

String Manipulation and Probability

(SMAP)

Final Project Report

Emily Melissa Sillars (ems2331)

Swapnil Paliwal (sp3991)

Andrew Magid (aam2302)

Tushar Arora (ta2673)

Table of Contents

1. Introduction	4
2. Language Tutorial	5
2.1 GettingStarted	5
2.2 Building SMAP	5
2.3 Running Simple SMAP program	6
3. Language Reference Manual	6
3.1 Tokens	6
3.2 Comments	6
3.3 Identifiers	6
3.4 Keywords	7
3.5 DataTypes	7
3.6 Operators	7
3.6.1 Binary Operators	
3.6.2 Unary Operators	
3.7 Types	8
3.7.1 Built-in types (Probability)	
3.7.2 Built-in types (List)	
3.8 ProgramStructure	9
3.9 Declarations	9
3.9.1 GlobalDeclarations	9
3.9.2 FunctionDeclarations	9
3.10 Statements	11
3.10.1 Conditional statement	11
3.10.2 While statement	11
3.10.3 For Statement	11
3.10.4 Return Statement	12
3.10.5 Statement Block	12
3.11 Expressions	12
3.11.1 Identifiers	13
3.11.2 Literals	13
3.11.3 Negation	14
3.11.4 Operations	14
3.11.5 Comparison	14
3.11.6 Assignment	14
3.11.7 FunctionCall	15
4. Standard Library	18
4.1 Built-in functions	18
5. Architectural Design	36
5.1 Scanner(scanner.ml)	36
5.2 Parser(parse.mly,ast.ml)	36
5.3 SemanticChecking(semant.ml, sast.ml)	36
5.4 CodeGeneration(codegen.ml)	36
6. Test Plan	37
6.1 TestingSuiteandAutomation	37

6.2 ExampleTestPrograms	
37 6.2.1 SMAPCode1	37
6.2.2 SMAPCode2	46
6.2.3 SMAPCode3	
7. Appendix	
7.1 Source Files	
7.1.1 makefile	
7.1.2 scanner.mll	
7.1.3 ast.ml	
7.1.4 parser.mly	
7.1.5 sast.ml	
7.1.6 codegen.ml	
7.1.7 list.c	
7.1.8 prob.c	
7.1.9 testall.sh	

Chapter 1

Introduction

String Manipulation and Probability (SMAP) is an imperative and statically typed language with C-like syntax. The goal of this project is to add in a special type, along with the basic C features, which reflects probability and so the language extends the functionality by adding a new and a very unique data type, called the prob type which in a limited manner incorporates this feature of probability into SMAP. The prob type is polymorphic which means any type can be made into a probabilistic type the prob type identifier and it models a discrete distribution by matching probabilities to values. The extended goal of this feature is to get incorporated into games where attacks, objects and other random things that the game picks can be based on this probability feature rather than making it completely random.

Chapter 2

Language Tutorial

2.1 Getting Started

SMAP requires Ocaml and LLVM libraries to build and execute.

Mac OS Distribution

(TO INSTALL BREW)

```
$ curl -fsSL -o install.sh https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh  
$ /bin/bash install.sh
```

(TO INSTALL OPAM AND LLVM)

```
$ brew install opam  
$ brew install llvm  
$ opam install llvm
```

Ubuntu Distribution

(TO INSTALL OPAM AND LLVM)

```
$ sudo apt-get install opam  
$ sudo apt-get install llvm  
$ opam install llvm
```

Other Linux Distributions:

(TO INSTALL BREW)

```
$ curl -fsSL -o install.sh https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh  
$ /bin/bash install.sh
```

(TO INSTALL OPAM AND LLVM)

```
$ brew install llvm  
$ opam depect conf-llvm.6.0.0  
$ opam install llvm  
$ export PATH=/usr/local/opt/llvm@6/bin:$PATH
```

2.2 Building SMAP

Docker Environment setup

(TO INSTALL DOCKER)

```
$ apt install docker.io
```

(ADD YOURSELF TO THE DOCKER GROUP)

```
$ sudo usermod -aG docker <username>
```

(TEST INSTALLATION)

```
$ docker run hello-world
```

(RUN INSIDE THE CLONED DIRECTORY)

```
$ docker run --rm -it -v 'pwd' :/home/smap -w=/home/smap/pl
```

Clone and Build SMAP

(CLONE REPO)

```
$ git clone https://github.com/magidandrew/smap.git
```

(RUN THE DOCKER INSIDE SMAP)

```
$ cd smap; docker run --rm -it -v 'pwd' :/home/smap -w=/home/smap/plt
```

(MAKE FILES)

```
$ make all
```

(RUN THE TEST FILES)

```
$ ./testall.sh
```

(CLEAN THE EXECUTABLES)

```
$ make clean
```

2.2 Running a single SMAP program

```
$ ./testfile.sh <file path>
```

Chapter 3

Language Reference Manual

Lexical Conventions

3.1 Tokens

There are five classes of tokens: identifiers, keywords, string literals, operators and other separators. Other tokens are ignored because they act as separators, i.e. they separate the tokens. Blanks, horizontal and vertical tabs, newlines, formfeeds and comments are part of the group of tokens that are ignored while parsing the program.

3.2 Comments

Single-line comments are denoted by the characters `//` which ignore all text following them until a newline is met. Multi-line comments are denoted by the characters `/*` and `*/` which introduce and terminate the multi-line comment, respectively. Everything between the paired multi-line comment characters is ignored. Comments do not nest and do not occur in literals.

```
test > ≡ test-intGlobalAssignment.smap
1  int y = 3;
2  int w = 300;
3  int main()
4  | //this is a single line comment
5  | printint(w);
6  | w = 5;
7  | w = 400;
8  | /* this is
9  | | a
10 | | multi-line
11 | | comment */
12 | printint(w);
13 |
```

3.3 Identifiers

Identifiers are a sequence of letters and digits. SMAP is case-sensitive since lowercase and uppercase letters are distinguished to allow for camel case, and the underscore `_` is included to allow for snake case. Identifiers must begin with an underscore or letter.

<code>okayToUseCamelCase</code>	✓
<code>okay_to_use_snake_case</code>	✓
<code>OkayToStartWithCapital</code>	✓
<code>_alsoOkay</code>	✓
<code>9_not_okay_to_start_with_number</code>	X
<code>&not_okay_non_alphanumeric-=@#%</code>	X

3.4 Keywords

The following identifiers are reserved keywords in SMAP:

<code>bool</code>	<code>elif</code>	<code>if</code>	<code>prob</code>	<code>void</code>
	<code>else</code>	<code>int</code>	<code>return</code>	<code>while</code>
<code>case</code>	<code>false</code>	<code>list</code>	<code>string</code>	<code>continue</code>
<code>char</code>	<code>float</code>	<code>null</code>	<code>switch</code>	
	<code>for</code>		<code>true</code>	

3.5 Data Types (Primitive)

Data Types	Description
int	A sequence of digits in decimal format. Cannot start with a zero.
bool	Either true or false , implemented as integers 0 or 1.
char	A text character in single quotes, for example 'a'. Escape characters '\n', '\0', '\\', '\'', '\\"', represent new line, null terminator, regular slash, regular single quote and regular double quote respectively.
float	A sequence of digits in decimal format containing a single . for the decimal point.
string	A sequence of characters in double quotes, for example, "hello". String literals are implemented as dynamic lists of characters.

3.6 Operators

3.6.1 Binary Operators

There are four basic categories of binary operators, as well as concatenation.

Arithmetic: +, -, *, /

Bitwise: &, |, ^, >>, <<

Comparison: >, <, >=, <=, ==, and !=

Logical: ||, &&

These expressions group left to right and follow the general form

```
type expr =
  Binop of expr * binary_op * expr
```

// where **binop_op** is a binary operator and

// the expressions on either side of the operation are **of the same type**

```
expr PLUS    expr      { Binop($1, Add, $3)      }
| expr MINUS  expr      { Binop($1, Sub, $3)      }
| expr TIMES  expr      { Binop($1, Mul, $3)     }
| expr DIVIDE expr      { Binop($1, Div, $3)     }
| expr COMPEQ expr      { Binop($1, CompEq, $3)  }
| expr COMPLT expr      { Binop($1, CompLt, $3)  }
| expr COMPLEQ expr     { Binop($1, CompLeq, $3) }
| expr COMPGT expr      { Binop($1, CompGt, $3)  }
| expr COMPGEQ expr     { Binop($1, CompEq, $3)  }
| expr COMPNEQ expr     { Binop($1, CompNeq, $3) }
| expr RSHIFT expr      { Binop($1, RShift, $3)  }
| expr LSHIFT expr      { Binop($1, LShift, $3)  }
| expr BITAND expr      { Binop($1, BitAnd, $3)  }
| expr BITOR  expr      { Binop($1, BitOr, $3)   }
| expr XOR    expr      { Binop($1, Xor, $3)    }
```

The only exceptions to the same type rule requirement are arithmetic operations between floats and ints (in which ints are implicitly cast to floats), concatenation operator between a string and float, string and int, or string and char (where floats, ints, and chars are implicitly cast to strings), and right and left shifts on lists for deleting elements, which require an argument of type int on the right hand side.

3.6.2 Assignment Operator

The assignment operator is a special case of binary operators.

```
/* assignment expressions (special case of binary) */
| expr ASSIGN expr      { Assign($1, Equal, $3)    }
| expr ADDEQUAL expr    { Assign($1, PlusEqual, $3) }
| expr MINUSEQUAL expr  { Assign($1, MinusEqual, $3) }
| expr TIMESEQUAL expr  { Assign($1, TimesEqual, $3) }
| expr DIVEQUAL expr    { Assign($1, DivEqual, $3)  }
```

3.6.3 Unary Operators

```

Unop of unary_op * expr
type unary_op = BitNot | Not | Bang | Octothorpe | Neg

| MINUS expr          { Unop(Neg, $2)    }
| BITNOT expr         { Unop(BitNot, $2) }
| NOT expr            { Unop(Not, $2)   }
| expr NOT            { Unop(Bang, $1)  }
| expr OCTOTHORPE    { Unop(Octothorpe, $1) }

```

Unary expressions group left to right and are of the following forms:

```

~ expr // BitNot; flip all the bits
! expr // Not; a logical not; non zero values become 0, zeroes become 1
expr!  // Bang; evaluate a probability type
expr#  //Octothorpe; return the list of probabilities contained in the probability type
-expr  //Neg; denotes a negative number

```

3.7 Types (Built-in types)

SMAP supports two fundamental types: primitive types and built-in data types.

Primitive types : int, float, bool, string and char.

Built-in types: list and prob

```

type typ = Int | Bool | Float | Void | Char | String | Prob | List
type typ_name = typ list
type bind = typ_name * string

```

The SMAP AST represents types as a list of built-in and primitive types. Type checking in `semant.ml` rules out invalid sequences of types. For example, `[Char,Char]` is an invalid sequence of types.

3.7.1 Built-in types - Probability

```

expr PROBCOLON expr { ProbColon($1,$3) }

```

The probability type is polymorphic; any type can be made into a probabilistic type with the **prob** type qualifier. The probability type models a discrete distribution by matching probabilities to values. Probability types are initialized with a pair of non-empty lists separated by a colon. The left hand side list represents probabilities and must be of type `list float`. The right hand side list represents values and must be of type `list <type>`, where `<type>` is the type declared after the **prob** type qualifier in the declaration. Once initialized, the list of values inside the probability type is immutable.

```
prob type-identifier id = list float : list type-identifier;  
// ^where the two type-identifiers must match^!  
  
prob int someNumbers = list float : list int;  
// note that int matches int!  
  
prob int randNum = [0.25, 0.25, 0.50] : [1, 87, 6];  
// ^ example with concrete types
```

Accessing Fields

```
expr LENGTH { Length($1) }
```

The “length” of a probability type returns the minimum of the lengths of its two lists and is retrieved with `.length`:

```
prob int aNum = [0.3, 0.4, 0.3 ] : [7, 5]; // aNum#.length is 3, and  
// aNum[0,1].length is 2  
int len = aNum.length; // len is min(3,2) = 2
```

The `#` operator extracts the list of probabilities:

```
prob int randNum = [0.25, 0.25, 0.50] : [1, 87, 6];  
list float vals = randNum#; // get the list of probabilities
```

```
. // inside randNum
// vals == randNum# == [0,25, 0.25, 0.50] // evaluates to true
```

The ! pulls out a value based on the underlying distribution:

```
int num = randNum! //num os 1, 87 or 6
```

Note that if the length of the two lists are unequal, the value-list or probability-list is treated as “trimmed” to the minimum of the two list lengths. When the probability list is treated as “trimmed” its remaining elements are treated as normalized to sum to one.

```
randNum = [0.60,0.40] : randNum[0,1]; // [0.60,0.40]:[1, 87, 6]
num = randNum! // num3 is 1 or 87 (6 trimmed off)
randNum = [0.20, 0.25, 0.55]: [4, 2]; // randNum# is treated like
[0.45,0.55]
num = randNum!; // num is 4 or 2
```

```
randNum = [0.60,0.40] : randNum[0,1]; // [0.60,0.40]:[1, 87, 6]
num = randNum! // num3 is 1 or 87 (6 trimmed off)
randNum = [0.20, 0.25, 0.55]: [4, 2]; // randNum# is treated like [0.45,0.55]
num = randNum!; // num is 4 or 2
```

Modifying Probability Types with Transformations

We use transformations to safely modify a probability type’s underlying distribution.

Transformations use the arithmetic operators +, -, *, and / as part of a “zipwith” function, and then normalize the resulting list of floats to represent a new distribution. Transformation expressions can be generalized into four forms, with a probability type required on the left side

of the operation.

probType OP anotherProbType

probType OP list float

probType OP list int

probType OP float

probType OP int

Let's take a look at these transformations in action.

Ex. probType OP anotherProbType:

```
prob int num = [0.25, 0.15, 0.15, 0.45] : [6, 87, 4, 302] ;  
prob char letter = [0.25, 0.75 ] : ['a','b'];  
letter = letter * num; // letter is [0.36,0.64] : ['a','b']
```

How did we get the result [0.36,0.64] : ['a','b']? First, letter and num's values lists are immutable. Since the letter is on the left hand side of the transformation, its values are chosen to appear in the resulting probability type. **Note that this means a transformation between two probability types is associative but not commutative!** Next, letter# and num# are zipped with operator *. Transformations use the probability type's length (the minimum of its two list lengths) when performing zipwith, so the expression can be broken down as follows.

```
letter * num  
zipwith letter# num# *  
zipwith [0.25, 0.75] [0.25, 0.15, 0.15, 0.45] *  
[0.25*0.25, 0.75*0.15]  
[0.0625, 0.1125]
```

Then the transformation normalizes the probabilities,

$N([0.0625, 0.1125]) = [0.0625/0.157, 0.1125/0.157] = [0.36,0.64]$

and final result is

[0.36,0.64] : ['a','b']

Ex. probType OP list float:

```
prob int changedNum = Num * [0.2, 0.3] ; // [0.53, 0.047]:Num[0,1]  
// zipwith Num# letter# *  
// zipwith [0.25, 0.15, 0.15, 0.45] [0.2, 0.3] *  
// [0.25*0.2, 0.15*0.3]  
// [0.05, 0.045]  
// Normalize([0.05, 0.045]) = [0.05/0.095, 0.045/0.095] = [0.53, 0.047]  
// final result is [0.53, 0.047] : [6, 87, 4, 302]
```

Since integers are implicitly converted to floats in float op int or int op float expressions, probType OP list int expressions are implicitly converted expressions of probType OP list float.

Ex. probType OP float:

Expressions of **probType OP float** are implicitly converted to expressions of **probType OP list float**, where the float transforming the probability is promoted to a list of equal length with a single repeated value.

```
letter = letter + 3 // [0.48, 0.52]:['a', 'b']  
// since letter.length = 2, 3 gets promoted to [3, 3]  
// now we can do zip [0.36, 0.64] [3,3] + = [3.36, 3.64]  
// Normalize([3.36, 3.64]) = [3.36/7, 3.64/7,] = [0.48, 0.52]  
// final result is [0.48, 0.52]:['a', 'b']
```

Since integers are implicitly converted to floats in float op int or int op float expressions, probType OP int expressions are implicitly converted expressions of probType OP float.

Normalization

When a probability type is first initialized, or whenever a probability type is transformed, its internal probability list is normalized to ensure it represents a valid distribution. There are three steps in the normalization process.

- 1) Check both the probability list and the value list are non-empty. If this check fails, throw an error.
- 2) Check the probability list contains only non-negative elements, and that the sum of the elements > 0 . If this check fails, throw an error.
- 3) Divide each element by the sum

3.7.2 Primitive types - List

A list is a collection of homogenous mutable data, i.e. all of the elements inside the list would be of the same type and can be modified. A list is also dynamic, i.e. the list size is mutable.

Lists are defined by the list keyword followed by the singular type that the list will contain. Lists can be created with an initial set of data or empty and can have elements added to them dynamically.

```
| LBRACKET expr_list_opt RBRACKET { List_lit(List.rev $2) }
```

Note that square brackets surround an optional `expr_list`. This allows the empty list to be parsed as `[]`.

Creating a list:

```
list int myints = [3, 2, 5];  
list int emptylist;
```

Indexing a list using positional notation:

```
list_elt: ID index index_list_opt { ListElement($1,$2,$3) }  
index: LBRACKET expr RBRACKET { Index($2) }  
index_list_opt:  
{ [] }
```

```
| index index_list_opt      { $1::$2 }
```

An indexed list element consists of an identifier followed by at least one index. The `index_list_opt` allows for multidimensional array indexing.

```
myints[0] == 3; //evaluates to true
myints[0] = 1;
myints[0] == 1; //evaluates to true
```

Oftentimes, we need to append a value at the end or beginning of the list. This is easily done with the `>` and `<` symbol within list indexing:

```
| id ADDHEAD ASSIGN expr      { ListAddHead($1,$4)      }
| id ADDTAIL ASSIGN expr      { ListAddTail($1,$4)     }
```

```
myints[>] = 42;
myints[<] = 37;
// myints now looks like: [37, 3, 2, 5, 42]
```

```
| expr LENGTH                { Length($1)                }
```

Getting the length of the list is possible by using the `.length` operator which returns the length as an integer.

```
myints.length == 5; //evaluates to true
```

```
| expr CONCAT expr           { Binop($1, Concat, $3)     }
```

We allow the ++ operator on the list type to concatenate two lists of the same type together.

```
list int numsToAdd = [100];
list int concatResult = myints ++ numsToAdd;
// concatResult == [37, 3, 2, 5, 42, 100]
```

Delete elements from a list using the right and left shift operators:

```
| expr RSHIFT expr      { Binop($1, RShift, $3) }
| expr LSHIFT expr     { Binop($1, LShift, $3) }
```

```
myInts << 1;
// myints now looks like: [3, 2, 5, 42]
myInts >> 2;
// myints now looks like: [3, 2]
```

3.8 Program Structure

Program is the top level node in the syntax tree. Since we parse bottom-up all the parsing should end here.

program: vdecl list * func decl list

A program contains multiple (top-level) variable declarations and then multiple function declarations with arguments if necessary.

```
vdecl:
  typ_decl ID SEMICOLON { Vdecl (($1, $2), Noexpr) }
| typ_decl ID ASSIGN expr SEMICOLON { Vdecl (($1, $2), $4) }
```

A type decl consists of a list of type specifiers and type qualifiers.

```
Typ_decl:
```

```

| spec_qual_list      {$1}

typ_spec:
  INT      { Int      }
| BOOL     { Bool     }
| FLOAT    { Float    }
| VOID     { Void     }
| CHAR     { Char     }
| STRING   { String   }

typ_qual:
| PROB     { Prob     }
| LIST     { List     }

spec_qual_list:
| typ_spec spec_qual_list_opt {$1::$2}
| typ_qual spec_qual_list     {$1::$2}

spec_qual_list_opt:
| /*nothing*/  {[]}
| spec_qual_list {$1}

```

3.9 Declarations

There are two types of declarations that SMAP supports. One is the variable declarations and the other is function declarations (which also includes the main function).

3.9.1 Variable Declarations

Variable declarations happen at the start of the .smap file, or at the very start of each function.

Five Type specifiers: **int, bool, float, string, and char**

Two type qualifiers: **prob, and list**

Types are declared with or without an initial value. In the case of no initial expression,

```
vdecl:
  typ_decl ID SEMICOLON { Vdecl (($1, $2),Noexpr) }
| typ_decl ID ASSIGN expr SEMICOLON { Vdecl (($1, $2),$4)}
```

Note that the type of these declarations are distinct. The standard syntax for declaring the various types of variables are :

```
prob list int specialList;      // one of a group of possible lists
list prob int listOfSpecialNums; //a list where each elt is
                                //one of a group of possible integers
list prob list list int num;    //a list of where each elt is
                                // one of a group of possible lists of int lists
prob list list int num;        // one of a group of possible lists of int lists
```

Type Initialization

Types are initialized after declaration using the assignment operator. Initialization can occur immediately after declaration.

```
<qualifier>* <type-specifier> identifier = initializer;
    or some time later in a separate statement:
<qualifier>* <type-specifier> identifier; //declared here
...
identifier = initializer;                //initialized here
```

```
/* identifier is either an indexed list element
or regular text identifier */
id:
  ID          { Id($1)          }
| list_elt   { $1              }
```

Note that an identifier can be either a regular text identifier, or a list_elt. This ensures that indexed elements within an array can be assigned to.

3.9.2 Function Declarations

The main function is the place where all the code written inside or outside gets executed (code outside has to be called inside the main to get executed).

The functions in SMAP have the following declaration in the ast.ml

```
type func_decl = { typ_name : typ_name;
                  fname : string;
                  formals : bind list;
                  locals : vdecl list;
                  body : stmt list;
                  }
```

Here the typ_name is the return type of the function, the fname is the name of the function, the formals is a list of arguments to the functions, the locals is the list of local variable declaration to the function and the body is a list of statements that gets executed when there is a function call.

Finding the main function gets handled in the semant file where the “find_func” function is responsible for checking for the existence of the function.

```
118 (* make sure a main function is defined *)
119 let find_func s =
120   try StringMap.find s function_decls
121     with Not_found -> raise (Failure ("unrecognized function " ^ s)) in
122
123   let _ = find_func "main" in (* Ensure "main" is defined *)
124
```

The function declarator consists of a return type followed by the function signature. The function signature lists the return type (void keyword is used if no return

type) , name of the function, and a list of parameters inside parentheses (if the function has 0 arguments, an empty pair of parentheses are used).

The function definition consists of the function signature, and function body. The latter is a block of statements denoted with curly braces.

***A note on Blocks:** Curly braces surrounding a group of statements denotes a “block” of statements. Entering a block signals entrance into a new, more local scope. Contents like bindings (identifiers and their values) declared inside a block are not visible outside the block. Statements inside a block are executed in sequence.*

To declare a function without immediately specifying its definition, write out the type signature followed by a semicolon.

return-type name (type-specifier arg1_id, ... type-specifier argn_id)

To declare and define a function or to define a function declared earlier, write the type signature followed by the body of the function enclosed in curly braces.

```
return-type name (type-specifier arg1_id, ... type-specifier argn_id){  
    statement;  
    ...  
    statement;  
    return identifier; //where identifier is of type return-type  
}
```

An example of a function declaration in the SMAP is as follows -

```
int buildSky(list char theSky){
    int j = 0;
    prob char particle = [0.2,0.2,0.6] : ['*', 'o', ' '];
    for(j=0; j < theSky.length; j=j+1){
        theSky[j] = particle!;
    }
}
```

An example of an invalid program in SMAP is as follows -

```
test > ≡ fail-nomain.smap
1     int sup(){
2     |     print("nothing");
3     |     }
4
```

3.10 Statements

Statements are usually executed sequentially, i.e. in the order they are defined within a program. There are 3 special control flow statements that usually jump out of the scope or exit the program. Those 3 special statements are - return, exit and continue.

All the statements are defined in the ast file as follows -


```

33
34 type stmt =
35   Block of stmt list
36 | Expr of expr
37 | Return of expr
38 | If of expr * stmt
39 | If_Else of stmt * stmt
40 | If_Elif of stmt * stmt * stmt list
41 | If_Elif_Else of stmt * stmt * stmt list * stmt
42 | For of expr * expr * expr * stmt
43 | While of expr * stmt
44 | Break
45 | Continue
46 | Elif of expr * stmt (*only use inside an if statement, never alone!*)
47

```

3.10.1 Conditional Statement

The conditional statements execute based on the boolean condition of the block. If the result of the test expression inside the block is true, then the following list of statements inside the block is executed but if the test expression is false then it either goes in the else if block (if it exists) or in the else block and executes the list of statements based on the conditions.

The SMAP interprets true as 1 and false as 0 for the conditions. An example of how the parser interprets it as is -

```

if (expr) {
    // this block only executes if expr evaluates to true
}
elif(expr2){ //expr2 is only evaluated if expr evaluates to false
    // this block only executes if expr2 evaluates to true
}
else{
    // this block only executes if none of the other blocks executed
}

```

```
}
```

An example code of the if - elif -else is as follows -

```
test > ≡ test-if-elif.smap
 1  int main() {
 2      int x = 5;
 3      if (x > 5) {
 4          print(1);
 5      }
 6      elif (x == 5) {
 7          print(42);
 8      }
 9      elif (x == 4) {
10          print(24);
11      }
12 }
```

The output of this block will be 42

3.10.2 While Statement

The code inside the while statement will get executed until the conditional statement returns false.

```
while(condition){
    //statements
}
```

An example code from SMAP for while statement -

```
test > ≡ test-while.smap
1   int main()
2   {
3       int i = 5;
4       while (i > 0) {
5           print(i);
6           i = i - 1;
7       }
8       return 0;
9   }
```

3.10.3 For Statement

A for loop is used to repeat a specific statement block a given number of times by initializing a counter and incrementing/decrementing the counter until the given condition becomes false. For loops can be used to execute a set of statements, once for each item in a given sequence.

The ast handles the for statements in the following manner -

```
for(scoped variable assignment; condition; statement){
    //statements
}
```

An example from the SMAP is as follows -

```
test > ≡ test-for.smap
1  int main() {
2      int i = 0;
3      for(i=0; i<5; i=i+1) {
4          print(i);
5      }
6      return 0;
7  }
```

3.10.4 Return Statement

The return statement exits the scope that it is in and either stores the return value to the caller of the function.

Return in ast -

```
type stmt =
    | Return of expr
```

3.10.5 Statement Block

Statement Block consists of statements that are inside the curly braces, the ones which need to be executed when there is a call to a function which has the block. SMAP uses semicolon as a delimiter to indicate the end of a statement.

```
block:  LBRACE stmt_list RBRACE      { Block(List.rev $2)}
```

3.11 Expressions

Expressions are a sequence of operators, data types and operands that are meant to be evaluated.

```
type expr =
  Binop of expr * binary_op * expr
| Unop of unary_op * expr
| Assign of expr * assign_op * expr
| Int_lit of int
| Bool_lit of bool
| Float_lit of float
| Char_lit of char
| String_lit of string
| List_lit of expr list
| Id of string
| Cast of typ_name * expr
| ListElement of string * expr * expr list
| ListAddHead of expr * expr
| ListAddTail of expr * expr
| Length of expr
| ProbColon of expr * expr
| FunCall of string * expr list
| Index of expr
| Noexpr
```

Noexpr appears inside optional expressions, for example the `expr_opt` inside the for loop predicate. It also appears inside `vdecls` when a declared variable is left uninitialized.

Noexpr inside `vdecl`:

```
vdecl:
  typ_decl ID SEMICOLON { vdecl (($1, $2),Noexpr) }
| typ_decl ID ASSIGN expr SEMICOLON { vdecl (($1, $2),$4)}
```

Optional expressions inside for loop:

```
| FOR LPAREN expr_opt SEMICOLON expr_opt SEMICOLON expr_opt RPAREN
block          { For($3,$5,$7,$9) }
```

Noexpr in optional expressions:

```
expr_opt:      /* zero or one expressions */
              { Noexpr      }
| expr         { $1         }

expr_list_opt: /* zero or more expressions */{ []      }
| expr_list   { $1         }

expr_list:     /* one or more expressions */
expr          { [$1]       }
| expr_list COMMA expr { $3 :: $1   }
```

3.11.5 Function Call

```
| ID LPAREN args_opt RPAREN      { FunCall ($1,$3)      }
```

Chapter 4

Standard Library

4.1 List functions

- **init_list (list *l)** - Initialises a list and gives it an INITIAL_LIST_CAPACITY of 10 elements.
- **check_resizing(list *l)** - this function checks if the list is at capacity and returns 1 if resizing is required.
- **check_empty (list *l)** - checks if the list is empty and returns a value of 1 if it is empty and 0 otherwise.
- **resize (list *l)** - this function resizes the list and increases the size by multiplying the initial size by 2.
- **push_back (list *l, char *item)** - add the item to the back of the list l. This function has a void return type.
- **push_front (list *l, char *item)** - add the item to the front of the list l. This function has a void return type.
- **del_back (list *l)** - deletes the last element from the list , returns 0 on success and -1 on failure.
- **del_front (list *l)** - deletes the first element from the list, returns 0 on success and -1 on failure.
- **del_at (list *l, int i)** - deletes the ith element from the list l and returns 0 on success and -1 on failure.
- **get_back(list *l)** - returns the last element from the list. This function has a return type of char.
- **get_front(list *l)** - returns the first element from the list. This function has a return type of char.
- **get_at(list *l, int i)** - return the ith element from the list. Returns a NULL of the i value is out of bounds or if the list is empty and char otherwise.
- **print_list_int(list *l)** - prints all the elements from the list.

4.1 Prob functions

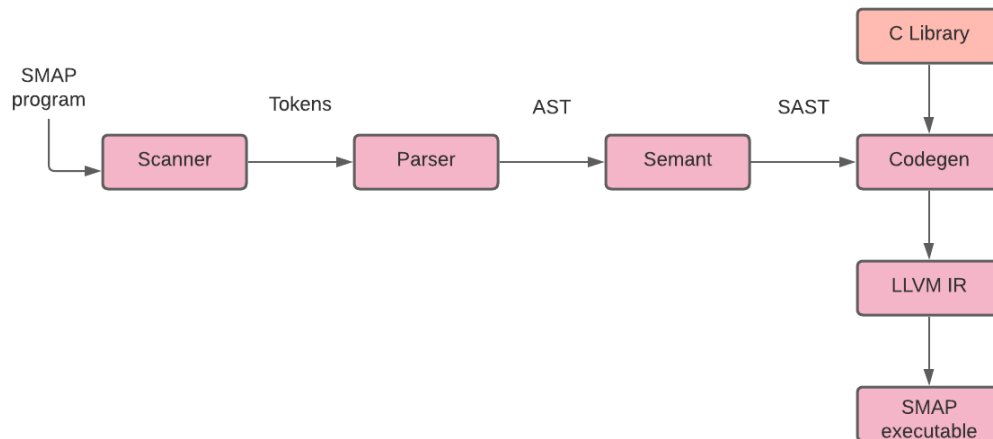
- **get_probs (prob *p)** -
- **get_vals (prob *p)** -
- **get_length (prob *p)** -
- **peek (prob *p)** -

4.2 String functions

- **corresponding_int(list* x, int index)** - Returns encoded version of the string at the index.
- **stringReverse(list* x)** - Reverse the input list and print a casted char of the values.
- **characterLocation(list* x, list* y)** - Returns the location of the character (taken from y) found in x.
- **stringLength(list* x)** - Returns the length of the list.

Chapter 6

Architecture Design



6.1 Scanner

The scanner takes in the SMAP program file (.smap) and generates tokens for all the code including the identifiers, keywords, operators, and literals specific to the SMAP language.

6.2 Parser

The parser takes in the tokens generated from the scanner and converts them into an abstract syntax tree (AST) based on the rules of SMAP grammar that are defined in the AST file.

6.3 Semantic Checking

This is the stage where the type check happens. The compiler checks the AST file for all the variables, functions and anything else that requires a check and converts it into an SAST file, which is similar to the AST but has grammar that has been type checked.

6.4 Code Generation

During code generation, semantically checked AST (which is the SAST now) is passed TO generate an intermediate representation (IR) of lower- level instructions. Outside library functions are also linked in this stage. Finally, the SMAP source code is converted into LLVM IR, which can then be compiled further into machine code.

Chapter 7

7.1 Testing Suite and Automation

SMAP was developed progressively over multiple branches. The main branch being the dune branch which implements all the prob type API and the list API including over 75 tests. All the team members contributed to the testing and the testing suite includes both the success and failing test cases.

7.2 Example Test program

7.2.1 SMAP Code 1

This code prints recursively. The code and the .ll file are -

SMAP CODE1 .smap file

//ToDo: Implement the military enigma

```
int character_location(list int x,int y){  
    int j = 0;  
    int i = 0;  
    for(i=0;i<26;i=i+1) {  
        if(x[i]==y) {  
            return i;  
        }  
    }  
    return j;  
}
```

```

list int rotor_shift(list int rotor_input,int position){

    int character_holder=rotor_input[0];

    int k=0;

    int l=0;

    int m=0;

    for(k=0;k<position;k=k+1){

        character_holder=rotor_input[0];

        for(l=0;l<25;l=l+1){

            m = l + 1;

            rotor_input[l]=rotor_input[m];

            if(l==24){

                rotor_input[25]=character_holder;

            }

        }

    }

    return rotor_input;

}

```

```

int return_cor_int(string what_message, int what_index) {

    int res = corresponding_int(what_message,what_index);

    return res;

}

```

```

list int initialize_input_text(string message_user,list int input_list_message) {

    int message_length_input = stringLength(message_user);

    int message_length_init = 0;

    int current_int_val = 0;

    for(message_length_init = 0; message_length_init <
message_length_input;message_length_init=message_length_init+1) {

        current_int_val = return_cor_int(message_user,message_length_init);

        input_list_message[message_length_init] = current_int_val;

    }

    return input_list_message;

}

```

```

int character_location_status(list int xx, int yy) {

    int my_response = character_location(xx,yy);

    return my_response;

}

```

```

int my_modulus_result(int base) {

    int modulus_results = modulus_operation(base,26);

```

```
    return modulus_results;
}
```

```
int rotor_direction_output(int input_output_char_loc, list int rotor_one_input_direct, list int
rotor_one_output_direct, list int rotor_two_input_direct, list int rotor_two_output_direct, list int
rotor_three_input_direct, list int rotor_three_output_direct){
```

```
    int index_reverse = 0;
```

```
    int character_location_holder=input_output_char_loc;
```

```
    int character_location_holder_two = 0;
```

```
    list int reflector= [0,1,2,3,4,5,6,3,8,9,10,6,12,10,12,8,4,1,5,19,2,21,21,9,0,19];
```

```
    //Getting character on rotor3 input tray
```

```
    character_location_holder=character_location(rotor_three_input_direct,rotor_three_output_direct[character_location_holder]);
```

```
    //Getting character on rotor2 input tray
```

```
    character_location_holder=character_location(rotor_two_input_direct,rotor_two_output_direct[character_location_holder]);
```

```
    //Getting character on rotor1 input tray
```

```
    character_location_holder=character_location(rotor_one_input_direct,rotor_one_output_direct[character_location_holder]);
```

```
    //Verified till above
```

//Getting the reflected element and the other reflected value

```
for(index_reverse=0;index_reverse<26;index_reverse=index_reverse+1){  
    if(characher_location_holder==index_reverse){  
  
    }  
    else{  
        if(reflector[characher_location_holder]==reflector[index_reverse]){  
            characher_location_holder_two=index_reverse;  
        }  
    }  
}
```

//Getting character on rotor1 output tray

```
characher_location_holder=character_location(rotor_one_output_direct,rotor_one_input_direct[characher_location_holder_two]);
```

//Getting character on rotor2 output tray

```
characher_location_holder=character_location(rotor_two_output_direct,rotor_two_input_direct[characher_location_holder]);
```

//Getting character on rotor3 output tray

```
characher_location_holder=character_location(rotor_three_output_direct,rotor_three_input_direct[characher_location_holder]);
```

//Cipher Character

```
return characher_location_holder;
```



```
count_rotor_three = count_rotor_three + 1;
```

```
if(count_rotor_three>0) {
```

```
    modulus_result = my_modulus_result(count_rotor_three);
```

```
    if(modulus_result == 0) {
```

```
        rotor_shift(rotor_two_e_input,1);
```

```
            rotor_shift(rotor_two_e_output,1);
```

```
        count_rotor_two = count_rotor_two + 1;
```

```
    if(count_rotor_two>0) {
```

```
        modulus_result_t = my_modulus_result(count_rotor_two);
```

```
        if(modulus_result_t == 0) {
```

```
            rotor_shift(rotor_one_e_input,1);
```

```
            rotor_shift(rotor_one_e_output,1);
```

```
            count_rotor_one = count_rotor_one + 1;
```

```
        }
```

```
    }
```

```
}
```

```
}
```

```
char_loc_holder=rotor_direction_output(char_loc_holder,rotor_one_e_input,rotor_one_e_output,rotor_two_e_input,
rotor_two_e_output,rotor_three_e_input,rotor_three_e_output);
```

```
    cipher_output_character[loop_counter_e_d]=input_output_char[char_loc_holder];
```

```
    }
```

```
    return cipher_output_character;
```

```
}
```

//ToDo: Implement the plugboard redirect

```
list<int> plugboard_redirect(list<int> text_redirect,int count_for_input, list<int> first_plugs, list<int> second_plugs, int
message_length_plug){
```

```
    //substitute a character with different character
```

```
    list<int> list_of_char_redirect = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25];
```

```
    list<int> sub_list_char = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25];
```

```
    list<int> temp_sub_val_holder = [1,1];
```

```
    list<int> loc_holder = [1,1];
```

```
    int loc_holding_for_text = 0;
```

```
    int counter=0;
```

```
    int string_redirect_length = 0;
```

//provide machine setting credentials

```
if(count_for_input==0) {  
    while(counter!=10) {  
        temp_sub_val_holder[0] = first_plugs[counter];  
        temp_sub_val_holder[1] = second_plugs[counter];  
        loc_holder[0] = character_location(list_of_char_redirect,list_of_char_redirect[temp_sub_val_holder[0]]);  
        loc_holder[1] = character_location(list_of_char_redirect,list_of_char_redirect[temp_sub_val_holder[1]]);  
        sub_list_char[(loc_holder[0])] = list_of_char_redirect[(loc_holder[1])];  
        sub_list_char[(loc_holder[1])] = list_of_char_redirect[(loc_holder[0])];  
        counter=counter+1;  
    }  
}
```

```
for(string_redirect_length=0;string_redirect_length<message_length_plug;string_redirect_length=string_re  
direct_length+1){  
    loc_holding_for_text = character_location(list_of_char_redirect,text_redirect[string_redirect_length]);  
    text_redirect[string_redirect_length] = sub_list_char[loc_holding_for_text];  
}
```

```
return text_redirect;
```


//ToDo: Rotor 3 initialize

```
list int rotor_three_out = [1,3,5,7,9,11,2,15,17,19,23,21,25,13,24,4,8,22,6,0,10,12,20,18,16,14];
```

```
list int rotor_three_in = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25];
```

//ToDo: Rotor 2 initialize

```
list int rotor_two_out = [0,9,3,10,18,8,17,20,23,1,11,7,22,19,12,2,16,6,25,13,15,24,5,21,14,4];
```

```
list int rotor_two_in = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25];
```

//ToDo: Rotor 1 initialize

```
list int rotor_one_out = [4,10,12,5,11,6,3,16,21,25,13,19,14,22,24,7,23,20,18,15,0,8,1,17,2,9];
```

```
list int rotor_one_in = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25];
```

//ToDo: Shift the inputs based on key and ring settings

//For Rotor - I

```
location_on_initialization_array=character_location(rotor_one_in,key[0]);
```

```
rotor_one_in = rotor_shift(rotor_one_in,location_on_initialization_array);
```

```
location_on_initialization_array=character_location(rotor_one_out,ring_setting[0]);
```

```
rotor_one_out = rotor_shift(rotor_one_out,location_on_initialization_array);
```

//For Rotor - II

```
location_on_initialization_array=character_location(rotor_two_in,key[1]);  
rotor_two_in = rotor_shift(rotor_two_in,location_on_initialization_array);
```

```
location_on_initialization_array=character_location(rotor_two_out,ring_setting[1]);  
rotor_two_out = rotor_shift(rotor_two_out,location_on_initialization_array);
```

//For Rotor - III

```
location_on_initialization_array=character_location(rotor_three_in,key[2]);  
rotor_three_in = rotor_shift(rotor_three_in,location_on_initialization_array);
```

```
location_on_initialization_array=character_location(rotor_three_out,ring_setting[2]);  
rotor_three_out = rotor_shift(rotor_three_out,location_on_initialization_array);
```

//ToDo: Initialize the message

```
input_text = initialize_input_text(messagez,input_text);
```

//ToDo: VERY IMPORTANT -> Plug re-direct the input message

```
input_text =  
plugboard_redirect(input_text,plugboard_input_count,first_plugs_init,second_plugs_init,length_input);
```

```
//ToDo: Initialize the encryption_decryption function

temp =
encryption_decryption(rotor_one_in,rotor_one_out,rotor_two_in,rotor_two_out,rotor_three_in,rotor_three_out,input
_text,length_input);

temp = plugboard_redirect(temp,plugboard_input_count,first_plugs_init,second_plugs_init,length_input);

print("Output text: ");

for(final_output_counter=0;final_output_counter<length_input;final_output_counter=final_output_counter
+1) {

corresponding_char(temp[final_output_counter]);

}

return 0;

}
```

