;
constant RECTANGLE_HEND29   : integer := 288;
constant RECTANGLE_HSTART30 : integer := 290;
constant RECTANGLE_HEND30   : integer := 298;

constant RECTANGLE_HSTART31 : integer := 300;
constant RECTANGLE_HEND31   : integer := 308;
```

```
constant RECTANGLE_HSTART32 : integer := 310;
constant RECTANGLE_HEND32   : integer := 318;
constant RECTANGLE_HSTART33 : integer := 320;
constant RECTANGLE_HEND33   : integer := 328;
constant RECTANGLE_HSTART34 : integer := 330;
constant RECTANGLE_HEND34   : integer := 338;
constant RECTANGLE_HSTART35 : integer := 340;
constant RECTANGLE_HEND35   : integer := 348;
constant RECTANGLE_HSTART36 : integer := 350;
constant RECTANGLE_HEND36   : integer := 358;
constant RECTANGLE_HSTART37 : integer := 360;
constant RECTANGLE_HEND37   : integer := 368;
constant RECTANGLE_HSTART38 : integer := 370;
constant RECTANGLE_HEND38   : integer := 378;
constant RECTANGLE_HSTART39 : integer := 380;
constant RECTANGLE_HEND39   : integer := 388;
constant RECTANGLE_HSTART40 : integer := 390;
constant RECTANGLE_HEND40   : integer := 398;

constant RECTANGLE_HSTART41 : integer := 400;
constant RECTANGLE_HEND41   : integer := 408;
constant RECTANGLE_HSTART42 : integer := 410;
constant RECTANGLE_HEND42   : integer := 418;
constant RECTANGLE_HSTART43 : integer := 420;
constant RECTANGLE_HEND43   : integer := 428;
constant RECTANGLE_HSTART44 : integer := 430;
constant RECTANGLE_HEND44   : integer := 438;
constant RECTANGLE_HSTART45 : integer := 440;
constant RECTANGLE_HEND45   : integer := 448;
constant RECTANGLE_HSTART46 : integer := 450;
constant RECTANGLE_HEND46   : integer := 458;
constant RECTANGLE_HSTART47 : integer := 460;
constant RECTANGLE_HEND47   : integer := 468;
constant RECTANGLE_HSTART48 : integer := 470;
constant RECTANGLE_HEND48   : integer := 478;
constant RECTANGLE_HSTART49 : integer := 480;
constant RECTANGLE_HEND49   : integer := 488;
constant RECTANGLE_HSTART50 : integer := 490;
constant RECTANGLE_HEND50   : integer := 498;


constant RECTANGLE_HSTART51 : integer := 500;
constant RECTANGLE_HEND51   : integer := 508;
constant RECTANGLE_HSTART52 : integer := 510;
constant RECTANGLE_HEND52   : integer := 518;
constant RECTANGLE_HSTART53 : integer := 520;
constant RECTANGLE_HEND53   : integer := 528;
constant RECTANGLE_HSTART54 : integer := 530;
constant RECTANGLE_HEND54   : integer := 538;
constant RECTANGLE_HSTART55 : integer := 540;
constant RECTANGLE_HEND55   : integer := 548;
constant RECTANGLE_HSTART56 : integer := 550;
constant RECTANGLE_HEND56   : integer := 558;
```

```
    constant RECTANGLE_HSTART57 : integer := 560;
    constant RECTANGLE_HEND57   : integer := 568;
    constant RECTANGLE_HSTART58 : integer := 570;
    constant RECTANGLE_HEND58   : integer := 578;
    constant RECTANGLE_HSTART59 : integer := 580;
    constant RECTANGLE_HEND59   : integer := 588;
    constant RECTANGLE_HSTART60 : integer := 590;
    constant RECTANGLE_HEND60   : integer := 598;

    constant RECTANGLE_HSTART61 : integer := 600;
    constant RECTANGLE_HEND61   : integer := 608;
    constant RECTANGLE_HSTART62 : integer := 610;
    constant RECTANGLE_HEND62   : integer := 618;
    constant RECTANGLE_HSTART63 : integer := 620;
    constant RECTANGLE_HEND63   : integer := 628;
    constant RECTANGLE_HSTART64 : integer := 630;
    constant RECTANGLE_HEND64   : integer := 638;

    constant RECTANGLE_VEND1,
RECTANGLE_VEND2,
RECTANGLE_VEND3,
RECTANGLE_VEND4,
RECTANGLE_VEND5,
RECTANGLE_VEND6,
RECTANGLE_VEND7,
RECTANGLE_VEND8,
RECTANGLE_VEND9,
RECTANGLE_VEND10,
RECTANGLE_VEND11,
RECTANGLE_VEND12,
RECTANGLE_VEND13,
RECTANGLE_VEND14,
RECTANGLE_VEND15,
RECTANGLE_VEND16,
RECTANGLE_VEND17,
RECTANGLE_VEND18,
RECTANGLE_VEND19,
RECTANGLE_VEND20,
RECTANGLE_VEND21,
RECTANGLE_VEND22,
RECTANGLE_VEND23,
RECTANGLE_VEND24,
RECTANGLE_VEND25,
RECTANGLE_VEND26,
RECTANGLE_VEND27,
RECTANGLE_VEND28,
RECTANGLE_VEND29,
RECTANGLE_VEND30,
RECTANGLE_VEND31,
RECTANGLE_VEND32,
RECTANGLE_VEND33,
RECTANGLE_VEND34,
RECTANGLE_VEND35,
```

```
RECTANGLE_VEND36,
RECTANGLE_VEND37,
RECTANGLE_VEND38,
RECTANGLE_VEND39,
RECTANGLE_VEND40,
RECTANGLE_VEND41,
RECTANGLE_VEND42,
RECTANGLE_VEND43,
RECTANGLE_VEND44,
RECTANGLE_VEND45,
RECTANGLE_VEND46,
RECTANGLE_VEND47,
RECTANGLE_VEND48,
RECTANGLE_VEND49,
RECTANGLE_VEND50,
RECTANGLE_VEND51,
RECTANGLE_VEND52,
RECTANGLE_VEND53,
RECTANGLE_VEND54,
RECTANGLE_VEND55,
RECTANGLE_VEND56,
RECTANGLE_VEND57,
RECTANGLE_VEND58,
RECTANGLE_VEND59,
RECTANGLE_VEND60,
RECTANGLE_VEND61,
RECTANGLE_VEND62,
RECTANGLE_VEND63,
RECTANGLE_VEND64: integer:=380;


signal RECTANGLE_VSTART1,
RECTANGLE_VSTART2,
RECTANGLE_VSTART3,
RECTANGLE_VSTART4,
RECTANGLE_VSTART5,
RECTANGLE_VSTART6,
RECTANGLE_VSTART7,
RECTANGLE_VSTART8,
RECTANGLE_VSTART9,
RECTANGLE_VSTART10,
RECTANGLE_VSTART11,
RECTANGLE_VSTART12,
RECTANGLE_VSTART13,
RECTANGLE_VSTART14,
RECTANGLE_VSTART15,
RECTANGLE_VSTART16,
RECTANGLE_VSTART17,
RECTANGLE_VSTART18,
RECTANGLE_VSTART19,
RECTANGLE_VSTART20,
RECTANGLE_VSTART21,
RECTANGLE_VSTART22,
```