SMAP Code1 .Il file

1. ; ModuleID = 'Smap'
2. source_filename = "Smap"
- 3.
4. %list = type { i8**, i32, i32 }
5. %prob = type { %list, %list, i32 }
- 6.
7. @test = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align 1
8. @newLine = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
9. @fmt = private unnamed_addr constant [3 x i8] c"%d\00", align 1
10. @fmt.1 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
11. @test.2 = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align 1
12. @newLine.3 = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
13. @fmt.4 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
14. @fmt.5 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
15. @test.6 = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align 1
16. @newLine.7 = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
17. @fmt.8 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
18. @fmt.9 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
19. @test.10 = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align 1
20. @newLine.11 = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
21. @fmt.12 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
22. @fmt.13 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
23. @test.14 = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align 1
24. @newLine.15 = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
25. @fmt.16 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
26. @fmt.17 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
27. @test.18 = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align 1
28. @newLine.19 = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
29. @fmt.20 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
30. @fmt.21 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
31. @test.22 = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align 1
32. @newLine.23 = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
33. @fmt.24 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
34. @fmt.25 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
35. @test.26 = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align 1
36. @newLine.27 = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
37. @fmt.28 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
38. @fmt.29 = private unnamed_addr constant [3 x i8] c"%f\00", align 1


```

39. @test.30 = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align
    1
40. @newLine.31 = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
41. @fmt.32 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
42. @fmt.33 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
43. @test.34 = private unnamed_addr constant [17 x i8] c"test test test!\0A\00", align
    1
44. @newLine.35 = private unnamed_addr constant [2 x i8] c"\0A\00", align 1
45. @fmt.36 = private unnamed_addr constant [3 x i8] c"%d\00", align 1
46. @fmt.37 = private unnamed_addr constant [3 x i8] c"%f\00", align 1
47.
48. declare i32 @printstr(i8*)
49.
50. declare i32 @corresponding_char(i32)
51.
52. declare i32 @printchar(i8)
53.
54. declare i32 @print_list_char(%list*)
55.
56. declare i32 @printb(i1)
57.
58. declare i32 @stringLength(%list*)
59.
60. declare i32 @stringReverse(%list*)
61.
62. declare i32 @absolute(i32)
63.
64. declare i32 @key_test(i32, i32)
65.
66. declare i32 @characterLocation(%list*, %list*)
67.
68. declare i32 @isCompDivisible(i32)
69.
70. declare i32 @divisible(i32, i32)
71.
72. declare i32 @power(i32, i32)
73.
74. declare i32 @ceilFloat(double)
75.
76. declare i32 @ascii(%list*)
77.
78. declare %list* @int_to_char(i32)
79.
80. declare i32 @corresponding_int(%list*, i32)

```

```

81.
82. declare { i8*, i32 } @testMakeStruct(i8*, i32)
83.
84. declare i32 @printint(i32)
85.
86. declare i32 @printf(i8*, ...)
87.
88. declare i32 @push_front(%list*, i8*, ...)
89.
90. declare i32 @push_back(%list*, i8*, ...)
91.
92. declare i32 @init_list(%list*, ...)
93.
94. declare i32 @init_prob(%prob*, %list*, %list*, ...)
95.
96. declare i8* @peek(%prob*, ...)
97.
98. declare i32 @list_length(%list*, ...)
99.
100.     declare i32 @modulus_operation(i32, i32, ...)
101.
102.     declare i32 @get_length(%prob*, ...)
103.
104.     declare i8* @get_at(%list*, i32, ...)
105.
106.     declare %list* @set_at(%list*, i32, i8*, ...)
107.
108.     declare i32 @print_list_int(%list*, ...)
109.
110.     declare i32 @print_list_float(%list*, ...)
111.
112.     declare %list* @get_vals(%prob*, ...)
113.
114.     declare %list* @get_probs(%prob*, ...)
115.
116.     declare i32 @print_prob_int_debug(%prob*, ...)
117.
118.     declare i32 @bad_add_head(%list*, i8*, ...)
119.
120.     declare i32 @very_bad_get_head(%list*, ...)
121.
122.     define i32 @main() {
123.     entry:
124.         %plugboard_input_count = alloca i32

```

```

125.      %final_output_counter = alloca i32
126.      %location_on_initialization_array = alloca i32
127.      %temp = alloca %list*
128.      %input_text = alloca %list*
129.      %messagez = alloca %list*
130.      %length_input = alloca i32
131.      %key = alloca %list*
132.      %ring_setting = alloca %list*
133.      %first_plugs_init = alloca %list*
134.      %second_plugs_init = alloca %list*
135.      %rotor_three_out = alloca %list*
136.      %rotor_three_in = alloca %list*
137.      %rotor_two_out = alloca %list*
138.      %rotor_two_in = alloca %list*
139.      %rotor_one_out = alloca %list*
140.      %rotor_one_in = alloca %list*
141.      store i32 0, i32* %plugboard_input_count
142.      store i32 0, i32* %final_output_counter
143.      store i32 0, i32* %location_on_initialization_array
144.      %mallocall = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
      (%list, %list* null, i32 1) to i32))
145.      %listAddr = bitcast i8* %mallocall to %list*
146.      %init_list = call i32 (%list*, ...) @init_list(%list* %listAddr)
147.      %tmp = alloca i32*
148.      %mallocall1 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
149.      %tmpAddr = bitcast i8* %mallocall1 to i32*
150.      store i32* %tmpAddr, i32** %tmp
151.      %tmpWithVal = load i32*, i32** %tmp
152.      store i32 1, i32* %tmpWithVal
153.      %asChar = bitcast i32* %tmpWithVal to i8*
154.      %push_back = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr, i8*
      %asChar)
155.      store %list* %listAddr, %list** %tmp
156.      %mallocall2 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
      (%list, %list* null, i32 1) to i32))
157.      %listAddr3 = bitcast i8* %mallocall2 to %list*
158.      %init_list4 = call i32 (%list*, ...) @init_list(%list* %listAddr3)
159.      %tmp5 = alloca i32*
160.      %mallocall6 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
161.      %tmpAddr7 = bitcast i8* %mallocall6 to i32*
162.      store i32* %tmpAddr7, i32** %tmp5
163.      %tmpWithVal8 = load i32*, i32** %tmp5

```

```

164.     store i32 1, i32* %tmpWithVal8
165.     %asChar9 = bitcast i32* %tmpWithVal8 to i8*
166.     %tmp10 = alloca i32*
167.     %alloca11 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
168.     %tmpAddr12 = bitcast i8* %alloca11 to i32*
169.     store i32* %tmpAddr12, i32** %tmp10
170.     %tmpWithVal13 = load i32*, i32** %tmp10
171.     store i32 1, i32* %tmpWithVal13
172.     %asChar14 = bitcast i32* %tmpWithVal13 to i8*
173.     %tmp15 = alloca i32*
174.     %alloca16 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
175.     %tmpAddr17 = bitcast i8* %alloca16 to i32*
176.     store i32* %tmpAddr17, i32** %tmp15
177.     %tmpWithVal18 = load i32*, i32** %tmp15
178.     store i32 1, i32* %tmpWithVal18
179.     %asChar19 = bitcast i32* %tmpWithVal18 to i8*
180.     %tmp20 = alloca i32*
181.     %alloca21 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
182.     %tmpAddr22 = bitcast i8* %alloca21 to i32*
183.     store i32* %tmpAddr22, i32** %tmp20
184.     %tmpWithVal23 = load i32*, i32** %tmp20
185.     store i32 1, i32* %tmpWithVal23
186.     %asChar24 = bitcast i32* %tmpWithVal23 to i8*
187.     %tmp25 = alloca i32*
188.     %alloca26 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
189.     %tmpAddr27 = bitcast i8* %alloca26 to i32*
190.     store i32* %tmpAddr27, i32** %tmp25
191.     %tmpWithVal28 = load i32*, i32** %tmp25
192.     store i32 1, i32* %tmpWithVal28
193.     %asChar29 = bitcast i32* %tmpWithVal28 to i8*
194.     %tmp30 = alloca i32*
195.     %alloca31 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
196.     %tmpAddr32 = bitcast i8* %alloca31 to i32*
197.     store i32* %tmpAddr32, i32** %tmp30
198.     %tmpWithVal33 = load i32*, i32** %tmp30
199.     store i32 1, i32* %tmpWithVal33
200.     %asChar34 = bitcast i32* %tmpWithVal33 to i8*
201.     %tmp35 = alloca i32*

```

```

202.      %alloca36 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
203.      %tmpAddr37 = bitcast i8* %alloca36 to i32*
204.      store i32* %tmpAddr37, i32** %tmp35
205.      %tmpWithVal38 = load i32*, i32** %tmp35
206.      store i32 1, i32* %tmpWithVal38
207.      %asChar39 = bitcast i32* %tmpWithVal38 to i8*
208.      %tmp40 = alloca i32*
209.      %alloca41 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
210.      %tmpAddr42 = bitcast i8* %alloca41 to i32*
211.      store i32* %tmpAddr42, i32** %tmp40
212.      %tmpWithVal43 = load i32*, i32** %tmp40
213.      store i32 1, i32* %tmpWithVal43
214.      %asChar44 = bitcast i32* %tmpWithVal43 to i8*
215.      %tmp45 = alloca i32*
216.      %alloca46 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
217.      %tmpAddr47 = bitcast i8* %alloca46 to i32*
218.      store i32* %tmpAddr47, i32** %tmp45
219.      %tmpWithVal48 = load i32*, i32** %tmp45
220.      store i32 1, i32* %tmpWithVal48
221.      %asChar49 = bitcast i32* %tmpWithVal48 to i8*
222.      %tmp50 = alloca i32*
223.      %alloca51 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
224.      %tmpAddr52 = bitcast i8* %alloca51 to i32*
225.      store i32* %tmpAddr52, i32** %tmp50
226.      %tmpWithVal53 = load i32*, i32** %tmp50
227.      store i32 1, i32* %tmpWithVal53
228.      %asChar54 = bitcast i32* %tmpWithVal53 to i8*
229.      %tmp55 = alloca i32*
230.      %alloca56 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
231.      %tmpAddr57 = bitcast i8* %alloca56 to i32*
232.      store i32* %tmpAddr57, i32** %tmp55
233.      %tmpWithVal58 = load i32*, i32** %tmp55
234.      store i32 1, i32* %tmpWithVal58
235.      %asChar59 = bitcast i32* %tmpWithVal58 to i8*
236.      %tmp60 = alloca i32*
237.      %alloca61 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
238.      %tmpAddr62 = bitcast i8* %alloca61 to i32*
239.      store i32* %tmpAddr62, i32** %tmp60

```

```

240.      %tmpWithVal63 = load i32*, i32** %tmp60
241.      store i32 1, i32* %tmpWithVal63
242.      %asChar64 = bitcast i32* %tmpWithVal63 to i8*
243.      %tmp65 = alloca i32*
244.      %allocaCall66 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
245.      %tmpAddr67 = bitcast i8* %allocaCall66 to i32*
246.      store i32* %tmpAddr67, i32** %tmp65
247.      %tmpWithVal68 = load i32*, i32** %tmp65
248.      store i32 1, i32* %tmpWithVal68
249.      %asChar69 = bitcast i32* %tmpWithVal68 to i8*
250.      %tmp70 = alloca i32*
251.      %allocaCall71 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
252.      %tmpAddr72 = bitcast i8* %allocaCall71 to i32*
253.      store i32* %tmpAddr72, i32** %tmp70
254.      %tmpWithVal73 = load i32*, i32** %tmp70
255.      store i32 1, i32* %tmpWithVal73
256.      %asChar74 = bitcast i32* %tmpWithVal73 to i8*
257.      %tmp75 = alloca i32*
258.      %allocaCall76 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
259.      %tmpAddr77 = bitcast i8* %allocaCall76 to i32*
260.      store i32* %tmpAddr77, i32** %tmp75
261.      %tmpWithVal78 = load i32*, i32** %tmp75
262.      store i32 1, i32* %tmpWithVal78
263.      %asChar79 = bitcast i32* %tmpWithVal78 to i8*
264.      %tmp80 = alloca i32*
265.      %allocaCall81 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
266.      %tmpAddr82 = bitcast i8* %allocaCall81 to i32*
267.      store i32* %tmpAddr82, i32** %tmp80
268.      %tmpWithVal83 = load i32*, i32** %tmp80
269.      store i32 1, i32* %tmpWithVal83
270.      %asChar84 = bitcast i32* %tmpWithVal83 to i8*
271.      %tmp85 = alloca i32*
272.      %allocaCall86 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
273.      %tmpAddr87 = bitcast i8* %allocaCall86 to i32*
274.      store i32* %tmpAddr87, i32** %tmp85
275.      %tmpWithVal88 = load i32*, i32** %tmp85
276.      store i32 1, i32* %tmpWithVal88
277.      %asChar89 = bitcast i32* %tmpWithVal88 to i8*
278.      %tmp90 = alloca i32*

```

```

279.      %alloca91 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
280.      %tmpAddr92 = bitcast i8* %alloca91 to i32*
281.      store i32* %tmpAddr92, i32** %tmp90
282.      %tmpWithVal93 = load i32*, i32** %tmp90
283.      store i32 1, i32* %tmpWithVal93
284.      %asChar94 = bitcast i32* %tmpWithVal93 to i8*
285.      %tmp95 = alloca i32*
286.      %alloca96 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
287.      %tmpAddr97 = bitcast i8* %alloca96 to i32*
288.      store i32* %tmpAddr97, i32** %tmp95
289.      %tmpWithVal98 = load i32*, i32** %tmp95
290.      store i32 1, i32* %tmpWithVal98
291.      %asChar99 = bitcast i32* %tmpWithVal98 to i8*
292.      %tmp100 = alloca i32*
293.      %alloca101 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
294.      %tmpAddr102 = bitcast i8* %alloca101 to i32*
295.      store i32* %tmpAddr102, i32** %tmp100
296.      %tmpWithVal103 = load i32*, i32** %tmp100
297.      store i32 1, i32* %tmpWithVal103
298.      %asChar104 = bitcast i32* %tmpWithVal103 to i8*
299.      %tmp105 = alloca i32*
300.      %alloca106 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
301.      %tmpAddr107 = bitcast i8* %alloca106 to i32*
302.      store i32* %tmpAddr107, i32** %tmp105
303.      %tmpWithVal108 = load i32*, i32** %tmp105
304.      store i32 1, i32* %tmpWithVal108
305.      %asChar109 = bitcast i32* %tmpWithVal108 to i8*
306.      %tmp110 = alloca i32*
307.      %alloca111 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
308.      %tmpAddr112 = bitcast i8* %alloca111 to i32*
309.      store i32* %tmpAddr112, i32** %tmp110
310.      %tmpWithVal113 = load i32*, i32** %tmp110
311.      store i32 1, i32* %tmpWithVal113
312.      %asChar114 = bitcast i32* %tmpWithVal113 to i8*
313.      %tmp115 = alloca i32*
314.      %alloca116 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
315.      %tmpAddr117 = bitcast i8* %alloca116 to i32*
316.      store i32* %tmpAddr117, i32** %tmp115

```

```

317.      %tmpWithVal118 = load i32*, i32** %tmp115
318.      store i32 1, i32* %tmpWithVal118
319.      %asChar119 = bitcast i32* %tmpWithVal118 to i8*
320.      %tmp120 = alloca i32*
321.      %allocaCall121 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
322.      %tmpAddr122 = bitcast i8* %allocaCall121 to i32*
323.      store i32* %tmpAddr122, i32** %tmp120
324.      %tmpWithVal123 = load i32*, i32** %tmp120
325.      store i32 1, i32* %tmpWithVal123
326.      %asChar124 = bitcast i32* %tmpWithVal123 to i8*
327.      %tmp125 = alloca i32*
328.      %allocaCall126 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
329.      %tmpAddr127 = bitcast i8* %allocaCall126 to i32*
330.      store i32* %tmpAddr127, i32** %tmp125
331.      %tmpWithVal128 = load i32*, i32** %tmp125
332.      store i32 1, i32* %tmpWithVal128
333.      %asChar129 = bitcast i32* %tmpWithVal128 to i8*
334.      %tmp130 = alloca i32*
335.      %allocaCall131 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
336.      %tmpAddr132 = bitcast i8* %allocaCall131 to i32*
337.      store i32* %tmpAddr132, i32** %tmp130
338.      %tmpWithVal133 = load i32*, i32** %tmp130
339.      store i32 1, i32* %tmpWithVal133
340.      %asChar134 = bitcast i32* %tmpWithVal133 to i8*
341.      %tmp135 = alloca i32*
342.      %allocaCall136 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
343.      %tmpAddr137 = bitcast i8* %allocaCall136 to i32*
344.      store i32* %tmpAddr137, i32** %tmp135
345.      %tmpWithVal138 = load i32*, i32** %tmp135
346.      store i32 1, i32* %tmpWithVal138
347.      %asChar139 = bitcast i32* %tmpWithVal138 to i8*
348.      %tmp140 = alloca i32*
349.      %allocaCall141 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
350.      %tmpAddr142 = bitcast i8* %allocaCall141 to i32*
351.      store i32* %tmpAddr142, i32** %tmp140
352.      %tmpWithVal143 = load i32*, i32** %tmp140
353.      store i32 1, i32* %tmpWithVal143
354.      %asChar144 = bitcast i32* %tmpWithVal143 to i8*
355.      %tmp145 = alloca i32*

```



```

356.      %alloca146 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
357.      %tmpAddr147 = bitcast i8* %alloca146 to i32*
358.      store i32* %tmpAddr147, i32** %tmp145
359.      %tmpWithVal148 = load i32*, i32** %tmp145
360.      store i32 1, i32* %tmpWithVal148
361.      %asChar149 = bitcast i32* %tmpWithVal148 to i8*
362.      %tmp150 = alloca i32*
363.      %alloca151 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
364.      %tmpAddr152 = bitcast i8* %alloca151 to i32*
365.      store i32* %tmpAddr152, i32** %tmp150
366.      %tmpWithVal153 = load i32*, i32** %tmp150
367.      store i32 1, i32* %tmpWithVal153
368.      %asChar154 = bitcast i32* %tmpWithVal153 to i8*
369.      %tmp155 = alloca i32*
370.      %alloca156 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
371.      %tmpAddr157 = bitcast i8* %alloca156 to i32*
372.      store i32* %tmpAddr157, i32** %tmp155
373.      %tmpWithVal158 = load i32*, i32** %tmp155
374.      store i32 1, i32* %tmpWithVal158
375.      %asChar159 = bitcast i32* %tmpWithVal158 to i8*
376.      %tmp160 = alloca i32*
377.      %alloca161 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
378.      %tmpAddr162 = bitcast i8* %alloca161 to i32*
379.      store i32* %tmpAddr162, i32** %tmp160
380.      %tmpWithVal163 = load i32*, i32** %tmp160
381.      store i32 1, i32* %tmpWithVal163
382.      %asChar164 = bitcast i32* %tmpWithVal163 to i8*
383.      %tmp165 = alloca i32*
384.      %alloca166 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
385.      %tmpAddr167 = bitcast i8* %alloca166 to i32*
386.      store i32* %tmpAddr167, i32** %tmp165
387.      %tmpWithVal168 = load i32*, i32** %tmp165
388.      store i32 1, i32* %tmpWithVal168
389.      %asChar169 = bitcast i32* %tmpWithVal168 to i8*
390.      %tmp170 = alloca i32*
391.      %alloca171 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
392.      %tmpAddr172 = bitcast i8* %alloca171 to i32*
393.      store i32* %tmpAddr172, i32** %tmp170

```

```

394.      %tmpWithVal173 = load i32*, i32** %tmp170
395.      store i32 1, i32* %tmpWithVal173
396.      %asChar174 = bitcast i32* %tmpWithVal173 to i8*
397.      %tmp175 = alloca i32*
398.      %allocaCall176 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
399.      %tmpAddr177 = bitcast i8* %allocaCall176 to i32*
400.      store i32* %tmpAddr177, i32** %tmp175
401.      %tmpWithVal178 = load i32*, i32** %tmp175
402.      store i32 1, i32* %tmpWithVal178
403.      %asChar179 = bitcast i32* %tmpWithVal178 to i8*
404.      %tmp180 = alloca i32*
405.      %allocaCall181 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
406.      %tmpAddr182 = bitcast i8* %allocaCall181 to i32*
407.      store i32* %tmpAddr182, i32** %tmp180
408.      %tmpWithVal183 = load i32*, i32** %tmp180
409.      store i32 1, i32* %tmpWithVal183
410.      %asChar184 = bitcast i32* %tmpWithVal183 to i8*
411.      %tmp185 = alloca i32*
412.      %allocaCall186 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
413.      %tmpAddr187 = bitcast i8* %allocaCall186 to i32*
414.      store i32* %tmpAddr187, i32** %tmp185
415.      %tmpWithVal188 = load i32*, i32** %tmp185
416.      store i32 1, i32* %tmpWithVal188
417.      %asChar189 = bitcast i32* %tmpWithVal188 to i8*
418.      %tmp190 = alloca i32*
419.      %allocaCall191 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
420.      %tmpAddr192 = bitcast i8* %allocaCall191 to i32*
421.      store i32* %tmpAddr192, i32** %tmp190
422.      %tmpWithVal193 = load i32*, i32** %tmp190
423.      store i32 1, i32* %tmpWithVal193
424.      %asChar194 = bitcast i32* %tmpWithVal193 to i8*
425.      %tmp195 = alloca i32*
426.      %allocaCall196 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
427.      %tmpAddr197 = bitcast i8* %allocaCall196 to i32*
428.      store i32* %tmpAddr197, i32** %tmp195
429.      %tmpWithVal198 = load i32*, i32** %tmp195
430.      store i32 1, i32* %tmpWithVal198
431.      %asChar199 = bitcast i32* %tmpWithVal198 to i8*
432.      %tmp200 = alloca i32*

```

```

433.      %alloca201 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
434.      %tmpAddr202 = bitcast i8* %alloca201 to i32*
435.      store i32* %tmpAddr202, i32** %tmp200
436.      %tmpWithVal203 = load i32*, i32** %tmp200
437.      store i32 1, i32* %tmpWithVal203
438.      %asChar204 = bitcast i32* %tmpWithVal203 to i8*
439.      %tmp205 = alloca i32*
440.      %alloca206 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
441.      %tmpAddr207 = bitcast i8* %alloca206 to i32*
442.      store i32* %tmpAddr207, i32** %tmp205
443.      %tmpWithVal208 = load i32*, i32** %tmp205
444.      store i32 1, i32* %tmpWithVal208
445.      %asChar209 = bitcast i32* %tmpWithVal208 to i8*
446.      %tmp210 = alloca i32*
447.      %alloca211 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
448.      %tmpAddr212 = bitcast i8* %alloca211 to i32*
449.      store i32* %tmpAddr212, i32** %tmp210
450.      %tmpWithVal213 = load i32*, i32** %tmp210
451.      store i32 1, i32* %tmpWithVal213
452.      %asChar214 = bitcast i32* %tmpWithVal213 to i8*
453.      %tmp215 = alloca i32*
454.      %alloca216 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
455.      %tmpAddr217 = bitcast i8* %alloca216 to i32*
456.      store i32* %tmpAddr217, i32** %tmp215
457.      %tmpWithVal218 = load i32*, i32** %tmp215
458.      store i32 1, i32* %tmpWithVal218
459.      %asChar219 = bitcast i32* %tmpWithVal218 to i8*
460.      %tmp220 = alloca i32*
461.      %alloca221 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
462.      %tmpAddr222 = bitcast i8* %alloca221 to i32*
463.      store i32* %tmpAddr222, i32** %tmp220
464.      %tmpWithVal223 = load i32*, i32** %tmp220
465.      store i32 1, i32* %tmpWithVal223
466.      %asChar224 = bitcast i32* %tmpWithVal223 to i8*
467.      %tmp225 = alloca i32*
468.      %alloca226 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
469.      %tmpAddr227 = bitcast i8* %alloca226 to i32*
470.      store i32* %tmpAddr227, i32** %tmp225

```

```

471.      %tmpWithVal228 = load i32*, i32** %tmp225
472.      store i32 1, i32* %tmpWithVal228
473.      %asChar229 = bitcast i32* %tmpWithVal228 to i8*
474.      %tmp230 = alloca i32*
475.      %allocaCall231 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
476.      %tmpAddr232 = bitcast i8* %allocaCall231 to i32*
477.      store i32* %tmpAddr232, i32** %tmp230
478.      %tmpWithVal233 = load i32*, i32** %tmp230
479.      store i32 1, i32* %tmpWithVal233
480.      %asChar234 = bitcast i32* %tmpWithVal233 to i8*
481.      %tmp235 = alloca i32*
482.      %allocaCall236 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
483.      %tmpAddr237 = bitcast i8* %allocaCall236 to i32*
484.      store i32* %tmpAddr237, i32** %tmp235
485.      %tmpWithVal238 = load i32*, i32** %tmp235
486.      store i32 1, i32* %tmpWithVal238
487.      %asChar239 = bitcast i32* %tmpWithVal238 to i8*
488.      %tmp240 = alloca i32*
489.      %allocaCall241 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
490.      %tmpAddr242 = bitcast i8* %allocaCall241 to i32*
491.      store i32* %tmpAddr242, i32** %tmp240
492.      %tmpWithVal243 = load i32*, i32** %tmp240
493.      store i32 1, i32* %tmpWithVal243
494.      %asChar244 = bitcast i32* %tmpWithVal243 to i8*
495.      %tmp245 = alloca i32*
496.      %allocaCall246 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
497.      %tmpAddr247 = bitcast i8* %allocaCall246 to i32*
498.      store i32* %tmpAddr247, i32** %tmp245
499.      %tmpWithVal248 = load i32*, i32** %tmp245
500.      store i32 1, i32* %tmpWithVal248
501.      %asChar249 = bitcast i32* %tmpWithVal248 to i8*
502.      %tmp250 = alloca i32*
503.      %allocaCall251 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
504.      %tmpAddr252 = bitcast i8* %allocaCall251 to i32*
505.      store i32* %tmpAddr252, i32** %tmp250
506.      %tmpWithVal253 = load i32*, i32** %tmp250
507.      store i32 1, i32* %tmpWithVal253
508.      %asChar254 = bitcast i32* %tmpWithVal253 to i8*
509.      %tmp255 = alloca i32*

```

```

510.      %alloca256 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
511.      %tmpAddr257 = bitcast i8* %alloca256 to i32*
512.      store i32* %tmpAddr257, i32** %tmp255
513.      %tmpWithVal258 = load i32*, i32** %tmp255
514.      store i32 1, i32* %tmpWithVal258
515.      %asChar259 = bitcast i32* %tmpWithVal258 to i8*
516.      %tmp260 = alloca i32*
517.      %alloca261 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
518.      %tmpAddr262 = bitcast i8* %alloca261 to i32*
519.      store i32* %tmpAddr262, i32** %tmp260
520.      %tmpWithVal263 = load i32*, i32** %tmp260
521.      store i32 1, i32* %tmpWithVal263
522.      %asChar264 = bitcast i32* %tmpWithVal263 to i8*
523.      %tmp265 = alloca i32*
524.      %alloca266 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
525.      %tmpAddr267 = bitcast i8* %alloca266 to i32*
526.      store i32* %tmpAddr267, i32** %tmp265
527.      %tmpWithVal268 = load i32*, i32** %tmp265
528.      store i32 1, i32* %tmpWithVal268
529.      %asChar269 = bitcast i32* %tmpWithVal268 to i8*
530.      %tmp270 = alloca i32*
531.      %alloca271 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
532.      %tmpAddr272 = bitcast i8* %alloca271 to i32*
533.      store i32* %tmpAddr272, i32** %tmp270
534.      %tmpWithVal273 = load i32*, i32** %tmp270
535.      store i32 1, i32* %tmpWithVal273
536.      %asChar274 = bitcast i32* %tmpWithVal273 to i8*
537.      %tmp275 = alloca i32*
538.      %alloca276 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
539.      %tmpAddr277 = bitcast i8* %alloca276 to i32*
540.      store i32* %tmpAddr277, i32** %tmp275
541.      %tmpWithVal278 = load i32*, i32** %tmp275
542.      store i32 1, i32* %tmpWithVal278
543.      %asChar279 = bitcast i32* %tmpWithVal278 to i8*
544.      %tmp280 = alloca i32*
545.      %alloca281 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
546.      %tmpAddr282 = bitcast i8* %alloca281 to i32*
547.      store i32* %tmpAddr282, i32** %tmp280

```

```

548.      %tmpWithVal283 = load i32*, i32** %tmp280
549.      store i32 1, i32* %tmpWithVal283
550.      %asChar284 = bitcast i32* %tmpWithVal283 to i8*
551.      %tmp285 = alloca i32*
552.      %allocaCall286 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
553.      %tmpAddr287 = bitcast i8* %allocaCall286 to i32*
554.      store i32* %tmpAddr287, i32** %tmp285
555.      %tmpWithVal288 = load i32*, i32** %tmp285
556.      store i32 1, i32* %tmpWithVal288
557.      %asChar289 = bitcast i32* %tmpWithVal288 to i8*
558.      %tmp290 = alloca i32*
559.      %allocaCall291 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
560.      %tmpAddr292 = bitcast i8* %allocaCall291 to i32*
561.      store i32* %tmpAddr292, i32** %tmp290
562.      %tmpWithVal293 = load i32*, i32** %tmp290
563.      store i32 1, i32* %tmpWithVal293
564.      %asChar294 = bitcast i32* %tmpWithVal293 to i8*
565.      %tmp295 = alloca i32*
566.      %allocaCall296 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
567.      %tmpAddr297 = bitcast i8* %allocaCall296 to i32*
568.      store i32* %tmpAddr297, i32** %tmp295
569.      %tmpWithVal298 = load i32*, i32** %tmp295
570.      store i32 1, i32* %tmpWithVal298
571.      %asChar299 = bitcast i32* %tmpWithVal298 to i8*
572.      %tmp300 = alloca i32*
573.      %allocaCall301 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
574.      %tmpAddr302 = bitcast i8* %allocaCall301 to i32*
575.      store i32* %tmpAddr302, i32** %tmp300
576.      %tmpWithVal303 = load i32*, i32** %tmp300
577.      store i32 1, i32* %tmpWithVal303
578.      %asChar304 = bitcast i32* %tmpWithVal303 to i8*
579.      %tmp305 = alloca i32*
580.      %allocaCall306 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
581.      %tmpAddr307 = bitcast i8* %allocaCall306 to i32*
582.      store i32* %tmpAddr307, i32** %tmp305
583.      %tmpWithVal308 = load i32*, i32** %tmp305
584.      store i32 1, i32* %tmpWithVal308
585.      %asChar309 = bitcast i32* %tmpWithVal308 to i8*
586.      %tmp310 = alloca i32*

```

```

587.      %alloca311 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
588.      %tmpAddr312 = bitcast i8* %alloca311 to i32*
589.      store i32* %tmpAddr312, i32** %tmp310
590.      %tmpWithVal313 = load i32*, i32** %tmp310
591.      store i32 1, i32* %tmpWithVal313
592.      %asChar314 = bitcast i32* %tmpWithVal313 to i8*
593.      %tmp315 = alloca i32*
594.      %alloca316 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
595.      %tmpAddr317 = bitcast i8* %alloca316 to i32*
596.      store i32* %tmpAddr317, i32** %tmp315
597.      %tmpWithVal318 = load i32*, i32** %tmp315
598.      store i32 1, i32* %tmpWithVal318
599.      %asChar319 = bitcast i32* %tmpWithVal318 to i8*
600.      %tmp320 = alloca i32*
601.      %alloca321 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
602.      %tmpAddr322 = bitcast i8* %alloca321 to i32*
603.      store i32* %tmpAddr322, i32** %tmp320
604.      %tmpWithVal323 = load i32*, i32** %tmp320
605.      store i32 1, i32* %tmpWithVal323
606.      %asChar324 = bitcast i32* %tmpWithVal323 to i8*
607.      %tmp325 = alloca i32*
608.      %alloca326 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
609.      %tmpAddr327 = bitcast i8* %alloca326 to i32*
610.      store i32* %tmpAddr327, i32** %tmp325
611.      %tmpWithVal328 = load i32*, i32** %tmp325
612.      store i32 1, i32* %tmpWithVal328
613.      %asChar329 = bitcast i32* %tmpWithVal328 to i8*
614.      %push_back330 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr3, i8* %asChar9)
615.      %push_back331 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr3, i8* %asChar14)
616.      %push_back332 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr3, i8* %asChar19)
617.      %push_back333 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr3, i8* %asChar24)
618.      %push_back334 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr3, i8* %asChar29)
619.      %push_back335 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr3, i8* %asChar34)

```

620. %push_back336 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar39)

621. %push_back337 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar44)

622. %push_back338 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar49)

623. %push_back339 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar54)

624. %push_back340 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar59)

625. %push_back341 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar64)

626. %push_back342 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar69)

627. %push_back343 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar74)

628. %push_back344 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar79)

629. %push_back345 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar84)

630. %push_back346 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar89)

631. %push_back347 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar94)

632. %push_back348 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar99)

633. %push_back349 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar104)

634. %push_back350 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar109)

635. %push_back351 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar114)

636. %push_back352 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar119)

637. %push_back353 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar124)

638. %push_back354 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar129)

639. %push_back355 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar134)

640. %push_back356 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar139)

641. %push_back357 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar144)

642. %push_back358 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar149)

643. %push_back359 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar154)

644. %push_back360 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar159)

645. %push_back361 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar164)

646. %push_back362 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar169)

647. %push_back363 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar174)

648. %push_back364 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar179)

649. %push_back365 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar184)

650. %push_back366 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar189)

651. %push_back367 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar194)

652. %push_back368 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar199)

653. %push_back369 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar204)

654. %push_back370 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar209)

655. %push_back371 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar214)

656. %push_back372 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar219)

657. %push_back373 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar224)

658. %push_back374 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar229)

659. %push_back375 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar234)

660. %push_back376 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar239)

661. %push_back377 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar244)

662. %push_back378 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar249)

663. %push_back379 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr3, i8* %asChar254)

```

664.      %push_back380 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar259)
665.      %push_back381 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar264)
666.      %push_back382 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar269)
667.      %push_back383 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar274)
668.      %push_back384 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar279)
669.      %push_back385 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar284)
670.      %push_back386 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar289)
671.      %push_back387 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar294)
672.      %push_back388 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar299)
673.      %push_back389 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar304)
674.      %push_back390 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar309)
675.      %push_back391 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar314)
676.      %push_back392 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar319)
677.      %push_back393 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar324)
678.      %push_back394 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr3, i8* %asChar329)
679.      store %list* %listAddr3, %list** %input_text
680.      %mallocall395 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
681.      %listAddr396 = bitcast i8* %mallocall395 to %list*
682.      %init_list397 = call i32 (%list*, ...) @init_list(%list* %listAddr396)
683.      %tmp398 = alloca i8*
684.      %tmpAddr400 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
          i8* null, i32 1) to i32))
685.      store i8* %tmpAddr400, i8** %tmp398
686.      %tmpWithVal401 = load i8*, i8** %tmp398
687.      store i8 90, i8* %tmpWithVal401
688.      %tmp402 = alloca i8*
689.      %tmpAddr404 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
          i8* null, i32 1) to i32))

```

```

690.      store i8* %tmpAddr404, i8** %tmp402
691.      %tmpWithVal405 = load i8*, i8** %tmp402
692.      store i8 69, i8* %tmpWithVal405
693.      %tmp406 = alloca i8*
694.      %tmpAddr408 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
695.      store i8* %tmpAddr408, i8** %tmp406
696.      %tmpWithVal409 = load i8*, i8** %tmp406
697.      store i8 82, i8* %tmpWithVal409
698.      %tmp410 = alloca i8*
699.      %tmpAddr412 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
700.      store i8* %tmpAddr412, i8** %tmp410
701.      %tmpWithVal413 = load i8*, i8** %tmp410
702.      store i8 79, i8* %tmpWithVal413
703.      %tmp414 = alloca i8*
704.      %tmpAddr416 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
705.      store i8* %tmpAddr416, i8** %tmp414
706.      %tmpWithVal417 = load i8*, i8** %tmp414
707.      store i8 83, i8* %tmpWithVal417
708.      %tmp418 = alloca i8*
709.      %tmpAddr420 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
710.      store i8* %tmpAddr420, i8** %tmp418
711.      %tmpWithVal421 = load i8*, i8** %tmp418
712.      store i8 73, i8* %tmpWithVal421
713.      %tmp422 = alloca i8*
714.      %tmpAddr424 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
715.      store i8* %tmpAddr424, i8** %tmp422
716.      %tmpWithVal425 = load i8*, i8** %tmp422
717.      store i8 88, i8* %tmpWithVal425
718.      %tmp426 = alloca i8*
719.      %tmpAddr428 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
720.      store i8* %tmpAddr428, i8** %tmp426
721.      %tmpWithVal429 = load i8*, i8** %tmp426
722.      store i8 72, i8* %tmpWithVal429
723.      %tmp430 = alloca i8*
724.      %tmpAddr432 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
725.      store i8* %tmpAddr432, i8** %tmp430
726.      %tmpWithVal433 = load i8*, i8** %tmp430

```

```

727.      store i8 85, i8* %tmpWithVal433
728.      %tmp434 = alloca i8*
729.      %tmpAddr436 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
730.      store i8* %tmpAddr436, i8** %tmp434
731.      %tmpWithVal437 = load i8*, i8** %tmp434
732.      store i8 78, i8* %tmpWithVal437
733.      %tmp438 = alloca i8*
734.      %tmpAddr440 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
735.      store i8* %tmpAddr440, i8** %tmp438
736.      %tmpWithVal441 = load i8*, i8** %tmp438
737.      store i8 68, i8* %tmpWithVal441
738.      %tmp442 = alloca i8*
739.      %tmpAddr444 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
740.      store i8* %tmpAddr444, i8** %tmp442
741.      %tmpWithVal445 = load i8*, i8** %tmp442
742.      store i8 82, i8* %tmpWithVal445
743.      %tmp446 = alloca i8*
744.      %tmpAddr448 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
745.      store i8* %tmpAddr448, i8** %tmp446
746.      %tmpWithVal449 = load i8*, i8** %tmp446
747.      store i8 69, i8* %tmpWithVal449
748.      %tmp450 = alloca i8*
749.      %tmpAddr452 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
750.      store i8* %tmpAddr452, i8** %tmp450
751.      %tmpWithVal453 = load i8*, i8** %tmp450
752.      store i8 69, i8* %tmpWithVal453
753.      %tmp454 = alloca i8*
754.      %tmpAddr456 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
755.      store i8* %tmpAddr456, i8** %tmp454
756.      %tmpWithVal457 = load i8*, i8** %tmp454
757.      store i8 68, i8* %tmpWithVal457
758.      %tmp458 = alloca i8*
759.      %tmpAddr460 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
760.      store i8* %tmpAddr460, i8** %tmp458
761.      %tmpWithVal461 = load i8*, i8** %tmp458
762.      store i8 72, i8* %tmpWithVal461
763.      %tmp462 = alloca i8*

```

```

764.      %tmpAddr464 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
765.      store i8* %tmpAddr464, i8** %tmp462
766.      %tmpWithVal465 = load i8*, i8** %tmp462
767.      store i8 79, i8* %tmpWithVal465
768.      %tmp466 = alloca i8*
769.      %tmpAddr468 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
770.      store i8* %tmpAddr468, i8** %tmp466
771.      %tmpWithVal469 = load i8*, i8** %tmp466
772.      store i8 85, i8* %tmpWithVal469
773.      %tmp470 = alloca i8*
774.      %tmpAddr472 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
775.      store i8* %tmpAddr472, i8** %tmp470
776.      %tmpWithVal473 = load i8*, i8** %tmp470
777.      store i8 82, i8* %tmpWithVal473
778.      %tmp474 = alloca i8*
779.      %tmpAddr476 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
780.      store i8* %tmpAddr476, i8** %tmp474
781.      %tmpWithVal477 = load i8*, i8** %tmp474
782.      store i8 83, i8* %tmpWithVal477
783.      %tmp478 = alloca i8*
784.      %tmpAddr480 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
785.      store i8* %tmpAddr480, i8** %tmp478
786.      %tmpWithVal481 = load i8*, i8** %tmp478
787.      store i8 87, i8* %tmpWithVal481
788.      %tmp482 = alloca i8*
789.      %tmpAddr484 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
790.      store i8* %tmpAddr484, i8** %tmp482
791.      %tmpWithVal485 = load i8*, i8** %tmp482
792.      store i8 69, i8* %tmpWithVal485
793.      %tmp486 = alloca i8*
794.      %tmpAddr488 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
795.      store i8* %tmpAddr488, i8** %tmp486
796.      %tmpWithVal489 = load i8*, i8** %tmp486
797.      store i8 65, i8* %tmpWithVal489
798.      %tmp490 = alloca i8*
799.      %tmpAddr492 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))

```

```

800.      store i8* %tmpAddr492, i8** %tmp490
801.      %tmpWithVal493 = load i8*, i8** %tmp490
802.      store i8 84, i8* %tmpWithVal493
803.      %tmp494 = alloca i8*
804.      %tmpAddr496 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
805.      store i8* %tmpAddr496, i8** %tmp494
806.      %tmpWithVal497 = load i8*, i8** %tmp494
807.      store i8 72, i8* %tmpWithVal497
808.      %tmp498 = alloca i8*
809.      %tmpAddr500 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
810.      store i8* %tmpAddr500, i8** %tmp498
811.      %tmpWithVal501 = load i8*, i8** %tmp498
812.      store i8 69, i8* %tmpWithVal501
813.      %tmp502 = alloca i8*
814.      %tmpAddr504 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
815.      store i8* %tmpAddr504, i8** %tmp502
816.      %tmpWithVal505 = load i8*, i8** %tmp502
817.      store i8 82, i8* %tmpWithVal505
818.      %tmp506 = alloca i8*
819.      %tmpAddr508 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
820.      store i8* %tmpAddr508, i8** %tmp506
821.      %tmpWithVal509 = load i8*, i8** %tmp506
822.      store i8 84, i8* %tmpWithVal509
823.      %tmp510 = alloca i8*
824.      %tmpAddr512 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
825.      store i8* %tmpAddr512, i8** %tmp510
826.      %tmpWithVal513 = load i8*, i8** %tmp510
827.      store i8 79, i8* %tmpWithVal513
828.      %tmp514 = alloca i8*
829.      %tmpAddr516 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
830.      store i8* %tmpAddr516, i8** %tmp514
831.      %tmpWithVal517 = load i8*, i8** %tmp514
832.      store i8 68, i8* %tmpWithVal517
833.      %tmp518 = alloca i8*
834.      %tmpAddr520 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
835.      store i8* %tmpAddr520, i8** %tmp518
836.      %tmpWithVal521 = load i8*, i8** %tmp518

```

```

837.      store i8 65, i8* %tmpWithVal521
838.      %tmp522 = alloca i8*
839.      %tmpAddr524 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
840.      store i8* %tmpAddr524, i8** %tmp522
841.      %tmpWithVal525 = load i8*, i8** %tmp522
842.      store i8 89, i8* %tmpWithVal525
843.      %tmp526 = alloca i8*
844.      %tmpAddr528 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
845.      store i8* %tmpAddr528, i8** %tmp526
846.      %tmpWithVal529 = load i8*, i8** %tmp526
847.      store i8 73, i8* %tmpWithVal529
848.      %tmp530 = alloca i8*
849.      %tmpAddr532 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
850.      store i8* %tmpAddr532, i8** %tmp530
851.      %tmpWithVal533 = load i8*, i8** %tmp530
852.      store i8 83, i8* %tmpWithVal533
853.      %tmp534 = alloca i8*
854.      %tmpAddr536 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
855.      store i8* %tmpAddr536, i8** %tmp534
856.      %tmpWithVal537 = load i8*, i8** %tmp534
857.      store i8 67, i8* %tmpWithVal537
858.      %tmp538 = alloca i8*
859.      %tmpAddr540 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
860.      store i8* %tmpAddr540, i8** %tmp538
861.      %tmpWithVal541 = load i8*, i8** %tmp538
862.      store i8 76, i8* %tmpWithVal541
863.      %tmp542 = alloca i8*
864.      %tmpAddr544 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
865.      store i8* %tmpAddr544, i8** %tmp542
866.      %tmpWithVal545 = load i8*, i8** %tmp542
867.      store i8 69, i8* %tmpWithVal545
868.      %tmp546 = alloca i8*
869.      %tmpAddr548 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
870.      store i8* %tmpAddr548, i8** %tmp546
871.      %tmpWithVal549 = load i8*, i8** %tmp546
872.      store i8 65, i8* %tmpWithVal549
873.      %tmp550 = alloca i8*

```

```

874.      %tmpAddr552 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
875.      store i8* %tmpAddr552, i8** %tmp550
876.      %tmpWithVal553 = load i8*, i8** %tmp550
877.      store i8 82, i8* %tmpWithVal553
878.      %tmp554 = alloca i8*
879.      %tmpAddr556 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
880.      store i8* %tmpAddr556, i8** %tmp554
881.      %tmpWithVal557 = load i8*, i8** %tmp554
882.      store i8 82, i8* %tmpWithVal557
883.      %tmp558 = alloca i8*
884.      %tmpAddr560 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
885.      store i8* %tmpAddr560, i8** %tmp558
886.      %tmpWithVal561 = load i8*, i8** %tmp558
887.      store i8 65, i8* %tmpWithVal561
888.      %tmp562 = alloca i8*
889.      %tmpAddr564 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
890.      store i8* %tmpAddr564, i8** %tmp562
891.      %tmpWithVal565 = load i8*, i8** %tmp562
892.      store i8 73, i8* %tmpWithVal565
893.      %tmp566 = alloca i8*
894.      %tmpAddr568 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
895.      store i8* %tmpAddr568, i8** %tmp566
896.      %tmpWithVal569 = load i8*, i8** %tmp566
897.      store i8 78, i8* %tmpWithVal569
898.      %tmp570 = alloca i8*
899.      %tmpAddr572 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
900.      store i8* %tmpAddr572, i8** %tmp570
901.      %tmpWithVal573 = load i8*, i8** %tmp570
902.      store i8 73, i8* %tmpWithVal573
903.      %tmp574 = alloca i8*
904.      %tmpAddr576 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
905.      store i8* %tmpAddr576, i8** %tmp574
906.      %tmpWithVal577 = load i8*, i8** %tmp574
907.      store i8 78, i8* %tmpWithVal577
908.      %tmp578 = alloca i8*
909.      %tmpAddr580 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))

```



```

910.      store i8* %tmpAddr580, i8** %tmp578
911.      %tmpWithVal581 = load i8*, i8** %tmp578
912.      store i8 84, i8* %tmpWithVal581
913.      %tmp582 = alloca i8*
914.      %tmpAddr584 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
915.      store i8* %tmpAddr584, i8** %tmp582
916.      %tmpWithVal585 = load i8*, i8** %tmp582
917.      store i8 72, i8* %tmpWithVal585
918.      %tmp586 = alloca i8*
919.      %tmpAddr588 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
920.      store i8* %tmpAddr588, i8** %tmp586
921.      %tmpWithVal589 = load i8*, i8** %tmp586
922.      store i8 69, i8* %tmpWithVal589
923.      %tmp590 = alloca i8*
924.      %tmpAddr592 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
925.      store i8* %tmpAddr592, i8** %tmp590
926.      %tmpWithVal593 = load i8*, i8** %tmp590
927.      store i8 69, i8* %tmpWithVal593
928.      %tmp594 = alloca i8*
929.      %tmpAddr596 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
930.      store i8* %tmpAddr596, i8** %tmp594
931.      %tmpWithVal597 = load i8*, i8** %tmp594
932.      store i8 86, i8* %tmpWithVal597
933.      %tmp598 = alloca i8*
934.      %tmpAddr600 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
935.      store i8* %tmpAddr600, i8** %tmp598
936.      %tmpWithVal601 = load i8*, i8** %tmp598
937.      store i8 69, i8* %tmpWithVal601
938.      %tmp602 = alloca i8*
939.      %tmpAddr604 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
940.      store i8* %tmpAddr604, i8** %tmp602
941.      %tmpWithVal605 = load i8*, i8** %tmp602
942.      store i8 78, i8* %tmpWithVal605
943.      %tmp606 = alloca i8*
944.      %tmpAddr608 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
945.      store i8* %tmpAddr608, i8** %tmp606
946.      %tmpWithVal609 = load i8*, i8** %tmp606

```

```

947.      store i8 73, i8* %tmpWithVal609
948.      %tmp610 = alloca i8*
949.      %tmpAddr612 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
950.      store i8* %tmpAddr612, i8** %tmp610
951.      %tmpWithVal613 = load i8*, i8** %tmp610
952.      store i8 78, i8* %tmpWithVal613
953.      %tmp614 = alloca i8*
954.      %tmpAddr616 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
955.      store i8* %tmpAddr616, i8** %tmp614
956.      %tmpWithVal617 = load i8*, i8** %tmp614
957.      store i8 71, i8* %tmpWithVal617
958.      %tmp618 = alloca i8*
959.      %tmpAddr620 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
960.      store i8* %tmpAddr620, i8** %tmp618
961.      %tmpWithVal621 = load i8*, i8** %tmp618
962.      store i8 72, i8* %tmpWithVal621
963.      %tmp622 = alloca i8*
964.      %tmpAddr624 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
965.      store i8* %tmpAddr624, i8** %tmp622
966.      %tmpWithVal625 = load i8*, i8** %tmp622
967.      store i8 69, i8* %tmpWithVal625
968.      %tmp626 = alloca i8*
969.      %tmpAddr628 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
970.      store i8* %tmpAddr628, i8** %tmp626
971.      %tmpWithVal629 = load i8*, i8** %tmp626
972.      store i8 73, i8* %tmpWithVal629
973.      %tmp630 = alloca i8*
974.      %tmpAddr632 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
975.      store i8* %tmpAddr632, i8** %tmp630
976.      %tmpWithVal633 = load i8*, i8** %tmp630
977.      store i8 76, i8* %tmpWithVal633
978.      %tmp634 = alloca i8*
979.      %tmpAddr636 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
980.      store i8* %tmpAddr636, i8** %tmp634
981.      %tmpWithVal637 = load i8*, i8** %tmp634
982.      store i8 72, i8* %tmpWithVal637
983.      %tmp638 = alloca i8*

```

```

984.      %tmpAddr640 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
985.      store i8* %tmpAddr640, i8** %tmp638
986.      %tmpWithVal641 = load i8*, i8** %tmp638
987.      store i8 73, i8* %tmpWithVal641
988.      %tmp642 = alloca i8*
989.      %tmpAddr644 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
990.      store i8* %tmpAddr644, i8** %tmp642
991.      %tmpWithVal645 = load i8*, i8** %tmp642
992.      store i8 84, i8* %tmpWithVal645
993.      %tmp646 = alloca i8*
994.      %tmpAddr648 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
995.      store i8* %tmpAddr648, i8** %tmp646
996.      %tmpWithVal649 = load i8*, i8** %tmp646
997.      store i8 76, i8* %tmpWithVal649
998.      %tmp650 = alloca i8*
999.      %tmpAddr652 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
1000.     store i8* %tmpAddr652, i8** %tmp650
1001.     %tmpWithVal653 = load i8*, i8** %tmp650
1002.     store i8 69, i8* %tmpWithVal653
1003.     %tmp654 = alloca i8*
1004.     %tmpAddr656 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
1005.     store i8* %tmpAddr656, i8** %tmp654
1006.     %tmpWithVal657 = load i8*, i8** %tmp654
1007.     store i8 82, i8* %tmpWithVal657
1008.     %push_back658 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr396, i8* %tmpWithVal401)
1009.     %push_back659 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr396, i8* %tmpWithVal405)
1010.     %push_back660 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr396, i8* %tmpWithVal409)
1011.     %push_back661 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr396, i8* %tmpWithVal413)
1012.     %push_back662 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr396, i8* %tmpWithVal417)
1013.     %push_back663 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr396, i8* %tmpWithVal421)
1014.     %push_back664 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr396, i8* %tmpWithVal425)

```

1015. %push_back665 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal429)

1016. %push_back666 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal433)

1017. %push_back667 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal437)

1018. %push_back668 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal441)

1019. %push_back669 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal445)

1020. %push_back670 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal449)

1021. %push_back671 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal453)

1022. %push_back672 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal457)

1023. %push_back673 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal461)

1024. %push_back674 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal465)

1025. %push_back675 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal469)

1026. %push_back676 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal473)

1027. %push_back677 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal477)

1028. %push_back678 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal481)

1029. %push_back679 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal485)

1030. %push_back680 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal489)

1031. %push_back681 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal493)

1032. %push_back682 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal497)

1033. %push_back683 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal501)

1034. %push_back684 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal505)

1035. %push_back685 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal509)

1036. %push_back686 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal513)

1037. %push_back687 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal517)

1038. %push_back688 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal521)

1039. %push_back689 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal525)

1040. %push_back690 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal529)

1041. %push_back691 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal533)

1042. %push_back692 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal537)

1043. %push_back693 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal541)

1044. %push_back694 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal545)

1045. %push_back695 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal549)

1046. %push_back696 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal553)

1047. %push_back697 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal557)

1048. %push_back698 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal561)

1049. %push_back699 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal565)

1050. %push_back700 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal569)

1051. %push_back701 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal573)

1052. %push_back702 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal577)

1053. %push_back703 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal581)

1054. %push_back704 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal585)

1055. %push_back705 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal589)

1056. %push_back706 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal593)

1057. %push_back707 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal597)

1058. %push_back708 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr396, i8* %tmpWithVal601)

```

1059.      %push_back709 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal605)
1060.      %push_back710 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal609)
1061.      %push_back711 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal613)
1062.      %push_back712 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal617)
1063.      %push_back713 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal621)
1064.      %push_back714 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal625)
1065.      %push_back715 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal629)
1066.      %push_back716 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal633)
1067.      %push_back717 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal637)
1068.      %push_back718 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal641)
1069.      %push_back719 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal645)
1070.      %push_back720 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal649)
1071.      %push_back721 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal653)
1072.      %push_back722 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr396, i8* %tmpWithVal657)
1073.      store %list* %listAddr396, %list** %messagez
1074.      %messagez723 = load %list*, %list** %messagez
1075.      %stringLength = call i32 @stringLength(%list* %messagez723)
1076.      store i32 %stringLength, i32* %length_input
1077.      %alloca724 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
1078.      %listAddr725 = bitcast i8* %alloca724 to %list*
1079.      %init_list726 = call i32 (%list*, ...) @init_list(%list* %listAddr725)
1080.      %tmp727 = alloca i32*
1081.      %alloca728 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1082.      %tmpAddr729 = bitcast i8* %alloca728 to i32*
1083.      store i32* %tmpAddr729, i32** %tmp727
1084.      %tmpWithVal730 = load i32*, i32** %tmp727
1085.      store i32 18, i32* %tmpWithVal730
1086.      %asChar731 = bitcast i32* %tmpWithVal730 to i8*

```

```

1087.    %tmp732 = alloca i32*
1088.    %alloca733 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1089.    %tmpAddr734 = bitcast i8* %alloca733 to i32*
1090.    store i32* %tmpAddr734, i32** %tmp732
1091.    %tmpWithVal735 = load i32*, i32** %tmp732
1092.    store i32 8, i32* %tmpWithVal735
1093.    %asChar736 = bitcast i32* %tmpWithVal735 to i8*
1094.    %tmp737 = alloca i32*
1095.    %alloca738 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1096.    %tmpAddr739 = bitcast i8* %alloca738 to i32*
1097.    store i32* %tmpAddr739, i32** %tmp737
1098.    %tmpWithVal740 = load i32*, i32** %tmp737
1099.    store i32 6, i32* %tmpWithVal740
1100.    %asChar741 = bitcast i32* %tmpWithVal740 to i8*
1101.    %push_back742 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr725, i8* %asChar731)
1102.    %push_back743 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr725, i8* %asChar736)
1103.    %push_back744 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr725, i8* %asChar741)
1104.    store %list* %listAddr725, %list** %key
1105.    %alloca745 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
      (%list, %list* null, i32 1) to i32))
1106.    %listAddr746 = bitcast i8* %alloca745 to %list*
1107.    %init_list747 = call i32 (%list*, ...) @init_list(%list* %listAddr746)
1108.    %tmp748 = alloca i32*
1109.    %alloca749 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1110.    %tmpAddr750 = bitcast i8* %alloca749 to i32*
1111.    store i32* %tmpAddr750, i32** %tmp748
1112.    %tmpWithVal751 = load i32*, i32** %tmp748
1113.    store i32 15, i32* %tmpWithVal751
1114.    %asChar752 = bitcast i32* %tmpWithVal751 to i8*
1115.    %tmp753 = alloca i32*
1116.    %alloca754 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1117.    %tmpAddr755 = bitcast i8* %alloca754 to i32*
1118.    store i32* %tmpAddr755, i32** %tmp753
1119.    %tmpWithVal756 = load i32*, i32** %tmp753
1120.    store i32 12, i32* %tmpWithVal756
1121.    %asChar757 = bitcast i32* %tmpWithVal756 to i8*
1122.    %tmp758 = alloca i32*

```

```

1123.      %alloca759 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1124.      %tmpAddr760 = bitcast i8* %alloca759 to i32*
1125.      store i32* %tmpAddr760, i32** %tmp758
1126.      %tmpWithVal761 = load i32*, i32** %tmp758
1127.      store i32 15, i32* %tmpWithVal761
1128.      %asChar762 = bitcast i32* %tmpWithVal761 to i8*
1129.      %push_back763 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr746, i8* %asChar752)
1130.      %push_back764 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr746, i8* %asChar757)
1131.      %push_back765 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr746, i8* %asChar762)
1132.      store %list* %listAddr746, %list** %ring_setting
1133.      %alloca766 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
1134.      %listAddr767 = bitcast i8* %alloca766 to %list*
1135.      %init_list768 = call i32 (%list*, ...) @init_list(%list* %listAddr767)
1136.      %tmp769 = alloca i32*
1137.      %alloca770 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1138.      %tmpAddr771 = bitcast i8* %alloca770 to i32*
1139.      store i32* %tmpAddr771, i32** %tmp769
1140.      %tmpWithVal772 = load i32*, i32** %tmp769
1141.      store i32 1, i32* %tmpWithVal772
1142.      %asChar773 = bitcast i32* %tmpWithVal772 to i8*
1143.      %tmp774 = alloca i32*
1144.      %alloca775 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1145.      %tmpAddr776 = bitcast i8* %alloca775 to i32*
1146.      store i32* %tmpAddr776, i32** %tmp774
1147.      %tmpWithVal777 = load i32*, i32** %tmp774
1148.      store i32 2, i32* %tmpWithVal777
1149.      %asChar778 = bitcast i32* %tmpWithVal777 to i8*
1150.      %tmp779 = alloca i32*
1151.      %alloca780 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1152.      %tmpAddr781 = bitcast i8* %alloca780 to i32*
1153.      store i32* %tmpAddr781, i32** %tmp779
1154.      %tmpWithVal782 = load i32*, i32** %tmp779
1155.      store i32 3, i32* %tmpWithVal782
1156.      %asChar783 = bitcast i32* %tmpWithVal782 to i8*
1157.      %tmp784 = alloca i32*

```



```

1158.      %alloca785 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1159.      %tmpAddr786 = bitcast i8* %alloca785 to i32*
1160.      store i32* %tmpAddr786, i32** %tmp784
1161.      %tmpWithVal787 = load i32*, i32** %tmp784
1162.      store i32 4, i32* %tmpWithVal787
1163.      %asChar788 = bitcast i32* %tmpWithVal787 to i8*
1164.      %tmp789 = alloca i32*
1165.      %alloca790 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1166.      %tmpAddr791 = bitcast i8* %alloca790 to i32*
1167.      store i32* %tmpAddr791, i32** %tmp789
1168.      %tmpWithVal792 = load i32*, i32** %tmp789
1169.      store i32 5, i32* %tmpWithVal792
1170.      %asChar793 = bitcast i32* %tmpWithVal792 to i8*
1171.      %tmp794 = alloca i32*
1172.      %alloca795 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1173.      %tmpAddr796 = bitcast i8* %alloca795 to i32*
1174.      store i32* %tmpAddr796, i32** %tmp794
1175.      %tmpWithVal797 = load i32*, i32** %tmp794
1176.      store i32 6, i32* %tmpWithVal797
1177.      %asChar798 = bitcast i32* %tmpWithVal797 to i8*
1178.      %tmp799 = alloca i32*
1179.      %alloca800 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1180.      %tmpAddr801 = bitcast i8* %alloca800 to i32*
1181.      store i32* %tmpAddr801, i32** %tmp799
1182.      %tmpWithVal802 = load i32*, i32** %tmp799
1183.      store i32 9, i32* %tmpWithVal802
1184.      %asChar803 = bitcast i32* %tmpWithVal802 to i8*
1185.      %tmp804 = alloca i32*
1186.      %alloca805 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1187.      %tmpAddr806 = bitcast i8* %alloca805 to i32*
1188.      store i32* %tmpAddr806, i32** %tmp804
1189.      %tmpWithVal807 = load i32*, i32** %tmp804
1190.      store i32 13, i32* %tmpWithVal807
1191.      %asChar808 = bitcast i32* %tmpWithVal807 to i8*
1192.      %tmp809 = alloca i32*
1193.      %alloca810 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1194.      %tmpAddr811 = bitcast i8* %alloca810 to i32*
1195.      store i32* %tmpAddr811, i32** %tmp809

```

```

1196.    %tmpWithVal812 = load i32*, i32** %tmp809
1197.    store i32 10, i32* %tmpWithVal812
1198.    %asChar813 = bitcast i32* %tmpWithVal812 to i8*
1199.    %tmp814 = alloca i32*
1200.    %allocaCall815 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1201.    %tmpAddr816 = bitcast i8* %allocaCall815 to i32*
1202.    store i32* %tmpAddr816, i32** %tmp814
1203.    %tmpWithVal817 = load i32*, i32** %tmp814
1204.    store i32 21, i32* %tmpWithVal817
1205.    %asChar818 = bitcast i32* %tmpWithVal817 to i8*
1206.    %push_back819 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar773)
1207.    %push_back820 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar778)
1208.    %push_back821 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar783)
1209.    %push_back822 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar788)
1210.    %push_back823 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar793)
1211.    %push_back824 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar798)
1212.    %push_back825 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar803)
1213.    %push_back826 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar808)
1214.    %push_back827 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar813)
1215.    %push_back828 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr767, i8* %asChar818)
1216.    store %list* %listAddr767, %list** %first_plugs_init
1217.    %allocaCall829 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
      (%list, %list* null, i32 1) to i32))
1218.    %listAddr830 = bitcast i8* %allocaCall829 to %list*
1219.    %init_list831 = call i32 (%list*, ...) @init_list(%list* %listAddr830)
1220.    %tmp832 = alloca i32*
1221.    %allocaCall833 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1222.    %tmpAddr834 = bitcast i8* %allocaCall833 to i32*
1223.    store i32* %tmpAddr834, i32** %tmp832
1224.    %tmpWithVal835 = load i32*, i32** %tmp832
1225.    store i32 18, i32* %tmpWithVal835
1226.    %asChar836 = bitcast i32* %tmpWithVal835 to i8*

```

```

1227.    %tmp837 = alloca i32*
1228.    %alloca838 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1229.    %tmpAddr839 = bitcast i8* %alloca838 to i32*
1230.    store i32* %tmpAddr839, i32** %tmp837
1231.    %tmpWithVal840 = load i32*, i32** %tmp837
1232.    store i32 8, i32* %tmpWithVal840
1233.    %asChar841 = bitcast i32* %tmpWithVal840 to i8*
1234.    %tmp842 = alloca i32*
1235.    %alloca843 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1236.    %tmpAddr844 = bitcast i8* %alloca843 to i32*
1237.    store i32* %tmpAddr844, i32** %tmp842
1238.    %tmpWithVal845 = load i32*, i32** %tmp842
1239.    store i32 12, i32* %tmpWithVal845
1240.    %asChar846 = bitcast i32* %tmpWithVal845 to i8*
1241.    %tmp847 = alloca i32*
1242.    %alloca848 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1243.    %tmpAddr849 = bitcast i8* %alloca848 to i32*
1244.    store i32* %tmpAddr849, i32** %tmp847
1245.    %tmpWithVal850 = load i32*, i32** %tmp847
1246.    store i32 16, i32* %tmpWithVal850
1247.    %asChar851 = bitcast i32* %tmpWithVal850 to i8*
1248.    %tmp852 = alloca i32*
1249.    %alloca853 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1250.    %tmpAddr854 = bitcast i8* %alloca853 to i32*
1251.    store i32* %tmpAddr854, i32** %tmp852
1252.    %tmpWithVal855 = load i32*, i32** %tmp852
1253.    store i32 20, i32* %tmpWithVal855
1254.    %asChar856 = bitcast i32* %tmpWithVal855 to i8*
1255.    %tmp857 = alloca i32*
1256.    %alloca858 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1257.    %tmpAddr859 = bitcast i8* %alloca858 to i32*
1258.    store i32* %tmpAddr859, i32** %tmp857
1259.    %tmpWithVal860 = load i32*, i32** %tmp857
1260.    store i32 24, i32* %tmpWithVal860
1261.    %asChar861 = bitcast i32* %tmpWithVal860 to i8*
1262.    %tmp862 = alloca i32*
1263.    %alloca863 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1264.    %tmpAddr864 = bitcast i8* %alloca863 to i32*

```

```

1265.    store i32* %tmpAddr864, i32** %tmp862
1266.    %tmpWithVal865 = load i32*, i32** %tmp862
1267.    store i32 11, i32* %tmpWithVal865
1268.    %asChar866 = bitcast i32* %tmpWithVal865 to i8*
1269.    %tmp867 = alloca i32*
1270.    %allocaCall868 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
1271.    %tmpAddr869 = bitcast i8* %allocaCall868 to i32*
1272.    store i32* %tmpAddr869, i32** %tmp867
1273.    %tmpWithVal870 = load i32*, i32** %tmp867
1274.    store i32 15, i32* %tmpWithVal870
1275.    %asChar871 = bitcast i32* %tmpWithVal870 to i8*
1276.    %tmp872 = alloca i32*
1277.    %allocaCall873 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
1278.    %tmpAddr874 = bitcast i8* %allocaCall873 to i32*
1279.    store i32* %tmpAddr874, i32** %tmp872
1280.    %tmpWithVal875 = load i32*, i32** %tmp872
1281.    store i32 25, i32* %tmpWithVal875
1282.    %asChar876 = bitcast i32* %tmpWithVal875 to i8*
1283.    %tmp877 = alloca i32*
1284.    %allocaCall878 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
1285.    %tmpAddr879 = bitcast i8* %allocaCall878 to i32*
1286.    store i32* %tmpAddr879, i32** %tmp877
1287.    %tmpWithVal880 = load i32*, i32** %tmp877
1288.    store i32 22, i32* %tmpWithVal880
1289.    %asChar881 = bitcast i32* %tmpWithVal880 to i8*
1290.    %push_back882 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr830, i8* %asChar836)
1291.    %push_back883 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr830, i8* %asChar841)
1292.    %push_back884 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr830, i8* %asChar846)
1293.    %push_back885 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr830, i8* %asChar851)
1294.    %push_back886 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr830, i8* %asChar856)
1295.    %push_back887 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr830, i8* %asChar861)
1296.    %push_back888 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr830, i8* %asChar866)
1297.    %push_back889 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr830, i8* %asChar871)

```

```

1298.      %push_back890 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr830, i8* %asChar876)
1299.      %push_back891 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr830, i8* %asChar881)
1300.      store %list* %listAddr830, %list** %second_plugs_init
1301.      %malloccall892 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
1302.      %listAddr893 = bitcast i8* %malloccall892 to %list*
1303.      %init_list894 = call i32 (%list*, ...) @init_list(%list* %listAddr893)
1304.      %tmp895 = alloca i32*
1305.      %malloccall896 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1306.      %tmpAddr897 = bitcast i8* %malloccall896 to i32*
1307.      store i32* %tmpAddr897, i32** %tmp895
1308.      %tmpWithVal898 = load i32*, i32** %tmp895
1309.      store i32 1, i32* %tmpWithVal898
1310.      %asChar899 = bitcast i32* %tmpWithVal898 to i8*
1311.      %tmp900 = alloca i32*
1312.      %malloccall901 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1313.      %tmpAddr902 = bitcast i8* %malloccall901 to i32*
1314.      store i32* %tmpAddr902, i32** %tmp900
1315.      %tmpWithVal903 = load i32*, i32** %tmp900
1316.      store i32 3, i32* %tmpWithVal903
1317.      %asChar904 = bitcast i32* %tmpWithVal903 to i8*
1318.      %tmp905 = alloca i32*
1319.      %malloccall906 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1320.      %tmpAddr907 = bitcast i8* %malloccall906 to i32*
1321.      store i32* %tmpAddr907, i32** %tmp905
1322.      %tmpWithVal908 = load i32*, i32** %tmp905
1323.      store i32 5, i32* %tmpWithVal908
1324.      %asChar909 = bitcast i32* %tmpWithVal908 to i8*
1325.      %tmp910 = alloca i32*
1326.      %malloccall911 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1327.      %tmpAddr912 = bitcast i8* %malloccall911 to i32*
1328.      store i32* %tmpAddr912, i32** %tmp910
1329.      %tmpWithVal913 = load i32*, i32** %tmp910
1330.      store i32 7, i32* %tmpWithVal913
1331.      %asChar914 = bitcast i32* %tmpWithVal913 to i8*
1332.      %tmp915 = alloca i32*
1333.      %malloccall916 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))

```

```

1334.    %tmpAddr917 = bitcast i8* %alloca916 to i32*
1335.    store i32* %tmpAddr917, i32** %tmp915
1336.    %tmpWithVal918 = load i32*, i32** %tmp915
1337.    store i32 9, i32* %tmpWithVal918
1338.    %asChar919 = bitcast i32* %tmpWithVal918 to i8*
1339.    %tmp920 = alloca i32*
1340.    %alloca921 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1341.    %tmpAddr922 = bitcast i8* %alloca921 to i32*
1342.    store i32* %tmpAddr922, i32** %tmp920
1343.    %tmpWithVal923 = load i32*, i32** %tmp920
1344.    store i32 11, i32* %tmpWithVal923
1345.    %asChar924 = bitcast i32* %tmpWithVal923 to i8*
1346.    %tmp925 = alloca i32*
1347.    %alloca926 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1348.    %tmpAddr927 = bitcast i8* %alloca926 to i32*
1349.    store i32* %tmpAddr927, i32** %tmp925
1350.    %tmpWithVal928 = load i32*, i32** %tmp925
1351.    store i32 2, i32* %tmpWithVal928
1352.    %asChar929 = bitcast i32* %tmpWithVal928 to i8*
1353.    %tmp930 = alloca i32*
1354.    %alloca931 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1355.    %tmpAddr932 = bitcast i8* %alloca931 to i32*
1356.    store i32* %tmpAddr932, i32** %tmp930
1357.    %tmpWithVal933 = load i32*, i32** %tmp930
1358.    store i32 15, i32* %tmpWithVal933
1359.    %asChar934 = bitcast i32* %tmpWithVal933 to i8*
1360.    %tmp935 = alloca i32*
1361.    %alloca936 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1362.    %tmpAddr937 = bitcast i8* %alloca936 to i32*
1363.    store i32* %tmpAddr937, i32** %tmp935
1364.    %tmpWithVal938 = load i32*, i32** %tmp935
1365.    store i32 17, i32* %tmpWithVal938
1366.    %asChar939 = bitcast i32* %tmpWithVal938 to i8*
1367.    %tmp940 = alloca i32*
1368.    %alloca941 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1369.    %tmpAddr942 = bitcast i8* %alloca941 to i32*
1370.    store i32* %tmpAddr942, i32** %tmp940
1371.    %tmpWithVal943 = load i32*, i32** %tmp940
1372.    store i32 19, i32* %tmpWithVal943

```

```

1373.    %asChar944 = bitcast i32* %tmpWithVal943 to i8*
1374.    %tmp945 = alloca i32*
1375.    %allocaCall946 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1376.    %tmpAddr947 = bitcast i8* %allocaCall946 to i32*
1377.    store i32* %tmpAddr947, i32** %tmp945
1378.    %tmpWithVal948 = load i32*, i32** %tmp945
1379.    store i32 23, i32* %tmpWithVal948
1380.    %asChar949 = bitcast i32* %tmpWithVal948 to i8*
1381.    %tmp950 = alloca i32*
1382.    %allocaCall951 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1383.    %tmpAddr952 = bitcast i8* %allocaCall951 to i32*
1384.    store i32* %tmpAddr952, i32** %tmp950
1385.    %tmpWithVal953 = load i32*, i32** %tmp950
1386.    store i32 21, i32* %tmpWithVal953
1387.    %asChar954 = bitcast i32* %tmpWithVal953 to i8*
1388.    %tmp955 = alloca i32*
1389.    %allocaCall956 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1390.    %tmpAddr957 = bitcast i8* %allocaCall956 to i32*
1391.    store i32* %tmpAddr957, i32** %tmp955
1392.    %tmpWithVal958 = load i32*, i32** %tmp955
1393.    store i32 25, i32* %tmpWithVal958
1394.    %asChar959 = bitcast i32* %tmpWithVal958 to i8*
1395.    %tmp960 = alloca i32*
1396.    %allocaCall961 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1397.    %tmpAddr962 = bitcast i8* %allocaCall961 to i32*
1398.    store i32* %tmpAddr962, i32** %tmp960
1399.    %tmpWithVal963 = load i32*, i32** %tmp960
1400.    store i32 13, i32* %tmpWithVal963
1401.    %asChar964 = bitcast i32* %tmpWithVal963 to i8*
1402.    %tmp965 = alloca i32*
1403.    %allocaCall966 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1404.    %tmpAddr967 = bitcast i8* %allocaCall966 to i32*
1405.    store i32* %tmpAddr967, i32** %tmp965
1406.    %tmpWithVal968 = load i32*, i32** %tmp965
1407.    store i32 24, i32* %tmpWithVal968
1408.    %asChar969 = bitcast i32* %tmpWithVal968 to i8*
1409.    %tmp970 = alloca i32*
1410.    %allocaCall971 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))

```

```

1411.    %tmpAddr972 = bitcast i8* %alloca971 to i32*
1412.    store i32* %tmpAddr972, i32** %tmp970
1413.    %tmpWithVal973 = load i32*, i32** %tmp970
1414.    store i32 4, i32* %tmpWithVal973
1415.    %asChar974 = bitcast i32* %tmpWithVal973 to i8*
1416.    %tmp975 = alloca i32*
1417.    %alloca976 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1418.    %tmpAddr977 = bitcast i8* %alloca976 to i32*
1419.    store i32* %tmpAddr977, i32** %tmp975
1420.    %tmpWithVal978 = load i32*, i32** %tmp975
1421.    store i32 8, i32* %tmpWithVal978
1422.    %asChar979 = bitcast i32* %tmpWithVal978 to i8*
1423.    %tmp980 = alloca i32*
1424.    %alloca981 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1425.    %tmpAddr982 = bitcast i8* %alloca981 to i32*
1426.    store i32* %tmpAddr982, i32** %tmp980
1427.    %tmpWithVal983 = load i32*, i32** %tmp980
1428.    store i32 22, i32* %tmpWithVal983
1429.    %asChar984 = bitcast i32* %tmpWithVal983 to i8*
1430.    %tmp985 = alloca i32*
1431.    %alloca986 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1432.    %tmpAddr987 = bitcast i8* %alloca986 to i32*
1433.    store i32* %tmpAddr987, i32** %tmp985
1434.    %tmpWithVal988 = load i32*, i32** %tmp985
1435.    store i32 6, i32* %tmpWithVal988
1436.    %asChar989 = bitcast i32* %tmpWithVal988 to i8*
1437.    %tmp990 = alloca i32*
1438.    %alloca991 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1439.    %tmpAddr992 = bitcast i8* %alloca991 to i32*
1440.    store i32* %tmpAddr992, i32** %tmp990
1441.    %tmpWithVal993 = load i32*, i32** %tmp990
1442.    store i32 0, i32* %tmpWithVal993
1443.    %asChar994 = bitcast i32* %tmpWithVal993 to i8*
1444.    %tmp995 = alloca i32*
1445.    %alloca996 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1446.    %tmpAddr997 = bitcast i8* %alloca996 to i32*
1447.    store i32* %tmpAddr997, i32** %tmp995
1448.    %tmpWithVal998 = load i32*, i32** %tmp995
1449.    store i32 10, i32* %tmpWithVal998

```



```

1450.    %asChar999 = bitcast i32* %tmpWithVal998 to i8*
1451.    %tmp1000 = alloca i32*
1452.    %malloccall1001 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1453.    %tmpAddr1002 = bitcast i8* %malloccall1001 to i32*
1454.    store i32* %tmpAddr1002, i32** %tmp1000
1455.    %tmpWithVal1003 = load i32*, i32** %tmp1000
1456.    store i32 12, i32* %tmpWithVal1003
1457.    %asChar1004 = bitcast i32* %tmpWithVal1003 to i8*
1458.    %tmp1005 = alloca i32*
1459.    %malloccall1006 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1460.    %tmpAddr1007 = bitcast i8* %malloccall1006 to i32*
1461.    store i32* %tmpAddr1007, i32** %tmp1005
1462.    %tmpWithVal1008 = load i32*, i32** %tmp1005
1463.    store i32 20, i32* %tmpWithVal1008
1464.    %asChar1009 = bitcast i32* %tmpWithVal1008 to i8*
1465.    %tmp1010 = alloca i32*
1466.    %malloccall1011 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1467.    %tmpAddr1012 = bitcast i8* %malloccall1011 to i32*
1468.    store i32* %tmpAddr1012, i32** %tmp1010
1469.    %tmpWithVal1013 = load i32*, i32** %tmp1010
1470.    store i32 18, i32* %tmpWithVal1013
1471.    %asChar1014 = bitcast i32* %tmpWithVal1013 to i8*
1472.    %tmp1015 = alloca i32*
1473.    %malloccall1016 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1474.    %tmpAddr1017 = bitcast i8* %malloccall1016 to i32*
1475.    store i32* %tmpAddr1017, i32** %tmp1015
1476.    %tmpWithVal1018 = load i32*, i32** %tmp1015
1477.    store i32 16, i32* %tmpWithVal1018
1478.    %asChar1019 = bitcast i32* %tmpWithVal1018 to i8*
1479.    %tmp1020 = alloca i32*
1480.    %malloccall1021 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1481.    %tmpAddr1022 = bitcast i8* %malloccall1021 to i32*
1482.    store i32* %tmpAddr1022, i32** %tmp1020
1483.    %tmpWithVal1023 = load i32*, i32** %tmp1020
1484.    store i32 14, i32* %tmpWithVal1023
1485.    %asChar1024 = bitcast i32* %tmpWithVal1023 to i8*
1486.    %push_back1025 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr893, i8* %asChar899)

```

1487. %push_back1026 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar904)

1488. %push_back1027 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar909)

1489. %push_back1028 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar914)

1490. %push_back1029 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar919)

1491. %push_back1030 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar924)

1492. %push_back1031 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar929)

1493. %push_back1032 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar934)

1494. %push_back1033 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar939)

1495. %push_back1034 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar944)

1496. %push_back1035 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar949)

1497. %push_back1036 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar954)

1498. %push_back1037 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar959)

1499. %push_back1038 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar964)

1500. %push_back1039 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar969)

1501. %push_back1040 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar974)

1502. %push_back1041 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar979)

1503. %push_back1042 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar984)

1504. %push_back1043 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar989)

1505. %push_back1044 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar994)

1506. %push_back1045 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar999)

1507. %push_back1046 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar1004)

1508. %push_back1047 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr893, i8* %asChar1009)

```

1509.      %push_back1048 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr893, i8* %asChar1014)
1510.      %push_back1049 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr893, i8* %asChar1019)
1511.      %push_back1050 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr893, i8* %asChar1024)
1512.      store %list* %listAddr893, %list** %rotor_three_out
1513.      %alloca1051 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
1514.      %listAddr1052 = bitcast i8* %alloca1051 to %list*
1515.      %init_list1053 = call i32 (%list*, ...) @init_list(%list* %listAddr1052)
1516.      %tmp1054 = alloca i32*
1517.      %alloca1055 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1518.      %tmpAddr1056 = bitcast i8* %alloca1055 to i32*
1519.      store i32* %tmpAddr1056, i32** %tmp1054
1520.      %tmpWithVal1057 = load i32*, i32** %tmp1054
1521.      store i32 0, i32* %tmpWithVal1057
1522.      %asChar1058 = bitcast i32* %tmpWithVal1057 to i8*
1523.      %tmp1059 = alloca i32*
1524.      %alloca1060 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1525.      %tmpAddr1061 = bitcast i8* %alloca1060 to i32*
1526.      store i32* %tmpAddr1061, i32** %tmp1059
1527.      %tmpWithVal1062 = load i32*, i32** %tmp1059
1528.      store i32 1, i32* %tmpWithVal1062
1529.      %asChar1063 = bitcast i32* %tmpWithVal1062 to i8*
1530.      %tmp1064 = alloca i32*
1531.      %alloca1065 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1532.      %tmpAddr1066 = bitcast i8* %alloca1065 to i32*
1533.      store i32* %tmpAddr1066, i32** %tmp1064
1534.      %tmpWithVal1067 = load i32*, i32** %tmp1064
1535.      store i32 2, i32* %tmpWithVal1067
1536.      %asChar1068 = bitcast i32* %tmpWithVal1067 to i8*
1537.      %tmp1069 = alloca i32*
1538.      %alloca1070 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1539.      %tmpAddr1071 = bitcast i8* %alloca1070 to i32*
1540.      store i32* %tmpAddr1071, i32** %tmp1069
1541.      %tmpWithVal1072 = load i32*, i32** %tmp1069
1542.      store i32 3, i32* %tmpWithVal1072
1543.      %asChar1073 = bitcast i32* %tmpWithVal1072 to i8*
1544.      %tmp1074 = alloca i32*

```

```

1545.      %malloccall1075 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1546.      %tmpAddr1076 = bitcast i8* %malloccall1075 to i32*
1547.      store i32* %tmpAddr1076, i32** %tmp1074
1548.      %tmpWithVal1077 = load i32*, i32** %tmp1074
1549.      store i32 4, i32* %tmpWithVal1077
1550.      %asChar1078 = bitcast i32* %tmpWithVal1077 to i8*
1551.      %tmp1079 = alloca i32*
1552.      %malloccall1080 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1553.      %tmpAddr1081 = bitcast i8* %malloccall1080 to i32*
1554.      store i32* %tmpAddr1081, i32** %tmp1079
1555.      %tmpWithVal1082 = load i32*, i32** %tmp1079
1556.      store i32 5, i32* %tmpWithVal1082
1557.      %asChar1083 = bitcast i32* %tmpWithVal1082 to i8*
1558.      %tmp1084 = alloca i32*
1559.      %malloccall1085 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1560.      %tmpAddr1086 = bitcast i8* %malloccall1085 to i32*
1561.      store i32* %tmpAddr1086, i32** %tmp1084
1562.      %tmpWithVal1087 = load i32*, i32** %tmp1084
1563.      store i32 6, i32* %tmpWithVal1087
1564.      %asChar1088 = bitcast i32* %tmpWithVal1087 to i8*
1565.      %tmp1089 = alloca i32*
1566.      %malloccall1090 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1567.      %tmpAddr1091 = bitcast i8* %malloccall1090 to i32*
1568.      store i32* %tmpAddr1091, i32** %tmp1089
1569.      %tmpWithVal1092 = load i32*, i32** %tmp1089
1570.      store i32 7, i32* %tmpWithVal1092
1571.      %asChar1093 = bitcast i32* %tmpWithVal1092 to i8*
1572.      %tmp1094 = alloca i32*
1573.      %malloccall1095 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1574.      %tmpAddr1096 = bitcast i8* %malloccall1095 to i32*
1575.      store i32* %tmpAddr1096, i32** %tmp1094
1576.      %tmpWithVal1097 = load i32*, i32** %tmp1094
1577.      store i32 8, i32* %tmpWithVal1097
1578.      %asChar1098 = bitcast i32* %tmpWithVal1097 to i8*
1579.      %tmp1099 = alloca i32*
1580.      %malloccall1100 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1581.      %tmpAddr1101 = bitcast i8* %malloccall1100 to i32*
1582.      store i32* %tmpAddr1101, i32** %tmp1099