```
RECTANGLE_VSTART23,
RECTANGLE_VSTART24,
RECTANGLE_VSTART25,
RECTANGLE_VSTART26,
RECTANGLE_VSTART27,
RECTANGLE_VSTART28,
RECTANGLE_VSTART29,
RECTANGLE_VSTART30,
RECTANGLE_VSTART31,
RECTANGLE_VSTART32,
RECTANGLE_VSTART33,
RECTANGLE_VSTART34,
RECTANGLE_VSTART35,
RECTANGLE_VSTART36,
RECTANGLE_VSTART37,
RECTANGLE_VSTART38,
RECTANGLE_VSTART39,
RECTANGLE_VSTART40,
RECTANGLE_VSTART41,
RECTANGLE_VSTART42,
RECTANGLE_VSTART43,
RECTANGLE_VSTART44,
RECTANGLE_VSTART45,
RECTANGLE_VSTART46,
RECTANGLE_VSTART47,
RECTANGLE_VSTART48,
RECTANGLE_VSTART49,
RECTANGLE_VSTART50,
RECTANGLE_VSTART51,
RECTANGLE_VSTART52,
RECTANGLE_VSTART53,
RECTANGLE_VSTART54,
RECTANGLE_VSTART55,
RECTANGLE_VSTART56,
RECTANGLE_VSTART57,
RECTANGLE_VSTART58,
RECTANGLE_VSTART59,
RECTANGLE_VSTART60,
RECTANGLE_VSTART61,
RECTANGLE_VSTART62,
RECTANGLE_VSTART63,
RECTANGLE_VSTART64: unsigned(15 downto 0);

signal vs1:unsigned(15 downto 0);
signal vs2:unsigned(15 downto 0);
signal vs3:unsigned(15 downto 0);
signal vs4:unsigned(15 downto 0);
signal vs5:unsigned(15 downto 0);
signal vs6:unsigned(15 downto 0);
signal vs7:unsigned(15 downto 0);
signal vs8:unsigned(15 downto 0);
signal vs9:unsigned(15 downto 0);
signal vs10:unsigned(15 downto 0);
```

```
signal vs11:unsigned(15 downto 0);
signal vs12:unsigned(15 downto 0);
signal vs13:unsigned(15 downto 0);
signal vs14:unsigned(15 downto 0);
signal vs15:unsigned(15 downto 0);
signal vs16:unsigned(15 downto 0);
signal vs17:unsigned(15 downto 0);
signal vs18:unsigned(15 downto 0);
signal vs19:unsigned(15 downto 0);
signal vs20:unsigned(15 downto 0);
signal vs21:unsigned(15 downto 0);
signal vs22:unsigned(15 downto 0);
signal vs23:unsigned(15 downto 0);
signal vs24:unsigned(15 downto 0);
signal vs25:unsigned(15 downto 0);
signal vs26:unsigned(15 downto 0);
signal vs27:unsigned(15 downto 0);
signal vs28:unsigned(15 downto 0);
signal vs29:unsigned(15 downto 0);
signal vs30:unsigned(15 downto 0);
signal vs31:unsigned(15 downto 0);
signal vs32:unsigned(15 downto 0);
signal vs33:unsigned(15 downto 0);
signal vs34:unsigned(15 downto 0);
signal vs35:unsigned(15 downto 0);
signal vs36:unsigned(15 downto 0);
signal vs37:unsigned(15 downto 0);
signal vs38:unsigned(15 downto 0);
signal vs39:unsigned(15 downto 0);
signal vs40:unsigned(15 downto 0);
signal vs41:unsigned(15 downto 0);
signal vs42:unsigned(15 downto 0);
signal vs43:unsigned(15 downto 0);
signal vs44:unsigned(15 downto 0);
signal vs45:unsigned(15 downto 0);
signal vs46:unsigned(15 downto 0);
signal vs47:unsigned(15 downto 0);
signal vs48:unsigned(15 downto 0);
signal vs49:unsigned(15 downto 0);
signal vs50:unsigned(15 downto 0);
signal vs51:unsigned(15 downto 0);
signal vs52:unsigned(15 downto 0);
signal vs53:unsigned(15 downto 0);
signal vs54:unsigned(15 downto 0);
signal vs55:unsigned(15 downto 0);
signal vs56:unsigned(15 downto 0);
signal vs57:unsigned(15 downto 0);
signal vs58:unsigned(15 downto 0);
signal vs59:unsigned(15 downto 0);
signal vs60:unsigned(15 downto 0);
signal vs61:unsigned(15 downto 0);
signal vs62:unsigned(15 downto 0);
signal vs63:unsigned(15 downto 0);
```

```
signal vs64:unsigned(15 downto 0);



    -- Signals for the video controller
    signal Hcount : unsigned(9 downto 0);    -- Horizontal position (0-800)
    signal Vcount : unsigned(9 downto 0);    -- Vertical position (0-524)
    signal EndOfLine, EndOfField : std_logic;

    signal vga_hblank, vga_hsync,    vga_vblank, vga_vsync : std_logic;    -- Sync. signals

    signal rectangle_h1, rectangle_v1, rectangle1 : std_logic;    -- rectangle area
    signal rectangle_h2, rectangle_v2, rectangle2 : std_logic;    -- rectangle area
    signal rectangle_h3, rectangle_v3, rectangle3 : std_logic;    -- rectangle area
    signal rectangle_h4, rectangle_v4, rectangle4 : std_logic;    -- rectangle area
    signal rectangle_h5, rectangle_v5, rectangle5 : std_logic;    -- rectangle area
    signal rectangle_h6, rectangle_v6, rectangle6 : std_logic;    -- rectangle area
    signal rectangle_h7, rectangle_v7, rectangle7 : std_logic;    -- rectangle area
    signal rectangle_h8, rectangle_v8, rectangle8 : std_logic;    -- rectangle area
    signal rectangle_h9, rectangle_v9, rectangle9 : std_logic;    -- rectangle area
    signal rectangle_h10, rectangle_v10, rectangle10 : std_logic;    -- rectangle area
    signal rectangle_h11, rectangle_v11, rectangle11 : std_logic;    -- rectangle area
    signal rectangle_h12, rectangle_v12, rectangle12 : std_logic;    -- rectangle area
    signal rectangle_h13, rectangle_v13, rectangle13 : std_logic;    -- rectangle area
    signal rectangle_h14, rectangle_v14, rectangle14 : std_logic;    -- rectangle area
    signal rectangle_h15, rectangle_v15, rectangle15 : std_logic;    -- rectangle area
    signal rectangle_h16, rectangle_v16, rectangle16 : std_logic;    -- rectangle area
    signal rectangle_h17, rectangle_v17, rectangle17 : std_logic;    -- rectangle area
    signal rectangle_h18, rectangle_v18, rectangle18 : std_logic;    -- rectangle area
    signal rectangle_h19, rectangle_v19, rectangle19 : std_logic;    -- rectangle area
    signal rectangle_h20, rectangle_v20, rectangle20 : std_logic;    -- rectangle area
    signal rectangle_h21, rectangle_v21, rectangle21 : std_logic;    -- rectangle area
    signal rectangle_h22, rectangle_v22, rectangle22 : std_logic;    -- rectangle area
    signal rectangle_h23, rectangle_v23, rectangle23 : std_logic;    -- rectangle area
    signal rectangle_h24, rectangle_v24, rectangle24 : std_logic;    -- rectangle area
    signal rectangle_h25, rectangle_v25, rectangle25 : std_logic;    -- rectangle area
    signal rectangle_h26, rectangle_v26, rectangle26 : std_logic;    -- rectangle area
    signal rectangle_h27, rectangle_v27, rectangle27 : std_logic;    -- rectangle area
    signal rectangle_h28, rectangle_v28, rectangle28 : std_logic;    -- rectangle area
    signal rectangle_h29, rectangle_v29, rectangle29 : std_logic;    -- rectangle area
    signal rectangle_h30, rectangle_v30, rectangle30 : std_logic;    -- rectangle area
    signal rectangle_h31, rectangle_v31, rectangle31 : std_logic;    -- rectangle area
    signal rectangle_h32, rectangle_v32, rectangle32 : std_logic;    -- rectangle area
    signal rectangle_h33, rectangle_v33, rectangle33 : std_logic;    -- rectangle area
    signal rectangle_h34, rectangle_v34, rectangle34 : std_logic;    -- rectangle area
    signal rectangle_h35, rectangle_v35, rectangle35 : std_logic;    -- rectangle area
    signal rectangle_h36, rectangle_v36, rectangle36 : std_logic;    -- rectangle area
    signal rectangle_h37, rectangle_v37, rectangle37 : std_logic;    -- rectangle area
    signal rectangle_h38, rectangle_v38, rectangle38 : std_logic;    -- rectangle area
    signal rectangle_h39, rectangle_v39, rectangle39 : std_logic;    -- rectangle area
    signal rectangle_h40, rectangle_v40, rectangle40 : std_logic;    -- rectangle area
    signal rectangle_h41, rectangle_v41, rectangle41 : std_logic;    -- rectangle area
```