```

```

1583.    %tmpWithVal1102 = load i32*, i32** %tmp1099
1584.    store i32 9, i32* %tmpWithVal1102
1585.    %asChar1103 = bitcast i32* %tmpWithVal1102 to i8*
1586.    %tmp1104 = alloca i32*
1587.    %allocaCall1105 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1588.    %tmpAddr1106 = bitcast i8* %allocaCall1105 to i32*
1589.    store i32* %tmpAddr1106, i32** %tmp1104
1590.    %tmpWithVal1107 = load i32*, i32** %tmp1104
1591.    store i32 10, i32* %tmpWithVal1107
1592.    %asChar1108 = bitcast i32* %tmpWithVal1107 to i8*
1593.    %tmp1109 = alloca i32*
1594.    %allocaCall1110 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1595.    %tmpAddr1111 = bitcast i8* %allocaCall1110 to i32*
1596.    store i32* %tmpAddr1111, i32** %tmp1109
1597.    %tmpWithVal1112 = load i32*, i32** %tmp1109
1598.    store i32 11, i32* %tmpWithVal1112
1599.    %asChar1113 = bitcast i32* %tmpWithVal1112 to i8*
1600.    %tmp1114 = alloca i32*
1601.    %allocaCall1115 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1602.    %tmpAddr1116 = bitcast i8* %allocaCall1115 to i32*
1603.    store i32* %tmpAddr1116, i32** %tmp1114
1604.    %tmpWithVal1117 = load i32*, i32** %tmp1114
1605.    store i32 12, i32* %tmpWithVal1117
1606.    %asChar1118 = bitcast i32* %tmpWithVal1117 to i8*
1607.    %tmp1119 = alloca i32*
1608.    %allocaCall1120 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1609.    %tmpAddr1121 = bitcast i8* %allocaCall1120 to i32*
1610.    store i32* %tmpAddr1121, i32** %tmp1119
1611.    %tmpWithVal1122 = load i32*, i32** %tmp1119
1612.    store i32 13, i32* %tmpWithVal1122
1613.    %asChar1123 = bitcast i32* %tmpWithVal1122 to i8*
1614.    %tmp1124 = alloca i32*
1615.    %allocaCall1125 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1616.    %tmpAddr1126 = bitcast i8* %allocaCall1125 to i32*
1617.    store i32* %tmpAddr1126, i32** %tmp1124
1618.    %tmpWithVal1127 = load i32*, i32** %tmp1124
1619.    store i32 14, i32* %tmpWithVal1127
1620.    %asChar1128 = bitcast i32* %tmpWithVal1127 to i8*
1621.    %tmp1129 = alloca i32*

```

```

1622.      %malloccall1130 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1623.      %tmpAddr1131 = bitcast i8* %malloccall1130 to i32*
1624.      store i32* %tmpAddr1131, i32** %tmp1129
1625.      %tmpWithVal1132 = load i32*, i32** %tmp1129
1626.      store i32 15, i32* %tmpWithVal1132
1627.      %asChar1133 = bitcast i32* %tmpWithVal1132 to i8*
1628.      %tmp1134 = alloca i32*
1629.      %malloccall1135 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1630.      %tmpAddr1136 = bitcast i8* %malloccall1135 to i32*
1631.      store i32* %tmpAddr1136, i32** %tmp1134
1632.      %tmpWithVal1137 = load i32*, i32** %tmp1134
1633.      store i32 16, i32* %tmpWithVal1137
1634.      %asChar1138 = bitcast i32* %tmpWithVal1137 to i8*
1635.      %tmp1139 = alloca i32*
1636.      %malloccall1140 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1637.      %tmpAddr1141 = bitcast i8* %malloccall1140 to i32*
1638.      store i32* %tmpAddr1141, i32** %tmp1139
1639.      %tmpWithVal1142 = load i32*, i32** %tmp1139
1640.      store i32 17, i32* %tmpWithVal1142
1641.      %asChar1143 = bitcast i32* %tmpWithVal1142 to i8*
1642.      %tmp1144 = alloca i32*
1643.      %malloccall1145 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1644.      %tmpAddr1146 = bitcast i8* %malloccall1145 to i32*
1645.      store i32* %tmpAddr1146, i32** %tmp1144
1646.      %tmpWithVal1147 = load i32*, i32** %tmp1144
1647.      store i32 18, i32* %tmpWithVal1147
1648.      %asChar1148 = bitcast i32* %tmpWithVal1147 to i8*
1649.      %tmp1149 = alloca i32*
1650.      %malloccall1150 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1651.      %tmpAddr1151 = bitcast i8* %malloccall1150 to i32*
1652.      store i32* %tmpAddr1151, i32** %tmp1149
1653.      %tmpWithVal1152 = load i32*, i32** %tmp1149
1654.      store i32 19, i32* %tmpWithVal1152
1655.      %asChar1153 = bitcast i32* %tmpWithVal1152 to i8*
1656.      %tmp1154 = alloca i32*
1657.      %malloccall1155 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1658.      %tmpAddr1156 = bitcast i8* %malloccall1155 to i32*
1659.      store i32* %tmpAddr1156, i32** %tmp1154

```

```

1660.    %tmpWithVal1157 = load i32*, i32** %tmp1154
1661.    store i32 20, i32* %tmpWithVal1157
1662.    %asChar1158 = bitcast i32* %tmpWithVal1157 to i8*
1663.    %tmp1159 = alloca i32*
1664.    %allocaCall1160 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1665.    %tmpAddr1161 = bitcast i8* %allocaCall1160 to i32*
1666.    store i32* %tmpAddr1161, i32** %tmp1159
1667.    %tmpWithVal1162 = load i32*, i32** %tmp1159
1668.    store i32 21, i32* %tmpWithVal1162
1669.    %asChar1163 = bitcast i32* %tmpWithVal1162 to i8*
1670.    %tmp1164 = alloca i32*
1671.    %allocaCall1165 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1672.    %tmpAddr1166 = bitcast i8* %allocaCall1165 to i32*
1673.    store i32* %tmpAddr1166, i32** %tmp1164
1674.    %tmpWithVal1167 = load i32*, i32** %tmp1164
1675.    store i32 22, i32* %tmpWithVal1167
1676.    %asChar1168 = bitcast i32* %tmpWithVal1167 to i8*
1677.    %tmp1169 = alloca i32*
1678.    %allocaCall1170 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1679.    %tmpAddr1171 = bitcast i8* %allocaCall1170 to i32*
1680.    store i32* %tmpAddr1171, i32** %tmp1169
1681.    %tmpWithVal1172 = load i32*, i32** %tmp1169
1682.    store i32 23, i32* %tmpWithVal1172
1683.    %asChar1173 = bitcast i32* %tmpWithVal1172 to i8*
1684.    %tmp1174 = alloca i32*
1685.    %allocaCall1175 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1686.    %tmpAddr1176 = bitcast i8* %allocaCall1175 to i32*
1687.    store i32* %tmpAddr1176, i32** %tmp1174
1688.    %tmpWithVal1177 = load i32*, i32** %tmp1174
1689.    store i32 24, i32* %tmpWithVal1177
1690.    %asChar1178 = bitcast i32* %tmpWithVal1177 to i8*
1691.    %tmp1179 = alloca i32*
1692.    %allocaCall1180 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1693.    %tmpAddr1181 = bitcast i8* %allocaCall1180 to i32*
1694.    store i32* %tmpAddr1181, i32** %tmp1179
1695.    %tmpWithVal1182 = load i32*, i32** %tmp1179
1696.    store i32 25, i32* %tmpWithVal1182
1697.    %asChar1183 = bitcast i32* %tmpWithVal1182 to i8*

```

1698. %push_back1184 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1058)
1699. %push_back1185 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1063)
1700. %push_back1186 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1068)
1701. %push_back1187 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1073)
1702. %push_back1188 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1078)
1703. %push_back1189 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1083)
1704. %push_back1190 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1088)
1705. %push_back1191 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1093)
1706. %push_back1192 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1098)
1707. %push_back1193 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1103)
1708. %push_back1194 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1108)
1709. %push_back1195 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1113)
1710. %push_back1196 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1118)
1711. %push_back1197 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1123)
1712. %push_back1198 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1128)
1713. %push_back1199 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1133)
1714. %push_back1200 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1138)
1715. %push_back1201 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1143)
1716. %push_back1202 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1148)
1717. %push_back1203 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1153)
1718. %push_back1204 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1158)
1719. %push_back1205 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr1052, i8* %asChar1163)


```

1720.      %push_back1206 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1052, i8* %asChar1168)
1721.      %push_back1207 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1052, i8* %asChar1173)
1722.      %push_back1208 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1052, i8* %asChar1178)
1723.      %push_back1209 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1052, i8* %asChar1183)
1724.      store %list* %listAddr1052, %list** %rotor_three_in
1725.      %mallocall1210 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
1726.      %listAddr1211 = bitcast i8* %mallocall1210 to %list*
1727.      %init_list1212 = call i32 (%list*, ...) @init_list(%list* %listAddr1211)
1728.      %tmp1213 = alloca i32*
1729.      %mallocall1214 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1730.      %tmpAddr1215 = bitcast i8* %mallocall1214 to i32*
1731.      store i32* %tmpAddr1215, i32** %tmp1213
1732.      %tmpWithVal1216 = load i32*, i32** %tmp1213
1733.      store i32 0, i32* %tmpWithVal1216
1734.      %asChar1217 = bitcast i32* %tmpWithVal1216 to i8*
1735.      %tmp1218 = alloca i32*
1736.      %mallocall1219 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1737.      %tmpAddr1220 = bitcast i8* %mallocall1219 to i32*
1738.      store i32* %tmpAddr1220, i32** %tmp1218
1739.      %tmpWithVal1221 = load i32*, i32** %tmp1218
1740.      store i32 9, i32* %tmpWithVal1221
1741.      %asChar1222 = bitcast i32* %tmpWithVal1221 to i8*
1742.      %tmp1223 = alloca i32*
1743.      %mallocall1224 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1744.      %tmpAddr1225 = bitcast i8* %mallocall1224 to i32*
1745.      store i32* %tmpAddr1225, i32** %tmp1223
1746.      %tmpWithVal1226 = load i32*, i32** %tmp1223
1747.      store i32 3, i32* %tmpWithVal1226
1748.      %asChar1227 = bitcast i32* %tmpWithVal1226 to i8*
1749.      %tmp1228 = alloca i32*
1750.      %mallocall1229 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1751.      %tmpAddr1230 = bitcast i8* %mallocall1229 to i32*
1752.      store i32* %tmpAddr1230, i32** %tmp1228
1753.      %tmpWithVal1231 = load i32*, i32** %tmp1228
1754.      store i32 10, i32* %tmpWithVal1231

```

```

1755.    %asChar1232 = bitcast i32* %tmpWithVal1231 to i8*
1756.    %tmp1233 = alloca i32*
1757.    %allocaCall1234 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1758.    %tmpAddr1235 = bitcast i8* %allocaCall1234 to i32*
1759.    store i32* %tmpAddr1235, i32** %tmp1233
1760.    %tmpWithVal1236 = load i32*, i32** %tmp1233
1761.    store i32 18, i32* %tmpWithVal1236
1762.    %asChar1237 = bitcast i32* %tmpWithVal1236 to i8*
1763.    %tmp1238 = alloca i32*
1764.    %allocaCall1239 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1765.    %tmpAddr1240 = bitcast i8* %allocaCall1239 to i32*
1766.    store i32* %tmpAddr1240, i32** %tmp1238
1767.    %tmpWithVal1241 = load i32*, i32** %tmp1238
1768.    store i32 8, i32* %tmpWithVal1241
1769.    %asChar1242 = bitcast i32* %tmpWithVal1241 to i8*
1770.    %tmp1243 = alloca i32*
1771.    %allocaCall1244 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1772.    %tmpAddr1245 = bitcast i8* %allocaCall1244 to i32*
1773.    store i32* %tmpAddr1245, i32** %tmp1243
1774.    %tmpWithVal1246 = load i32*, i32** %tmp1243
1775.    store i32 17, i32* %tmpWithVal1246
1776.    %asChar1247 = bitcast i32* %tmpWithVal1246 to i8*
1777.    %tmp1248 = alloca i32*
1778.    %allocaCall1249 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1779.    %tmpAddr1250 = bitcast i8* %allocaCall1249 to i32*
1780.    store i32* %tmpAddr1250, i32** %tmp1248
1781.    %tmpWithVal1251 = load i32*, i32** %tmp1248
1782.    store i32 20, i32* %tmpWithVal1251
1783.    %asChar1252 = bitcast i32* %tmpWithVal1251 to i8*
1784.    %tmp1253 = alloca i32*
1785.    %allocaCall1254 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1786.    %tmpAddr1255 = bitcast i8* %allocaCall1254 to i32*
1787.    store i32* %tmpAddr1255, i32** %tmp1253
1788.    %tmpWithVal1256 = load i32*, i32** %tmp1253
1789.    store i32 23, i32* %tmpWithVal1256
1790.    %asChar1257 = bitcast i32* %tmpWithVal1256 to i8*
1791.    %tmp1258 = alloca i32*
1792.    %allocaCall1259 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))

```

```

1793.    %tmpAddr1260 = bitcast i8* %alloca1259 to i32*
1794.    store i32* %tmpAddr1260, i32** %tmp1258
1795.    %tmpWithVal1261 = load i32*, i32** %tmp1258
1796.    store i32 1, i32* %tmpWithVal1261
1797.    %asChar1262 = bitcast i32* %tmpWithVal1261 to i8*
1798.    %tmp1263 = alloca i32*
1799.    %alloca1264 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1800.    %tmpAddr1265 = bitcast i8* %alloca1264 to i32*
1801.    store i32* %tmpAddr1265, i32** %tmp1263
1802.    %tmpWithVal1266 = load i32*, i32** %tmp1263
1803.    store i32 11, i32* %tmpWithVal1266
1804.    %asChar1267 = bitcast i32* %tmpWithVal1266 to i8*
1805.    %tmp1268 = alloca i32*
1806.    %alloca1269 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1807.    %tmpAddr1270 = bitcast i8* %alloca1269 to i32*
1808.    store i32* %tmpAddr1270, i32** %tmp1268
1809.    %tmpWithVal1271 = load i32*, i32** %tmp1268
1810.    store i32 7, i32* %tmpWithVal1271
1811.    %asChar1272 = bitcast i32* %tmpWithVal1271 to i8*
1812.    %tmp1273 = alloca i32*
1813.    %alloca1274 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1814.    %tmpAddr1275 = bitcast i8* %alloca1274 to i32*
1815.    store i32* %tmpAddr1275, i32** %tmp1273
1816.    %tmpWithVal1276 = load i32*, i32** %tmp1273
1817.    store i32 22, i32* %tmpWithVal1276
1818.    %asChar1277 = bitcast i32* %tmpWithVal1276 to i8*
1819.    %tmp1278 = alloca i32*
1820.    %alloca1279 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1821.    %tmpAddr1280 = bitcast i8* %alloca1279 to i32*
1822.    store i32* %tmpAddr1280, i32** %tmp1278
1823.    %tmpWithVal1281 = load i32*, i32** %tmp1278
1824.    store i32 19, i32* %tmpWithVal1281
1825.    %asChar1282 = bitcast i32* %tmpWithVal1281 to i8*
1826.    %tmp1283 = alloca i32*
1827.    %alloca1284 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1828.    %tmpAddr1285 = bitcast i8* %alloca1284 to i32*
1829.    store i32* %tmpAddr1285, i32** %tmp1283
1830.    %tmpWithVal1286 = load i32*, i32** %tmp1283
1831.    store i32 12, i32* %tmpWithVal1286

```

```

1832.    %asChar1287 = bitcast i32* %tmpWithVal1286 to i8*
1833.    %tmp1288 = alloca i32*
1834.    %malloccall1289 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1835.    %tmpAddr1290 = bitcast i8* %malloccall1289 to i32*
1836.    store i32* %tmpAddr1290, i32** %tmp1288
1837.    %tmpWithVal1291 = load i32*, i32** %tmp1288
1838.    store i32 2, i32* %tmpWithVal1291
1839.    %asChar1292 = bitcast i32* %tmpWithVal1291 to i8*
1840.    %tmp1293 = alloca i32*
1841.    %malloccall1294 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1842.    %tmpAddr1295 = bitcast i8* %malloccall1294 to i32*
1843.    store i32* %tmpAddr1295, i32** %tmp1293
1844.    %tmpWithVal1296 = load i32*, i32** %tmp1293
1845.    store i32 16, i32* %tmpWithVal1296
1846.    %asChar1297 = bitcast i32* %tmpWithVal1296 to i8*
1847.    %tmp1298 = alloca i32*
1848.    %malloccall1299 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1849.    %tmpAddr1300 = bitcast i8* %malloccall1299 to i32*
1850.    store i32* %tmpAddr1300, i32** %tmp1298
1851.    %tmpWithVal1301 = load i32*, i32** %tmp1298
1852.    store i32 6, i32* %tmpWithVal1301
1853.    %asChar1302 = bitcast i32* %tmpWithVal1301 to i8*
1854.    %tmp1303 = alloca i32*
1855.    %malloccall1304 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1856.    %tmpAddr1305 = bitcast i8* %malloccall1304 to i32*
1857.    store i32* %tmpAddr1305, i32** %tmp1303
1858.    %tmpWithVal1306 = load i32*, i32** %tmp1303
1859.    store i32 25, i32* %tmpWithVal1306
1860.    %asChar1307 = bitcast i32* %tmpWithVal1306 to i8*
1861.    %tmp1308 = alloca i32*
1862.    %malloccall1309 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1863.    %tmpAddr1310 = bitcast i8* %malloccall1309 to i32*
1864.    store i32* %tmpAddr1310, i32** %tmp1308
1865.    %tmpWithVal1311 = load i32*, i32** %tmp1308
1866.    store i32 13, i32* %tmpWithVal1311
1867.    %asChar1312 = bitcast i32* %tmpWithVal1311 to i8*
1868.    %tmp1313 = alloca i32*
1869.    %malloccall1314 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))

```

```

1870.    %tmpAddr1315 = bitcast i8* %alloca1314 to i32*
1871.    store i32* %tmpAddr1315, i32** %tmp1313
1872.    %tmpWithVal1316 = load i32*, i32** %tmp1313
1873.    store i32 15, i32* %tmpWithVal1316
1874.    %asChar1317 = bitcast i32* %tmpWithVal1316 to i8*
1875.    %tmp1318 = alloca i32*
1876.    %alloca1319 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1877.    %tmpAddr1320 = bitcast i8* %alloca1319 to i32*
1878.    store i32* %tmpAddr1320, i32** %tmp1318
1879.    %tmpWithVal1321 = load i32*, i32** %tmp1318
1880.    store i32 24, i32* %tmpWithVal1321
1881.    %asChar1322 = bitcast i32* %tmpWithVal1321 to i8*
1882.    %tmp1323 = alloca i32*
1883.    %alloca1324 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1884.    %tmpAddr1325 = bitcast i8* %alloca1324 to i32*
1885.    store i32* %tmpAddr1325, i32** %tmp1323
1886.    %tmpWithVal1326 = load i32*, i32** %tmp1323
1887.    store i32 5, i32* %tmpWithVal1326
1888.    %asChar1327 = bitcast i32* %tmpWithVal1326 to i8*
1889.    %tmp1328 = alloca i32*
1890.    %alloca1329 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1891.    %tmpAddr1330 = bitcast i8* %alloca1329 to i32*
1892.    store i32* %tmpAddr1330, i32** %tmp1328
1893.    %tmpWithVal1331 = load i32*, i32** %tmp1328
1894.    store i32 21, i32* %tmpWithVal1331
1895.    %asChar1332 = bitcast i32* %tmpWithVal1331 to i8*
1896.    %tmp1333 = alloca i32*
1897.    %alloca1334 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1898.    %tmpAddr1335 = bitcast i8* %alloca1334 to i32*
1899.    store i32* %tmpAddr1335, i32** %tmp1333
1900.    %tmpWithVal1336 = load i32*, i32** %tmp1333
1901.    store i32 14, i32* %tmpWithVal1336
1902.    %asChar1337 = bitcast i32* %tmpWithVal1336 to i8*
1903.    %tmp1338 = alloca i32*
1904.    %alloca1339 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1905.    %tmpAddr1340 = bitcast i8* %alloca1339 to i32*
1906.    store i32* %tmpAddr1340, i32** %tmp1338
1907.    %tmpWithVal1341 = load i32*, i32** %tmp1338
1908.    store i32 4, i32* %tmpWithVal1341

```

```

1909.      %asChar1342 = bitcast i32* %tmpWithVal1341 to i8*
1910.      %push_back1343 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1217)
1911.      %push_back1344 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1222)
1912.      %push_back1345 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1227)
1913.      %push_back1346 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1232)
1914.      %push_back1347 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1237)
1915.      %push_back1348 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1242)
1916.      %push_back1349 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1247)
1917.      %push_back1350 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1252)
1918.      %push_back1351 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1257)
1919.      %push_back1352 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1262)
1920.      %push_back1353 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1267)
1921.      %push_back1354 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1272)
1922.      %push_back1355 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1277)
1923.      %push_back1356 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1282)
1924.      %push_back1357 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1287)
1925.      %push_back1358 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1292)
1926.      %push_back1359 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1297)
1927.      %push_back1360 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1302)
1928.      %push_back1361 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1307)
1929.      %push_back1362 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1312)
1930.      %push_back1363 = call i32 (@push_back(%list*
      %listAddr1211, i8* %asChar1317)

```

```

1931.      %push_back1364 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1211, i8* %asChar1322)
1932.      %push_back1365 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1211, i8* %asChar1327)
1933.      %push_back1366 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1211, i8* %asChar1332)
1934.      %push_back1367 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1211, i8* %asChar1337)
1935.      %push_back1368 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1211, i8* %asChar1342)
1936.      store %list* %listAddr1211, %list** %rotor_two_out
1937.      %alloca1369 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
1938.      %listAddr1370 = bitcast i8* %alloca1369 to %list*
1939.      %init_list1371 = call i32 (%list*, ...) @init_list(%list* %listAddr1370)
1940.      %tmp1372 = alloca i32*
1941.      %alloca1373 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1942.      %tmpAddr1374 = bitcast i8* %alloca1373 to i32*
1943.      store i32* %tmpAddr1374, i32** %tmp1372
1944.      %tmpWithVal1375 = load i32*, i32** %tmp1372
1945.      store i32 0, i32* %tmpWithVal1375
1946.      %asChar1376 = bitcast i32* %tmpWithVal1375 to i8*
1947.      %tmp1377 = alloca i32*
1948.      %alloca1378 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1949.      %tmpAddr1379 = bitcast i8* %alloca1378 to i32*
1950.      store i32* %tmpAddr1379, i32** %tmp1377
1951.      %tmpWithVal1380 = load i32*, i32** %tmp1377
1952.      store i32 1, i32* %tmpWithVal1380
1953.      %asChar1381 = bitcast i32* %tmpWithVal1380 to i8*
1954.      %tmp1382 = alloca i32*
1955.      %alloca1383 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1956.      %tmpAddr1384 = bitcast i8* %alloca1383 to i32*
1957.      store i32* %tmpAddr1384, i32** %tmp1382
1958.      %tmpWithVal1385 = load i32*, i32** %tmp1382
1959.      store i32 2, i32* %tmpWithVal1385
1960.      %asChar1386 = bitcast i32* %tmpWithVal1385 to i8*
1961.      %tmp1387 = alloca i32*
1962.      %alloca1388 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
1963.      %tmpAddr1389 = bitcast i8* %alloca1388 to i32*
1964.      store i32* %tmpAddr1389, i32** %tmp1387

```

```

1965.    %tmpWithVal1390 = load i32*, i32** %tmp1387
1966.    store i32 3, i32* %tmpWithVal1390
1967.    %asChar1391 = bitcast i32* %tmpWithVal1390 to i8*
1968.    %tmp1392 = alloca i32*
1969.    %allocaCall1393 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1970.    %tmpAddr1394 = bitcast i8* %allocaCall1393 to i32*
1971.    store i32* %tmpAddr1394, i32** %tmp1392
1972.    %tmpWithVal1395 = load i32*, i32** %tmp1392
1973.    store i32 4, i32* %tmpWithVal1395
1974.    %asChar1396 = bitcast i32* %tmpWithVal1395 to i8*
1975.    %tmp1397 = alloca i32*
1976.    %allocaCall1398 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1977.    %tmpAddr1399 = bitcast i8* %allocaCall1398 to i32*
1978.    store i32* %tmpAddr1399, i32** %tmp1397
1979.    %tmpWithVal1400 = load i32*, i32** %tmp1397
1980.    store i32 5, i32* %tmpWithVal1400
1981.    %asChar1401 = bitcast i32* %tmpWithVal1400 to i8*
1982.    %tmp1402 = alloca i32*
1983.    %allocaCall1403 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1984.    %tmpAddr1404 = bitcast i8* %allocaCall1403 to i32*
1985.    store i32* %tmpAddr1404, i32** %tmp1402
1986.    %tmpWithVal1405 = load i32*, i32** %tmp1402
1987.    store i32 6, i32* %tmpWithVal1405
1988.    %asChar1406 = bitcast i32* %tmpWithVal1405 to i8*
1989.    %tmp1407 = alloca i32*
1990.    %allocaCall1408 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1991.    %tmpAddr1409 = bitcast i8* %allocaCall1408 to i32*
1992.    store i32* %tmpAddr1409, i32** %tmp1407
1993.    %tmpWithVal1410 = load i32*, i32** %tmp1407
1994.    store i32 7, i32* %tmpWithVal1410
1995.    %asChar1411 = bitcast i32* %tmpWithVal1410 to i8*
1996.    %tmp1412 = alloca i32*
1997.    %allocaCall1413 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
1998.    %tmpAddr1414 = bitcast i8* %allocaCall1413 to i32*
1999.    store i32* %tmpAddr1414, i32** %tmp1412
2000.    %tmpWithVal1415 = load i32*, i32** %tmp1412
2001.    store i32 8, i32* %tmpWithVal1415
2002.    %asChar1416 = bitcast i32* %tmpWithVal1415 to i8*
2003.    %tmp1417 = alloca i32*

```



```

2004.      %malloccall1418 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2005.      %tmpAddr1419 = bitcast i8* %malloccall1418 to i32*
2006.      store i32* %tmpAddr1419, i32** %tmp1417
2007.      %tmpWithVal1420 = load i32*, i32** %tmp1417
2008.      store i32 9, i32* %tmpWithVal1420
2009.      %asChar1421 = bitcast i32* %tmpWithVal1420 to i8*
2010.      %tmp1422 = alloca i32*
2011.      %malloccall1423 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2012.      %tmpAddr1424 = bitcast i8* %malloccall1423 to i32*
2013.      store i32* %tmpAddr1424, i32** %tmp1422
2014.      %tmpWithVal1425 = load i32*, i32** %tmp1422
2015.      store i32 10, i32* %tmpWithVal1425
2016.      %asChar1426 = bitcast i32* %tmpWithVal1425 to i8*
2017.      %tmp1427 = alloca i32*
2018.      %malloccall1428 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2019.      %tmpAddr1429 = bitcast i8* %malloccall1428 to i32*
2020.      store i32* %tmpAddr1429, i32** %tmp1427
2021.      %tmpWithVal1430 = load i32*, i32** %tmp1427
2022.      store i32 11, i32* %tmpWithVal1430
2023.      %asChar1431 = bitcast i32* %tmpWithVal1430 to i8*
2024.      %tmp1432 = alloca i32*
2025.      %malloccall1433 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2026.      %tmpAddr1434 = bitcast i8* %malloccall1433 to i32*
2027.      store i32* %tmpAddr1434, i32** %tmp1432
2028.      %tmpWithVal1435 = load i32*, i32** %tmp1432
2029.      store i32 12, i32* %tmpWithVal1435
2030.      %asChar1436 = bitcast i32* %tmpWithVal1435 to i8*
2031.      %tmp1437 = alloca i32*
2032.      %malloccall1438 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2033.      %tmpAddr1439 = bitcast i8* %malloccall1438 to i32*
2034.      store i32* %tmpAddr1439, i32** %tmp1437
2035.      %tmpWithVal1440 = load i32*, i32** %tmp1437
2036.      store i32 13, i32* %tmpWithVal1440
2037.      %asChar1441 = bitcast i32* %tmpWithVal1440 to i8*
2038.      %tmp1442 = alloca i32*
2039.      %malloccall1443 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2040.      %tmpAddr1444 = bitcast i8* %malloccall1443 to i32*
2041.      store i32* %tmpAddr1444, i32** %tmp1442

```

```

2042.    %tmpWithVal1445 = load i32*, i32** %tmp1442
2043.    store i32 14, i32* %tmpWithVal1445
2044.    %asChar1446 = bitcast i32* %tmpWithVal1445 to i8*
2045.    %tmp1447 = alloca i32*
2046.    %allocaCall1448 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2047.    %tmpAddr1449 = bitcast i8* %allocaCall1448 to i32*
2048.    store i32* %tmpAddr1449, i32** %tmp1447
2049.    %tmpWithVal1450 = load i32*, i32** %tmp1447
2050.    store i32 15, i32* %tmpWithVal1450
2051.    %asChar1451 = bitcast i32* %tmpWithVal1450 to i8*
2052.    %tmp1452 = alloca i32*
2053.    %allocaCall1453 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2054.    %tmpAddr1454 = bitcast i8* %allocaCall1453 to i32*
2055.    store i32* %tmpAddr1454, i32** %tmp1452
2056.    %tmpWithVal1455 = load i32*, i32** %tmp1452
2057.    store i32 16, i32* %tmpWithVal1455
2058.    %asChar1456 = bitcast i32* %tmpWithVal1455 to i8*
2059.    %tmp1457 = alloca i32*
2060.    %allocaCall1458 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2061.    %tmpAddr1459 = bitcast i8* %allocaCall1458 to i32*
2062.    store i32* %tmpAddr1459, i32** %tmp1457
2063.    %tmpWithVal1460 = load i32*, i32** %tmp1457
2064.    store i32 17, i32* %tmpWithVal1460
2065.    %asChar1461 = bitcast i32* %tmpWithVal1460 to i8*
2066.    %tmp1462 = alloca i32*
2067.    %allocaCall1463 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2068.    %tmpAddr1464 = bitcast i8* %allocaCall1463 to i32*
2069.    store i32* %tmpAddr1464, i32** %tmp1462
2070.    %tmpWithVal1465 = load i32*, i32** %tmp1462
2071.    store i32 18, i32* %tmpWithVal1465
2072.    %asChar1466 = bitcast i32* %tmpWithVal1465 to i8*
2073.    %tmp1467 = alloca i32*
2074.    %allocaCall1468 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2075.    %tmpAddr1469 = bitcast i8* %allocaCall1468 to i32*
2076.    store i32* %tmpAddr1469, i32** %tmp1467
2077.    %tmpWithVal1470 = load i32*, i32** %tmp1467
2078.    store i32 19, i32* %tmpWithVal1470
2079.    %asChar1471 = bitcast i32* %tmpWithVal1470 to i8*
2080.    %tmp1472 = alloca i32*

```

```

2081.      %alloca1473 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2082.      %tmpAddr1474 = bitcast i8* %alloca1473 to i32*
2083.      store i32* %tmpAddr1474, i32** %tmp1472
2084.      %tmpWithVal1475 = load i32*, i32** %tmp1472
2085.      store i32 20, i32* %tmpWithVal1475
2086.      %asChar1476 = bitcast i32* %tmpWithVal1475 to i8*
2087.      %tmp1477 = alloca i32*
2088.      %alloca1478 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2089.      %tmpAddr1479 = bitcast i8* %alloca1478 to i32*
2090.      store i32* %tmpAddr1479, i32** %tmp1477
2091.      %tmpWithVal1480 = load i32*, i32** %tmp1477
2092.      store i32 21, i32* %tmpWithVal1480
2093.      %asChar1481 = bitcast i32* %tmpWithVal1480 to i8*
2094.      %tmp1482 = alloca i32*
2095.      %alloca1483 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2096.      %tmpAddr1484 = bitcast i8* %alloca1483 to i32*
2097.      store i32* %tmpAddr1484, i32** %tmp1482
2098.      %tmpWithVal1485 = load i32*, i32** %tmp1482
2099.      store i32 22, i32* %tmpWithVal1485
2100.     %asChar1486 = bitcast i32* %tmpWithVal1485 to i8*
2101.     %tmp1487 = alloca i32*
2102.     %alloca1488 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2103.     %tmpAddr1489 = bitcast i8* %alloca1488 to i32*
2104.     store i32* %tmpAddr1489, i32** %tmp1487
2105.     %tmpWithVal1490 = load i32*, i32** %tmp1487
2106.     store i32 23, i32* %tmpWithVal1490
2107.     %asChar1491 = bitcast i32* %tmpWithVal1490 to i8*
2108.     %tmp1492 = alloca i32*
2109.     %alloca1493 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2110.     %tmpAddr1494 = bitcast i8* %alloca1493 to i32*
2111.     store i32* %tmpAddr1494, i32** %tmp1492
2112.     %tmpWithVal1495 = load i32*, i32** %tmp1492
2113.     store i32 24, i32* %tmpWithVal1495
2114.     %asChar1496 = bitcast i32* %tmpWithVal1495 to i8*
2115.     %tmp1497 = alloca i32*
2116.     %alloca1498 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2117.     %tmpAddr1499 = bitcast i8* %alloca1498 to i32*
2118.     store i32* %tmpAddr1499, i32** %tmp1497

```

```

2119.    %tmpWithVal1500 = load i32*, i32** %tmp1497
2120.    store i32 25, i32* %tmpWithVal1500
2121.    %asChar1501 = bitcast i32* %tmpWithVal1500 to i8*
2122.    %push_back1502 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1376)
2123.    %push_back1503 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1381)
2124.    %push_back1504 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1386)
2125.    %push_back1505 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1391)
2126.    %push_back1506 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1396)
2127.    %push_back1507 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1401)
2128.    %push_back1508 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1406)
2129.    %push_back1509 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1411)
2130.    %push_back1510 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1416)
2131.    %push_back1511 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1421)
2132.    %push_back1512 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1426)
2133.    %push_back1513 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1431)
2134.    %push_back1514 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1436)
2135.    %push_back1515 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1441)
2136.    %push_back1516 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1446)
2137.    %push_back1517 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1451)
2138.    %push_back1518 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1456)
2139.    %push_back1519 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1461)
2140.    %push_back1520 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1466)
2141.    %push_back1521 = call i32 (%list*, i8*, ...) @push_back(%list*
    %listAddr1370, i8* %asChar1471)

```

```

2142.    %push_back1522 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1370, i8* %asChar1476)
2143.    %push_back1523 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1370, i8* %asChar1481)
2144.    %push_back1524 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1370, i8* %asChar1486)
2145.    %push_back1525 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1370, i8* %asChar1491)
2146.    %push_back1526 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1370, i8* %asChar1496)
2147.    %push_back1527 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1370, i8* %asChar1501)
2148.    store %list* %listAddr1370, %list** %rotor_two_in
2149.    %alloca1528 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
        (%list, %list* null, i32 1) to i32))
2150.    %listAddr1529 = bitcast i8* %alloca1528 to %list*
2151.    %init_list1530 = call i32 (%list*, ...) @init_list(%list* %listAddr1529)
2152.    %tmp1531 = alloca i32*
2153.    %alloca1532 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
        (i32, i32* null, i32 1) to i32))
2154.    %tmpAddr1533 = bitcast i8* %alloca1532 to i32*
2155.    store i32* %tmpAddr1533, i32** %tmp1531
2156.    %tmpWithVal1534 = load i32*, i32** %tmp1531
2157.    store i32 4, i32* %tmpWithVal1534
2158.    %asChar1535 = bitcast i32* %tmpWithVal1534 to i8*
2159.    %tmp1536 = alloca i32*
2160.    %alloca1537 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
        (i32, i32* null, i32 1) to i32))
2161.    %tmpAddr1538 = bitcast i8* %alloca1537 to i32*
2162.    store i32* %tmpAddr1538, i32** %tmp1536
2163.    %tmpWithVal1539 = load i32*, i32** %tmp1536
2164.    store i32 10, i32* %tmpWithVal1539
2165.    %asChar1540 = bitcast i32* %tmpWithVal1539 to i8*
2166.    %tmp1541 = alloca i32*
2167.    %alloca1542 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
        (i32, i32* null, i32 1) to i32))
2168.    %tmpAddr1543 = bitcast i8* %alloca1542 to i32*
2169.    store i32* %tmpAddr1543, i32** %tmp1541
2170.    %tmpWithVal1544 = load i32*, i32** %tmp1541
2171.    store i32 12, i32* %tmpWithVal1544
2172.    %asChar1545 = bitcast i32* %tmpWithVal1544 to i8*
2173.    %tmp1546 = alloca i32*
2174.    %alloca1547 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
        (i32, i32* null, i32 1) to i32))

```

```

2175.    %tmpAddr1548 = bitcast i8* %alloca1547 to i32*
2176.    store i32* %tmpAddr1548, i32** %tmp1546
2177.    %tmpWithVal1549 = load i32*, i32** %tmp1546
2178.    store i32 5, i32* %tmpWithVal1549
2179.    %asChar1550 = bitcast i32* %tmpWithVal1549 to i8*
2180.    %tmp1551 = alloca i32*
2181.    %alloca1552 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2182.    %tmpAddr1553 = bitcast i8* %alloca1552 to i32*
2183.    store i32* %tmpAddr1553, i32** %tmp1551
2184.    %tmpWithVal1554 = load i32*, i32** %tmp1551
2185.    store i32 11, i32* %tmpWithVal1554
2186.    %asChar1555 = bitcast i32* %tmpWithVal1554 to i8*
2187.    %tmp1556 = alloca i32*
2188.    %alloca1557 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2189.    %tmpAddr1558 = bitcast i8* %alloca1557 to i32*
2190.    store i32* %tmpAddr1558, i32** %tmp1556
2191.    %tmpWithVal1559 = load i32*, i32** %tmp1556
2192.    store i32 6, i32* %tmpWithVal1559
2193.    %asChar1560 = bitcast i32* %tmpWithVal1559 to i8*
2194.    %tmp1561 = alloca i32*
2195.    %alloca1562 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2196.    %tmpAddr1563 = bitcast i8* %alloca1562 to i32*
2197.    store i32* %tmpAddr1563, i32** %tmp1561
2198.    %tmpWithVal1564 = load i32*, i32** %tmp1561
2199.    store i32 3, i32* %tmpWithVal1564
2200.    %asChar1565 = bitcast i32* %tmpWithVal1564 to i8*
2201.    %tmp1566 = alloca i32*
2202.    %alloca1567 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2203.    %tmpAddr1568 = bitcast i8* %alloca1567 to i32*
2204.    store i32* %tmpAddr1568, i32** %tmp1566
2205.    %tmpWithVal1569 = load i32*, i32** %tmp1566
2206.    store i32 16, i32* %tmpWithVal1569
2207.    %asChar1570 = bitcast i32* %tmpWithVal1569 to i8*
2208.    %tmp1571 = alloca i32*
2209.    %alloca1572 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2210.    %tmpAddr1573 = bitcast i8* %alloca1572 to i32*
2211.    store i32* %tmpAddr1573, i32** %tmp1571
2212.    %tmpWithVal1574 = load i32*, i32** %tmp1571
2213.    store i32 21, i32* %tmpWithVal1574

```

```

2214.    %asChar1575 = bitcast i32* %tmpWithVal1574 to i8*
2215.    %tmp1576 = alloca i32*
2216.    %allocaCall1577 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2217.    %tmpAddr1578 = bitcast i8* %allocaCall1577 to i32*
2218.    store i32* %tmpAddr1578, i32** %tmp1576
2219.    %tmpWithVal1579 = load i32*, i32** %tmp1576
2220.    store i32 25, i32* %tmpWithVal1579
2221.    %asChar1580 = bitcast i32* %tmpWithVal1579 to i8*
2222.    %tmp1581 = alloca i32*
2223.    %allocaCall1582 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2224.    %tmpAddr1583 = bitcast i8* %allocaCall1582 to i32*
2225.    store i32* %tmpAddr1583, i32** %tmp1581
2226.    %tmpWithVal1584 = load i32*, i32** %tmp1581
2227.    store i32 13, i32* %tmpWithVal1584
2228.    %asChar1585 = bitcast i32* %tmpWithVal1584 to i8*
2229.    %tmp1586 = alloca i32*
2230.    %allocaCall1587 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2231.    %tmpAddr1588 = bitcast i8* %allocaCall1587 to i32*
2232.    store i32* %tmpAddr1588, i32** %tmp1586
2233.    %tmpWithVal1589 = load i32*, i32** %tmp1586
2234.    store i32 19, i32* %tmpWithVal1589
2235.    %asChar1590 = bitcast i32* %tmpWithVal1589 to i8*
2236.    %tmp1591 = alloca i32*
2237.    %allocaCall1592 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2238.    %tmpAddr1593 = bitcast i8* %allocaCall1592 to i32*
2239.    store i32* %tmpAddr1593, i32** %tmp1591
2240.    %tmpWithVal1594 = load i32*, i32** %tmp1591
2241.    store i32 14, i32* %tmpWithVal1594
2242.    %asChar1595 = bitcast i32* %tmpWithVal1594 to i8*
2243.    %tmp1596 = alloca i32*
2244.    %allocaCall1597 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2245.    %tmpAddr1598 = bitcast i8* %allocaCall1597 to i32*
2246.    store i32* %tmpAddr1598, i32** %tmp1596
2247.    %tmpWithVal1599 = load i32*, i32** %tmp1596
2248.    store i32 22, i32* %tmpWithVal1599
2249.    %asChar1600 = bitcast i32* %tmpWithVal1599 to i8*
2250.    %tmp1601 = alloca i32*
2251.    %allocaCall1602 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))