```
    signal rectangle_h42, rectangle_v42, rectangle42 : std_logic;    -- rectangle area
    signal rectangle_h43, rectangle_v43, rectangle43 : std_logic;    -- rectangle area
    signal rectangle_h44, rectangle_v44, rectangle44 : std_logic;    -- rectangle area
    signal rectangle_h45, rectangle_v45, rectangle45 : std_logic;    -- rectangle area
    signal rectangle_h46, rectangle_v46, rectangle46 : std_logic;    -- rectangle area
    signal rectangle_h47, rectangle_v47, rectangle47 : std_logic;    -- rectangle area
    signal rectangle_h48, rectangle_v48, rectangle48 : std_logic;    -- rectangle area
    signal rectangle_h49, rectangle_v49, rectangle49 : std_logic;    -- rectangle area
    signal rectangle_h50, rectangle_v50, rectangle50 : std_logic;    -- rectangle area
    signal rectangle_h51, rectangle_v51, rectangle51 : std_logic;    -- rectangle area
    signal rectangle_h52, rectangle_v52, rectangle52 : std_logic;    -- rectangle area
    signal rectangle_h53, rectangle_v53, rectangle53 : std_logic;    -- rectangle area
    signal rectangle_h54, rectangle_v54, rectangle54 : std_logic;    -- rectangle area
    signal rectangle_h55, rectangle_v55, rectangle55 : std_logic;    -- rectangle area
    signal rectangle_h56, rectangle_v56, rectangle56 : std_logic;    -- rectangle area
    signal rectangle_h57, rectangle_v57, rectangle57 : std_logic;    -- rectangle area
    signal rectangle_h58, rectangle_v58, rectangle58 : std_logic;    -- rectangle area
    signal rectangle_h59, rectangle_v59, rectangle59 : std_logic;    -- rectangle area
    signal rectangle_h60, rectangle_v60, rectangle60 : std_logic;    -- rectangle area
    signal rectangle_h61, rectangle_v61, rectangle61 : std_logic;    -- rectangle area
    signal rectangle_h62, rectangle_v62, rectangle62 : std_logic;    -- rectangle area
    signal rectangle_h63, rectangle_v63, rectangle63 : std_logic;    -- rectangle area
    signal rectangle_h64, rectangle_v64, rectangle64 : std_logic;    -- rectangle area

begin

clk0:    process (clk)
    begin
        if rising_edge(clk) then
            clk25 <= not clk25;
        end if;
    end process;

peripheral: process(clk)
  begin
        if rising_edge(clk) then
            if reset_n = '0' then

              vs1<=(others=>'0');
              vs2<=(others=>'0');
              vs3<=(others=>'0');
              vs4<=(others=>'0');
              vs5<=(others=>'0');
              vs6<=(others=>'0');
              vs7<=(others=>'0');
              vs8<=(others=>'0');
              vs9<=(others=>'0');
              vs10<=(others=>'0');
              vs11<=(others=>'0');
              vs12<=(others=>'0');
              vs13<=(others=>'0');
              vs14<=(others=>'0');
              vs15<=(others=>'0');
```

```
  vs16<=(others=>'0');
  vs17<=(others=>'0');
  vs18<=(others=>'0');
  vs19<=(others=>'0');
  vs20<=(others=>'0');
  vs21<=(others=>'0');
  vs22<=(others=>'0');
  vs23<=(others=>'0');
  vs24<=(others=>'0');
  vs25<=(others=>'0');
  vs26<=(others=>'0');
  vs27<=(others=>'0');
  vs28<=(others=>'0');
  vs29<=(others=>'0');
  vs30<=(others=>'0');
  vs31<=(others=>'0');
  vs32<=(others=>'0');
  vs33<=(others=>'0');
  vs34<=(others=>'0');
  vs35<=(others=>'0');
  vs36<=(others=>'0');
  vs37<=(others=>'0');
  vs38<=(others=>'0');
  vs39<=(others=>'0');
  vs40<=(others=>'0');
  vs41<=(others=>'0');
  vs42<=(others=>'0');
  vs43<=(others=>'0');
  vs44<=(others=>'0');
  vs45<=(others=>'0');
  vs46<=(others=>'0');
  vs47<=(others=>'0');
  vs48<=(others=>'0');
  vs49<=(others=>'0');
  vs50<=(others=>'0');
  vs51<=(others=>'0');
  vs52<=(others=>'0');
  vs53<=(others=>'0');
  vs54<=(others=>'0');
  vs55<=(others=>'0');
  vs56<=(others=>'0');
  vs57<=(others=>'0');
  vs58<=(others=>'0');
  vs59<=(others=>'0');
  vs60<=(others=>'0');
  vs61<=(others=>'0');
  vs62<=(others=>'0');
  vs63<=(others=>'0');
  vs64<=(others=>'0');
else
    if chipselect = '1' then
        if write = '1' then
            if address="0000000"
```

```
then      vs1 <= writedata(15 downto 0);
elsif address="0000001"
then      vs2<= writedata(15 downto 0);
elsif address="0000010"
then      vs3<= writedata(15 downto 0);
elsif address="0000011"
then      vs4<= writedata(15 downto 0);
elsif address="0000100"
then      vs5<= writedata(15 downto 0);
elsif address="0000101"
then      vs6<= writedata(15 downto 0);
elsif address="0000110"
then      vs7<= writedata(15 downto 0);
elsif address="0000111"
then      vs8<= writedata(15 downto 0);
elsif address="0001000"
   then      vs9 <= writedata(15 downto 0);
elsif address="0001001"
   then      vs10<= writedata(15 downto 0);
elsif address="0001010"
   then      vs11<= writedata(15 downto 0);
elsif address="0001011"
   then      vs12<= writedata(15 downto 0);
elsif address="0001100"
   then      vs13<= writedata(15 downto 0);
elsif address="0001101"
   then      vs14<= writedata(15 downto 0);
elsif address="0001110"
   then      vs15<= writedata(15 downto 0);
elsif address="0001111"
   then      vs16<= writedata(15 downto 0);

elsif address="0010000"
then      vs17 <= writedata(15 downto 0);
elsif address="0010001"
then      vs18<= writedata(15 downto 0);
elsif address="0010010"
then      vs19<= writedata(15 downto 0);
elsif address="0010011"
then      vs20<= writedata(15 downto 0);
elsif address="0010100"
then      vs21<= writedata(15 downto 0);
elsif address="0010101"
then      vs22<= writedata(15 downto 0);
elsif address="0010110"
then      vs23<= writedata(15 downto 0);
elsif address="0010111"
then      vs24<= writedata(15 downto 0);
elsif address="0011000"
   then      vs25 <= writedata(15 downto 0);
elsif address="0011001"
   then      vs26<= writedata(15 downto 0);
```