```

```

2252.    %tmpAddr1603 = bitcast i8* %alloca1602 to i32*
2253.    store i32* %tmpAddr1603, i32** %tmp1601
2254.    %tmpWithVal1604 = load i32*, i32** %tmp1601
2255.    store i32 24, i32* %tmpWithVal1604
2256.    %asChar1605 = bitcast i32* %tmpWithVal1604 to i8*
2257.    %tmp1606 = alloca i32*
2258.    %alloca1607 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2259.    %tmpAddr1608 = bitcast i8* %alloca1607 to i32*
2260.    store i32* %tmpAddr1608, i32** %tmp1606
2261.    %tmpWithVal1609 = load i32*, i32** %tmp1606
2262.    store i32 7, i32* %tmpWithVal1609
2263.    %asChar1610 = bitcast i32* %tmpWithVal1609 to i8*
2264.    %tmp1611 = alloca i32*
2265.    %alloca1612 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2266.    %tmpAddr1613 = bitcast i8* %alloca1612 to i32*
2267.    store i32* %tmpAddr1613, i32** %tmp1611
2268.    %tmpWithVal1614 = load i32*, i32** %tmp1611
2269.    store i32 23, i32* %tmpWithVal1614
2270.    %asChar1615 = bitcast i32* %tmpWithVal1614 to i8*
2271.    %tmp1616 = alloca i32*
2272.    %alloca1617 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2273.    %tmpAddr1618 = bitcast i8* %alloca1617 to i32*
2274.    store i32* %tmpAddr1618, i32** %tmp1616
2275.    %tmpWithVal1619 = load i32*, i32** %tmp1616
2276.    store i32 20, i32* %tmpWithVal1619
2277.    %asChar1620 = bitcast i32* %tmpWithVal1619 to i8*
2278.    %tmp1621 = alloca i32*
2279.    %alloca1622 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2280.    %tmpAddr1623 = bitcast i8* %alloca1622 to i32*
2281.    store i32* %tmpAddr1623, i32** %tmp1621
2282.    %tmpWithVal1624 = load i32*, i32** %tmp1621
2283.    store i32 18, i32* %tmpWithVal1624
2284.    %asChar1625 = bitcast i32* %tmpWithVal1624 to i8*
2285.    %tmp1626 = alloca i32*
2286.    %alloca1627 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2287.    %tmpAddr1628 = bitcast i8* %alloca1627 to i32*
2288.    store i32* %tmpAddr1628, i32** %tmp1626
2289.    %tmpWithVal1629 = load i32*, i32** %tmp1626
2290.    store i32 15, i32* %tmpWithVal1629

```



```

2291.    %asChar1630 = bitcast i32* %tmpWithVal1629 to i8*
2292.    %tmp1631 = alloca i32*
2293.    %allocaCall1632 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2294.    %tmpAddr1633 = bitcast i8* %allocaCall1632 to i32*
2295.    store i32* %tmpAddr1633, i32** %tmp1631
2296.    %tmpWithVal1634 = load i32*, i32** %tmp1631
2297.    store i32 0, i32* %tmpWithVal1634
2298.    %asChar1635 = bitcast i32* %tmpWithVal1634 to i8*
2299.    %tmp1636 = alloca i32*
2300.    %allocaCall1637 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2301.    %tmpAddr1638 = bitcast i8* %allocaCall1637 to i32*
2302.    store i32* %tmpAddr1638, i32** %tmp1636
2303.    %tmpWithVal1639 = load i32*, i32** %tmp1636
2304.    store i32 8, i32* %tmpWithVal1639
2305.    %asChar1640 = bitcast i32* %tmpWithVal1639 to i8*
2306.    %tmp1641 = alloca i32*
2307.    %allocaCall1642 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2308.    %tmpAddr1643 = bitcast i8* %allocaCall1642 to i32*
2309.    store i32* %tmpAddr1643, i32** %tmp1641
2310.    %tmpWithVal1644 = load i32*, i32** %tmp1641
2311.    store i32 1, i32* %tmpWithVal1644
2312.    %asChar1645 = bitcast i32* %tmpWithVal1644 to i8*
2313.    %tmp1646 = alloca i32*
2314.    %allocaCall1647 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2315.    %tmpAddr1648 = bitcast i8* %allocaCall1647 to i32*
2316.    store i32* %tmpAddr1648, i32** %tmp1646
2317.    %tmpWithVal1649 = load i32*, i32** %tmp1646
2318.    store i32 17, i32* %tmpWithVal1649
2319.    %asChar1650 = bitcast i32* %tmpWithVal1649 to i8*
2320.    %tmp1651 = alloca i32*
2321.    %allocaCall1652 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2322.    %tmpAddr1653 = bitcast i8* %allocaCall1652 to i32*
2323.    store i32* %tmpAddr1653, i32** %tmp1651
2324.    %tmpWithVal1654 = load i32*, i32** %tmp1651
2325.    store i32 2, i32* %tmpWithVal1654
2326.    %asChar1655 = bitcast i32* %tmpWithVal1654 to i8*
2327.    %tmp1656 = alloca i32*
2328.    %allocaCall1657 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))

```

```

2329.      %tmpAddr1658 = bitcast i8* %malloccall1657 to i32*
2330.      store i32* %tmpAddr1658, i32** %tmp1656
2331.      %tmpWithVal1659 = load i32*, i32** %tmp1656
2332.      store i32 9, i32* %tmpWithVal1659
2333.      %asChar1660 = bitcast i32* %tmpWithVal1659 to i8*
2334.      %push_back1661 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1535)
2335.      %push_back1662 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1540)
2336.      %push_back1663 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1545)
2337.      %push_back1664 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1550)
2338.      %push_back1665 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1555)
2339.      %push_back1666 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1560)
2340.      %push_back1667 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1565)
2341.      %push_back1668 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1570)
2342.      %push_back1669 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1575)
2343.      %push_back1670 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1580)
2344.      %push_back1671 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1585)
2345.      %push_back1672 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1590)
2346.      %push_back1673 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1595)
2347.      %push_back1674 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1600)
2348.      %push_back1675 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1605)
2349.      %push_back1676 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1610)
2350.      %push_back1677 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1615)
2351.      %push_back1678 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1620)
2352.      %push_back1679 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr1529, i8* %asChar1625)

```

```

2353.      %push_back1680 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1529, i8* %asChar1630)
2354.      %push_back1681 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1529, i8* %asChar1635)
2355.      %push_back1682 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1529, i8* %asChar1640)
2356.      %push_back1683 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1529, i8* %asChar1645)
2357.      %push_back1684 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1529, i8* %asChar1650)
2358.      %push_back1685 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1529, i8* %asChar1655)
2359.      %push_back1686 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1529, i8* %asChar1660)
2360.      store %list* %listAddr1529, %list** %rotor_one_out
2361.      %mallocall1687 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
2362.      %listAddr1688 = bitcast i8* %mallocall1687 to %list*
2363.      %init_list1689 = call i32 (%list*, ...) @init_list(%list* %listAddr1688)
2364.      %tmp1690 = alloca i32*
2365.      %mallocall1691 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2366.      %tmpAddr1692 = bitcast i8* %mallocall1691 to i32*
2367.      store i32* %tmpAddr1692, i32** %tmp1690
2368.      %tmpWithVal1693 = load i32*, i32** %tmp1690
2369.      store i32 0, i32* %tmpWithVal1693
2370.      %asChar1694 = bitcast i32* %tmpWithVal1693 to i8*
2371.      %tmp1695 = alloca i32*
2372.      %mallocall1696 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2373.      %tmpAddr1697 = bitcast i8* %mallocall1696 to i32*
2374.      store i32* %tmpAddr1697, i32** %tmp1695
2375.      %tmpWithVal1698 = load i32*, i32** %tmp1695
2376.      store i32 1, i32* %tmpWithVal1698
2377.      %asChar1699 = bitcast i32* %tmpWithVal1698 to i8*
2378.      %tmp1700 = alloca i32*
2379.      %mallocall1701 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2380.      %tmpAddr1702 = bitcast i8* %mallocall1701 to i32*
2381.      store i32* %tmpAddr1702, i32** %tmp1700
2382.      %tmpWithVal1703 = load i32*, i32** %tmp1700
2383.      store i32 2, i32* %tmpWithVal1703
2384.      %asChar1704 = bitcast i32* %tmpWithVal1703 to i8*
2385.      %tmp1705 = alloca i32*

```

```

2386.      %malloccall1706 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2387.      %tmpAddr1707 = bitcast i8* %malloccall1706 to i32*
2388.      store i32* %tmpAddr1707, i32** %tmp1705
2389.      %tmpWithVal1708 = load i32*, i32** %tmp1705
2390.      store i32 3, i32* %tmpWithVal1708
2391.      %asChar1709 = bitcast i32* %tmpWithVal1708 to i8*
2392.      %tmp1710 = alloca i32*
2393.      %malloccall1711 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2394.      %tmpAddr1712 = bitcast i8* %malloccall1711 to i32*
2395.      store i32* %tmpAddr1712, i32** %tmp1710
2396.      %tmpWithVal1713 = load i32*, i32** %tmp1710
2397.      store i32 4, i32* %tmpWithVal1713
2398.      %asChar1714 = bitcast i32* %tmpWithVal1713 to i8*
2399.      %tmp1715 = alloca i32*
2400.      %malloccall1716 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2401.      %tmpAddr1717 = bitcast i8* %malloccall1716 to i32*
2402.      store i32* %tmpAddr1717, i32** %tmp1715
2403.      %tmpWithVal1718 = load i32*, i32** %tmp1715
2404.      store i32 5, i32* %tmpWithVal1718
2405.      %asChar1719 = bitcast i32* %tmpWithVal1718 to i8*
2406.      %tmp1720 = alloca i32*
2407.      %malloccall1721 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2408.      %tmpAddr1722 = bitcast i8* %malloccall1721 to i32*
2409.      store i32* %tmpAddr1722, i32** %tmp1720
2410.      %tmpWithVal1723 = load i32*, i32** %tmp1720
2411.      store i32 6, i32* %tmpWithVal1723
2412.      %asChar1724 = bitcast i32* %tmpWithVal1723 to i8*
2413.      %tmp1725 = alloca i32*
2414.      %malloccall1726 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2415.      %tmpAddr1727 = bitcast i8* %malloccall1726 to i32*
2416.      store i32* %tmpAddr1727, i32** %tmp1725
2417.      %tmpWithVal1728 = load i32*, i32** %tmp1725
2418.      store i32 7, i32* %tmpWithVal1728
2419.      %asChar1729 = bitcast i32* %tmpWithVal1728 to i8*
2420.      %tmp1730 = alloca i32*
2421.      %malloccall1731 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2422.      %tmpAddr1732 = bitcast i8* %malloccall1731 to i32*
2423.      store i32* %tmpAddr1732, i32** %tmp1730

```

```

2424.    %tmpWithVal1733 = load i32*, i32** %tmp1730
2425.    store i32 8, i32* %tmpWithVal1733
2426.    %asChar1734 = bitcast i32* %tmpWithVal1733 to i8*
2427.    %tmp1735 = alloca i32*
2428.    %allocaCall1736 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2429.    %tmpAddr1737 = bitcast i8* %allocaCall1736 to i32*
2430.    store i32* %tmpAddr1737, i32** %tmp1735
2431.    %tmpWithVal1738 = load i32*, i32** %tmp1735
2432.    store i32 9, i32* %tmpWithVal1738
2433.    %asChar1739 = bitcast i32* %tmpWithVal1738 to i8*
2434.    %tmp1740 = alloca i32*
2435.    %allocaCall1741 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2436.    %tmpAddr1742 = bitcast i8* %allocaCall1741 to i32*
2437.    store i32* %tmpAddr1742, i32** %tmp1740
2438.    %tmpWithVal1743 = load i32*, i32** %tmp1740
2439.    store i32 10, i32* %tmpWithVal1743
2440.    %asChar1744 = bitcast i32* %tmpWithVal1743 to i8*
2441.    %tmp1745 = alloca i32*
2442.    %allocaCall1746 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2443.    %tmpAddr1747 = bitcast i8* %allocaCall1746 to i32*
2444.    store i32* %tmpAddr1747, i32** %tmp1745
2445.    %tmpWithVal1748 = load i32*, i32** %tmp1745
2446.    store i32 11, i32* %tmpWithVal1748
2447.    %asChar1749 = bitcast i32* %tmpWithVal1748 to i8*
2448.    %tmp1750 = alloca i32*
2449.    %allocaCall1751 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2450.    %tmpAddr1752 = bitcast i8* %allocaCall1751 to i32*
2451.    store i32* %tmpAddr1752, i32** %tmp1750
2452.    %tmpWithVal1753 = load i32*, i32** %tmp1750
2453.    store i32 12, i32* %tmpWithVal1753
2454.    %asChar1754 = bitcast i32* %tmpWithVal1753 to i8*
2455.    %tmp1755 = alloca i32*
2456.    %allocaCall1756 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2457.    %tmpAddr1757 = bitcast i8* %allocaCall1756 to i32*
2458.    store i32* %tmpAddr1757, i32** %tmp1755
2459.    %tmpWithVal1758 = load i32*, i32** %tmp1755
2460.    store i32 13, i32* %tmpWithVal1758
2461.    %asChar1759 = bitcast i32* %tmpWithVal1758 to i8*
2462.    %tmp1760 = alloca i32*

```

```

2463.      %malloccall1761 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2464.      %tmpAddr1762 = bitcast i8* %malloccall1761 to i32*
2465.      store i32* %tmpAddr1762, i32** %tmp1760
2466.      %tmpWithVal1763 = load i32*, i32** %tmp1760
2467.      store i32 14, i32* %tmpWithVal1763
2468.      %asChar1764 = bitcast i32* %tmpWithVal1763 to i8*
2469.      %tmp1765 = alloca i32*
2470.      %malloccall1766 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2471.      %tmpAddr1767 = bitcast i8* %malloccall1766 to i32*
2472.      store i32* %tmpAddr1767, i32** %tmp1765
2473.      %tmpWithVal1768 = load i32*, i32** %tmp1765
2474.      store i32 15, i32* %tmpWithVal1768
2475.      %asChar1769 = bitcast i32* %tmpWithVal1768 to i8*
2476.      %tmp1770 = alloca i32*
2477.      %malloccall1771 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2478.      %tmpAddr1772 = bitcast i8* %malloccall1771 to i32*
2479.      store i32* %tmpAddr1772, i32** %tmp1770
2480.      %tmpWithVal1773 = load i32*, i32** %tmp1770
2481.      store i32 16, i32* %tmpWithVal1773
2482.      %asChar1774 = bitcast i32* %tmpWithVal1773 to i8*
2483.      %tmp1775 = alloca i32*
2484.      %malloccall1776 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2485.      %tmpAddr1777 = bitcast i8* %malloccall1776 to i32*
2486.      store i32* %tmpAddr1777, i32** %tmp1775
2487.      %tmpWithVal1778 = load i32*, i32** %tmp1775
2488.      store i32 17, i32* %tmpWithVal1778
2489.      %asChar1779 = bitcast i32* %tmpWithVal1778 to i8*
2490.      %tmp1780 = alloca i32*
2491.      %malloccall1781 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2492.      %tmpAddr1782 = bitcast i8* %malloccall1781 to i32*
2493.      store i32* %tmpAddr1782, i32** %tmp1780
2494.      %tmpWithVal1783 = load i32*, i32** %tmp1780
2495.      store i32 18, i32* %tmpWithVal1783
2496.      %asChar1784 = bitcast i32* %tmpWithVal1783 to i8*
2497.      %tmp1785 = alloca i32*
2498.      %malloccall1786 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2499.      %tmpAddr1787 = bitcast i8* %malloccall1786 to i32*
2500.      store i32* %tmpAddr1787, i32** %tmp1785

```

```

2501.    %tmpWithVal1788 = load i32*, i32** %tmp1785
2502.    store i32 19, i32* %tmpWithVal1788
2503.    %asChar1789 = bitcast i32* %tmpWithVal1788 to i8*
2504.    %tmp1790 = alloca i32*
2505.    %allocaCall1791 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2506.    %tmpAddr1792 = bitcast i8* %allocaCall1791 to i32*
2507.    store i32* %tmpAddr1792, i32** %tmp1790
2508.    %tmpWithVal1793 = load i32*, i32** %tmp1790
2509.    store i32 20, i32* %tmpWithVal1793
2510.    %asChar1794 = bitcast i32* %tmpWithVal1793 to i8*
2511.    %tmp1795 = alloca i32*
2512.    %allocaCall1796 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2513.    %tmpAddr1797 = bitcast i8* %allocaCall1796 to i32*
2514.    store i32* %tmpAddr1797, i32** %tmp1795
2515.    %tmpWithVal1798 = load i32*, i32** %tmp1795
2516.    store i32 21, i32* %tmpWithVal1798
2517.    %asChar1799 = bitcast i32* %tmpWithVal1798 to i8*
2518.    %tmp1800 = alloca i32*
2519.    %allocaCall1801 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2520.    %tmpAddr1802 = bitcast i8* %allocaCall1801 to i32*
2521.    store i32* %tmpAddr1802, i32** %tmp1800
2522.    %tmpWithVal1803 = load i32*, i32** %tmp1800
2523.    store i32 22, i32* %tmpWithVal1803
2524.    %asChar1804 = bitcast i32* %tmpWithVal1803 to i8*
2525.    %tmp1805 = alloca i32*
2526.    %allocaCall1806 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2527.    %tmpAddr1807 = bitcast i8* %allocaCall1806 to i32*
2528.    store i32* %tmpAddr1807, i32** %tmp1805
2529.    %tmpWithVal1808 = load i32*, i32** %tmp1805
2530.    store i32 23, i32* %tmpWithVal1808
2531.    %asChar1809 = bitcast i32* %tmpWithVal1808 to i8*
2532.    %tmp1810 = alloca i32*
2533.    %allocaCall1811 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2534.    %tmpAddr1812 = bitcast i8* %allocaCall1811 to i32*
2535.    store i32* %tmpAddr1812, i32** %tmp1810
2536.    %tmpWithVal1813 = load i32*, i32** %tmp1810
2537.    store i32 24, i32* %tmpWithVal1813
2538.    %asChar1814 = bitcast i32* %tmpWithVal1813 to i8*
2539.    %tmp1815 = alloca i32*

```

```

2540.      %malloccall1816 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
2541.      %tmpAddr1817 = bitcast i8* %malloccall1816 to i32*
2542.      store i32* %tmpAddr1817, i32** %tmp1815
2543.      %tmpWithVal1818 = load i32*, i32** %tmp1815
2544.      store i32 25, i32* %tmpWithVal1818
2545.      %asChar1819 = bitcast i32* %tmpWithVal1818 to i8*
2546.      %push_back1820 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1694)
2547.      %push_back1821 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1699)
2548.      %push_back1822 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1704)
2549.      %push_back1823 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1709)
2550.      %push_back1824 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1714)
2551.      %push_back1825 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1719)
2552.      %push_back1826 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1724)
2553.      %push_back1827 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1729)
2554.      %push_back1828 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1734)
2555.      %push_back1829 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1739)
2556.      %push_back1830 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1744)
2557.      %push_back1831 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1749)
2558.      %push_back1832 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1754)
2559.      %push_back1833 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1759)
2560.      %push_back1834 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1764)
2561.      %push_back1835 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1769)
2562.      %push_back1836 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1774)
2563.      %push_back1837 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1779)

```



```

2564.      %push_back1838 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1784)
2565.      %push_back1839 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1789)
2566.      %push_back1840 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1794)
2567.      %push_back1841 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1799)
2568.      %push_back1842 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1804)
2569.      %push_back1843 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1809)
2570.      %push_back1844 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1814)
2571.      %push_back1845 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr1688, i8* %asChar1819)
2572.      store %list* %listAddr1688, %list** %rotor_one_in
2573.      %key1846 = load %list*, %list** %key
2574.      %get_at = call i8* (%list*, i32, ...) @get_at(%list* %key1846, i32 0)
2575.      %eltAsPtr = bitcast i8* %get_at to i32*
2576.      %eltDeref = load i32, i32* %eltAsPtr
2577.      %rotor_one_in1847 = load %list*, %list** %rotor_one_in
2578.      %character_location_result = call i32 @character_location(%list*
          %rotor_one_in1847, i32 %eltDeref)
2579.      store i32 %character_location_result, i32*
          %location_on_initialization_array
2580.      %location_on_initialization_array1848 = load i32, i32*
          %location_on_initialization_array
2581.      %rotor_one_in1849 = load %list*, %list** %rotor_one_in
2582.      %rotor_shift_result = call %list* @rotor_shift(%list* %rotor_one_in1849,
          i32 %location_on_initialization_array1848)
2583.      store %list* %rotor_shift_result, %list** %rotor_one_in
2584.      %ring_setting1850 = load %list*, %list** %ring_setting
2585.      %get_at1851 = call i8* (%list*, i32, ...) @get_at(%list*
          %ring_setting1850, i32 0)
2586.      %eltAsPtr1852 = bitcast i8* %get_at1851 to i32*
2587.      %eltDeref1853 = load i32, i32* %eltAsPtr1852
2588.      %rotor_one_out1854 = load %list*, %list** %rotor_one_out
2589.      %character_location_result1855 = call i32 @character_location(%list*
          %rotor_one_out1854, i32 %eltDeref1853)
2590.      store i32 %character_location_result1855, i32*
          %location_on_initialization_array
2591.      %location_on_initialization_array1856 = load i32, i32*
          %location_on_initialization_array

```

```

2592.      %rotor_one_out1857 = load %list*, %list** %rotor_one_out
2593.      %rotor_shift_result1858 = call %list* @rotor_shift(%list*
      %rotor_one_out1857, i32 %location_on_initialization_array1856)
2594.      store %list* %rotor_shift_result1858, %list** %rotor_one_out
2595.      %key1859 = load %list*, %list** %key
2596.      %get_at1860 = call i8* (%list*, i32, ...) @get_at(%list* %key1859, i32 1)
2597.      %eltAsPtr1861 = bitcast i8* %get_at1860 to i32*
2598.      %eltDeref1862 = load i32, i32* %eltAsPtr1861
2599.      %rotor_two_in1863 = load %list*, %list** %rotor_two_in
2600.      %character_location_result1864 = call i32 @character_location(%list*
      %rotor_two_in1863, i32 %eltDeref1862)
2601.      store i32 %character_location_result1864, i32*
      %location_on_initialization_array
2602.      %location_on_initialization_array1865 = load i32, i32*
      %location_on_initialization_array
2603.      %rotor_two_in1866 = load %list*, %list** %rotor_two_in
2604.      %rotor_shift_result1867 = call %list* @rotor_shift(%list*
      %rotor_two_in1866, i32 %location_on_initialization_array1865)
2605.      store %list* %rotor_shift_result1867, %list** %rotor_two_in
2606.      %ring_setting1868 = load %list*, %list** %ring_setting
2607.      %get_at1869 = call i8* (%list*, i32, ...) @get_at(%list*
      %ring_setting1868, i32 1)
2608.      %eltAsPtr1870 = bitcast i8* %get_at1869 to i32*
2609.      %eltDeref1871 = load i32, i32* %eltAsPtr1870
2610.      %rotor_two_out1872 = load %list*, %list** %rotor_two_out
2611.      %character_location_result1873 = call i32 @character_location(%list*
      %rotor_two_out1872, i32 %eltDeref1871)
2612.      store i32 %character_location_result1873, i32*
      %location_on_initialization_array
2613.      %location_on_initialization_array1874 = load i32, i32*
      %location_on_initialization_array
2614.      %rotor_two_out1875 = load %list*, %list** %rotor_two_out
2615.      %rotor_shift_result1876 = call %list* @rotor_shift(%list*
      %rotor_two_out1875, i32 %location_on_initialization_array1874)
2616.      store %list* %rotor_shift_result1876, %list** %rotor_two_out
2617.      %key1877 = load %list*, %list** %key
2618.      %get_at1878 = call i8* (%list*, i32, ...) @get_at(%list* %key1877, i32 2)
2619.      %eltAsPtr1879 = bitcast i8* %get_at1878 to i32*
2620.      %eltDeref1880 = load i32, i32* %eltAsPtr1879
2621.      %rotor_three_in1881 = load %list*, %list** %rotor_three_in
2622.      %character_location_result1882 = call i32 @character_location(%list*
      %rotor_three_in1881, i32 %eltDeref1880)
2623.      store i32 %character_location_result1882, i32*
      %location_on_initialization_array

```

```

2624.      %location_on_initialization_array1883 = load i32, i32*
          %location_on_initialization_array
2625.      %rotor_three_in1884 = load %list*, %list** %rotor_three_in
2626.      %rotor_shift_result1885 = call %list* @rotor_shift(%list*
          %rotor_three_in1884, i32 %location_on_initialization_array1883)
2627.      store %list* %rotor_shift_result1885, %list** %rotor_three_in
2628.      %ring_setting1886 = load %list*, %list** %ring_setting
2629.      %get_at1887 = call i8* (%list*, i32, ...) @get_at(%list*
          %ring_setting1886, i32 2)
2630.      %eltAsPtr1888 = bitcast i8* %get_at1887 to i32*
2631.      %eltDeref1889 = load i32, i32* %eltAsPtr1888
2632.      %rotor_three_out1890 = load %list*, %list** %rotor_three_out
2633.      %character_location_result1891 = call i32 @character_location(%list*
          %rotor_three_out1890, i32 %eltDeref1889)
2634.      store i32 %character_location_result1891, i32*
          %location_on_initialization_array
2635.      %location_on_initialization_array1892 = load i32, i32*
          %location_on_initialization_array
2636.      %rotor_three_out1893 = load %list*, %list** %rotor_three_out
2637.      %rotor_shift_result1894 = call %list* @rotor_shift(%list*
          %rotor_three_out1893, i32 %location_on_initialization_array1892)
2638.      store %list* %rotor_shift_result1894, %list** %rotor_three_out
2639.      %input_text1895 = load %list*, %list** %input_text
2640.      %messagez1896 = load %list*, %list** %messagez
2641.      %initialize_input_text_result = call %list* @initialize_input_text(%list*
          %messagez1896, %list* %input_text1895)
2642.      store %list* %initialize_input_text_result, %list** %input_text
2643.      %length_input1897 = load i32, i32* %length_input
2644.      %second_plugs_init1898 = load %list*, %list** %second_plugs_init
2645.      %first_plugs_init1899 = load %list*, %list** %first_plugs_init
2646.      %plugboard_input_count1900 = load i32, i32* %plugboard_input_count
2647.      %input_text1901 = load %list*, %list** %input_text
2648.      %plugboard_redirect_result = call %list* @plugboard_redirect(%list*
          %input_text1901, i32 %plugboard_input_count1900, %list*
          %first_plugs_init1899, %list* %second_plugs_init1898, i32 %length_input1897)
2649.      store %list* %plugboard_redirect_result, %list** %input_text
2650.      %length_input1902 = load i32, i32* %length_input
2651.      %input_text1903 = load %list*, %list** %input_text
2652.      %rotor_three_out1904 = load %list*, %list** %rotor_three_out
2653.      %rotor_three_in1905 = load %list*, %list** %rotor_three_in
2654.      %rotor_two_out1906 = load %list*, %list** %rotor_two_out
2655.      %rotor_two_in1907 = load %list*, %list** %rotor_two_in
2656.      %rotor_one_out1908 = load %list*, %list** %rotor_one_out
2657.      %rotor_one_in1909 = load %list*, %list** %rotor_one_in

```

```

2658.      %encryption_decryption_result = call %list*
          @encryption_decryption(%list* %rotor_one_in1909, %list* %rotor_one_out1908,
          %list* %rotor_two_in1907, %list* %rotor_two_out1906, %list*
          %rotor_three_in1905, %list* %rotor_three_out1904, %list* %input_text1903, i32
          %length_input1902)
2659.      store %list* %encryption_decryption_result, %list** %temp
2660.      %length_input1910 = load i32, i32* %length_input
2661.      %second_plugs_init1911 = load %list*, %list** %second_plugs_init
2662.      %first_plugs_init1912 = load %list*, %list** %first_plugs_init
2663.      %plugboard_input_count1913 = load i32, i32* %plugboard_input_count
2664.      %temp1914 = load %list*, %list** %temp
2665.      %plugboard_redirect_result1915 = call %list*
          @plugboard_redirect(%list* %temp1914, i32 %plugboard_input_count1913,
          %list* %first_plugs_init1912, %list* %second_plugs_init1911, i32
          %length_input1910)
2666.      store %list* %plugboard_redirect_result1915, %list** %temp
2667.      %mallocall1916 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
2668.      %listAddr1917 = bitcast i8* %mallocall1916 to %list*
2669.      %init_list1918 = call i32 (%list*, ...) @init_list(%list* %listAddr1917)
2670.      %tmp1919 = alloca i8*
2671.      %tmpAddr1921 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
          i8* null, i32 1) to i32))
2672.      store i8* %tmpAddr1921, i8** %tmp1919
2673.      %tmpWithVal1922 = load i8*, i8** %tmp1919
2674.      store i8 79, i8* %tmpWithVal1922
2675.      %tmp1923 = alloca i8*
2676.      %tmpAddr1925 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
          i8* null, i32 1) to i32))
2677.      store i8* %tmpAddr1925, i8** %tmp1923
2678.      %tmpWithVal1926 = load i8*, i8** %tmp1923
2679.      store i8 117, i8* %tmpWithVal1926
2680.      %tmp1927 = alloca i8*
2681.      %tmpAddr1929 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
          i8* null, i32 1) to i32))
2682.      store i8* %tmpAddr1929, i8** %tmp1927
2683.      %tmpWithVal1930 = load i8*, i8** %tmp1927
2684.      store i8 116, i8* %tmpWithVal1930
2685.      %tmp1931 = alloca i8*
2686.      %tmpAddr1933 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
          i8* null, i32 1) to i32))
2687.      store i8* %tmpAddr1933, i8** %tmp1931
2688.      %tmpWithVal1934 = load i8*, i8** %tmp1931
2689.      store i8 112, i8* %tmpWithVal1934

```

```

2690.    %tmp1935 = alloca i8*
2691.    %tmpAddr1937 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
2692.    store i8* %tmpAddr1937, i8** %tmp1935
2693.    %tmpWithVal1938 = load i8*, i8** %tmp1935
2694.    store i8 117, i8* %tmpWithVal1938
2695.    %tmp1939 = alloca i8*
2696.    %tmpAddr1941 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
2697.    store i8* %tmpAddr1941, i8** %tmp1939
2698.    %tmpWithVal1942 = load i8*, i8** %tmp1939
2699.    store i8 116, i8* %tmpWithVal1942
2700.    %tmp1943 = alloca i8*
2701.    %tmpAddr1945 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
2702.    store i8* %tmpAddr1945, i8** %tmp1943
2703.    %tmpWithVal1946 = load i8*, i8** %tmp1943
2704.    store i8 32, i8* %tmpWithVal1946
2705.    %tmp1947 = alloca i8*
2706.    %tmpAddr1949 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
2707.    store i8* %tmpAddr1949, i8** %tmp1947
2708.    %tmpWithVal1950 = load i8*, i8** %tmp1947
2709.    store i8 116, i8* %tmpWithVal1950
2710.    %tmp1951 = alloca i8*
2711.    %tmpAddr1953 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
2712.    store i8* %tmpAddr1953, i8** %tmp1951
2713.    %tmpWithVal1954 = load i8*, i8** %tmp1951
2714.    store i8 101, i8* %tmpWithVal1954
2715.    %tmp1955 = alloca i8*
2716.    %tmpAddr1957 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
2717.    store i8* %tmpAddr1957, i8** %tmp1955
2718.    %tmpWithVal1958 = load i8*, i8** %tmp1955
2719.    store i8 120, i8* %tmpWithVal1958
2720.    %tmp1959 = alloca i8*
2721.    %tmpAddr1961 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
2722.    store i8* %tmpAddr1961, i8** %tmp1959
2723.    %tmpWithVal1962 = load i8*, i8** %tmp1959
2724.    store i8 116, i8* %tmpWithVal1962
2725.    %tmp1963 = alloca i8*

```

```

2726.      %tmpAddr1965 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
2727.      store i8* %tmpAddr1965, i8** %tmp1963
2728.      %tmpWithVal1966 = load i8*, i8** %tmp1963
2729.      store i8 58, i8* %tmpWithVal1966
2730.      %tmp1967 = alloca i8*
2731.      %tmpAddr1969 = tail call i8* @malloc(i32 ptrtoint (i8* getelementptr (i8,
      i8* null, i32 1) to i32))
2732.      store i8* %tmpAddr1969, i8** %tmp1967
2733.      %tmpWithVal1970 = load i8*, i8** %tmp1967
2734.      store i8 32, i8* %tmpWithVal1970
2735.      %push_back1971 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1922)
2736.      %push_back1972 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1926)
2737.      %push_back1973 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1930)
2738.      %push_back1974 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1934)
2739.      %push_back1975 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1938)
2740.      %push_back1976 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1942)
2741.      %push_back1977 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1946)
2742.      %push_back1978 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1950)
2743.      %push_back1979 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1954)
2744.      %push_back1980 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1958)
2745.      %push_back1981 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1962)
2746.      %push_back1982 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1966)
2747.      %push_back1983 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr1917, i8* %tmpWithVal1970)
2748.      %print_list_char = call i32 @print_list_char(%list* %listAddr1917)
2749.      store i32 0, i32* %final_output_counter
2750.      br label %while
2751.
2752.      while:                                ; preds = %while_body, %entry
2753.      %final_output_counter1991 = load i32, i32* %final_output_counter
2754.      %length_input1992 = load i32, i32* %length_input

```

```

2755.      %tmp1993 = icmp slt i32 %final_output_counter1991,
          %length_input1992
2756.      br i1 %tmp1993, label %while_body, label %merge
2757.
2758.      while_body:                                ; preds = %while
2759.      %temp1984 = load %list*, %list** %temp
2760.      %final_output_counter1985 = load i32, i32* %final_output_counter
2761.      %get_at1986 = call i8* (%list*, i32, ...) @get_at(%list* %temp1984, i32
          %final_output_counter1985)
2762.      %eltAsPtr1987 = bitcast i8* %get_at1986 to i32*
2763.      %eltDeref1988 = load i32, i32* %eltAsPtr1987
2764.      %corresponding_char = call i32 @corresponding_char(i32
          %eltDeref1988)
2765.      %final_output_counter1989 = load i32, i32* %final_output_counter
2766.      %tmp1990 = add i32 %final_output_counter1989, 1
2767.      store i32 %tmp1990, i32* %final_output_counter
2768.      br label %while
2769.
2770.      merge:                                       ; preds = %while
2771.      ret i32 0
2772.      }
2773.
2774.      define %list* @plugboard_redirect(%list* %text_redirect, i32
          %count_for_input, %list* %first_plugs, %list* %second_plugs, i32
          %message_length_plug) {
2775.      entry:
2776.      %text_redirect1 = alloca %list*
2777.      store %list* %text_redirect, %list** %text_redirect1
2778.      %count_for_input2 = alloca i32
2779.      store i32 %count_for_input, i32* %count_for_input2
2780.      %first_plugs3 = alloca %list*
2781.      store %list* %first_plugs, %list** %first_plugs3
2782.      %second_plugs4 = alloca %list*
2783.      store %list* %second_plugs, %list** %second_plugs4
2784.      %message_length_plug5 = alloca i32
2785.      store i32 %message_length_plug, i32* %message_length_plug5
2786.      %list_of_char_redirect = alloca %list*
2787.      %sub_list_char = alloca %list*
2788.      %temp_sub_val_holder = alloca %list*
2789.      %loc_holder = alloca %list*
2790.      %loc_holding_for_text = alloca i32
2791.      %counter = alloca i32
2792.      %string_redirect_length = alloca i32

```

```

2793.      %malloccall = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
2794.      %listAddr = bitcast i8* %malloccall to %list*
2795.      %init_list = call i32 (%list*, ...) @init_list(%list* %listAddr)
2796.      %tmp = alloca i32*
2797.      %malloccall6 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
          i32* null, i32 1) to i32))
2798.      %tmpAddr = bitcast i8* %malloccall6 to i32*
2799.      store i32* %tmpAddr, i32** %tmp
2800.      %tmpWithVal = load i32*, i32** %tmp
2801.      store i32 0, i32* %tmpWithVal
2802.      %asChar = bitcast i32* %tmpWithVal to i8*
2803.      %tmp7 = alloca i32*
2804.      %malloccall8 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
          i32* null, i32 1) to i32))
2805.      %tmpAddr9 = bitcast i8* %malloccall8 to i32*
2806.      store i32* %tmpAddr9, i32** %tmp7
2807.      %tmpWithVal10 = load i32*, i32** %tmp7
2808.      store i32 1, i32* %tmpWithVal10
2809.      %asChar11 = bitcast i32* %tmpWithVal10 to i8*
2810.      %tmp12 = alloca i32*
2811.      %malloccall13 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
          i32* null, i32 1) to i32))
2812.      %tmpAddr14 = bitcast i8* %malloccall13 to i32*
2813.      store i32* %tmpAddr14, i32** %tmp12
2814.      %tmpWithVal15 = load i32*, i32** %tmp12
2815.      store i32 2, i32* %tmpWithVal15
2816.      %asChar16 = bitcast i32* %tmpWithVal15 to i8*
2817.      %tmp17 = alloca i32*
2818.      %malloccall18 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
          i32* null, i32 1) to i32))
2819.      %tmpAddr19 = bitcast i8* %malloccall18 to i32*
2820.      store i32* %tmpAddr19, i32** %tmp17
2821.      %tmpWithVal20 = load i32*, i32** %tmp17
2822.      store i32 3, i32* %tmpWithVal20
2823.      %asChar21 = bitcast i32* %tmpWithVal20 to i8*
2824.      %tmp22 = alloca i32*
2825.      %malloccall23 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
          i32* null, i32 1) to i32))
2826.      %tmpAddr24 = bitcast i8* %malloccall23 to i32*
2827.      store i32* %tmpAddr24, i32** %tmp22
2828.      %tmpWithVal25 = load i32*, i32** %tmp22
2829.      store i32 4, i32* %tmpWithVal25
2830.      %asChar26 = bitcast i32* %tmpWithVal25 to i8*

```



```

2831.    %tmp27 = alloca i32*
2832.    %alloca128 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2833.    %tmpAddr29 = bitcast i8* %alloca128 to i32*
2834.    store i32* %tmpAddr29, i32** %tmp27
2835.    %tmpWithVal30 = load i32*, i32** %tmp27
2836.    store i32 5, i32* %tmpWithVal30
2837.    %asChar31 = bitcast i32* %tmpWithVal30 to i8*
2838.    %tmp32 = alloca i32*
2839.    %alloca133 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2840.    %tmpAddr34 = bitcast i8* %alloca133 to i32*
2841.    store i32* %tmpAddr34, i32** %tmp32
2842.    %tmpWithVal35 = load i32*, i32** %tmp32
2843.    store i32 6, i32* %tmpWithVal35
2844.    %asChar36 = bitcast i32* %tmpWithVal35 to i8*
2845.    %tmp37 = alloca i32*
2846.    %alloca138 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2847.    %tmpAddr39 = bitcast i8* %alloca138 to i32*
2848.    store i32* %tmpAddr39, i32** %tmp37
2849.    %tmpWithVal40 = load i32*, i32** %tmp37
2850.    store i32 7, i32* %tmpWithVal40
2851.    %asChar41 = bitcast i32* %tmpWithVal40 to i8*
2852.    %tmp42 = alloca i32*
2853.    %alloca143 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2854.    %tmpAddr44 = bitcast i8* %alloca143 to i32*
2855.    store i32* %tmpAddr44, i32** %tmp42
2856.    %tmpWithVal45 = load i32*, i32** %tmp42
2857.    store i32 8, i32* %tmpWithVal45
2858.    %asChar46 = bitcast i32* %tmpWithVal45 to i8*
2859.    %tmp47 = alloca i32*
2860.    %alloca148 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2861.    %tmpAddr49 = bitcast i8* %alloca148 to i32*
2862.    store i32* %tmpAddr49, i32** %tmp47
2863.    %tmpWithVal50 = load i32*, i32** %tmp47
2864.    store i32 9, i32* %tmpWithVal50
2865.    %asChar51 = bitcast i32* %tmpWithVal50 to i8*
2866.    %tmp52 = alloca i32*
2867.    %alloca153 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2868.    %tmpAddr54 = bitcast i8* %alloca153 to i32*