```
  elsif address="0011010"
  then      vs27 <= writedata(15 downto 0);
  elsif address="0011011"
  then      vs28<= writedata(15 downto 0);
  elsif address="0011100"
  then      vs29<= writedata(15 downto 0);
  elsif address="0011101"
  then      vs30<= writedata(15 downto 0);
  elsif address="0011110"
  then      vs31<= writedata(15 downto 0);
  elsif address="0011111"
  then      vs32<= writedata(15 downto 0);
  elsif address="0100000"
  then      vs33<= writedata(15 downto 0);
  elsif address="0100001"
  then      vs34<= writedata(15 downto 0);
  elsif address="0100010"
     then      vs35 <= writedata(15 downto 0);
  elsif address="0100011"
     then      vs36<= writedata(15 downto 0);

  elsif address="0100100"
  then      vs37 <= writedata(15 downto 0);
  elsif address="0100101"
  then      vs38<= writedata(15 downto 0);
  elsif address="0100110"
  then      vs39<= writedata(15 downto 0);
  elsif address="0100111"
  then      vs40<= writedata(15 downto 0);
  elsif address="0101000"
  then      vs41<= writedata(15 downto 0);
elsif address="0101001"
  then      vs42<= writedata(15 downto 0);
  elsif address="0101010"
  then      vs43<= writedata(15 downto 0);
  elsif address="0101011"
  then      vs44<= writedata(15 downto 0);
  elsif address="0101100"
     then      vs45 <= writedata(15 downto 0);
  elsif address="0101101"
     then      vs46<= writedata(15 downto 0);

  elsif address="0101110"
    then    vs47 <= writedata(15 downto 0);
  elsif address="0101111"
  then      vs48<= writedata(15 downto 0);
  elsif address="0110000"
  then      vs49<= writedata(15 downto 0);
  elsif address="0110001"
  then      vs50<= writedata(15 downto 0);
  elsif address="0110010"
  then      vs51<= writedata(15 downto 0);
  elsif address="0110011"
```

```
                then      vs52<= writedata(15 downto 0);
                elsif address="0110100"
                then      vs53<= writedata(15 downto 0);
                elsif address="0110101"
                then      vs54<= writedata(15 downto 0);
                elsif address="0110110"
                then    vs55 <= writedata(15 downto 0);
     elsif address="0110111"
                  then      vs56<= writedata(15 downto 0);

              elsif address="0111000"
                 then    vs57 <= writedata(15 downto 0);
       elsif address="0111001"
                then      vs58<= writedata(15 downto 0);
           elsif address="0111010"
                then      vs59<= writedata(15 downto 0);
elsif address="0111011"
                then      vs60<= writedata(15 downto 0);
                elsif address="0111100"
                then      vs61<= writedata(15 downto 0);
             elsif address="0111101"
                then      vs62<= writedata(15 downto 0);
                elsif address="0111110"
                then      vs63<= writedata(15 downto 0);
                elsif address="0111111"
                then      vs64<= writedata(15 downto 0);
                end if;


         end if;
    end if;
    if EndOfLine='1' and EndOfField='1'
    then RECTANGLE_VSTART1<=vs1;
            RECTANGLE_VSTART2<=vs2;
            RECTANGLE_VSTART3<=vs3;
            RECTANGLE_VSTART4<=vs4;
            RECTANGLE_VSTART5<=vs5;
            RECTANGLE_VSTART6<=vs6;
            RECTANGLE_VSTART7<=vs7;
            RECTANGLE_VSTART8<=vs8;
            RECTANGLE_VSTART9<=vs9;
            RECTANGLE_VSTART10<=vs10;
            RECTANGLE_VSTART11<=vs11;
            RECTANGLE_VSTART12<=vs12;
            RECTANGLE_VSTART13<=vs13;
            RECTANGLE_VSTART14<=vs14;
            RECTANGLE_VSTART15<=vs15;
            RECTANGLE_VSTART16<=vs16;

            RECTANGLE_VSTART17<=vs17;
            RECTANGLE_VSTART18<=vs18;
            RECTANGLE_VSTART19<=vs19;
            RECTANGLE_VSTART20<=vs20;
```

```
                RECTANGLE_VSTART21<=vs21;
                RECTANGLE_VSTART22<=vs22;
                RECTANGLE_VSTART23<=vs23;
                RECTANGLE_VSTART24<=vs24;
                RECTANGLE_VSTART25<=vs25;
                RECTANGLE_VSTART26<=vs26;
                RECTANGLE_VSTART27<=vs27;
                RECTANGLE_VSTART28<=vs28;
                RECTANGLE_VSTART29<=vs29;
                RECTANGLE_VSTART30<=vs30;

                RECTANGLE_VSTART31<=vs31;
                RECTANGLE_VSTART32<=vs32;
                RECTANGLE_VSTART33<=vs33;
                RECTANGLE_VSTART34<=vs34;
                RECTANGLE_VSTART35<=vs35;
                RECTANGLE_VSTART36<=vs36;
                RECTANGLE_VSTART37<=vs37;
                RECTANGLE_VSTART38<=vs38;
                RECTANGLE_VSTART39<=vs39;
                RECTANGLE_VSTART40<=vs40;

                RECTANGLE_VSTART41<=vs41;
                RECTANGLE_VSTART42<=vs42;
                RECTANGLE_VSTART43<=vs43;
                RECTANGLE_VSTART44<=vs44;
                RECTANGLE_VSTART45<=vs45;
                RECTANGLE_VSTART46<=vs46;
                RECTANGLE_VSTART47<=vs47;
                RECTANGLE_VSTART48<=vs48;
                RECTANGLE_VSTART49<=vs49;
                RECTANGLE_VSTART50<=vs50;

                RECTANGLE_VSTART51<=vs51;
                RECTANGLE_VSTART52<=vs52;
                RECTANGLE_VSTART53<=vs53;
                RECTANGLE_VSTART54<=vs54;
                RECTANGLE_VSTART55<=vs55;
                RECTANGLE_VSTART56<=vs56;
                RECTANGLE_VSTART57<=vs57;
                RECTANGLE_VSTART58<=vs58;
                RECTANGLE_VSTART59<=vs59;
                RECTANGLE_VSTART60<=vs60;
                RECTANGLE_VSTART61<=vs61;
                RECTANGLE_VSTART62<=vs62;
                RECTANGLE_VSTART63<=vs63;
                RECTANGLE_VSTART64<=vs64;

            end if;
        end if;
    end if;
end process peripheral;
```

```
-- Horizontal and vertical counters

HCounter : process (clk25)
begin
   if rising_edge(clk25) then
      if reset_n = '0' then
         Hcount <= (others => '0');
      elsif EndOfLine = '1' then
         Hcount <= (others => '0');
      else
         Hcount <= Hcount + 1;
      end if;
   end if;
end process HCounter;

EndOfLine <= '1' when Hcount = HTOTAL - 1 else '0';

VCounter: process (clk25)
begin
   if rising_edge(clk25) then
      if reset_n = '0' then
         Vcount <= (others => '0');
      elsif EndOfLine = '1' then
         if EndOfField = '1' then
            Vcount <= (others => '0');
         else
            Vcount <= Vcount + 1;
         end if;
      end if;
   end if;
end process VCounter;

EndOfField <= '1' when Vcount = VTOTAL - 1 else '0';

-- State machines to generate HSYNC, VSYNC, HBLANK, and VBLANK

HSyncGen : process (clk25)
begin
   if rising_edge(clk25) then
      if reset_n = '0' or EndOfLine = '1' then
         vga_hsync <= '1';
      elsif Hcount = HSYNC - 1 then
         vga_hsync <= '0';
      end if;
   end if;
end process HSyncGen;
```