```

```

2869.    store i32* %tmpAddr54, i32** %tmp52
2870.    %tmpWithVal55 = load i32*, i32** %tmp52
2871.    store i32 10, i32* %tmpWithVal55
2872.    %asChar56 = bitcast i32* %tmpWithVal55 to i8*
2873.    %tmp57 = alloca i32*
2874.    %malloccall58 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2875.    %tmpAddr59 = bitcast i8* %malloccall58 to i32*
2876.    store i32* %tmpAddr59, i32** %tmp57
2877.    %tmpWithVal60 = load i32*, i32** %tmp57
2878.    store i32 11, i32* %tmpWithVal60
2879.    %asChar61 = bitcast i32* %tmpWithVal60 to i8*
2880.    %tmp62 = alloca i32*
2881.    %malloccall63 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2882.    %tmpAddr64 = bitcast i8* %malloccall63 to i32*
2883.    store i32* %tmpAddr64, i32** %tmp62
2884.    %tmpWithVal65 = load i32*, i32** %tmp62
2885.    store i32 12, i32* %tmpWithVal65
2886.    %asChar66 = bitcast i32* %tmpWithVal65 to i8*
2887.    %tmp67 = alloca i32*
2888.    %malloccall68 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2889.    %tmpAddr69 = bitcast i8* %malloccall68 to i32*
2890.    store i32* %tmpAddr69, i32** %tmp67
2891.    %tmpWithVal70 = load i32*, i32** %tmp67
2892.    store i32 13, i32* %tmpWithVal70
2893.    %asChar71 = bitcast i32* %tmpWithVal70 to i8*
2894.    %tmp72 = alloca i32*
2895.    %malloccall73 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2896.    %tmpAddr74 = bitcast i8* %malloccall73 to i32*
2897.    store i32* %tmpAddr74, i32** %tmp72
2898.    %tmpWithVal75 = load i32*, i32** %tmp72
2899.    store i32 14, i32* %tmpWithVal75
2900.    %asChar76 = bitcast i32* %tmpWithVal75 to i8*
2901.    %tmp77 = alloca i32*
2902.    %malloccall78 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2903.    %tmpAddr79 = bitcast i8* %malloccall78 to i32*
2904.    store i32* %tmpAddr79, i32** %tmp77
2905.    %tmpWithVal80 = load i32*, i32** %tmp77
2906.    store i32 15, i32* %tmpWithVal80
2907.    %asChar81 = bitcast i32* %tmpWithVal80 to i8*

```

```

2908.    %tmp82 = alloca i32*
2909.    %allocaCall83 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2910.    %tmpAddr84 = bitcast i8* %allocaCall83 to i32*
2911.    store i32* %tmpAddr84, i32** %tmp82
2912.    %tmpWithVal85 = load i32*, i32** %tmp82
2913.    store i32 16, i32* %tmpWithVal85
2914.    %asChar86 = bitcast i32* %tmpWithVal85 to i8*
2915.    %tmp87 = alloca i32*
2916.    %allocaCall88 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2917.    %tmpAddr89 = bitcast i8* %allocaCall88 to i32*
2918.    store i32* %tmpAddr89, i32** %tmp87
2919.    %tmpWithVal90 = load i32*, i32** %tmp87
2920.    store i32 17, i32* %tmpWithVal90
2921.    %asChar91 = bitcast i32* %tmpWithVal90 to i8*
2922.    %tmp92 = alloca i32*
2923.    %allocaCall93 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2924.    %tmpAddr94 = bitcast i8* %allocaCall93 to i32*
2925.    store i32* %tmpAddr94, i32** %tmp92
2926.    %tmpWithVal95 = load i32*, i32** %tmp92
2927.    store i32 18, i32* %tmpWithVal95
2928.    %asChar96 = bitcast i32* %tmpWithVal95 to i8*
2929.    %tmp97 = alloca i32*
2930.    %allocaCall98 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
2931.    %tmpAddr99 = bitcast i8* %allocaCall98 to i32*
2932.    store i32* %tmpAddr99, i32** %tmp97
2933.    %tmpWithVal100 = load i32*, i32** %tmp97
2934.    store i32 19, i32* %tmpWithVal100
2935.    %asChar101 = bitcast i32* %tmpWithVal100 to i8*
2936.    %tmp102 = alloca i32*
2937.    %allocaCall103 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
2938.    %tmpAddr104 = bitcast i8* %allocaCall103 to i32*
2939.    store i32* %tmpAddr104, i32** %tmp102
2940.    %tmpWithVal105 = load i32*, i32** %tmp102
2941.    store i32 20, i32* %tmpWithVal105
2942.    %asChar106 = bitcast i32* %tmpWithVal105 to i8*
2943.    %tmp107 = alloca i32*
2944.    %allocaCall108 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
2945.    %tmpAddr109 = bitcast i8* %allocaCall108 to i32*

```

```

2946.    store i32* %tmpAddr109, i32** %tmp107
2947.    %tmpWithVal110 = load i32*, i32** %tmp107
2948.    store i32 21, i32* %tmpWithVal110
2949.    %asChar111 = bitcast i32* %tmpWithVal110 to i8*
2950.    %tmp112 = alloca i32*
2951.    %alloca113 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2952.    %tmpAddr114 = bitcast i8* %alloca113 to i32*
2953.    store i32* %tmpAddr114, i32** %tmp112
2954.    %tmpWithVal115 = load i32*, i32** %tmp112
2955.    store i32 22, i32* %tmpWithVal115
2956.    %asChar116 = bitcast i32* %tmpWithVal115 to i8*
2957.    %tmp117 = alloca i32*
2958.    %alloca118 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2959.    %tmpAddr119 = bitcast i8* %alloca118 to i32*
2960.    store i32* %tmpAddr119, i32** %tmp117
2961.    %tmpWithVal120 = load i32*, i32** %tmp117
2962.    store i32 23, i32* %tmpWithVal120
2963.    %asChar121 = bitcast i32* %tmpWithVal120 to i8*
2964.    %tmp122 = alloca i32*
2965.    %alloca123 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2966.    %tmpAddr124 = bitcast i8* %alloca123 to i32*
2967.    store i32* %tmpAddr124, i32** %tmp122
2968.    %tmpWithVal125 = load i32*, i32** %tmp122
2969.    store i32 24, i32* %tmpWithVal125
2970.    %asChar126 = bitcast i32* %tmpWithVal125 to i8*
2971.    %tmp127 = alloca i32*
2972.    %alloca128 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
2973.    %tmpAddr129 = bitcast i8* %alloca128 to i32*
2974.    store i32* %tmpAddr129, i32** %tmp127
2975.    %tmpWithVal130 = load i32*, i32** %tmp127
2976.    store i32 25, i32* %tmpWithVal130
2977.    %asChar131 = bitcast i32* %tmpWithVal130 to i8*
2978.    %push_back = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr, i8*
      %asChar)
2979.    %push_back132 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar11)
2980.    %push_back133 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar16)
2981.    %push_back134 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar21)

```

2982. %push_back135 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar26)

2983. %push_back136 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar31)

2984. %push_back137 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar36)

2985. %push_back138 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar41)

2986. %push_back139 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar46)

2987. %push_back140 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar51)

2988. %push_back141 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar56)

2989. %push_back142 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar61)

2990. %push_back143 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar66)

2991. %push_back144 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar71)

2992. %push_back145 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar76)

2993. %push_back146 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar81)

2994. %push_back147 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar86)

2995. %push_back148 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar91)

2996. %push_back149 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar96)

2997. %push_back150 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar101)

2998. %push_back151 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar106)

2999. %push_back152 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar111)

3000. %push_back153 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar116)

3001. %push_back154 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar121)

3002. %push_back155 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar126)

3003. %push_back156 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar131)

```

3004.      store %list* %listAddr, %list** %list_of_char_redirect
3005.      %alloca157 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
      (%list, %list* null, i32 1) to i32))
3006.      %listAddr158 = bitcast i8* %alloca157 to i32*
3007.      %init_list159 = call i32 (@init_list(%list* %listAddr158))
3008.      %tmp160 = alloca i32*
3009.      %alloca161 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3010.      %tmpAddr162 = bitcast i8* %alloca161 to i32*
3011.      store i32* %tmpAddr162, i32** %tmp160
3012.      %tmpWithVal163 = load i32*, i32** %tmp160
3013.      store i32 0, i32* %tmpWithVal163
3014.      %asChar164 = bitcast i32* %tmpWithVal163 to i8*
3015.      %tmp165 = alloca i32*
3016.      %alloca166 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3017.      %tmpAddr167 = bitcast i8* %alloca166 to i32*
3018.      store i32* %tmpAddr167, i32** %tmp165
3019.      %tmpWithVal168 = load i32*, i32** %tmp165
3020.      store i32 1, i32* %tmpWithVal168
3021.      %asChar169 = bitcast i32* %tmpWithVal168 to i8*
3022.      %tmp170 = alloca i32*
3023.      %alloca171 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3024.      %tmpAddr172 = bitcast i8* %alloca171 to i32*
3025.      store i32* %tmpAddr172, i32** %tmp170
3026.      %tmpWithVal173 = load i32*, i32** %tmp170
3027.      store i32 2, i32* %tmpWithVal173
3028.      %asChar174 = bitcast i32* %tmpWithVal173 to i8*
3029.      %tmp175 = alloca i32*
3030.      %alloca176 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3031.      %tmpAddr177 = bitcast i8* %alloca176 to i32*
3032.      store i32* %tmpAddr177, i32** %tmp175
3033.      %tmpWithVal178 = load i32*, i32** %tmp175
3034.      store i32 3, i32* %tmpWithVal178
3035.      %asChar179 = bitcast i32* %tmpWithVal178 to i8*
3036.      %tmp180 = alloca i32*
3037.      %alloca181 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3038.      %tmpAddr182 = bitcast i8* %alloca181 to i32*
3039.      store i32* %tmpAddr182, i32** %tmp180
3040.      %tmpWithVal183 = load i32*, i32** %tmp180
3041.      store i32 4, i32* %tmpWithVal183

```

```

3042.    %asChar184 = bitcast i32* %tmpWithVal183 to i8*
3043.    %tmp185 = alloca i32*
3044.    %allocaCall186 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3045.    %tmpAddr187 = bitcast i8* %allocaCall186 to i32*
3046.    store i32* %tmpAddr187, i32** %tmp185
3047.    %tmpWithVal188 = load i32*, i32** %tmp185
3048.    store i32 5, i32* %tmpWithVal188
3049.    %asChar189 = bitcast i32* %tmpWithVal188 to i8*
3050.    %tmp190 = alloca i32*
3051.    %allocaCall191 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3052.    %tmpAddr192 = bitcast i8* %allocaCall191 to i32*
3053.    store i32* %tmpAddr192, i32** %tmp190
3054.    %tmpWithVal193 = load i32*, i32** %tmp190
3055.    store i32 6, i32* %tmpWithVal193
3056.    %asChar194 = bitcast i32* %tmpWithVal193 to i8*
3057.    %tmp195 = alloca i32*
3058.    %allocaCall196 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3059.    %tmpAddr197 = bitcast i8* %allocaCall196 to i32*
3060.    store i32* %tmpAddr197, i32** %tmp195
3061.    %tmpWithVal198 = load i32*, i32** %tmp195
3062.    store i32 7, i32* %tmpWithVal198
3063.    %asChar199 = bitcast i32* %tmpWithVal198 to i8*
3064.    %tmp200 = alloca i32*
3065.    %allocaCall201 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3066.    %tmpAddr202 = bitcast i8* %allocaCall201 to i32*
3067.    store i32* %tmpAddr202, i32** %tmp200
3068.    %tmpWithVal203 = load i32*, i32** %tmp200
3069.    store i32 8, i32* %tmpWithVal203
3070.    %asChar204 = bitcast i32* %tmpWithVal203 to i8*
3071.    %tmp205 = alloca i32*
3072.    %allocaCall206 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3073.    %tmpAddr207 = bitcast i8* %allocaCall206 to i32*
3074.    store i32* %tmpAddr207, i32** %tmp205
3075.    %tmpWithVal208 = load i32*, i32** %tmp205
3076.    store i32 9, i32* %tmpWithVal208
3077.    %asChar209 = bitcast i32* %tmpWithVal208 to i8*
3078.    %tmp210 = alloca i32*
3079.    %allocaCall211 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))

```

```

3080.    %tmpAddr212 = bitcast i8* %alloca211 to i32*
3081.    store i32* %tmpAddr212, i32** %tmp210
3082.    %tmpWithVal213 = load i32*, i32** %tmp210
3083.    store i32 10, i32* %tmpWithVal213
3084.    %asChar214 = bitcast i32* %tmpWithVal213 to i8*
3085.    %tmp215 = alloca i32*
3086.    %alloca216 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3087.    %tmpAddr217 = bitcast i8* %alloca216 to i32*
3088.    store i32* %tmpAddr217, i32** %tmp215
3089.    %tmpWithVal218 = load i32*, i32** %tmp215
3090.    store i32 11, i32* %tmpWithVal218
3091.    %asChar219 = bitcast i32* %tmpWithVal218 to i8*
3092.    %tmp220 = alloca i32*
3093.    %alloca221 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3094.    %tmpAddr222 = bitcast i8* %alloca221 to i32*
3095.    store i32* %tmpAddr222, i32** %tmp220
3096.    %tmpWithVal223 = load i32*, i32** %tmp220
3097.    store i32 12, i32* %tmpWithVal223
3098.    %asChar224 = bitcast i32* %tmpWithVal223 to i8*
3099.    %tmp225 = alloca i32*
3100.    %alloca226 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3101.    %tmpAddr227 = bitcast i8* %alloca226 to i32*
3102.    store i32* %tmpAddr227, i32** %tmp225
3103.    %tmpWithVal228 = load i32*, i32** %tmp225
3104.    store i32 13, i32* %tmpWithVal228
3105.    %asChar229 = bitcast i32* %tmpWithVal228 to i8*
3106.    %tmp230 = alloca i32*
3107.    %alloca231 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3108.    %tmpAddr232 = bitcast i8* %alloca231 to i32*
3109.    store i32* %tmpAddr232, i32** %tmp230
3110.    %tmpWithVal233 = load i32*, i32** %tmp230
3111.    store i32 14, i32* %tmpWithVal233
3112.    %asChar234 = bitcast i32* %tmpWithVal233 to i8*
3113.    %tmp235 = alloca i32*
3114.    %alloca236 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3115.    %tmpAddr237 = bitcast i8* %alloca236 to i32*
3116.    store i32* %tmpAddr237, i32** %tmp235
3117.    %tmpWithVal238 = load i32*, i32** %tmp235
3118.    store i32 15, i32* %tmpWithVal238

```



```

3119.    %asChar239 = bitcast i32* %tmpWithVal238 to i8*
3120.    %tmp240 = alloca i32*
3121.    %malloccall241 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3122.    %tmpAddr242 = bitcast i8* %malloccall241 to i32*
3123.    store i32* %tmpAddr242, i32** %tmp240
3124.    %tmpWithVal243 = load i32*, i32** %tmp240
3125.    store i32 16, i32* %tmpWithVal243
3126.    %asChar244 = bitcast i32* %tmpWithVal243 to i8*
3127.    %tmp245 = alloca i32*
3128.    %malloccall246 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3129.    %tmpAddr247 = bitcast i8* %malloccall246 to i32*
3130.    store i32* %tmpAddr247, i32** %tmp245
3131.    %tmpWithVal248 = load i32*, i32** %tmp245
3132.    store i32 17, i32* %tmpWithVal248
3133.    %asChar249 = bitcast i32* %tmpWithVal248 to i8*
3134.    %tmp250 = alloca i32*
3135.    %malloccall251 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3136.    %tmpAddr252 = bitcast i8* %malloccall251 to i32*
3137.    store i32* %tmpAddr252, i32** %tmp250
3138.    %tmpWithVal253 = load i32*, i32** %tmp250
3139.    store i32 18, i32* %tmpWithVal253
3140.    %asChar254 = bitcast i32* %tmpWithVal253 to i8*
3141.    %tmp255 = alloca i32*
3142.    %malloccall256 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3143.    %tmpAddr257 = bitcast i8* %malloccall256 to i32*
3144.    store i32* %tmpAddr257, i32** %tmp255
3145.    %tmpWithVal258 = load i32*, i32** %tmp255
3146.    store i32 19, i32* %tmpWithVal258
3147.    %asChar259 = bitcast i32* %tmpWithVal258 to i8*
3148.    %tmp260 = alloca i32*
3149.    %malloccall261 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3150.    %tmpAddr262 = bitcast i8* %malloccall261 to i32*
3151.    store i32* %tmpAddr262, i32** %tmp260
3152.    %tmpWithVal263 = load i32*, i32** %tmp260
3153.    store i32 20, i32* %tmpWithVal263
3154.    %asChar264 = bitcast i32* %tmpWithVal263 to i8*
3155.    %tmp265 = alloca i32*
3156.    %malloccall266 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))

```

```

3157.    %tmpAddr267 = bitcast i8* %alloca266 to i32*
3158.    store i32* %tmpAddr267, i32** %tmp265
3159.    %tmpWithVal268 = load i32*, i32** %tmp265
3160.    store i32 21, i32* %tmpWithVal268
3161.    %asChar269 = bitcast i32* %tmpWithVal268 to i8*
3162.    %tmp270 = alloca i32*
3163.    %alloca271 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3164.    %tmpAddr272 = bitcast i8* %alloca271 to i32*
3165.    store i32* %tmpAddr272, i32** %tmp270
3166.    %tmpWithVal273 = load i32*, i32** %tmp270
3167.    store i32 22, i32* %tmpWithVal273
3168.    %asChar274 = bitcast i32* %tmpWithVal273 to i8*
3169.    %tmp275 = alloca i32*
3170.    %alloca276 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3171.    %tmpAddr277 = bitcast i8* %alloca276 to i32*
3172.    store i32* %tmpAddr277, i32** %tmp275
3173.    %tmpWithVal278 = load i32*, i32** %tmp275
3174.    store i32 23, i32* %tmpWithVal278
3175.    %asChar279 = bitcast i32* %tmpWithVal278 to i8*
3176.    %tmp280 = alloca i32*
3177.    %alloca281 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3178.    %tmpAddr282 = bitcast i8* %alloca281 to i32*
3179.    store i32* %tmpAddr282, i32** %tmp280
3180.    %tmpWithVal283 = load i32*, i32** %tmp280
3181.    store i32 24, i32* %tmpWithVal283
3182.    %asChar284 = bitcast i32* %tmpWithVal283 to i8*
3183.    %tmp285 = alloca i32*
3184.    %alloca286 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3185.    %tmpAddr287 = bitcast i8* %alloca286 to i32*
3186.    store i32* %tmpAddr287, i32** %tmp285
3187.    %tmpWithVal288 = load i32*, i32** %tmp285
3188.    store i32 25, i32* %tmpWithVal288
3189.    %asChar289 = bitcast i32* %tmpWithVal288 to i8*
3190.    %push_back290 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr158, i8* %asChar164)
3191.    %push_back291 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr158, i8* %asChar169)
3192.    %push_back292 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr158, i8* %asChar174)

```

3193. %push_back293 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar179)

3194. %push_back294 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar184)

3195. %push_back295 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar189)

3196. %push_back296 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar194)

3197. %push_back297 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar199)

3198. %push_back298 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar204)

3199. %push_back299 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar209)

3200. %push_back300 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar214)

3201. %push_back301 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar219)

3202. %push_back302 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar224)

3203. %push_back303 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar229)

3204. %push_back304 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar234)

3205. %push_back305 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar239)

3206. %push_back306 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar244)

3207. %push_back307 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar249)

3208. %push_back308 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar254)

3209. %push_back309 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar259)

3210. %push_back310 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar264)

3211. %push_back311 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar269)

3212. %push_back312 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar274)

3213. %push_back313 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar279)

3214. %push_back314 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr158, i8* %asChar284)

```

3215.      %push_back315 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr158, i8* %asChar289)
3216.      store %list* %listAddr158, %list** %sub_list_char
3217.      %mallocall316 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
3218.      %listAddr317 = bitcast i8* %mallocall316 to %list*
3219.      %init_list318 = call i32 (%list*, ...) @init_list(%list* %listAddr317)
3220.      %tmp319 = alloca i32*
3221.      %mallocall320 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3222.      %tmpAddr321 = bitcast i8* %mallocall320 to i32*
3223.      store i32* %tmpAddr321, i32** %tmp319
3224.      %tmpWithVal322 = load i32*, i32** %tmp319
3225.      store i32 1, i32* %tmpWithVal322
3226.      %asChar323 = bitcast i32* %tmpWithVal322 to i8*
3227.      %tmp324 = alloca i32*
3228.      %mallocall325 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3229.      %tmpAddr326 = bitcast i8* %mallocall325 to i32*
3230.      store i32* %tmpAddr326, i32** %tmp324
3231.      %tmpWithVal327 = load i32*, i32** %tmp324
3232.      store i32 1, i32* %tmpWithVal327
3233.      %asChar328 = bitcast i32* %tmpWithVal327 to i8*
3234.      %push_back329 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr317, i8* %asChar323)
3235.      %push_back330 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr317, i8* %asChar328)
3236.      store %list* %listAddr317, %list** %temp_sub_val_holder
3237.      %mallocall331 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
3238.      %listAddr332 = bitcast i8* %mallocall331 to %list*
3239.      %init_list333 = call i32 (%list*, ...) @init_list(%list* %listAddr332)
3240.      %tmp334 = alloca i32*
3241.      %mallocall335 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3242.      %tmpAddr336 = bitcast i8* %mallocall335 to i32*
3243.      store i32* %tmpAddr336, i32** %tmp334
3244.      %tmpWithVal337 = load i32*, i32** %tmp334
3245.      store i32 1, i32* %tmpWithVal337
3246.      %asChar338 = bitcast i32* %tmpWithVal337 to i8*
3247.      %tmp339 = alloca i32*
3248.      %mallocall340 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3249.      %tmpAddr341 = bitcast i8* %mallocall340 to i32*

```

```

3250.     store i32* %tmpAddr341, i32** %tmp339
3251.     %tmpWithVal342 = load i32*, i32** %tmp339
3252.     store i32 1, i32* %tmpWithVal342
3253.     %asChar343 = bitcast i32* %tmpWithVal342 to i8*
3254.     %push_back344 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr332, i8* %asChar338)
3255.     %push_back345 = call i32 (%list*, i8*, ...) @push_back(%list*
        %listAddr332, i8* %asChar343)
3256.     store %list* %listAddr332, %list** %loc_holder
3257.     store i32 0, i32* %loc_holding_for_text
3258.     store i32 0, i32* %counter
3259.     store i32 0, i32* %string_redirect_length
3260.     br label %if
3261.
3262.     if:                                ; preds = %entry
3263.     %count_for_input442 = load i32, i32* %count_for_input2
3264.     %tmp443 = icmp eq i32 %count_for_input442, 0
3265.     br i1 %tmp443, label %if_body, label %merge
3266.
3267.     if_body:                            ; preds = %if
3268.     br label %while
3269.
3270.     merge:                              ; preds = %if, %merge441
3271.     store i32 0, i32* %string_redirect_length
3272.     br label %while444
3273.
3274.     while:                              ; preds = %while_body, %if_body
3275.     %counter439 = load i32, i32* %counter
3276.     %tmp440 = icmp ne i32 %counter439, 10
3277.     br i1 %tmp440, label %while_body, label %merge441
3278.
3279.     while_body:                         ; preds = %while
3280.     %temp_sub_val_holder346 = load %list*, %list** %temp_sub_val_holder
3281.     %first_plugs347 = load %list*, %list** %first_plugs3
3282.     %counter348 = load i32, i32* %counter
3283.     %get_at = call i8* (%list*, i32, ...) @get_at(%list* %first_plugs347, i32
        %counter348)
3284.     %eltAsPtr = bitcast i8* %get_at to i32*
3285.     %eltDeref = load i32, i32* %eltAsPtr
3286.     %tmp349 = alloca i32*
3287.     %mallocall350 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
        (i32, i32* null, i32 1) to i32))
3288.     %tmpAddr351 = bitcast i8* %mallocall350 to i32*
3289.     store i32* %tmpAddr351, i32** %tmp349

```

```

3290.    %tmpWithVal352 = load i32*, i32** %tmp349
3291.    store i32 %eltDeref, i32* %tmpWithVal352
3292.    %asChar353 = bitcast i32* %tmpWithVal352 to i8*
3293.    %set_at = call %list* (%list*, i32, i8*, ...) @set_at(%list*
    %temp_sub_val_holder346, i32 0, i8* %asChar353)
3294.    %temp_sub_val_holder354 = load %list*, %list** %temp_sub_val_holder
3295.    %second_plugs355 = load %list*, %list** %second_plugs4
3296.    %counter356 = load i32, i32* %counter
3297.    %get_at357 = call i8* (%list*, i32, ...) @get_at(%list*
    %second_plugs355, i32 %counter356)
3298.    %eltAsPtr358 = bitcast i8* %get_at357 to i32*
3299.    %eltDeref359 = load i32, i32* %eltAsPtr358
3300.    %tmp360 = alloca i32*
3301.    %mallocall361 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3302.    %tmpAddr362 = bitcast i8* %mallocall361 to i32*
3303.    store i32* %tmpAddr362, i32** %tmp360
3304.    %tmpWithVal363 = load i32*, i32** %tmp360
3305.    store i32 %eltDeref359, i32* %tmpWithVal363
3306.    %asChar364 = bitcast i32* %tmpWithVal363 to i8*
3307.    %set_at365 = call %list* (%list*, i32, i8*, ...) @set_at(%list*
    %temp_sub_val_holder354, i32 1, i8* %asChar364)
3308.    %loc_holder366 = load %list*, %list** %loc_holder
3309.    %list_of_char_redirect367 = load %list*, %list** %list_of_char_redirect
3310.    %temp_sub_val_holder368 = load %list*, %list** %temp_sub_val_holder
3311.    %get_at369 = call i8* (%list*, i32, ...) @get_at(%list*
    %temp_sub_val_holder368, i32 0)
3312.    %eltAsPtr370 = bitcast i8* %get_at369 to i32*
3313.    %eltDeref371 = load i32, i32* %eltAsPtr370
3314.    %get_at372 = call i8* (%list*, i32, ...) @get_at(%list*
    %list_of_char_redirect367, i32 %eltDeref371)
3315.    %eltAsPtr373 = bitcast i8* %get_at372 to i32*
3316.    %eltDeref374 = load i32, i32* %eltAsPtr373
3317.    %list_of_char_redirect375 = load %list*, %list** %list_of_char_redirect
3318.    %character_location_result = call i32 @character_location(%list*
    %list_of_char_redirect375, i32 %eltDeref374)
3319.    %tmp376 = alloca i32*
3320.    %mallocall377 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3321.    %tmpAddr378 = bitcast i8* %mallocall377 to i32*
3322.    store i32* %tmpAddr378, i32** %tmp376
3323.    %tmpWithVal379 = load i32*, i32** %tmp376
3324.    store i32 %character_location_result, i32* %tmpWithVal379
3325.    %asChar380 = bitcast i32* %tmpWithVal379 to i8*

```

```

3326.      %set_at381 = call %list* (%list*, i32, i8*, ...) @set_at(%list*
        %loc_holder366, i32 0, i8* %asChar380)
3327.      %loc_holder382 = load %list*, %list** %loc_holder
3328.      %list_of_char_redirect383 = load %list*, %list** %list_of_char_redirect
3329.      %temp_sub_val_holder384 = load %list*, %list** %temp_sub_val_holder
3330.      %get_at385 = call i8* (%list*, i32, ...) @get_at(%list*
        %temp_sub_val_holder384, i32 1)
3331.      %eltAsPtr386 = bitcast i8* %get_at385 to i32*
3332.      %eltDeref387 = load i32, i32* %eltAsPtr386
3333.      %get_at388 = call i8* (%list*, i32, ...) @get_at(%list*
        %list_of_char_redirect383, i32 %eltDeref387)
3334.      %eltAsPtr389 = bitcast i8* %get_at388 to i32*
3335.      %eltDeref390 = load i32, i32* %eltAsPtr389
3336.      %list_of_char_redirect391 = load %list*, %list** %list_of_char_redirect
3337.      %character_location_result392 = call i32 @character_location(%list*
        %list_of_char_redirect391, i32 %eltDeref390)
3338.      %tmp393 = alloca i32*
3339.      %mallocall394 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
        (i32, i32* null, i32 1) to i32))
3340.      %tmpAddr395 = bitcast i8* %mallocall394 to i32*
3341.      store i32* %tmpAddr395, i32** %tmp393
3342.      %tmpWithVal396 = load i32*, i32** %tmp393
3343.      store i32 %character_location_result392, i32* %tmpWithVal396
3344.      %asChar397 = bitcast i32* %tmpWithVal396 to i8*
3345.      %set_at398 = call %list* (%list*, i32, i8*, ...) @set_at(%list*
        %loc_holder382, i32 1, i8* %asChar397)
3346.      %sub_list_char399 = load %list*, %list** %sub_list_char
3347.      %loc_holder400 = load %list*, %list** %loc_holder
3348.      %get_at401 = call i8* (%list*, i32, ...) @get_at(%list* %loc_holder400,
        i32 0)
3349.      %eltAsPtr402 = bitcast i8* %get_at401 to i32*
3350.      %eltDeref403 = load i32, i32* %eltAsPtr402
3351.      %list_of_char_redirect404 = load %list*, %list** %list_of_char_redirect
3352.      %loc_holder405 = load %list*, %list** %loc_holder
3353.      %get_at406 = call i8* (%list*, i32, ...) @get_at(%list* %loc_holder405,
        i32 1)
3354.      %eltAsPtr407 = bitcast i8* %get_at406 to i32*
3355.      %eltDeref408 = load i32, i32* %eltAsPtr407
3356.      %get_at409 = call i8* (%list*, i32, ...) @get_at(%list*
        %list_of_char_redirect404, i32 %eltDeref408)
3357.      %eltAsPtr410 = bitcast i8* %get_at409 to i32*
3358.      %eltDeref411 = load i32, i32* %eltAsPtr410
3359.      %tmp412 = alloca i32*

```

```

3360.      %alloca413 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3361.      %tmpAddr414 = bitcast i8* %alloca413 to i32*
3362.      store i32* %tmpAddr414, i32** %tmp412
3363.      %tmpWithVal415 = load i32*, i32** %tmp412
3364.      store i32 %eltDeref411, i32* %tmpWithVal415
3365.      %asChar416 = bitcast i32* %tmpWithVal415 to i8*
3366.      %set_at417 = call %list* (%list*, i32, i8*, ...) @set_at(%list*
          %sub_list_char399, i32 %eltDeref403, i8* %asChar416)
3367.      %sub_list_char418 = load %list*, %list** %sub_list_char
3368.      %loc_holder419 = load %list*, %list** %loc_holder
3369.      %get_at420 = call i8* (%list*, i32, ...) @get_at(%list* %loc_holder419,
          i32 1)
3370.      %eltAsPtr421 = bitcast i8* %get_at420 to i32*
3371.      %eltDeref422 = load i32, i32* %eltAsPtr421
3372.      %list_of_char_redirect423 = load %list*, %list** %list_of_char_redirect
3373.      %loc_holder424 = load %list*, %list** %loc_holder
3374.      %get_at425 = call i8* (%list*, i32, ...) @get_at(%list* %loc_holder424,
          i32 0)
3375.      %eltAsPtr426 = bitcast i8* %get_at425 to i32*
3376.      %eltDeref427 = load i32, i32* %eltAsPtr426
3377.      %get_at428 = call i8* (%list*, i32, ...) @get_at(%list*
          %list_of_char_redirect423, i32 %eltDeref427)
3378.      %eltAsPtr429 = bitcast i8* %get_at428 to i32*
3379.      %eltDeref430 = load i32, i32* %eltAsPtr429
3380.      %tmp431 = alloca i32*
3381.      %alloca432 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3382.      %tmpAddr433 = bitcast i8* %alloca432 to i32*
3383.      store i32* %tmpAddr433, i32** %tmp431
3384.      %tmpWithVal434 = load i32*, i32** %tmp431
3385.      store i32 %eltDeref430, i32* %tmpWithVal434
3386.      %asChar435 = bitcast i32* %tmpWithVal434 to i8*
3387.      %set_at436 = call %list* (%list*, i32, i8*, ...) @set_at(%list*
          %sub_list_char418, i32 %eltDeref422, i8* %asChar435)
3388.      %counter437 = load i32, i32* %counter
3389.      %tmp438 = add i32 %counter437, 1
3390.      store i32 %tmp438, i32* %counter
3391.      br label %while
3392.
3393. merge441:                                ; preds = %while
3394.      br label %merge
3395.
3396. while444:                                ; preds = %while_body445, %merge

```



```

3397.    %string_redirect_length468 = load i32, i32* %string_redirect_length
3398.    %message_length_plug469 = load i32, i32* %message_length_plug5
3399.    %tmp470 = icmp slt i32 %string_redirect_length468,
        %message_length_plug469
3400.    br i1 %tmp470, label %while_body445, label %merge471
3401.
3402.    while_body445:                                ; preds = %while444
3403.    %text_redirect446 = load %list*, %list** %text_redirect1
3404.    %string_redirect_length447 = load i32, i32* %string_redirect_length
3405.    %get_at448 = call i8* (%list*, i32, ...) @get_at(%list* %text_redirect446,
        i32 %string_redirect_length447)
3406.    %eltAsPtr449 = bitcast i8* %get_at448 to i32*
3407.    %eltDeref450 = load i32, i32* %eltAsPtr449
3408.    %list_of_char_redirect451 = load %list*, %list** %list_of_char_redirect
3409.    %character_location_result452 = call i32 @character_location(%list*
        %list_of_char_redirect451, i32 %eltDeref450)
3410.    store i32 %character_location_result452, i32* %loc_holding_for_text
3411.    %text_redirect453 = load %list*, %list** %text_redirect1
3412.    %string_redirect_length454 = load i32, i32* %string_redirect_length
3413.    %sub_list_char455 = load %list*, %list** %sub_list_char
3414.    %loc_holding_for_text456 = load i32, i32* %loc_holding_for_text
3415.    %get_at457 = call i8* (%list*, i32, ...) @get_at(%list* %sub_list_char455,
        i32 %loc_holding_for_text456)
3416.    %eltAsPtr458 = bitcast i8* %get_at457 to i32*
3417.    %eltDeref459 = load i32, i32* %eltAsPtr458
3418.    %tmp460 = alloca i32*
3419.    %mallocall461 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
        (i32, i32* null, i32 1) to i32))
3420.    %tmpAddr462 = bitcast i8* %mallocall461 to i32*
3421.    store i32* %tmpAddr462, i32** %tmp460
3422.    %tmpWithVal463 = load i32*, i32** %tmp460
3423.    store i32 %eltDeref459, i32* %tmpWithVal463
3424.    %asChar464 = bitcast i32* %tmpWithVal463 to i8*
3425.    %set_at465 = call %list* (%list*, i32, i8*, ...) @set_at(%list*
        %text_redirect453, i32 %string_redirect_length454, i8* %asChar464)
3426.    %string_redirect_length466 = load i32, i32* %string_redirect_length
3427.    %tmp467 = add i32 %string_redirect_length466, 1
3428.    store i32 %tmp467, i32* %string_redirect_length
3429.    br label %while444
3430.
3431.    merge471:                                      ; preds = %while444
3432.    %text_redirect472 = load %list*, %list** %text_redirect1
3433.    ret %list* %text_redirect472
3434.    }

```

```

3435.
3436.     define %list* @encryption_decryption(%list* %rotor_one_e_input, %list*
        %rotor_one_e_output, %list* %rotor_two_e_input, %list* %rotor_two_e_output,
        %list* %rotor_three_e_input, %list* %rotor_three_e_output, %list* %text, i32
        %length_input_text) {
3437.     entry:
3438.         %rotor_one_e_input1 = alloca %list*
3439.         store %list* %rotor_one_e_input, %list** %rotor_one_e_input1
3440.         %rotor_one_e_output2 = alloca %list*
3441.         store %list* %rotor_one_e_output, %list** %rotor_one_e_output2
3442.         %rotor_two_e_input3 = alloca %list*
3443.         store %list* %rotor_two_e_input, %list** %rotor_two_e_input3
3444.         %rotor_two_e_output4 = alloca %list*
3445.         store %list* %rotor_two_e_output, %list** %rotor_two_e_output4
3446.         %rotor_three_e_input5 = alloca %list*
3447.         store %list* %rotor_three_e_input, %list** %rotor_three_e_input5
3448.         %rotor_three_e_output6 = alloca %list*
3449.         store %list* %rotor_three_e_output, %list** %rotor_three_e_output6
3450.         %text7 = alloca %list*
3451.         store %list* %text, %list** %text7
3452.         %length_input_text8 = alloca i32
3453.         store i32 %length_input_text, i32* %length_input_text8
3454.         %modulus_result = alloca i32
3455.         %modulus_result_t = alloca i32
3456.         %char_loc_holder = alloca i32
3457.         %count_rotor_one = alloca i32
3458.         %count_rotor_two = alloca i32
3459.         %count_rotor_three = alloca i32
3460.         %text_len = alloca i32
3461.         %loop_counter_e_d = alloca i32
3462.         %input_output_char = alloca %list*
3463.         %cipher_output_character = alloca %list*
3464.         store i32 0, i32* %modulus_result
3465.         store i32 0, i32* %modulus_result_t
3466.         store i32 0, i32* %char_loc_holder
3467.         store i32 0, i32* %count_rotor_one
3468.         store i32 0, i32* %count_rotor_two
3469.         store i32 0, i32* %count_rotor_three
3470.         %length_input_text9 = load i32, i32* %length_input_text8
3471.         store i32 %length_input_text9, i32* %text_len
3472.         store i32 0, i32* %loop_counter_e_d
3473.         %malloccall = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
            (%list, %list* null, i32 1) to i32))
3474.         %listAddr = bitcast i8* %malloccall to %list*

```

```

3475.    %init_list = call i32 (%list*, ...) @init_list(%list* %listAddr)
3476.    %tmp = alloca i32*
3477.    %alloca10 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3478.    %tmpAddr = bitcast i8* %alloca10 to i32*
3479.    store i32* %tmpAddr, i32** %tmp
3480.    %tmpWithVal = load i32*, i32** %tmp
3481.    store i32 0, i32* %tmpWithVal
3482.    %asChar = bitcast i32* %tmpWithVal to i8*
3483.    %tmp11 = alloca i32*
3484.    %alloca12 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3485.    %tmpAddr13 = bitcast i8* %alloca12 to i32*
3486.    store i32* %tmpAddr13, i32** %tmp11
3487.    %tmpWithVal14 = load i32*, i32** %tmp11
3488.    store i32 1, i32* %tmpWithVal14
3489.    %asChar15 = bitcast i32* %tmpWithVal14 to i8*
3490.    %tmp16 = alloca i32*
3491.    %alloca17 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3492.    %tmpAddr18 = bitcast i8* %alloca17 to i32*
3493.    store i32* %tmpAddr18, i32** %tmp16
3494.    %tmpWithVal19 = load i32*, i32** %tmp16
3495.    store i32 2, i32* %tmpWithVal19
3496.    %asChar20 = bitcast i32* %tmpWithVal19 to i8*
3497.    %tmp21 = alloca i32*
3498.    %alloca22 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3499.    %tmpAddr23 = bitcast i8* %alloca22 to i32*
3500.    store i32* %tmpAddr23, i32** %tmp21
3501.    %tmpWithVal24 = load i32*, i32** %tmp21
3502.    store i32 3, i32* %tmpWithVal24
3503.    %asChar25 = bitcast i32* %tmpWithVal24 to i8*
3504.    %tmp26 = alloca i32*
3505.    %alloca27 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3506.    %tmpAddr28 = bitcast i8* %alloca27 to i32*
3507.    store i32* %tmpAddr28, i32** %tmp26
3508.    %tmpWithVal29 = load i32*, i32** %tmp26
3509.    store i32 4, i32* %tmpWithVal29
3510.    %asChar30 = bitcast i32* %tmpWithVal29 to i8*
3511.    %tmp31 = alloca i32*
3512.    %alloca32 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))

```

```

3513.    %tmpAddr33 = bitcast i8* %alloca32 to i32*
3514.    store i32* %tmpAddr33, i32** %tmp31
3515.    %tmpWithVal34 = load i32*, i32** %tmp31
3516.    store i32 5, i32* %tmpWithVal34
3517.    %asChar35 = bitcast i32* %tmpWithVal34 to i8*
3518.    %tmp36 = alloca i32*
3519.    %alloca37 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3520.    %tmpAddr38 = bitcast i8* %alloca37 to i32*
3521.    store i32* %tmpAddr38, i32** %tmp36
3522.    %tmpWithVal39 = load i32*, i32** %tmp36
3523.    store i32 6, i32* %tmpWithVal39
3524.    %asChar40 = bitcast i32* %tmpWithVal39 to i8*
3525.    %tmp41 = alloca i32*
3526.    %alloca42 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3527.    %tmpAddr43 = bitcast i8* %alloca42 to i32*
3528.    store i32* %tmpAddr43, i32** %tmp41
3529.    %tmpWithVal44 = load i32*, i32** %tmp41
3530.    store i32 7, i32* %tmpWithVal44
3531.    %asChar45 = bitcast i32* %tmpWithVal44 to i8*
3532.    %tmp46 = alloca i32*
3533.    %alloca47 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3534.    %tmpAddr48 = bitcast i8* %alloca47 to i32*
3535.    store i32* %tmpAddr48, i32** %tmp46
3536.    %tmpWithVal49 = load i32*, i32** %tmp46
3537.    store i32 8, i32* %tmpWithVal49
3538.    %asChar50 = bitcast i32* %tmpWithVal49 to i8*
3539.    %tmp51 = alloca i32*
3540.    %alloca52 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3541.    %tmpAddr53 = bitcast i8* %alloca52 to i32*
3542.    store i32* %tmpAddr53, i32** %tmp51
3543.    %tmpWithVal54 = load i32*, i32** %tmp51
3544.    store i32 9, i32* %tmpWithVal54
3545.    %asChar55 = bitcast i32* %tmpWithVal54 to i8*
3546.    %tmp56 = alloca i32*
3547.    %alloca57 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3548.    %tmpAddr58 = bitcast i8* %alloca57 to i32*
3549.    store i32* %tmpAddr58, i32** %tmp56
3550.    %tmpWithVal59 = load i32*, i32** %tmp56
3551.    store i32 10, i32* %tmpWithVal59