```
HBlankGen : process (clk25)
begin
   if rising_edge(clk25) then
      if reset_n = '0' then
         vga_hblank <= '1';
      elsif Hcount = HSYNC + HBACK_PORCH then
         vga_hblank <= '0';
      elsif Hcount = HSYNC + HBACK_PORCH + HACTIVE then
         vga_hblank <= '1';
      end if;
   end if;
end process HBlankGen;

VSyncGen : process (clk25)
begin
   if rising_edge(clk25) then
      if reset_n = '0' then
         vga_vsync <= '1';
      elsif EndOfLine ='1' then
         if EndOfField = '1' then
            vga_vsync <= '1';
         elsif Vcount = VSYNC - 1 then
            vga_vsync <= '0';
         end if;
      end if;
   end if;
end process VSyncGen;

VBlankGen : process (clk25)
begin
   if rising_edge(clk25) then
      if reset_n = '0' then
         vga_vblank <= '1';
      elsif EndOfLine = '1' then
         if Vcount = VSYNC + VBACK_PORCH - 1 then
            vga_vblank <= '0';
         elsif Vcount = VSYNC + VBACK_PORCH + VACTIVE - 1 then
            vga_vblank <= '1';
         end if;
      end if;
   end if;
end process VBlankGen;

-- Rectangle generator

RectangleHGen : process (clk25)
begin
   if rising_edge(clk25) then

      if reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART1) then
         rectangle_h1 <= '1';
      elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND1 then
         rectangle_h1 <= '0';
```

```
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART2) then
   rectangle_h2 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND2 then
   rectangle_h2 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART3) then
   rectangle_h3 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND3 then
   rectangle_h3 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART4) then
   rectangle_h4 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND4 then
   rectangle_h4 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART5) then
   rectangle_h5 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND5 then
   rectangle_h5 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART6) then
   rectangle_h6 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND6 then
   rectangle_h6 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART7) then
   rectangle_h7 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND7 then
   rectangle_h7 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART8) then
   rectangle_h8 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND8 then
   rectangle_h8 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART9) then
   rectangle_h9 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND9 then
   rectangle_h9 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART10) then
   rectangle_h10 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND10 then
```

```
    rectangle_h10 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART11) then
    rectangle_h11 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND11 then
    rectangle_h11 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART12) then
    rectangle_h12 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND12 then
    rectangle_h12 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART13) then
    rectangle_h13 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND13 then
    rectangle_h13 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART14) then
    rectangle_h14 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND14 then
    rectangle_h14 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART15) then
    rectangle_h15 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND15 then
    rectangle_h15 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART16) then
    rectangle_h16 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND16 then
    rectangle_h16 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART17) then
    rectangle_h17 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND17 then
    rectangle_h17 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART18) then
    rectangle_h18 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND18 then
    rectangle_h18 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART19) then
    rectangle_h19 <= '1';
```

```
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND19 then
   rectangle_h19 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART20) then
   rectangle_h20 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND20 then
   rectangle_h20 <= '0';
end if;


if    reset_n = '0' or    (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART21) then
   rectangle_h21 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND21 then
   rectangle_h21 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART22) then
   rectangle_h22 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND22 then
   rectangle_h22 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART23) then
   rectangle_h23 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND23 then
   rectangle_h23 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART24) then
   rectangle_h24 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND24 then
   rectangle_h24 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART25) then
   rectangle_h25 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND25 then
   rectangle_h25 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART26) then
   rectangle_h26 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND26 then
   rectangle_h26 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART27) then
   rectangle_h27 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND27 then
   rectangle_h27 <= '0';
end if;
```

```
if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART28) then
   rectangle_h28 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND28 then
   rectangle_h28 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART29) then
   rectangle_h29 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND29 then
   rectangle_h29 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART30) then
   rectangle_h30 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND30 then
   rectangle_h30 <= '0';
end if;

if    reset_n = '0' or    (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART31) then
   rectangle_h31 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND31 then
   rectangle_h31 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART32) then
   rectangle_h32 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND32 then
   rectangle_h32 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART33) then
   rectangle_h33 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND33 then
   rectangle_h33 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART34) then
   rectangle_h34 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND34 then
   rectangle_h34 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART35) then
   rectangle_h35 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND35 then
   rectangle_h35 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART36) then
   rectangle_h36 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND36 then
   rectangle_h36 <= '0';
end if;
```

```
if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART37) then
   rectangle_h37 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND37 then
   rectangle_h37 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART38) then
   rectangle_h38 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND38 then
   rectangle_h38 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART39) then
   rectangle_h39 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND39 then
   rectangle_h39 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART40) then
   rectangle_h40 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND40 then
   rectangle_h40 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART41) then
   rectangle_h41 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND41 then
   rectangle_h41 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART42) then
   rectangle_h42 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND42 then
   rectangle_h42 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART43) then
   rectangle_h43 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND43 then
   rectangle_h43 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART44) then
   rectangle_h44 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND44 then
   rectangle_h44 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART45) then
   rectangle_h45 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND45 then
   rectangle_h45 <= '0';
```

```
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART46) then
   rectangle_h46 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND46 then
   rectangle_h46 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART47) then
   rectangle_h47 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND47 then
   rectangle_h47 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART48) then
   rectangle_h48 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND48 then
   rectangle_h48 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART49) then
   rectangle_h49 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND49 then
   rectangle_h49 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART50) then
   rectangle_h50 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND50 then
   rectangle_h50 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART51) then
   rectangle_h51 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND51 then
   rectangle_h51 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART52) then
   rectangle_h52 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND52 then
   rectangle_h52 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART53) then
   rectangle_h53 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND53 then
   rectangle_h53 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART54) then
   rectangle_h54 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND54 then
```

```
    rectangle_h54 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART55) then
    rectangle_h55 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND55 then
    rectangle_h55 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART56) then
    rectangle_h56 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND56 then
    rectangle_h56 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART57) then
    rectangle_h57 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND57 then
    rectangle_h57 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART58) then
    rectangle_h58 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND58 then
    rectangle_h58 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART59) then
    rectangle_h59 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND59 then
    rectangle_h59 <= '0';
end if;

if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART60) then
    rectangle_h60 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND60 then
    rectangle_h60 <= '0';
end if;



if     reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART61) then
    rectangle_h61 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND61 then
    rectangle_h61 <= '0';
end if;

if reset_n = '0' or    (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART62) then
    rectangle_h62 <= '1';
elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND62 then
    rectangle_h62 <= '0';
end if;
```

```
        if reset_n = '0' or    (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART63) then
            rectangle_h63 <= '1';
        elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND63 then
            rectangle_h63 <= '0';
        end if;

        if    reset_n = '0' or (Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HSTART64) then
            rectangle_h64 <= '1';
        elsif Hcount = HSYNC + HBACK_PORCH + RECTANGLE_HEND64 then
            rectangle_h64 <= '0';
        end if;

end if;

end process RectangleHGen;



RectangleVGen : process (clk25)
begin
    if rising_edge(clk25) then
        if reset_n = '0' then
            rectangle_v1 <= '0';
            rectangle_v2 <= '0';
            rectangle_v3 <= '0';
            rectangle_v4 <= '0';
            rectangle_v5 <= '0';
            rectangle_v6 <= '0';
            rectangle_v7 <= '0';
            rectangle_v8 <= '0';
            rectangle_v9 <= '0';
            rectangle_v10 <= '0';
            rectangle_v11 <= '0';
            rectangle_v12 <= '0';
            rectangle_v13 <= '0';
            rectangle_v14 <= '0';
            rectangle_v15 <= '0';
            rectangle_v16 <= '0';
            rectangle_v17 <= '0';
            rectangle_v18 <= '0';
            rectangle_v19 <= '0';
            rectangle_v20 <= '0';
            rectangle_v21 <= '0';
            rectangle_v22 <= '0';
            rectangle_v23 <= '0';
            rectangle_v24 <= '0';
            rectangle_v25 <= '0';
            rectangle_v26 <= '0';
            rectangle_v27 <= '0';
            rectangle_v28 <= '0';
            rectangle_v29 <= '0';
            rectangle_v30 <= '0';
            rectangle_v31 <= '0';
```

```
    rectangle_v32 <= '0';
    rectangle_v33 <= '0';
    rectangle_v34 <= '0';
    rectangle_v35 <= '0';
    rectangle_v36 <= '0';
    rectangle_v37 <= '0';
    rectangle_v38 <= '0';
    rectangle_v39 <= '0';
    rectangle_v40 <= '0';
    rectangle_v41 <= '0';
    rectangle_v42 <= '0';
    rectangle_v43 <= '0';
    rectangle_v44 <= '0';
    rectangle_v45 <= '0';
    rectangle_v46 <= '0';
    rectangle_v47 <= '0';
    rectangle_v48 <= '0';
    rectangle_v49 <= '0';
    rectangle_v50 <= '0';
    rectangle_v51 <= '0';
    rectangle_v52 <= '0';
    rectangle_v53 <= '0';
    rectangle_v54 <= '0';
    rectangle_v55 <= '0';
    rectangle_v56 <= '0';
    rectangle_v57 <= '0';
    rectangle_v58 <= '0';
    rectangle_v59 <= '0';
    rectangle_v60 <= '0';
    rectangle_v61 <= '0';
    rectangle_v62 <= '0';
    rectangle_v63 <= '0';
    rectangle_v64 <= '0';



elsif EndOfLine = '1' then

if    Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART1 then
    rectangle_v1 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND1 then
    rectangle_v1 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART2) then
    rectangle_v2 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND2 then
    rectangle_v2 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART3) then
    rectangle_v3 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND3 then
```

```
    rectangle_v3 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART4) then
    rectangle_v4 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND4 then
    rectangle_v4 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART5) then
    rectangle_v5 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND5 then
    rectangle_v5 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART6) then
    rectangle_v6 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND6 then
    rectangle_v6 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART7) then
    rectangle_v7 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND7 then
    rectangle_v7 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART8) then
    rectangle_v8 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND8 then
    rectangle_v8 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART9) then
    rectangle_v9 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND9 then
    rectangle_v9 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART10) then
    rectangle_v10 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND10 then
    rectangle_v10 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART11) then
    rectangle_v11 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND11 then
    rectangle_v11 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART12) then
    rectangle_v12 <= '1';
```

```
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND12 then
   rectangle_v12 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART13) then
   rectangle_v13 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND13 then
   rectangle_v13 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART14) then
   rectangle_v14 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND14 then
   rectangle_v14 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART15) then
   rectangle_v15 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND15 then
   rectangle_v15 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART16) then
   rectangle_v16 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND16 then
   rectangle_v16 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART17) then
   rectangle_v17 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND17 then
   rectangle_v17 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART18) then
   rectangle_v18 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND18 then
   rectangle_v18 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART19) then
   rectangle_v19 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND19 then
   rectangle_v19 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART20) then
   rectangle_v20 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND20 then
   rectangle_v20 <= '0';
end if;
```

```
if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART21) then
   rectangle_v21 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND21 then
   rectangle_v21 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART22) then
   rectangle_v22 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND22 then
   rectangle_v22 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART23) then
   rectangle_v23 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND23 then
   rectangle_v23 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART24) then
   rectangle_v24 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND24 then
   rectangle_v24 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART25) then
   rectangle_v25 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND25 then
   rectangle_v25 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART26) then
   rectangle_v26 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND26 then
   rectangle_v26 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART27) then
   rectangle_v27 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND27 then
   rectangle_v27 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART28) then
   rectangle_v28 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND28 then
   rectangle_v28 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART29) then
   rectangle_v29 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND29 then
   rectangle_v29 <= '0';
end if;
```

```
if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART30) then
   rectangle_v30 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND30 then
   rectangle_v30 <= '0';
end if;

if    (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART31) then
   rectangle_v31 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND31 then
   rectangle_v31 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART32) then
   rectangle_v32 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND32 then
   rectangle_v32 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART33) then
   rectangle_v33 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND33 then
   rectangle_v33 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART34) then
   rectangle_v34 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND34 then
   rectangle_v34 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART35) then
   rectangle_v35 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND35 then
   rectangle_v35 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART36) then
   rectangle_v36 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND36 then
   rectangle_v36 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART37) then
   rectangle_v37 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND37 then
   rectangle_v37 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART38) then
   rectangle_v38 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND38 then
   rectangle_v38 <= '0';
```

```
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART39) then
   rectangle_v39 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND39 then
   rectangle_v39 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART40) then
   rectangle_v40 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND40 then
   rectangle_v40 <= '0';
end if;

if(Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART41) then
   rectangle_v41 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND41 then
   rectangle_v41 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART42) then
   rectangle_v42 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND42 then
   rectangle_v42 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART43) then
   rectangle_v43 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND43 then
   rectangle_v43 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART44) then
   rectangle_v44 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND44 then
   rectangle_v44 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART45) then
   rectangle_v45 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND45 then
   rectangle_v45 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART46) then
   rectangle_v46 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND46 then
   rectangle_v46 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART47) then
   rectangle_v47 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND47 then
```

```
    rectangle_v47 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART48) then
    rectangle_v48 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND48 then
    rectangle_v48 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART49) then
    rectangle_v49 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND49 then
    rectangle_v49 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART50) then
    rectangle_v50 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND50 then
    rectangle_v50 <= '0';
end if;

if    (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART51) then
    rectangle_v51 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND51 then
    rectangle_v51 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART52) then
    rectangle_v52 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND52 then
    rectangle_v52 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART53) then
    rectangle_v53 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND53 then
    rectangle_v53 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART54) then
    rectangle_v54 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND54 then
    rectangle_v54 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART55) then
    rectangle_v55 <= '1';
elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND55 then
    rectangle_v55 <= '0';
end if;

if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART56) then
    rectangle_v56 <= '1';
```