```

```

3552.    %asChar60 = bitcast i32* %tmpWithVal59 to i8*
3553.    %tmp61 = alloca i32*
3554.    %allocaCall62 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3555.    %tmpAddr63 = bitcast i8* %allocaCall62 to i32*
3556.    store i32* %tmpAddr63, i32** %tmp61
3557.    %tmpWithVal64 = load i32*, i32** %tmp61
3558.    store i32 11, i32* %tmpWithVal64
3559.    %asChar65 = bitcast i32* %tmpWithVal64 to i8*
3560.    %tmp66 = alloca i32*
3561.    %allocaCall67 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3562.    %tmpAddr68 = bitcast i8* %allocaCall67 to i32*
3563.    store i32* %tmpAddr68, i32** %tmp66
3564.    %tmpWithVal69 = load i32*, i32** %tmp66
3565.    store i32 12, i32* %tmpWithVal69
3566.    %asChar70 = bitcast i32* %tmpWithVal69 to i8*
3567.    %tmp71 = alloca i32*
3568.    %allocaCall72 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3569.    %tmpAddr73 = bitcast i8* %allocaCall72 to i32*
3570.    store i32* %tmpAddr73, i32** %tmp71
3571.    %tmpWithVal74 = load i32*, i32** %tmp71
3572.    store i32 13, i32* %tmpWithVal74
3573.    %asChar75 = bitcast i32* %tmpWithVal74 to i8*
3574.    %tmp76 = alloca i32*
3575.    %allocaCall77 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3576.    %tmpAddr78 = bitcast i8* %allocaCall77 to i32*
3577.    store i32* %tmpAddr78, i32** %tmp76
3578.    %tmpWithVal79 = load i32*, i32** %tmp76
3579.    store i32 14, i32* %tmpWithVal79
3580.    %asChar80 = bitcast i32* %tmpWithVal79 to i8*
3581.    %tmp81 = alloca i32*
3582.    %allocaCall82 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3583.    %tmpAddr83 = bitcast i8* %allocaCall82 to i32*
3584.    store i32* %tmpAddr83, i32** %tmp81
3585.    %tmpWithVal84 = load i32*, i32** %tmp81
3586.    store i32 15, i32* %tmpWithVal84
3587.    %asChar85 = bitcast i32* %tmpWithVal84 to i8*
3588.    %tmp86 = alloca i32*
3589.    %allocaCall87 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))

```

```

3590.    %tmpAddr88 = bitcast i8* %alloca87 to i32*
3591.    store i32* %tmpAddr88, i32** %tmp86
3592.    %tmpWithVal89 = load i32*, i32** %tmp86
3593.    store i32 16, i32* %tmpWithVal89
3594.    %asChar90 = bitcast i32* %tmpWithVal89 to i8*
3595.    %tmp91 = alloca i32*
3596.    %alloca92 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3597.    %tmpAddr93 = bitcast i8* %alloca92 to i32*
3598.    store i32* %tmpAddr93, i32** %tmp91
3599.    %tmpWithVal94 = load i32*, i32** %tmp91
3600.    store i32 17, i32* %tmpWithVal94
3601.    %asChar95 = bitcast i32* %tmpWithVal94 to i8*
3602.    %tmp96 = alloca i32*
3603.    %alloca97 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
3604.    %tmpAddr98 = bitcast i8* %alloca97 to i32*
3605.    store i32* %tmpAddr98, i32** %tmp96
3606.    %tmpWithVal99 = load i32*, i32** %tmp96
3607.    store i32 18, i32* %tmpWithVal99
3608.    %asChar100 = bitcast i32* %tmpWithVal99 to i8*
3609.    %tmp101 = alloca i32*
3610.    %alloca102 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3611.    %tmpAddr103 = bitcast i8* %alloca102 to i32*
3612.    store i32* %tmpAddr103, i32** %tmp101
3613.    %tmpWithVal104 = load i32*, i32** %tmp101
3614.    store i32 19, i32* %tmpWithVal104
3615.    %asChar105 = bitcast i32* %tmpWithVal104 to i8*
3616.    %tmp106 = alloca i32*
3617.    %alloca107 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3618.    %tmpAddr108 = bitcast i8* %alloca107 to i32*
3619.    store i32* %tmpAddr108, i32** %tmp106
3620.    %tmpWithVal109 = load i32*, i32** %tmp106
3621.    store i32 20, i32* %tmpWithVal109
3622.    %asChar110 = bitcast i32* %tmpWithVal109 to i8*
3623.    %tmp111 = alloca i32*
3624.    %alloca112 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3625.    %tmpAddr113 = bitcast i8* %alloca112 to i32*
3626.    store i32* %tmpAddr113, i32** %tmp111
3627.    %tmpWithVal114 = load i32*, i32** %tmp111
3628.    store i32 21, i32* %tmpWithVal114

```

```

3629.      %asChar115 = bitcast i32* %tmpWithVal114 to i8*
3630.      %tmp116 = alloca i32*
3631.      %malloccall117 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3632.      %tmpAddr118 = bitcast i8* %malloccall117 to i32*
3633.      store i32* %tmpAddr118, i32** %tmp116
3634.      %tmpWithVal119 = load i32*, i32** %tmp116
3635.      store i32 22, i32* %tmpWithVal119
3636.      %asChar120 = bitcast i32* %tmpWithVal119 to i8*
3637.      %tmp121 = alloca i32*
3638.      %malloccall122 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3639.      %tmpAddr123 = bitcast i8* %malloccall122 to i32*
3640.      store i32* %tmpAddr123, i32** %tmp121
3641.      %tmpWithVal124 = load i32*, i32** %tmp121
3642.      store i32 23, i32* %tmpWithVal124
3643.      %asChar125 = bitcast i32* %tmpWithVal124 to i8*
3644.      %tmp126 = alloca i32*
3645.      %malloccall127 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3646.      %tmpAddr128 = bitcast i8* %malloccall127 to i32*
3647.      store i32* %tmpAddr128, i32** %tmp126
3648.      %tmpWithVal129 = load i32*, i32** %tmp126
3649.      store i32 24, i32* %tmpWithVal129
3650.      %asChar130 = bitcast i32* %tmpWithVal129 to i8*
3651.      %tmp131 = alloca i32*
3652.      %malloccall132 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3653.      %tmpAddr133 = bitcast i8* %malloccall132 to i32*
3654.      store i32* %tmpAddr133, i32** %tmp131
3655.      %tmpWithVal134 = load i32*, i32** %tmp131
3656.      store i32 25, i32* %tmpWithVal134
3657.      %asChar135 = bitcast i32* %tmpWithVal134 to i8*
3658.      %push_back = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr, i8*
      %asChar)
3659.      %push_back136 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar15)
3660.      %push_back137 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar20)
3661.      %push_back138 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar25)
3662.      %push_back139 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar30)

```

3663. %push_back140 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar35)

3664. %push_back141 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar40)

3665. %push_back142 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar45)

3666. %push_back143 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar50)

3667. %push_back144 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar55)

3668. %push_back145 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar60)

3669. %push_back146 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar65)

3670. %push_back147 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar70)

3671. %push_back148 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar75)

3672. %push_back149 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar80)

3673. %push_back150 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar85)

3674. %push_back151 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar90)

3675. %push_back152 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar95)

3676. %push_back153 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar100)

3677. %push_back154 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar105)

3678. %push_back155 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar110)

3679. %push_back156 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar115)

3680. %push_back157 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar120)

3681. %push_back158 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar125)

3682. %push_back159 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar130)

3683. %push_back160 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
i8* %asChar135)

3684. store %list* %listAddr, %list** %input_output_char


```

3685.      %alloca161 = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
          (%list, %list* null, i32 1) to i32))
3686.      %listAddr162 = bitcast i8* %alloca161 to %list*
3687.      %init_list163 = call i32 (%list*, ...) @init_list(%list* %listAddr162)
3688.      %tmp164 = alloca i32*
3689.      %alloca165 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3690.      %tmpAddr166 = bitcast i8* %alloca165 to i32*
3691.      store i32* %tmpAddr166, i32** %tmp164
3692.      %tmpWithVal167 = load i32*, i32** %tmp164
3693.      store i32 1, i32* %tmpWithVal167
3694.      %asChar168 = bitcast i32* %tmpWithVal167 to i8*
3695.      %tmp169 = alloca i32*
3696.      %alloca170 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3697.      %tmpAddr171 = bitcast i8* %alloca170 to i32*
3698.      store i32* %tmpAddr171, i32** %tmp169
3699.      %tmpWithVal172 = load i32*, i32** %tmp169
3700.      store i32 1, i32* %tmpWithVal172
3701.      %asChar173 = bitcast i32* %tmpWithVal172 to i8*
3702.      %tmp174 = alloca i32*
3703.      %alloca175 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3704.      %tmpAddr176 = bitcast i8* %alloca175 to i32*
3705.      store i32* %tmpAddr176, i32** %tmp174
3706.      %tmpWithVal177 = load i32*, i32** %tmp174
3707.      store i32 1, i32* %tmpWithVal177
3708.      %asChar178 = bitcast i32* %tmpWithVal177 to i8*
3709.      %tmp179 = alloca i32*
3710.      %alloca180 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3711.      %tmpAddr181 = bitcast i8* %alloca180 to i32*
3712.      store i32* %tmpAddr181, i32** %tmp179
3713.      %tmpWithVal182 = load i32*, i32** %tmp179
3714.      store i32 1, i32* %tmpWithVal182
3715.      %asChar183 = bitcast i32* %tmpWithVal182 to i8*
3716.      %tmp184 = alloca i32*
3717.      %alloca185 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
          (i32, i32* null, i32 1) to i32))
3718.      %tmpAddr186 = bitcast i8* %alloca185 to i32*
3719.      store i32* %tmpAddr186, i32** %tmp184
3720.      %tmpWithVal187 = load i32*, i32** %tmp184
3721.      store i32 1, i32* %tmpWithVal187
3722.      %asChar188 = bitcast i32* %tmpWithVal187 to i8*

```

```

3723.    %tmp189 = alloca i32*
3724.    %alloca190 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3725.    %tmpAddr191 = bitcast i8* %alloca190 to i32*
3726.    store i32* %tmpAddr191, i32** %tmp189
3727.    %tmpWithVal192 = load i32*, i32** %tmp189
3728.    store i32 1, i32* %tmpWithVal192
3729.    %asChar193 = bitcast i32* %tmpWithVal192 to i8*
3730.    %tmp194 = alloca i32*
3731.    %alloca195 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3732.    %tmpAddr196 = bitcast i8* %alloca195 to i32*
3733.    store i32* %tmpAddr196, i32** %tmp194
3734.    %tmpWithVal197 = load i32*, i32** %tmp194
3735.    store i32 1, i32* %tmpWithVal197
3736.    %asChar198 = bitcast i32* %tmpWithVal197 to i8*
3737.    %tmp199 = alloca i32*
3738.    %alloca200 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3739.    %tmpAddr201 = bitcast i8* %alloca200 to i32*
3740.    store i32* %tmpAddr201, i32** %tmp199
3741.    %tmpWithVal202 = load i32*, i32** %tmp199
3742.    store i32 1, i32* %tmpWithVal202
3743.    %asChar203 = bitcast i32* %tmpWithVal202 to i8*
3744.    %tmp204 = alloca i32*
3745.    %alloca205 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3746.    %tmpAddr206 = bitcast i8* %alloca205 to i32*
3747.    store i32* %tmpAddr206, i32** %tmp204
3748.    %tmpWithVal207 = load i32*, i32** %tmp204
3749.    store i32 1, i32* %tmpWithVal207
3750.    %asChar208 = bitcast i32* %tmpWithVal207 to i8*
3751.    %tmp209 = alloca i32*
3752.    %alloca210 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3753.    %tmpAddr211 = bitcast i8* %alloca210 to i32*
3754.    store i32* %tmpAddr211, i32** %tmp209
3755.    %tmpWithVal212 = load i32*, i32** %tmp209
3756.    store i32 1, i32* %tmpWithVal212
3757.    %asChar213 = bitcast i32* %tmpWithVal212 to i8*
3758.    %tmp214 = alloca i32*
3759.    %alloca215 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3760.    %tmpAddr216 = bitcast i8* %alloca215 to i32*

```

```

3761.    store i32* %tmpAddr216, i32** %tmp214
3762.    %tmpWithVal217 = load i32*, i32** %tmp214
3763.    store i32 1, i32* %tmpWithVal217
3764.    %asChar218 = bitcast i32* %tmpWithVal217 to i8*
3765.    %tmp219 = alloca i32*
3766.    %mallocall220 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3767.    %tmpAddr221 = bitcast i8* %mallocall220 to i32*
3768.    store i32* %tmpAddr221, i32** %tmp219
3769.    %tmpWithVal222 = load i32*, i32** %tmp219
3770.    store i32 1, i32* %tmpWithVal222
3771.    %asChar223 = bitcast i32* %tmpWithVal222 to i8*
3772.    %tmp224 = alloca i32*
3773.    %mallocall225 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3774.    %tmpAddr226 = bitcast i8* %mallocall225 to i32*
3775.    store i32* %tmpAddr226, i32** %tmp224
3776.    %tmpWithVal227 = load i32*, i32** %tmp224
3777.    store i32 1, i32* %tmpWithVal227
3778.    %asChar228 = bitcast i32* %tmpWithVal227 to i8*
3779.    %tmp229 = alloca i32*
3780.    %mallocall230 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3781.    %tmpAddr231 = bitcast i8* %mallocall230 to i32*
3782.    store i32* %tmpAddr231, i32** %tmp229
3783.    %tmpWithVal232 = load i32*, i32** %tmp229
3784.    store i32 1, i32* %tmpWithVal232
3785.    %asChar233 = bitcast i32* %tmpWithVal232 to i8*
3786.    %tmp234 = alloca i32*
3787.    %mallocall235 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3788.    %tmpAddr236 = bitcast i8* %mallocall235 to i32*
3789.    store i32* %tmpAddr236, i32** %tmp234
3790.    %tmpWithVal237 = load i32*, i32** %tmp234
3791.    store i32 1, i32* %tmpWithVal237
3792.    %asChar238 = bitcast i32* %tmpWithVal237 to i8*
3793.    %tmp239 = alloca i32*
3794.    %mallocall240 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3795.    %tmpAddr241 = bitcast i8* %mallocall240 to i32*
3796.    store i32* %tmpAddr241, i32** %tmp239
3797.    %tmpWithVal242 = load i32*, i32** %tmp239
3798.    store i32 1, i32* %tmpWithVal242
3799.    %asChar243 = bitcast i32* %tmpWithVal242 to i8*

```

```

3800.    %tmp244 = alloca i32*
3801.    %alloca245 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3802.    %tmpAddr246 = bitcast i8* %alloca245 to i32*
3803.    store i32* %tmpAddr246, i32** %tmp244
3804.    %tmpWithVal247 = load i32*, i32** %tmp244
3805.    store i32 1, i32* %tmpWithVal247
3806.    %asChar248 = bitcast i32* %tmpWithVal247 to i8*
3807.    %tmp249 = alloca i32*
3808.    %alloca250 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3809.    %tmpAddr251 = bitcast i8* %alloca250 to i32*
3810.    store i32* %tmpAddr251, i32** %tmp249
3811.    %tmpWithVal252 = load i32*, i32** %tmp249
3812.    store i32 1, i32* %tmpWithVal252
3813.    %asChar253 = bitcast i32* %tmpWithVal252 to i8*
3814.    %tmp254 = alloca i32*
3815.    %alloca255 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3816.    %tmpAddr256 = bitcast i8* %alloca255 to i32*
3817.    store i32* %tmpAddr256, i32** %tmp254
3818.    %tmpWithVal257 = load i32*, i32** %tmp254
3819.    store i32 1, i32* %tmpWithVal257
3820.    %asChar258 = bitcast i32* %tmpWithVal257 to i8*
3821.    %tmp259 = alloca i32*
3822.    %alloca260 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3823.    %tmpAddr261 = bitcast i8* %alloca260 to i32*
3824.    store i32* %tmpAddr261, i32** %tmp259
3825.    %tmpWithVal262 = load i32*, i32** %tmp259
3826.    store i32 1, i32* %tmpWithVal262
3827.    %asChar263 = bitcast i32* %tmpWithVal262 to i8*
3828.    %tmp264 = alloca i32*
3829.    %alloca265 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3830.    %tmpAddr266 = bitcast i8* %alloca265 to i32*
3831.    store i32* %tmpAddr266, i32** %tmp264
3832.    %tmpWithVal267 = load i32*, i32** %tmp264
3833.    store i32 1, i32* %tmpWithVal267
3834.    %asChar268 = bitcast i32* %tmpWithVal267 to i8*
3835.    %tmp269 = alloca i32*
3836.    %alloca270 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3837.    %tmpAddr271 = bitcast i8* %alloca270 to i32*

```

```

3838.    store i32* %tmpAddr271, i32** %tmp269
3839.    %tmpWithVal272 = load i32*, i32** %tmp269
3840.    store i32 1, i32* %tmpWithVal272
3841.    %asChar273 = bitcast i32* %tmpWithVal272 to i8*
3842.    %tmp274 = alloca i32*
3843.    %malloccall275 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3844.    %tmpAddr276 = bitcast i8* %malloccall275 to i32*
3845.    store i32* %tmpAddr276, i32** %tmp274
3846.    %tmpWithVal277 = load i32*, i32** %tmp274
3847.    store i32 1, i32* %tmpWithVal277
3848.    %asChar278 = bitcast i32* %tmpWithVal277 to i8*
3849.    %tmp279 = alloca i32*
3850.    %malloccall280 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3851.    %tmpAddr281 = bitcast i8* %malloccall280 to i32*
3852.    store i32* %tmpAddr281, i32** %tmp279
3853.    %tmpWithVal282 = load i32*, i32** %tmp279
3854.    store i32 1, i32* %tmpWithVal282
3855.    %asChar283 = bitcast i32* %tmpWithVal282 to i8*
3856.    %tmp284 = alloca i32*
3857.    %malloccall285 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3858.    %tmpAddr286 = bitcast i8* %malloccall285 to i32*
3859.    store i32* %tmpAddr286, i32** %tmp284
3860.    %tmpWithVal287 = load i32*, i32** %tmp284
3861.    store i32 1, i32* %tmpWithVal287
3862.    %asChar288 = bitcast i32* %tmpWithVal287 to i8*
3863.    %tmp289 = alloca i32*
3864.    %malloccall290 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3865.    %tmpAddr291 = bitcast i8* %malloccall290 to i32*
3866.    store i32* %tmpAddr291, i32** %tmp289
3867.    %tmpWithVal292 = load i32*, i32** %tmp289
3868.    store i32 1, i32* %tmpWithVal292
3869.    %asChar293 = bitcast i32* %tmpWithVal292 to i8*
3870.    %tmp294 = alloca i32*
3871.    %malloccall295 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3872.    %tmpAddr296 = bitcast i8* %malloccall295 to i32*
3873.    store i32* %tmpAddr296, i32** %tmp294
3874.    %tmpWithVal297 = load i32*, i32** %tmp294
3875.    store i32 1, i32* %tmpWithVal297
3876.    %asChar298 = bitcast i32* %tmpWithVal297 to i8*

```

```

3877.    %tmp299 = alloca i32*
3878.    %alloca300 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3879.    %tmpAddr301 = bitcast i8* %alloca300 to i32*
3880.    store i32* %tmpAddr301, i32** %tmp299
3881.    %tmpWithVal302 = load i32*, i32** %tmp299
3882.    store i32 1, i32* %tmpWithVal302
3883.    %asChar303 = bitcast i32* %tmpWithVal302 to i8*
3884.    %tmp304 = alloca i32*
3885.    %alloca305 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3886.    %tmpAddr306 = bitcast i8* %alloca305 to i32*
3887.    store i32* %tmpAddr306, i32** %tmp304
3888.    %tmpWithVal307 = load i32*, i32** %tmp304
3889.    store i32 1, i32* %tmpWithVal307
3890.    %asChar308 = bitcast i32* %tmpWithVal307 to i8*
3891.    %tmp309 = alloca i32*
3892.    %alloca310 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3893.    %tmpAddr311 = bitcast i8* %alloca310 to i32*
3894.    store i32* %tmpAddr311, i32** %tmp309
3895.    %tmpWithVal312 = load i32*, i32** %tmp309
3896.    store i32 1, i32* %tmpWithVal312
3897.    %asChar313 = bitcast i32* %tmpWithVal312 to i8*
3898.    %tmp314 = alloca i32*
3899.    %alloca315 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3900.    %tmpAddr316 = bitcast i8* %alloca315 to i32*
3901.    store i32* %tmpAddr316, i32** %tmp314
3902.    %tmpWithVal317 = load i32*, i32** %tmp314
3903.    store i32 1, i32* %tmpWithVal317
3904.    %asChar318 = bitcast i32* %tmpWithVal317 to i8*
3905.    %tmp319 = alloca i32*
3906.    %alloca320 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3907.    %tmpAddr321 = bitcast i8* %alloca320 to i32*
3908.    store i32* %tmpAddr321, i32** %tmp319
3909.    %tmpWithVal322 = load i32*, i32** %tmp319
3910.    store i32 1, i32* %tmpWithVal322
3911.    %asChar323 = bitcast i32* %tmpWithVal322 to i8*
3912.    %tmp324 = alloca i32*
3913.    %alloca325 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3914.    %tmpAddr326 = bitcast i8* %alloca325 to i32*

```

```

3915.    store i32* %tmpAddr326, i32** %tmp324
3916.    %tmpWithVal327 = load i32*, i32** %tmp324
3917.    store i32 1, i32* %tmpWithVal327
3918.    %asChar328 = bitcast i32* %tmpWithVal327 to i8*
3919.    %tmp329 = alloca i32*
3920.    %allocaCall330 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3921.    %tmpAddr331 = bitcast i8* %allocaCall330 to i32*
3922.    store i32* %tmpAddr331, i32** %tmp329
3923.    %tmpWithVal332 = load i32*, i32** %tmp329
3924.    store i32 1, i32* %tmpWithVal332
3925.    %asChar333 = bitcast i32* %tmpWithVal332 to i8*
3926.    %tmp334 = alloca i32*
3927.    %allocaCall335 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3928.    %tmpAddr336 = bitcast i8* %allocaCall335 to i32*
3929.    store i32* %tmpAddr336, i32** %tmp334
3930.    %tmpWithVal337 = load i32*, i32** %tmp334
3931.    store i32 1, i32* %tmpWithVal337
3932.    %asChar338 = bitcast i32* %tmpWithVal337 to i8*
3933.    %tmp339 = alloca i32*
3934.    %allocaCall340 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3935.    %tmpAddr341 = bitcast i8* %allocaCall340 to i32*
3936.    store i32* %tmpAddr341, i32** %tmp339
3937.    %tmpWithVal342 = load i32*, i32** %tmp339
3938.    store i32 1, i32* %tmpWithVal342
3939.    %asChar343 = bitcast i32* %tmpWithVal342 to i8*
3940.    %tmp344 = alloca i32*
3941.    %allocaCall345 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3942.    %tmpAddr346 = bitcast i8* %allocaCall345 to i32*
3943.    store i32* %tmpAddr346, i32** %tmp344
3944.    %tmpWithVal347 = load i32*, i32** %tmp344
3945.    store i32 1, i32* %tmpWithVal347
3946.    %asChar348 = bitcast i32* %tmpWithVal347 to i8*
3947.    %tmp349 = alloca i32*
3948.    %allocaCall350 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
3949.    %tmpAddr351 = bitcast i8* %allocaCall350 to i32*
3950.    store i32* %tmpAddr351, i32** %tmp349
3951.    %tmpWithVal352 = load i32*, i32** %tmp349
3952.    store i32 1, i32* %tmpWithVal352
3953.    %asChar353 = bitcast i32* %tmpWithVal352 to i8*

```

```

3954.    %tmp354 = alloca i32*
3955.    %alloca355 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3956.    %tmpAddr356 = bitcast i8* %alloca355 to i32*
3957.    store i32* %tmpAddr356, i32** %tmp354
3958.    %tmpWithVal357 = load i32*, i32** %tmp354
3959.    store i32 1, i32* %tmpWithVal357
3960.    %asChar358 = bitcast i32* %tmpWithVal357 to i8*
3961.    %tmp359 = alloca i32*
3962.    %alloca360 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3963.    %tmpAddr361 = bitcast i8* %alloca360 to i32*
3964.    store i32* %tmpAddr361, i32** %tmp359
3965.    %tmpWithVal362 = load i32*, i32** %tmp359
3966.    store i32 1, i32* %tmpWithVal362
3967.    %asChar363 = bitcast i32* %tmpWithVal362 to i8*
3968.    %tmp364 = alloca i32*
3969.    %alloca365 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3970.    %tmpAddr366 = bitcast i8* %alloca365 to i32*
3971.    store i32* %tmpAddr366, i32** %tmp364
3972.    %tmpWithVal367 = load i32*, i32** %tmp364
3973.    store i32 1, i32* %tmpWithVal367
3974.    %asChar368 = bitcast i32* %tmpWithVal367 to i8*
3975.    %tmp369 = alloca i32*
3976.    %alloca370 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3977.    %tmpAddr371 = bitcast i8* %alloca370 to i32*
3978.    store i32* %tmpAddr371, i32** %tmp369
3979.    %tmpWithVal372 = load i32*, i32** %tmp369
3980.    store i32 1, i32* %tmpWithVal372
3981.    %asChar373 = bitcast i32* %tmpWithVal372 to i8*
3982.    %tmp374 = alloca i32*
3983.    %alloca375 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3984.    %tmpAddr376 = bitcast i8* %alloca375 to i32*
3985.    store i32* %tmpAddr376, i32** %tmp374
3986.    %tmpWithVal377 = load i32*, i32** %tmp374
3987.    store i32 1, i32* %tmpWithVal377
3988.    %asChar378 = bitcast i32* %tmpWithVal377 to i8*
3989.    %tmp379 = alloca i32*
3990.    %alloca380 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3991.    %tmpAddr381 = bitcast i8* %alloca380 to i32*

```



```

3992.    store i32* %tmpAddr381, i32** %tmp379
3993.    %tmpWithVal382 = load i32*, i32** %tmp379
3994.    store i32 1, i32* %tmpWithVal382
3995.    %asChar383 = bitcast i32* %tmpWithVal382 to i8*
3996.    %tmp384 = alloca i32*
3997.    %alloca385 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
3998.    %tmpAddr386 = bitcast i8* %alloca385 to i32*
3999.    store i32* %tmpAddr386, i32** %tmp384
4000.    %tmpWithVal387 = load i32*, i32** %tmp384
4001.    store i32 1, i32* %tmpWithVal387
4002.    %asChar388 = bitcast i32* %tmpWithVal387 to i8*
4003.    %tmp389 = alloca i32*
4004.    %alloca390 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4005.    %tmpAddr391 = bitcast i8* %alloca390 to i32*
4006.    store i32* %tmpAddr391, i32** %tmp389
4007.    %tmpWithVal392 = load i32*, i32** %tmp389
4008.    store i32 1, i32* %tmpWithVal392
4009.    %asChar393 = bitcast i32* %tmpWithVal392 to i8*
4010.    %tmp394 = alloca i32*
4011.    %alloca395 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4012.    %tmpAddr396 = bitcast i8* %alloca395 to i32*
4013.    store i32* %tmpAddr396, i32** %tmp394
4014.    %tmpWithVal397 = load i32*, i32** %tmp394
4015.    store i32 1, i32* %tmpWithVal397
4016.    %asChar398 = bitcast i32* %tmpWithVal397 to i8*
4017.    %tmp399 = alloca i32*
4018.    %alloca400 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4019.    %tmpAddr401 = bitcast i8* %alloca400 to i32*
4020.    store i32* %tmpAddr401, i32** %tmp399
4021.    %tmpWithVal402 = load i32*, i32** %tmp399
4022.    store i32 1, i32* %tmpWithVal402
4023.    %asChar403 = bitcast i32* %tmpWithVal402 to i8*
4024.    %tmp404 = alloca i32*
4025.    %alloca405 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4026.    %tmpAddr406 = bitcast i8* %alloca405 to i32*
4027.    store i32* %tmpAddr406, i32** %tmp404
4028.    %tmpWithVal407 = load i32*, i32** %tmp404
4029.    store i32 1, i32* %tmpWithVal407
4030.    %asChar408 = bitcast i32* %tmpWithVal407 to i8*

```

```

4031.    %tmp409 = alloca i32*
4032.    %alloca410 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4033.    %tmpAddr411 = bitcast i8* %alloca410 to i32*
4034.    store i32* %tmpAddr411, i32** %tmp409
4035.    %tmpWithVal412 = load i32*, i32** %tmp409
4036.    store i32 1, i32* %tmpWithVal412
4037.    %asChar413 = bitcast i32* %tmpWithVal412 to i8*
4038.    %tmp414 = alloca i32*
4039.    %alloca415 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4040.    %tmpAddr416 = bitcast i8* %alloca415 to i32*
4041.    store i32* %tmpAddr416, i32** %tmp414
4042.    %tmpWithVal417 = load i32*, i32** %tmp414
4043.    store i32 1, i32* %tmpWithVal417
4044.    %asChar418 = bitcast i32* %tmpWithVal417 to i8*
4045.    %tmp419 = alloca i32*
4046.    %alloca420 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4047.    %tmpAddr421 = bitcast i8* %alloca420 to i32*
4048.    store i32* %tmpAddr421, i32** %tmp419
4049.    %tmpWithVal422 = load i32*, i32** %tmp419
4050.    store i32 1, i32* %tmpWithVal422
4051.    %asChar423 = bitcast i32* %tmpWithVal422 to i8*
4052.    %tmp424 = alloca i32*
4053.    %alloca425 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4054.    %tmpAddr426 = bitcast i8* %alloca425 to i32*
4055.    store i32* %tmpAddr426, i32** %tmp424
4056.    %tmpWithVal427 = load i32*, i32** %tmp424
4057.    store i32 1, i32* %tmpWithVal427
4058.    %asChar428 = bitcast i32* %tmpWithVal427 to i8*
4059.    %tmp429 = alloca i32*
4060.    %alloca430 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4061.    %tmpAddr431 = bitcast i8* %alloca430 to i32*
4062.    store i32* %tmpAddr431, i32** %tmp429
4063.    %tmpWithVal432 = load i32*, i32** %tmp429
4064.    store i32 1, i32* %tmpWithVal432
4065.    %asChar433 = bitcast i32* %tmpWithVal432 to i8*
4066.    %tmp434 = alloca i32*
4067.    %alloca435 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4068.    %tmpAddr436 = bitcast i8* %alloca435 to i32*

```

```

4069.    store i32* %tmpAddr436, i32** %tmp434
4070.    %tmpWithVal437 = load i32*, i32** %tmp434
4071.    store i32 1, i32* %tmpWithVal437
4072.    %asChar438 = bitcast i32* %tmpWithVal437 to i8*
4073.    %tmp439 = alloca i32*
4074.    %malloccall440 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4075.    %tmpAddr441 = bitcast i8* %malloccall440 to i32*
4076.    store i32* %tmpAddr441, i32** %tmp439
4077.    %tmpWithVal442 = load i32*, i32** %tmp439
4078.    store i32 1, i32* %tmpWithVal442
4079.    %asChar443 = bitcast i32* %tmpWithVal442 to i8*
4080.    %tmp444 = alloca i32*
4081.    %malloccall445 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4082.    %tmpAddr446 = bitcast i8* %malloccall445 to i32*
4083.    store i32* %tmpAddr446, i32** %tmp444
4084.    %tmpWithVal447 = load i32*, i32** %tmp444
4085.    store i32 1, i32* %tmpWithVal447
4086.    %asChar448 = bitcast i32* %tmpWithVal447 to i8*
4087.    %tmp449 = alloca i32*
4088.    %malloccall450 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4089.    %tmpAddr451 = bitcast i8* %malloccall450 to i32*
4090.    store i32* %tmpAddr451, i32** %tmp449
4091.    %tmpWithVal452 = load i32*, i32** %tmp449
4092.    store i32 1, i32* %tmpWithVal452
4093.    %asChar453 = bitcast i32* %tmpWithVal452 to i8*
4094.    %tmp454 = alloca i32*
4095.    %malloccall455 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4096.    %tmpAddr456 = bitcast i8* %malloccall455 to i32*
4097.    store i32* %tmpAddr456, i32** %tmp454
4098.    %tmpWithVal457 = load i32*, i32** %tmp454
4099.    store i32 1, i32* %tmpWithVal457
4100.    %asChar458 = bitcast i32* %tmpWithVal457 to i8*
4101.    %tmp459 = alloca i32*
4102.    %malloccall460 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4103.    %tmpAddr461 = bitcast i8* %malloccall460 to i32*
4104.    store i32* %tmpAddr461, i32** %tmp459
4105.    %tmpWithVal462 = load i32*, i32** %tmp459
4106.    store i32 1, i32* %tmpWithVal462
4107.    %asChar463 = bitcast i32* %tmpWithVal462 to i8*

```

```

4108.    %tmp464 = alloca i32*
4109.    %alloca465 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4110.    %tmpAddr466 = bitcast i8* %alloca465 to i32*
4111.    store i32* %tmpAddr466, i32** %tmp464
4112.    %tmpWithVal467 = load i32*, i32** %tmp464
4113.    store i32 1, i32* %tmpWithVal467
4114.    %asChar468 = bitcast i32* %tmpWithVal467 to i8*
4115.    %tmp469 = alloca i32*
4116.    %alloca470 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4117.    %tmpAddr471 = bitcast i8* %alloca470 to i32*
4118.    store i32* %tmpAddr471, i32** %tmp469
4119.    %tmpWithVal472 = load i32*, i32** %tmp469
4120.    store i32 1, i32* %tmpWithVal472
4121.    %asChar473 = bitcast i32* %tmpWithVal472 to i8*
4122.    %tmp474 = alloca i32*
4123.    %alloca475 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4124.    %tmpAddr476 = bitcast i8* %alloca475 to i32*
4125.    store i32* %tmpAddr476, i32** %tmp474
4126.    %tmpWithVal477 = load i32*, i32** %tmp474
4127.    store i32 1, i32* %tmpWithVal477
4128.    %asChar478 = bitcast i32* %tmpWithVal477 to i8*
4129.    %tmp479 = alloca i32*
4130.    %alloca480 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4131.    %tmpAddr481 = bitcast i8* %alloca480 to i32*
4132.    store i32* %tmpAddr481, i32** %tmp479
4133.    %tmpWithVal482 = load i32*, i32** %tmp479
4134.    store i32 1, i32* %tmpWithVal482
4135.    %asChar483 = bitcast i32* %tmpWithVal482 to i8*
4136.    %tmp484 = alloca i32*
4137.    %alloca485 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4138.    %tmpAddr486 = bitcast i8* %alloca485 to i32*
4139.    store i32* %tmpAddr486, i32** %tmp484
4140.    %tmpWithVal487 = load i32*, i32** %tmp484
4141.    store i32 1, i32* %tmpWithVal487
4142.    %asChar488 = bitcast i32* %tmpWithVal487 to i8*
4143.    %push_back489 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr162, i8* %asChar168)
4144.    %push_back490 = call i32 (%list*, i8*, ...) @push_back(%list*
      %listAddr162, i8* %asChar173)

```

4145. %push_back491 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar178)

4146. %push_back492 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar183)

4147. %push_back493 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar188)

4148. %push_back494 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar193)

4149. %push_back495 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar198)

4150. %push_back496 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar203)

4151. %push_back497 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar208)

4152. %push_back498 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar213)

4153. %push_back499 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar218)

4154. %push_back500 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar223)

4155. %push_back501 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar228)

4156. %push_back502 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar233)

4157. %push_back503 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar238)

4158. %push_back504 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar243)

4159. %push_back505 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar248)

4160. %push_back506 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar253)

4161. %push_back507 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar258)

4162. %push_back508 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar263)

4163. %push_back509 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar268)

4164. %push_back510 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar273)

4165. %push_back511 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar278)

4166. %push_back512 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar283)

4167. %push_back513 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar288)

4168. %push_back514 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar293)

4169. %push_back515 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar298)

4170. %push_back516 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar303)

4171. %push_back517 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar308)

4172. %push_back518 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar313)

4173. %push_back519 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar318)

4174. %push_back520 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar323)

4175. %push_back521 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar328)

4176. %push_back522 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar333)

4177. %push_back523 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar338)

4178. %push_back524 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar343)

4179. %push_back525 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar348)

4180. %push_back526 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar353)

4181. %push_back527 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar358)

4182. %push_back528 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar363)

4183. %push_back529 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar368)

4184. %push_back530 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar373)

4185. %push_back531 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar378)

4186. %push_back532 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar383)

4187. %push_back533 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar388)

4188. %push_back534 = call i32 (%list*, i8*, ...) @push_back(%list*
 %listAddr162, i8* %asChar393)

```

4189.      %push_back535 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar398)
4190.      %push_back536 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar403)
4191.      %push_back537 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar408)
4192.      %push_back538 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar413)
4193.      %push_back539 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar418)
4194.      %push_back540 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar423)
4195.      %push_back541 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar428)
4196.      %push_back542 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar433)
4197.      %push_back543 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar438)
4198.      %push_back544 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar443)
4199.      %push_back545 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar448)
4200.      %push_back546 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar453)
4201.      %push_back547 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar458)
4202.      %push_back548 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar463)
4203.      %push_back549 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar468)
4204.      %push_back550 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar473)
4205.      %push_back551 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar478)
4206.      %push_back552 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar483)
4207.      %push_back553 = call i32 (%list*, i8*, ...) @push_back(%list*
          %listAddr162, i8* %asChar488)
4208.      store %list* %listAddr162, %list** %cipher_output_character
4209.      store i32 0, i32* %loop_counter_e_d
4210.      br label %while
4211.
4212.      while:                                ; preds = %merge, %entry
4213.      %loop_counter_e_d615 = load i32, i32* %loop_counter_e_d

```

```

4214.    %text_len616 = load i32, i32* %text_len
4215.    %tmp617 = icmp slt i32 %loop_counter_e_d615, %text_len616
4216.    br i1 %tmp617, label %while_body, label %merge618
4217.
4218.    while_body:                                ; preds = %while
4219.    %text554 = load %list*, %list** %text7
4220.    %loop_counter_e_d555 = load i32, i32* %loop_counter_e_d
4221.    %get_at = call i8* (%list*, i32, ...) @get_at(%list* %text554, i32
    %loop_counter_e_d555)
4222.    %eltAsPtr = bitcast i8* %get_at to i32*
4223.    %eltDeref = load i32, i32* %eltAsPtr
4224.    %input_output_char556 = load %list*, %list** %input_output_char
4225.    %character_location_status_result = call i32
    @character_location_status(%list* %input_output_char556, i32 %eltDeref)
4226.    store i32 %character_location_status_result, i32* %char_loc_holder
4227.    %rotor_three_e_input557 = load %list*, %list** %rotor_three_e_input5
4228.    %rotor_shift_result = call %list* @rotor_shift(%list*
    %rotor_three_e_input557, i32 1)
4229.    %rotor_three_e_output558 = load %list*, %list**
    %rotor_three_e_output6
4230.    %rotor_shift_result559 = call %list* @rotor_shift(%list*
    %rotor_three_e_output558, i32 1)
4231.    %count_rotor_three560 = load i32, i32* %count_rotor_three
4232.    %tmp561 = add i32 %count_rotor_three560, 1
4233.    store i32 %tmp561, i32* %count_rotor_three
4234.    br label %if
4235.
4236.    if:                                          ; preds = %while_body
4237.    %count_rotor_three592 = load i32, i32* %count_rotor_three
4238.    %tmp593 = icmp sgt i32 %count_rotor_three592, 0
4239.    br i1 %tmp593, label %if_body, label %merge
4240.
4241.    if_body:                                    ; preds = %if
4242.    %count_rotor_three562 = load i32, i32* %count_rotor_three
4243.    %my_modulus_result_result = call i32 @my_modulus_result(i32
    %count_rotor_three562)
4244.    store i32 %my_modulus_result_result, i32* %modulus_result
4245.    br label %if563
4246.
4247.    merge:                                      ; preds = %if, %merge565
4248.    %rotor_three_e_output594 = load %list*, %list**
    %rotor_three_e_output6
4249.    %rotor_three_e_input595 = load %list*, %list** %rotor_three_e_input5
4250.    %rotor_two_e_output596 = load %list*, %list** %rotor_two_e_output4