```
    elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND56 then
       rectangle_v56 <= '0';
    end if;

    if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART57) then
       rectangle_v57 <= '1';
    elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND57 then
       rectangle_v57 <= '0';
    end if;

    if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART58) then
       rectangle_v58 <= '1';
    elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND58 then
       rectangle_v58 <= '0';
    end if;

    if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART59) then
       rectangle_v59 <= '1';
    elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND59 then
       rectangle_v59 <= '0';
    end if;

    if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART60) then
       rectangle_v60 <= '1';
    elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND60 then
       rectangle_v60 <= '0';
    end if;

    if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART61) then
       rectangle_v61 <= '1';
    elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND61 then
       rectangle_v61 <= '0';
    end if;

    if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART62) then
       rectangle_v62 <= '1';
    elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND62 then
       rectangle_v62 <= '0';
    end if;

    if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART63) then
       rectangle_v63 <= '1';
    elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND63 then
       rectangle_v63 <= '0';
    end if;

    if (Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VSTART64) then
       rectangle_v64 <= '1';
    elsif Vcount = VSYNC + VBACK_PORCH-1 + RECTANGLE_VEND64 then
       rectangle_v64 <= '0';
    end if;

end if;
```

```
    end if;
end process RectangleVGen;


  rectangle1 <= rectangle_h1 and rectangle_v1;
  rectangle2 <= rectangle_h2 and rectangle_v2;
  rectangle3 <= rectangle_h3 and rectangle_v3;
  rectangle4 <= rectangle_h4 and rectangle_v4;
  rectangle5 <= rectangle_h5 and rectangle_v5;
  rectangle6 <= rectangle_h6 and rectangle_v6;
  rectangle7 <= rectangle_h7 and rectangle_v7;
  rectangle8 <= rectangle_h8 and rectangle_v8;
  rectangle9 <= rectangle_h9 and rectangle_v9;
  rectangle10 <= rectangle_h10 and rectangle_v10;
  rectangle11 <= rectangle_h11 and rectangle_v11;
  rectangle12 <= rectangle_h12 and rectangle_v12;
  rectangle13 <= rectangle_h13 and rectangle_v13;
  rectangle14 <= rectangle_h14 and rectangle_v14;
  rectangle15 <= rectangle_h15 and rectangle_v15;
  rectangle16 <= rectangle_h16 and rectangle_v16;
  rectangle17 <= rectangle_h17 and rectangle_v17;
  rectangle18 <= rectangle_h18 and rectangle_v18;
  rectangle19 <= rectangle_h19 and rectangle_v19;
  rectangle20 <= rectangle_h20 and rectangle_v20;
  rectangle21 <= rectangle_h21 and rectangle_v21;
  rectangle22 <= rectangle_h22 and rectangle_v22;
  rectangle23 <= rectangle_h23 and rectangle_v23;
  rectangle24 <= rectangle_h24 and rectangle_v24;
  rectangle25 <= rectangle_h25 and rectangle_v25;
  rectangle26 <= rectangle_h26 and rectangle_v26;
  rectangle27 <= rectangle_h27 and rectangle_v27;
  rectangle28 <= rectangle_h28 and rectangle_v28;
  rectangle29 <= rectangle_h29 and rectangle_v29;
  rectangle30 <= rectangle_h30 and rectangle_v30;
  rectangle31 <= rectangle_h31 and rectangle_v31;
  rectangle32 <= rectangle_h32 and rectangle_v32;
  rectangle33 <= rectangle_h33 and rectangle_v33;
  rectangle34 <= rectangle_h34 and rectangle_v34;
  rectangle35 <= rectangle_h35 and rectangle_v35;
  rectangle36 <= rectangle_h36 and rectangle_v36;
  rectangle37 <= rectangle_h37 and rectangle_v37;
  rectangle38 <= rectangle_h38 and rectangle_v38;
  rectangle39 <= rectangle_h39 and rectangle_v39;
  rectangle40 <= rectangle_h40 and rectangle_v40;
  rectangle41 <= rectangle_h41 and rectangle_v41;
  rectangle42 <= rectangle_h42 and rectangle_v42;
  rectangle43 <= rectangle_h43 and rectangle_v43;
  rectangle44 <= rectangle_h44 and rectangle_v44;
  rectangle45 <= rectangle_h45 and rectangle_v45;
  rectangle46 <= rectangle_h46 and rectangle_v46;
  rectangle47 <= rectangle_h47 and rectangle_v47;
  rectangle48 <= rectangle_h48 and rectangle_v48;
  rectangle49 <= rectangle_h49 and rectangle_v49;
```

```
    rectangle50 <= rectangle_h50 and rectangle_v50;
    rectangle51 <= rectangle_h51 and rectangle_v51;
    rectangle52 <= rectangle_h52 and rectangle_v52;
    rectangle53 <= rectangle_h53 and rectangle_v53;
    rectangle54 <= rectangle_h54 and rectangle_v54;
    rectangle55 <= rectangle_h55 and rectangle_v55;
    rectangle56 <= rectangle_h56 and rectangle_v56;
    rectangle57 <= rectangle_h57 and rectangle_v57;
    rectangle58 <= rectangle_h58 and rectangle_v58;
    rectangle59 <= rectangle_h59 and rectangle_v59;
    rectangle60 <= rectangle_h60 and rectangle_v60;
    rectangle61 <= rectangle_h61 and rectangle_v61;
    rectangle62 <= rectangle_h62 and rectangle_v62;
    rectangle63 <= rectangle_h63 and rectangle_v63;
    rectangle64 <= rectangle_h64 and rectangle_v64;
    -- Registered video signals going to the video DAC

    VideoOut: process (clk25, reset_n)
    begin
        if reset_n = '0' then
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
        elsif clk25'event and clk25 = '1' then

if (rectangle1 ='1'or rectangle2 ='1'or rectangle3 ='1'or rectangle4 ='1'or rectangle5 ='1'
or rectangle6 ='1'or rectangle7 ='1'or    rectangle8 ='1'or    rectangle9 ='1'
or    rectangle10 ='1'or rectangle11 ='1'or rectangle12 ='1'or    rectangle13 ='1'
or rectangle14 ='1'or rectangle15 ='1'or    rectangle16 ='1'or    rectangle17 ='1'
or    rectangle18 ='1'or    rectangle19 ='1'or    rectangle20 ='1'or    rectangle21 ='1'
or    rectangle22 ='1'or    rectangle23 ='1'or    rectangle24 ='1'or    rectangle25 ='1'
or rectangle26 ='1'or rectangle27 ='1'or    rectangle28 ='1'or    rectangle29 ='1'
or    rectangle30 ='1'or    rectangle31 ='1'or    rectangle32 ='1'or    rectangle33 ='1'
or    rectangle34 ='1'or    rectangle35 ='1'or    rectangle36 ='1'or    rectangle37 ='1'
or    rectangle38 ='1'or rectangle39 ='1'or    rectangle40 ='1'or    rectangle41 ='1'
or    rectangle42 ='1'or    rectangle43 ='1'or    rectangle44 ='1'or rectangle45 ='1'
or    rectangle46 ='1'or    rectangle47 ='1'or rectangle48 ='1'or rectangle49 ='1'
or rectangle50 ='1'or rectangle51 ='1' or    rectangle52 ='1'or    rectangle53 ='1'
or    rectangle54 ='1'or    rectangle55 ='1'or    rectangle56 ='1'or    rectangle57 ='1'
or    rectangle58 ='1'or rectangle59 ='1'or    rectangle60 ='1'
or rectangle61 ='1'or rectangle62 ='1'
or    rectangle63 ='1'or    rectangle64 ='1') then

                VGA_R <= "1111111111";
                VGA_G <= "1111111111";
                VGA_B <= "1111111111";


            elsif vga_hblank = '0' and vga_vblank ='0' then
                VGA_R <= "0000000000";
                VGA_G <= "0000000000";
                VGA_B <= "0000000000";
            else
```

```
            VGA_R <= "0000000000";
            VGA_G <= "0000000000";
            VGA_B <= "0000000000";
        end if;
    end if;
end process VideoOut;



VGA_CLK <= clk25;
VGA_HS <= not vga_hsync;
VGA_VS <= not vga_vsync;
VGA_SYNC <= '0';
VGA_BLANK <= not (vga_hsync or vga_vsync);

end rtl;
```

6.12 Pseudo Code for double precision multiplication

```
Double x, Double y;
If x==0 || y==0;
        Return 0.0;
Else
  Bit operation to extract sign,exponent,fraction;
      fraction|=0x0010000000000000;
  temp=(fraction_x>>27)*(fraction_y>>27);// integer multiplication
  test=(0x8000000000000&temp)>>51; // test the position of first 1;
  If test==1{
            fraction_result=(0x7FFFFFFFFFFF&temp)<<1;
            exp_result=exp_x+exp_y-1022;
        }
  Else {
            fraction_result=(0x3FFFFFFFFFFF&temp)<<2;
            exp_result=exp_x+exp_y-1023;
        }
        Return outcome=(sign_result<<63)| (exp_result<<52)| fraction_result;
```

6.13 Pseudo Code for double precision add/subtraction:

```
Double x, Double y;
If x==0; return y;
Else if   y==0;return x;
Else
  If sign_x == sign_y; do addition;
      Bit operation to extract sign,exponent,fraction;
          fraction|=0x0010000000000000;
```

```
                    temp=(fraction_x)+(fraction_y>>(exp_x-exp_y));
        find the position of first '1' in temp to adjust exp_result and fraction_result;
  else do substraction;
        Bit operation to extract sign,exponent,fraction;
            Determine which exponent is bigger; // here we suppose exp_x is bigger;
        sign_result=signx;
                    temp=(fraction_x)-(fraction_y>>(exp_x-exp_y));
        find the position of first '1' in temp to adjust exp_result and fraction_result;
Return outcome=(sign_result<<63)| (exp_result<<52)| fraction_result;
```

6.14 C code for custom floating point multiplication and addition/substraction

```
inline double doubleadd(double x1, double y1){

    if(x1==0.0)
    {
      return y1;
    }

    else if(y1==0.0)
    {
      return x1;
    }
    else//none of 0
    {

  int test,acc,sign_result;
    long long temp,checkx,checky;
    unsigned long long mask_result=0x0000000000000000;
    unsigned long long exp_result;



    unsigned long long realx;
    unsigned long long * maskx=(unsigned long long *) &x1;
    long long signx,expx;
    signx=(0x8000000000000000&*maskx)>>63;
    expx= (0x7FF0000000000000&*maskx)>>52;
    realx= 0x000FFFFFFFFFFFFF&*maskx;
    //checkx=realx+expx;
    realx|=0x0010000000000000;

    unsigned long long realy;
    unsigned long long * masky=(unsigned long long *) &y1;
    long long signy,expy;
    signy=(0x8000000000000000&*masky)>>63;
    expy= (0x7FF0000000000000&*masky)>>52;
    realy= 0x000FFFFFFFFFFFFF&*masky;
    //checky=realy+expy;
```

```
realy|=0x0010000000000000;


  if(signx==signy)
  {
        sign_result=signx;
        if(expx>=expy)
        {
              temp=(realx)+(realy>>(expx-expy));
              test=(0x20000000000000&temp)>>53;
              if(test==1){
                    mask_result=(0x1FFFFFFFFFFFF&temp)>>1;
                    exp_result=expx+1;
              }
              else{
                    mask_result=(0x0FFFFFFFFFFFF&temp);
                    exp_result=expx;
              }
        }

        else//expx<expy
        {
              temp=(realy)+(realx>>(expy-expx));
              test=(0x20000000000000&temp)>>53;
              if(test==1){
                    mask_result=(0x1FFFFFFFFFFFF&temp)>>1;
                    exp_result=expy+1;
              }
              else{
                    mask_result=(0x0FFFFFFFFFFFF&temp);
                    exp_result=expy;
              }
        }
  }
  else//signx!=signy
  {
        if(expx>expy)
        {

              sign_result=signx;
              temp=(realx)-(realy>>(expx-expy));
              if(temp==0)    return 0;
              acc=0;test=(0x10000000000000&temp)>>52;
              while(test!=1){
                    temp<<=1;
                    test=(0x10000000000000&temp)>>52;
                    acc++;
              }
```

```
        mask_result=(0x0FFFFFFFFFFFF&temp);
            exp_result=expx-acc;
        /*test=(0x10000000000000&temp)>>52;
        if(test==0){
              mask_result=(0x07FFFFFFFFFFF&temp)<<1;
                  exp_result=expx-1;
        }
        else{
              mask_result=(0x0FFFFFFFFFFFF&temp);
                  exp_result=expx;
        }*/

}
else if(expx<expy)
{
        sign_result=signy;
        temp=(realy)-(realx>>(expy-expx));
        if(temp==0)    return 0;
        acc=0;test=(0x10000000000000&temp)>>52;
        while(test!=1){
              temp<<=1;
              test=(0x10000000000000&temp)>>52;
              acc++;
        }
        mask_result=(0x0FFFFFFFFFFFF&temp);
            exp_result=expy-acc;
}
else//expx=expy
{
        if(realx>=realy){
              sign_result=signx;
              temp=(realx)-(realy);
              if(temp==0)    return 0;
              acc=0;test=(0x10000000000000&temp)>>52;
              while(test!=1){
                    temp<<=1;
                    test=(0x10000000000000&temp)>>52;
                    acc++;
              }
              mask_result=(0x0FFFFFFFFFFFF&temp);
                  exp_result=expx-acc;
        }
        else{
              sign_result=signy;
              temp=(realy)-(realx);
              if(temp==0)    return 0;
              acc=0;test=(0x10000000000000&temp)>>52;
              while(test!=1){
```

```
                        temp<<=1;
                        test=(0x10000000000000&temp)>>52;
                        acc++;
                    }
                    mask_result=(0x0FFFFFFFFFFFFF&temp);
                        exp_result=expx-acc;
                }
            }
        }
        exp_result=(exp_result<<52);
        mask_result|=exp_result;
        if(sign_result==1)
            mask_result|=0x8000000000000000;
        else
             mask_result&=0x7FFFFFFFFFFFFFFF;
        double *result=(double *)&mask_result;
        double outcome=*result;
        return outcome;
   }

}

/*printf("x1 is %f, dewindow is %f,y1 is %f, buf is %f\n",
        x1,dewindow[k],y1,((*buf) [ch] [( (k + ( ((i+1)>>1) <<6) ) + bufOffset[ch]) & 0x3ff]));*/


inline double doublemul(double x1, double y1){


      if(x1==0.0||y1==0.0){
        double outcome=0;
        return outcome;
      }
      else
      {
  int test,sign_result;
      long long temp,checkx,checky;
      unsigned long long mask_result=0x0000000000000000;
      unsigned long long exp_result;



      unsigned long long realx;
      unsigned long long * maskx=(unsigned long long *) &x1;
      long signx,expx;
      signx=(0x8000000000000000&*maskx)>>63;
      expx= (0x7FF0000000000000&*maskx)>>52;
      realx= 0x000FFFFFFFFFFFFF&*maskx;
```

```
    //checkx=realx+expx;
    realx|=0x0010000000000000;

    unsigned long long realy;
    unsigned long long * masky=(unsigned long long *) &y1;
    long signy,expy;
    signy=(0x8000000000000000&*masky)>>63;
    expy= (0x7FF0000000000000&*masky)>>52;
    realy= 0x000FFFFFFFFFFFFF&*masky;
    //checky=realy+expy;
    realy|=0x0010000000000000;

    temp=(realx>>27)*(realy>>27);
    test=(0x8000000000000&temp)>>51;
    if(test==1){
        mask_result=(0x7FFFFFFFFFFFF&temp)<<1;
        exp_result=expx+expy-1022;//1074;
    }
    else{
        mask_result=(0x3FFFFFFFFFFFF&temp)<<2;
        exp_result=expx+expy-1023;//1075;
    }

    exp_result=(exp_result<<52);
    mask_result|=exp_result;
    sign_result=signx+signy;
    if(sign_result==1)
        mask_result|=0x8000000000000000;
    else
        mask_result&=0x7FFFFFFFFFFFFFFF;

    //double* result=&mask_result;
    double *result=(double *)&mask_result;
    double outcome=*result;
    return outcome;
  }

}
```