```



```

4251.      %rotor_two_e_input597 = load %list*, %list** %rotor_two_e_input3
4252.      %rotor_one_e_output598 = load %list*, %list** %rotor_one_e_output2
4253.      %rotor_one_e_input599 = load %list*, %list** %rotor_one_e_input1
4254.      %char_loc_holder600 = load i32, i32* %char_loc_holder
4255.      %rotor_direction_output_result = call i32 @rotor_direction_output(i32
      %char_loc_holder600, %list* %rotor_one_e_input599, %list*
      %rotor_one_e_output598, %list* %rotor_two_e_input597, %list*
      %rotor_two_e_output596, %list* %rotor_three_e_input595, %list*
      %rotor_three_e_output594)
4256.      store i32 %rotor_direction_output_result, i32* %char_loc_holder
4257.      %cipher_output_character601 = load %list*, %list**
      %cipher_output_character
4258.      %loop_counter_e_d602 = load i32, i32* %loop_counter_e_d
4259.      %input_output_char603 = load %list*, %list** %input_output_char
4260.      %char_loc_holder604 = load i32, i32* %char_loc_holder
4261.      %get_at605 = call i8* (%list*, i32, ...) @get_at(%list*
      %input_output_char603, i32 %char_loc_holder604)
4262.      %eltAsPtr606 = bitcast i8* %get_at605 to i32*
4263.      %eltDeref607 = load i32, i32* %eltAsPtr606
4264.      %tmp608 = alloca i32*
4265.      %mallocall609 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4266.      %tmpAddr610 = bitcast i8* %mallocall609 to i32*
4267.      store i32* %tmpAddr610, i32** %tmp608
4268.      %tmpWithVal611 = load i32*, i32** %tmp608
4269.      store i32 %eltDeref607, i32* %tmpWithVal611
4270.      %asChar612 = bitcast i32* %tmpWithVal611 to i8*
4271.      %set_at = call %list* (%list*, i32, i8*, ...) @set_at(%list*
      %cipher_output_character601, i32 %loop_counter_e_d602, i8* %asChar612)
4272.      %loop_counter_e_d613 = load i32, i32* %loop_counter_e_d
4273.      %tmp614 = add i32 %loop_counter_e_d613, 1
4274.      store i32 %tmp614, i32* %loop_counter_e_d
4275.      br label %while
4276.
4277.      if563:                                ; preds = %if_body
4278.      %modulus_result590 = load i32, i32* %modulus_result
4279.      %tmp591 = icmp eq i32 %modulus_result590, 0
4280.      br i1 %tmp591, label %if_body564, label %merge565
4281.
4282.      if_body564:                            ; preds = %if563
4283.      %rotor_two_e_input566 = load %list*, %list** %rotor_two_e_input3
4284.      %rotor_shift_result567 = call %list* @rotor_shift(%list*
      %rotor_two_e_input566, i32 1)
4285.      %rotor_two_e_output568 = load %list*, %list** %rotor_two_e_output4

```

```

4286.    %rotor_shift_result569 = call %list* @rotor_shift(%list*
        %rotor_two_e_output568, i32 1)
4287.    %count_rotor_two570 = load i32, i32* %count_rotor_two
4288.    %tmp571 = add i32 %count_rotor_two570, 1
4289.    store i32 %tmp571, i32* %count_rotor_two
4290.    br label %if572
4291.
4292. merge565:                                ; preds = %if563, %merge574
4293.    br label %merge
4294.
4295. if572:                                    ; preds = %if_body564
4296.    %count_rotor_two588 = load i32, i32* %count_rotor_two
4297.    %tmp589 = icmp sgt i32 %count_rotor_two588, 0
4298.    br i1 %tmp589, label %if_body573, label %merge574
4299.
4300. if_body573:                              ; preds = %if572
4301.    %count_rotor_two575 = load i32, i32* %count_rotor_two
4302.    %my_modulus_result_result576 = call i32 @my_modulus_result(i32
        %count_rotor_two575)
4303.    store i32 %my_modulus_result_result576, i32* %modulus_result_t
4304.    br label %if577
4305.
4306. merge574:                                ; preds = %if572, %merge579
4307.    br label %merge565
4308.
4309. if577:                                    ; preds = %if_body573
4310.    %modulus_result_t586 = load i32, i32* %modulus_result_t
4311.    %tmp587 = icmp eq i32 %modulus_result_t586, 0
4312.    br i1 %tmp587, label %if_body578, label %merge579
4313.
4314. if_body578:                              ; preds = %if577
4315.    %rotor_one_e_input580 = load %list*, %list** %rotor_one_e_input1
4316.    %rotor_shift_result581 = call %list* @rotor_shift(%list*
        %rotor_one_e_input580, i32 1)
4317.    %rotor_one_e_output582 = load %list*, %list** %rotor_one_e_output2
4318.    %rotor_shift_result583 = call %list* @rotor_shift(%list*
        %rotor_one_e_output582, i32 1)
4319.    %count_rotor_one584 = load i32, i32* %count_rotor_one
4320.    %tmp585 = add i32 %count_rotor_one584, 1
4321.    store i32 %tmp585, i32* %count_rotor_one
4322.    br label %merge579
4323.
4324. merge579:                                ; preds = %if577, %if_body578
4325.    br label %merge574

```

```

4326.
4327.     merge618:                                ; preds = %while
4328.     %cipher_output_character619 = load %list*, %list**
        %cipher_output_character
4329.     ret %list* %cipher_output_character619
4330.     }
4331.
4332.     define i32 @rotor_direction_output(i32 %input_output_char_loc, %list*
        %rotor_one_input_direct, %list* %rotor_one_output_direct, %list*
        %rotor_two_input_direct, %list* %rotor_two_output_direct, %list*
        %rotor_three_input_direct, %list* %rotor_three_output_direct) {
4333.     entry:
4334.     %input_output_char_loc1 = alloca i32
4335.     store i32 %input_output_char_loc, i32* %input_output_char_loc1
4336.     %rotor_one_input_direct2 = alloca %list*
4337.     store %list* %rotor_one_input_direct, %list** %rotor_one_input_direct2
4338.     %rotor_one_output_direct3 = alloca %list*
4339.     store %list* %rotor_one_output_direct, %list**
        %rotor_one_output_direct3
4340.     %rotor_two_input_direct4 = alloca %list*
4341.     store %list* %rotor_two_input_direct, %list** %rotor_two_input_direct4
4342.     %rotor_two_output_direct5 = alloca %list*
4343.     store %list* %rotor_two_output_direct, %list**
        %rotor_two_output_direct5
4344.     %rotor_three_input_direct6 = alloca %list*
4345.     store %list* %rotor_three_input_direct, %list**
        %rotor_three_input_direct6
4346.     %rotor_three_output_direct7 = alloca %list*
4347.     store %list* %rotor_three_output_direct, %list**
        %rotor_three_output_direct7
4348.     %index_reverse = alloca i32
4349.     %character_location_holder = alloca i32
4350.     %character_location_holder_two = alloca i32
4351.     %reflector = alloca %list*
4352.     store i32 0, i32* %index_reverse
4353.     %input_output_char_loc8 = load i32, i32* %input_output_char_loc1
4354.     store i32 %input_output_char_loc8, i32* %character_location_holder
4355.     store i32 0, i32* %character_location_holder_two
4356.     %mallocall = tail call i8* @malloc(i32 ptrtoint (%list* getelementptr
        (%list, %list* null, i32 1) to i32))
4357.     %listAddr = bitcast i8* %mallocall to %list*
4358.     %init_list = call i32 (%list*, ...) @init_list(%list* %listAddr)
4359.     %tmp = alloca i32*

```

```

4360.      %malloccall9 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4361.      %tmpAddr = bitcast i8* %malloccall9 to i32*
4362.      store i32* %tmpAddr, i32** %tmp
4363.      %tmpWithVal = load i32*, i32** %tmp
4364.      store i32 0, i32* %tmpWithVal
4365.      %asChar = bitcast i32* %tmpWithVal to i8*
4366.      %tmp10 = alloca i32*
4367.      %malloccall11 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4368.      %tmpAddr12 = bitcast i8* %malloccall11 to i32*
4369.      store i32* %tmpAddr12, i32** %tmp10
4370.      %tmpWithVal13 = load i32*, i32** %tmp10
4371.      store i32 1, i32* %tmpWithVal13
4372.      %asChar14 = bitcast i32* %tmpWithVal13 to i8*
4373.      %tmp15 = alloca i32*
4374.      %malloccall16 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4375.      %tmpAddr17 = bitcast i8* %malloccall16 to i32*
4376.      store i32* %tmpAddr17, i32** %tmp15
4377.      %tmpWithVal18 = load i32*, i32** %tmp15
4378.      store i32 2, i32* %tmpWithVal18
4379.      %asChar19 = bitcast i32* %tmpWithVal18 to i8*
4380.      %tmp20 = alloca i32*
4381.      %malloccall21 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4382.      %tmpAddr22 = bitcast i8* %malloccall21 to i32*
4383.      store i32* %tmpAddr22, i32** %tmp20
4384.      %tmpWithVal23 = load i32*, i32** %tmp20
4385.      store i32 3, i32* %tmpWithVal23
4386.      %asChar24 = bitcast i32* %tmpWithVal23 to i8*
4387.      %tmp25 = alloca i32*
4388.      %malloccall26 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4389.      %tmpAddr27 = bitcast i8* %malloccall26 to i32*
4390.      store i32* %tmpAddr27, i32** %tmp25
4391.      %tmpWithVal28 = load i32*, i32** %tmp25
4392.      store i32 4, i32* %tmpWithVal28
4393.      %asChar29 = bitcast i32* %tmpWithVal28 to i8*
4394.      %tmp30 = alloca i32*
4395.      %malloccall31 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4396.      %tmpAddr32 = bitcast i8* %malloccall31 to i32*
4397.      store i32* %tmpAddr32, i32** %tmp30

```

```

4398.    %tmpWithVal33 = load i32*, i32** %tmp30
4399.    store i32 5, i32* %tmpWithVal33
4400.    %asChar34 = bitcast i32* %tmpWithVal33 to i8*
4401.    %tmp35 = alloca i32*
4402.    %allocaCall36 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4403.    %tmpAddr37 = bitcast i8* %allocaCall36 to i32*
4404.    store i32* %tmpAddr37, i32** %tmp35
4405.    %tmpWithVal38 = load i32*, i32** %tmp35
4406.    store i32 6, i32* %tmpWithVal38
4407.    %asChar39 = bitcast i32* %tmpWithVal38 to i8*
4408.    %tmp40 = alloca i32*
4409.    %allocaCall41 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4410.    %tmpAddr42 = bitcast i8* %allocaCall41 to i32*
4411.    store i32* %tmpAddr42, i32** %tmp40
4412.    %tmpWithVal43 = load i32*, i32** %tmp40
4413.    store i32 3, i32* %tmpWithVal43
4414.    %asChar44 = bitcast i32* %tmpWithVal43 to i8*
4415.    %tmp45 = alloca i32*
4416.    %allocaCall46 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4417.    %tmpAddr47 = bitcast i8* %allocaCall46 to i32*
4418.    store i32* %tmpAddr47, i32** %tmp45
4419.    %tmpWithVal48 = load i32*, i32** %tmp45
4420.    store i32 8, i32* %tmpWithVal48
4421.    %asChar49 = bitcast i32* %tmpWithVal48 to i8*
4422.    %tmp50 = alloca i32*
4423.    %allocaCall51 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4424.    %tmpAddr52 = bitcast i8* %allocaCall51 to i32*
4425.    store i32* %tmpAddr52, i32** %tmp50
4426.    %tmpWithVal53 = load i32*, i32** %tmp50
4427.    store i32 9, i32* %tmpWithVal53
4428.    %asChar54 = bitcast i32* %tmpWithVal53 to i8*
4429.    %tmp55 = alloca i32*
4430.    %allocaCall56 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4431.    %tmpAddr57 = bitcast i8* %allocaCall56 to i32*
4432.    store i32* %tmpAddr57, i32** %tmp55
4433.    %tmpWithVal58 = load i32*, i32** %tmp55
4434.    store i32 10, i32* %tmpWithVal58
4435.    %asChar59 = bitcast i32* %tmpWithVal58 to i8*
4436.    %tmp60 = alloca i32*

```

```

4437.      %malloccall61 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4438.      %tmpAddr62 = bitcast i8* %malloccall61 to i32*
4439.      store i32* %tmpAddr62, i32** %tmp60
4440.      %tmpWithVal63 = load i32*, i32** %tmp60
4441.      store i32 6, i32* %tmpWithVal63
4442.      %asChar64 = bitcast i32* %tmpWithVal63 to i8*
4443.      %tmp65 = alloca i32*
4444.      %malloccall66 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4445.      %tmpAddr67 = bitcast i8* %malloccall66 to i32*
4446.      store i32* %tmpAddr67, i32** %tmp65
4447.      %tmpWithVal68 = load i32*, i32** %tmp65
4448.      store i32 12, i32* %tmpWithVal68
4449.      %asChar69 = bitcast i32* %tmpWithVal68 to i8*
4450.      %tmp70 = alloca i32*
4451.      %malloccall71 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4452.      %tmpAddr72 = bitcast i8* %malloccall71 to i32*
4453.      store i32* %tmpAddr72, i32** %tmp70
4454.      %tmpWithVal73 = load i32*, i32** %tmp70
4455.      store i32 10, i32* %tmpWithVal73
4456.      %asChar74 = bitcast i32* %tmpWithVal73 to i8*
4457.      %tmp75 = alloca i32*
4458.      %malloccall76 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4459.      %tmpAddr77 = bitcast i8* %malloccall76 to i32*
4460.      store i32* %tmpAddr77, i32** %tmp75
4461.      %tmpWithVal78 = load i32*, i32** %tmp75
4462.      store i32 12, i32* %tmpWithVal78
4463.      %asChar79 = bitcast i32* %tmpWithVal78 to i8*
4464.      %tmp80 = alloca i32*
4465.      %malloccall81 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4466.      %tmpAddr82 = bitcast i8* %malloccall81 to i32*
4467.      store i32* %tmpAddr82, i32** %tmp80
4468.      %tmpWithVal83 = load i32*, i32** %tmp80
4469.      store i32 8, i32* %tmpWithVal83
4470.      %asChar84 = bitcast i32* %tmpWithVal83 to i8*
4471.      %tmp85 = alloca i32*
4472.      %malloccall86 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
      i32* null, i32 1) to i32))
4473.      %tmpAddr87 = bitcast i8* %malloccall86 to i32*
4474.      store i32* %tmpAddr87, i32** %tmp85

```

```

4475.    %tmpWithVal88 = load i32*, i32** %tmp85
4476.    store i32 4, i32* %tmpWithVal88
4477.    %asChar89 = bitcast i32* %tmpWithVal88 to i8*
4478.    %tmp90 = alloca i32*
4479.    %allocaCall91 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4480.    %tmpAddr92 = bitcast i8* %allocaCall91 to i32*
4481.    store i32* %tmpAddr92, i32** %tmp90
4482.    %tmpWithVal93 = load i32*, i32** %tmp90
4483.    store i32 1, i32* %tmpWithVal93
4484.    %asChar94 = bitcast i32* %tmpWithVal93 to i8*
4485.    %tmp95 = alloca i32*
4486.    %allocaCall96 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4487.    %tmpAddr97 = bitcast i8* %allocaCall96 to i32*
4488.    store i32* %tmpAddr97, i32** %tmp95
4489.    %tmpWithVal98 = load i32*, i32** %tmp95
4490.    store i32 5, i32* %tmpWithVal98
4491.    %asChar99 = bitcast i32* %tmpWithVal98 to i8*
4492.    %tmp100 = alloca i32*
4493.    %allocaCall101 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
4494.    %tmpAddr102 = bitcast i8* %allocaCall101 to i32*
4495.    store i32* %tmpAddr102, i32** %tmp100
4496.    %tmpWithVal103 = load i32*, i32** %tmp100
4497.    store i32 19, i32* %tmpWithVal103
4498.    %asChar104 = bitcast i32* %tmpWithVal103 to i8*
4499.    %tmp105 = alloca i32*
4500.    %allocaCall106 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
4501.    %tmpAddr107 = bitcast i8* %allocaCall106 to i32*
4502.    store i32* %tmpAddr107, i32** %tmp105
4503.    %tmpWithVal108 = load i32*, i32** %tmp105
4504.    store i32 2, i32* %tmpWithVal108
4505.    %asChar109 = bitcast i32* %tmpWithVal108 to i8*
4506.    %tmp110 = alloca i32*
4507.    %allocaCall111 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
4508.    %tmpAddr112 = bitcast i8* %allocaCall111 to i32*
4509.    store i32* %tmpAddr112, i32** %tmp110
4510.    %tmpWithVal113 = load i32*, i32** %tmp110
4511.    store i32 21, i32* %tmpWithVal113
4512.    %asChar114 = bitcast i32* %tmpWithVal113 to i8*
4513.    %tmp115 = alloca i32*

```

```

4514.      %malloccall116 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4515.      %tmpAddr117 = bitcast i8* %malloccall116 to i32*
4516.      store i32* %tmpAddr117, i32** %tmp115
4517.      %tmpWithVal118 = load i32*, i32** %tmp115
4518.      store i32 21, i32* %tmpWithVal118
4519.      %asChar119 = bitcast i32* %tmpWithVal118 to i8*
4520.      %tmp120 = alloca i32*
4521.      %malloccall121 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4522.      %tmpAddr122 = bitcast i8* %malloccall121 to i32*
4523.      store i32* %tmpAddr122, i32** %tmp120
4524.      %tmpWithVal123 = load i32*, i32** %tmp120
4525.      store i32 9, i32* %tmpWithVal123
4526.      %asChar124 = bitcast i32* %tmpWithVal123 to i8*
4527.      %tmp125 = alloca i32*
4528.      %malloccall126 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4529.      %tmpAddr127 = bitcast i8* %malloccall126 to i32*
4530.      store i32* %tmpAddr127, i32** %tmp125
4531.      %tmpWithVal128 = load i32*, i32** %tmp125
4532.      store i32 0, i32* %tmpWithVal128
4533.      %asChar129 = bitcast i32* %tmpWithVal128 to i8*
4534.      %tmp130 = alloca i32*
4535.      %malloccall131 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
4536.      %tmpAddr132 = bitcast i8* %malloccall131 to i32*
4537.      store i32* %tmpAddr132, i32** %tmp130
4538.      %tmpWithVal133 = load i32*, i32** %tmp130
4539.      store i32 19, i32* %tmpWithVal133
4540.      %asChar134 = bitcast i32* %tmpWithVal133 to i8*
4541.      %push_back = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr, i8*
      %asChar)
4542.      %push_back135 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar14)
4543.      %push_back136 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar19)
4544.      %push_back137 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar24)
4545.      %push_back138 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar29)
4546.      %push_back139 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar34)

```



```

4547.      %push_back140 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar39)
4548.      %push_back141 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar44)
4549.      %push_back142 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar49)
4550.      %push_back143 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar54)
4551.      %push_back144 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar59)
4552.      %push_back145 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar64)
4553.      %push_back146 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar69)
4554.      %push_back147 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar74)
4555.      %push_back148 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar79)
4556.      %push_back149 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar84)
4557.      %push_back150 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar89)
4558.      %push_back151 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar94)
4559.      %push_back152 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar99)
4560.      %push_back153 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar104)
4561.      %push_back154 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar109)
4562.      %push_back155 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar114)
4563.      %push_back156 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar119)
4564.      %push_back157 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar124)
4565.      %push_back158 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar129)
4566.      %push_back159 = call i32 (%list*, i8*, ...) @push_back(%list* %listAddr,
      i8* %asChar134)
4567.      store %list* %listAddr, %list** %reflector
4568.      %rotor_three_output_direct160 = load %list*, %list**
      %rotor_three_output_direct7

```

```

4569.      %character_location_holder161 = load i32, i32*
          %character_location_holder
4570.      %get_at = call i8* (%list*, i32, ...) @get_at(%list*
          %rotor_three_output_direct160, i32 %character_location_holder161)
4571.      %eltAsPtr = bitcast i8* %get_at to i32*
4572.      %eltDeref = load i32, i32* %eltAsPtr
4573.      %rotor_three_input_direct162 = load %list*, %list**
          %rotor_three_input_direct6
4574.      %character_location_result = call i32 @character_location(%list*
          %rotor_three_input_direct162, i32 %eltDeref)
4575.      store i32 %character_location_result, i32* %character_location_holder
4576.      %rotor_two_output_direct163 = load %list*, %list**
          %rotor_two_output_direct5
4577.      %character_location_holder164 = load i32, i32*
          %character_location_holder
4578.      %get_at165 = call i8* (%list*, i32, ...) @get_at(%list*
          %rotor_two_output_direct163, i32 %character_location_holder164)
4579.      %eltAsPtr166 = bitcast i8* %get_at165 to i32*
4580.      %eltDeref167 = load i32, i32* %eltAsPtr166
4581.      %rotor_two_input_direct168 = load %list*, %list**
          %rotor_two_input_direct4
4582.      %character_location_result169 = call i32 @character_location(%list*
          %rotor_two_input_direct168, i32 %eltDeref167)
4583.      store i32 %character_location_result169, i32*
          %character_location_holder
4584.      %rotor_one_output_direct170 = load %list*, %list**
          %rotor_one_output_direct3
4585.      %character_location_holder171 = load i32, i32*
          %character_location_holder
4586.      %get_at172 = call i8* (%list*, i32, ...) @get_at(%list*
          %rotor_one_output_direct170, i32 %character_location_holder171)
4587.      %eltAsPtr173 = bitcast i8* %get_at172 to i32*
4588.      %eltDeref174 = load i32, i32* %eltAsPtr173
4589.      %rotor_one_input_direct175 = load %list*, %list**
          %rotor_one_input_direct2
4590.      %character_location_result176 = call i32 @character_location(%list*
          %rotor_one_input_direct175, i32 %eltDeref174)
4591.      store i32 %character_location_result176, i32*
          %character_location_holder
4592.      store i32 0, i32* %index_reverse
4593.      br label %while
4594.
4595.      while:                                ; preds = %merge, %entry
4596.      %index_reverse197 = load i32, i32* %index_reverse

```

```

4597.    %tmp198 = icmp slt i32 %index_reverse197, 26
4598.    br i1 %tmp198, label %while_body, label %merge199
4599.
4600.    while_body:                                ; preds = %while
4601.    br label %if
4602.
4603.    if:                                          ; preds = %while_body
4604.    %characher_location_holder192 = load i32, i32*
    %characher_location_holder
4605.    %index_reverse193 = load i32, i32* %index_reverse
4606.    %tmp194 = icmp eq i32 %characher_location_holder192,
    %index_reverse193
4607.    br i1 %tmp194, label %if_body, label %else_body
4608.
4609.    if_body:                                    ; preds = %if
4610.    br label %merge
4611.
4612.    else_body:                                  ; preds = %if
4613.    br label %if177
4614.
4615.    merge:                                      ; preds = %merge179, %if_body
4616.    %index_reverse195 = load i32, i32* %index_reverse
4617.    %tmp196 = add i32 %index_reverse195, 1
4618.    store i32 %tmp196, i32* %index_reverse
4619.    br label %while
4620.
4621.    if177:                                      ; preds = %else_body
4622.    %reflector181 = load %list*, %list** %reflector
4623.    %characher_location_holder182 = load i32, i32*
    %characher_location_holder
4624.    %get_at183 = call i8* (%list*, i32, ...) @get_at(%list* %reflector181, i32
    %characher_location_holder182)
4625.    %eltAsPtr184 = bitcast i8* %get_at183 to i32*
4626.    %eltDeref185 = load i32, i32* %eltAsPtr184
4627.    %reflector186 = load %list*, %list** %reflector
4628.    %index_reverse187 = load i32, i32* %index_reverse
4629.    %get_at188 = call i8* (%list*, i32, ...) @get_at(%list* %reflector186, i32
    %index_reverse187)
4630.    %eltAsPtr189 = bitcast i8* %get_at188 to i32*
4631.    %eltDeref190 = load i32, i32* %eltAsPtr189
4632.    %tmp191 = icmp eq i32 %eltDeref185, %eltDeref190
4633.    br i1 %tmp191, label %if_body178, label %merge179
4634.
4635.    if_body178:                                ; preds = %if177

```

```

4636.    %index_reverse180 = load i32, i32* %index_reverse
4637.    store i32 %index_reverse180, i32* %character_location_holder_two
4638.    br label %merge179
4639.
4640.    merge179:                                ; preds = %if177, %if_body178
4641.    br label %merge
4642.
4643.    merge199:                                ; preds = %while
4644.    %rotor_one_input_direct200 = load %list*, %list**
    %rotor_one_input_direct2
4645.    %character_location_holder_two201 = load i32, i32*
    %character_location_holder_two
4646.    %get_at202 = call i8* (%list*, i32, ...) @get_at(%list*
    %rotor_one_input_direct200, i32 %character_location_holder_two201)
4647.    %eltAsPtr203 = bitcast i8* %get_at202 to i32*
4648.    %eltDeref204 = load i32, i32* %eltAsPtr203
4649.    %rotor_one_output_direct205 = load %list*, %list**
    %rotor_one_output_direct3
4650.    %character_location_result206 = call i32 @character_location(%list*
    %rotor_one_output_direct205, i32 %eltDeref204)
4651.    store i32 %character_location_result206, i32*
    %character_location_holder
4652.    %rotor_two_input_direct207 = load %list*, %list**
    %rotor_two_input_direct4
4653.    %character_location_holder208 = load i32, i32*
    %character_location_holder
4654.    %get_at209 = call i8* (%list*, i32, ...) @get_at(%list*
    %rotor_two_input_direct207, i32 %character_location_holder208)
4655.    %eltAsPtr210 = bitcast i8* %get_at209 to i32*
4656.    %eltDeref211 = load i32, i32* %eltAsPtr210
4657.    %rotor_two_output_direct212 = load %list*, %list**
    %rotor_two_output_direct5
4658.    %character_location_result213 = call i32 @character_location(%list*
    %rotor_two_output_direct212, i32 %eltDeref211)
4659.    store i32 %character_location_result213, i32*
    %character_location_holder
4660.    %rotor_three_input_direct214 = load %list*, %list**
    %rotor_three_input_direct6
4661.    %character_location_holder215 = load i32, i32*
    %character_location_holder
4662.    %get_at216 = call i8* (%list*, i32, ...) @get_at(%list*
    %rotor_three_input_direct214, i32 %character_location_holder215)
4663.    %eltAsPtr217 = bitcast i8* %get_at216 to i32*
4664.    %eltDeref218 = load i32, i32* %eltAsPtr217

```

```

4665.      %rotor_three_output_direct219 = load %list*, %list**
          %rotor_three_output_direct7
4666.      %character_location_result220 = call i32 @character_location(%list*
          %rotor_three_output_direct219, i32 %eltDeref218)
4667.      store i32 %character_location_result220, i32*
          %character_location_holder
4668.      %character_location_holder221 = load i32, i32*
          %character_location_holder
4669.      ret i32 %character_location_holder221
4670.      }
4671.
4672.      define i32 @my_modulus_result(i32 %base) {
4673.      entry:
4674.      %base1 = alloca i32
4675.      store i32 %base, i32* %base1
4676.      %modulus_results = alloca i32
4677.      %base2 = load i32, i32* %base1
4678.      %modulus_operation = call i32 (i32, i32, ...) @modulus_operation(i32
          %base2, i32 26)
4679.      store i32 %modulus_operation, i32* %modulus_results
4680.      %modulus_results3 = load i32, i32* %modulus_results
4681.      ret i32 %modulus_results3
4682.      }
4683.
4684.      define i32 @character_location_status(%list* %xx, i32 %yy) {
4685.      entry:
4686.      %xx1 = alloca %list*
4687.      store %list* %xx, %list** %xx1
4688.      %yy2 = alloca i32
4689.      store i32 %yy, i32* %yy2
4690.      %my_response = alloca i32
4691.      %yy3 = load i32, i32* %yy2
4692.      %xx4 = load %list*, %list** %xx1
4693.      %character_location_result = call i32 @character_location(%list* %xx4,
          i32 %yy3)
4694.      store i32 %character_location_result, i32* %my_response
4695.      %my_response5 = load i32, i32* %my_response
4696.      ret i32 %my_response5
4697.      }
4698.
4699.      define %list* @initialize_input_text(%list* %message_user, %list*
          %input_list_message) {
4700.      entry:
4701.      %message_user1 = alloca %list*

```

```

4702.    store %list* %message_user, %list** %message_user1
4703.    %input_list_message2 = alloca %list*
4704.    store %list* %input_list_message, %list** %input_list_message2
4705.    %message_length_input = alloca i32
4706.    %message_length_init = alloca i32
4707.    %current_int_val = alloca i32
4708.    %message_user3 = load %list*, %list** %message_user1
4709.    %stringLength = call i32 @stringLength(%list* %message_user3)
4710.    store i32 %stringLength, i32* %message_length_input
4711.    store i32 0, i32* %message_length_init
4712.    store i32 0, i32* %current_int_val
4713.    store i32 0, i32* %message_length_init
4714.    br label %while
4715.
4716.    while:                                ; preds = %while_body, %entry
4717.    %message_length_init11 = load i32, i32* %message_length_init
4718.    %message_length_input12 = load i32, i32* %message_length_input
4719.    %tmp13 = icmp slt i32 %message_length_init11,
    %message_length_input12
4720.    br i1 %tmp13, label %while_body, label %merge
4721.
4722.    while_body:                            ; preds = %while
4723.    %message_length_init4 = load i32, i32* %message_length_init
4724.    %message_user5 = load %list*, %list** %message_user1
4725.    %return_cor_int_result = call i32 @return_cor_int(%list*
    %message_user5, i32 %message_length_init4)
4726.    store i32 %return_cor_int_result, i32* %current_int_val
4727.    %input_list_message6 = load %list*, %list** %input_list_message2
4728.    %message_length_init7 = load i32, i32* %message_length_init
4729.    %current_int_val8 = load i32, i32* %current_int_val
4730.    %tmp = alloca i32*
4731.    %mallocall = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4732.    %tmpAddr = bitcast i8* %mallocall to i32*
4733.    store i32* %tmpAddr, i32** %tmp
4734.    %tmpWithVal = load i32*, i32** %tmp
4735.    store i32 %current_int_val8, i32* %tmpWithVal
4736.    %asChar = bitcast i32* %tmpWithVal to i8*
4737.    %set_at = call %list* (%list*, i32, i8*, ...) @set_at(%list*
    %input_list_message6, i32 %message_length_init7, i8* %asChar)
4738.    %message_length_init9 = load i32, i32* %message_length_init
4739.    %tmp10 = add i32 %message_length_init9, 1
4740.    store i32 %tmp10, i32* %message_length_init
4741.    br label %while

```

```

4742.
4743. merge:                                ; preds = %while
4744.   %input_list_message14 = load %list*, %list** %input_list_message2
4745.   ret %list* %input_list_message14
4746. }
4747.
4748. define i32 @return_cor_int(%list* %what_message, i32 %what_index) {
4749. entry:
4750.   %what_message1 = alloca %list*
4751.   store %list* %what_message, %list** %what_message1
4752.   %what_index2 = alloca i32
4753.   store i32 %what_index, i32* %what_index2
4754.   %res = alloca i32
4755.   %what_index3 = load i32, i32* %what_index2
4756.   %what_message4 = load %list*, %list** %what_message1
4757.   %corresponding_int = call i32 @corresponding_int(%list*
      %what_message4, i32 %what_index3)
4758.   store i32 %corresponding_int, i32* %res
4759.   %res5 = load i32, i32* %res
4760.   ret i32 %res5
4761. }
4762.
4763. define %list* @rotor_shift(%list* %rotor_input, i32 %position) {
4764. entry:
4765.   %rotor_input1 = alloca %list*
4766.   store %list* %rotor_input, %list** %rotor_input1
4767.   %position2 = alloca i32
4768.   store i32 %position, i32* %position2
4769.   %character_holder = alloca i32
4770.   %k = alloca i32
4771.   %l = alloca i32
4772.   %m = alloca i32
4773.   %rotor_input3 = load %list*, %list** %rotor_input1
4774.   %get_at = call i8* (%list*, i32, ...) @get_at(%list* %rotor_input3, i32 0)
4775.   %eltAsPtr = bitcast i8* %get_at to i32*
4776.   %eltDeref = load i32, i32* %eltAsPtr
4777.   store i32 %eltDeref, i32* %character_holder
4778.   store i32 0, i32* %k
4779.   store i32 0, i32* %l
4780.   store i32 0, i32* %m
4781.   store i32 0, i32* %k
4782.   br label %while
4783.
4784. while:                                ; preds = %merge33, %entry

```

```

4785.    %k36 = load i32, i32* %k
4786.    %position37 = load i32, i32* %position2
4787.    %tmp38 = icmp slt i32 %k36, %position37
4788.    br i1 %tmp38, label %while_body, label %merge39
4789.
4790.    while_body:                                ; preds = %while
4791.    %rotor_input4 = load %list*, %list** %rotor_input1
4792.    %get_at5 = call i8* (%list*, i32, ...) @get_at(%list* %rotor_input4, i32 0)
4793.    %eltAsPtr6 = bitcast i8* %get_at5 to i32*
4794.    %eltDeref7 = load i32, i32* %eltAsPtr6
4795.    store i32 %eltDeref7, i32* %character_holder
4796.    store i32 0, i32* %l
4797.    br label %while8
4798.
4799.    while8:                                    ; preds = %merge, %while_body
4800.    %l31 = load i32, i32* %l
4801.    %tmp32 = icmp slt i32 %l31, 25
4802.    br i1 %tmp32, label %while_body9, label %merge33
4803.
4804.    while_body9:                              ; preds = %while8
4805.    %l10 = load i32, i32* %l
4806.    %tmp = add i32 %l10, 1
4807.    store i32 %tmp, i32* %m
4808.    %rotor_input11 = load %list*, %list** %rotor_input1
4809.    %l12 = load i32, i32* %l
4810.    %rotor_input13 = load %list*, %list** %rotor_input1
4811.    %m14 = load i32, i32* %m
4812.    %get_at15 = call i8* (%list*, i32, ...) @get_at(%list* %rotor_input13, i32
    %m14)
4813.    %eltAsPtr16 = bitcast i8* %get_at15 to i32*
4814.    %eltDeref17 = load i32, i32* %eltAsPtr16
4815.    %tmp18 = alloca i32*
4816.    %allocaall = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4817.    %tmpAddr = bitcast i8* %allocaall to i32*
4818.    store i32* %tmpAddr, i32** %tmp18
4819.    %tmpWithVal = load i32*, i32** %tmp18
4820.    store i32 %eltDeref17, i32* %tmpWithVal
4821.    %asChar = bitcast i32* %tmpWithVal to i8*
4822.    %set_at = call %list* (%list*, i32, i8*, ...) @set_at(%list* %rotor_input11,
    i32 %l12, i8* %asChar)
4823.    br label %if
4824.
4825.    if:                                        ; preds = %while_body9

```



```

4826.    %l27 = load i32, i32* %l
4827.    %tmp28 = icmp eq i32 %l27, 24
4828.    br i1 %tmp28, label %if_body, label %merge
4829.
4830.    if_body:                                ; preds = %if
4831.    %rotor_input19 = load %list*, %list** %rotor_input1
4832.    %character_holder20 = load i32, i32* %character_holder
4833.    %tmp21 = alloca i32*
4834.    %mallocall22 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32,
    i32* null, i32 1) to i32))
4835.    %tmpAddr23 = bitcast i8* %mallocall22 to i32*
4836.    store i32* %tmpAddr23, i32** %tmp21
4837.    %tmpWithVal24 = load i32*, i32** %tmp21
4838.    store i32 %character_holder20, i32* %tmpWithVal24
4839.    %asChar25 = bitcast i32* %tmpWithVal24 to i8*
4840.    %set_at26 = call %list* (%list*, i32, i8*, ...) @set_at(%list*
    %rotor_input19, i32 25, i8* %asChar25)
4841.    br label %merge
4842.
4843.    merge:                                  ; preds = %if, %if_body
4844.    %l29 = load i32, i32* %l
4845.    %tmp30 = add i32 %l29, 1
4846.    store i32 %tmp30, i32* %l
4847.    br label %while8
4848.
4849.    merge33:                                ; preds = %while8
4850.    %k34 = load i32, i32* %k
4851.    %tmp35 = add i32 %k34, 1
4852.    store i32 %tmp35, i32* %k
4853.    br label %while
4854.
4855.    merge39:                                ; preds = %while
4856.    %rotor_input40 = load %list*, %list** %rotor_input1
4857.    ret %list* %rotor_input40
4858.    }
4859.
4860.    define i32 @character_location(%list* %x, i32 %y) {
4861.    entry:
4862.    %x1 = alloca %list*
4863.    store %list* %x, %list** %x1
4864.    %y2 = alloca i32
4865.    store i32 %y, i32* %y2
4866.    %j = alloca i32
4867.    %i = alloca i32

```

```

4868.    store i32 0, i32* %j
4869.    store i32 0, i32* %i
4870.    store i32 0, i32* %i
4871.    br label %while
4872.
4873.    while:                                ; preds = %merge, %entry
4874.    %i9 = load i32, i32* %i
4875.    %tmp10 = icmp slt i32 %i9, 26
4876.    br i1 %tmp10, label %while_body, label %merge11
4877.
4878.    while_body:                            ; preds = %while
4879.    br label %if
4880.
4881.    if:                                    ; preds = %while_body
4882.    %x4 = load %list*, %list** %x1
4883.    %i5 = load i32, i32* %i
4884.    %get_at = call i8* (%list*, i32, ...) @get_at(%list* %x4, i32 %i5)
4885.    %eltAsPtr = bitcast i8* %get_at to i32*
4886.    %eltDeref = load i32, i32* %eltAsPtr
4887.    %y6 = load i32, i32* %y2
4888.    %tmp = icmp eq i32 %eltDeref, %y6
4889.    br i1 %tmp, label %if_body, label %merge
4890.
4891.    if_body:                               ; preds = %if
4892.    %i3 = load i32, i32* %i
4893.    ret i32 %i3
4894.
4895.    merge:                                 ; preds = %if
4896.    %i7 = load i32, i32* %i
4897.    %tmp8 = add i32 %i7, 1
4898.    store i32 %tmp8, i32* %i
4899.    br label %while
4900.
4901.    merge11:                              ; preds = %while
4902.    %j12 = load i32, i32* %j
4903.    ret i32 %j12
4904.    }
4905.
4906.    declare noalias i8* @malloc(i32)

```

Chapter 8

Makefile

```
CC = cc
CFLAGS = -g -Wall
LIBS = -lm -lncurses

.PHONY : dune
dune:
    dune build src/smap.exe

.PHONY : dunehello
dunehello :
    dune exec src/smap.exe test/hello.smap
# calc : parser.cmo scanner.cmo calc.cmo
#   ocamlc -w A -o calc $^

parser : parser.cmo scanner.cmo

%.cmo : %.ml
    ocamlc -w A -c $<

%.cmi : %.mli
    ocamlc -w A -c $<

scanner.ml : scanner.mll
    ocamllex $^

parser.ml parser.mli : parser.mly
    ocamlyacc $^

# Depedencies from ocamldep
# calc.cmo : scanner.cmo parser.cmi ast.cmi
# calc.cmx : scanner.cmx parser.cmx ast.cmi
parser.cmo : ast.cmi parser.cmi
parser.cmx : ast.cmi parser.cmi
scanner.cmo : parser.cmi
scanner.cmx : parser.cmx

#####
```

```

.PHONY : clean
clean :
    rm -rf *.cmi *.cmo parser.ml parser.mli scanner.ml calc.out calc *.o *.dSYM; make
clean2

.PHONY : smap
smap : smap.native

# Everything below the line was copied from the microc makefile
# I changed the make clean rule to a make clean2 rule
# so it wouldn't clash with our current make clean rule
# I also changed the microc.native rule to be smap.native

##### microc makefile stuff #####

# "make test" Compiles everything and runs the regression tests
.PHONY : testhello
testhello :
    ./testhello.sh

.PHONY : test
test : all testall.sh
    ./testall.sh

# "make all" builds the executable as well as the "printbig" library designed
# to test linking external code

.PHONY : all
all : smap.native prob.o list.o testMakeStruct.o polymorphicPrint.o c_deps

# "make microc.native" compiles the compiler
#
# The _tags file controls the operation of ocamlbuild, e.g., by including
# packages, enabling warnings
#
# See https://github.com/ocaml/ocamlbuild/blob/master/manual/manual.adoc

smap.native :
    opam config exec -- \
    ocamlbuild -use-ocamlfind src/smap.native

```

```

# "make clean" removes all generated files

.PHONY : clean2
clean2 :
    ocamlbuild -clean
    rm -rf testall.log ocamlllvm *.diff

c_deps: list.o prob.o

c_tests: prob list

# Testing the "printbig" example
list.o:
    $(CC) $(CFLAGS) -c -o list.o runtime/list.c $(LIBS)

list: list.o runtime/list.c
    $(CC) $(CFLAGS) -o list -DBUILD_TEST $(LIBS)

prob.o:
    $(CC) $(CFLAGS) -c -o prob.o runtime/prob.c $(LIBS)

prob: list.o runtime/prob.c
    $(CC) $(CFLAGS) list.o -o prob -DBUILD_TEST runtime/prob.c $(LIBS)

printbig : printbig.c
    cc -o printbig -DBUILD_TEST printbig.c

polymorphicPrint.o :
    $(CC) $(CFLAGS) -c -o polymorphicPrint.o runtime/polymorphicPrint.c

testMakeStruct.o :
    $(CC) $(CFLAGS) -c -o testMakeStruct.o runtime/testMakeStruct.c

ncurses.o :
    $(CC) $(CFLAGS) -c -o ncurses.o runtime/ncurses.c

# Building the tarball

TESTS = \
    add1 arith1 arith2 arith3 fib float1 float2 float3 for1 for2 func1 \
    func2 func3 func4 func5 func6 func7 func8 func9 gcd2 gcd global1 \
    global2 global3 hello if1 if2 if3 if4 if5 if6 local1 local2 ops1 \

```

```

ops2 printbig var1 var2 while1 while2

FAILS = \
assign1 assign2 assign3 dead1 dead2 expr1 expr2 expr3 float1 float2 \
for1 for2 for3 for4 for5 func1 func2 func3 func4 func5 func6 func7 \
func8 func9 global1 global2 if1 if2 if3 nomain printbig printb print \
return1 return2 while1 while2

TESTFILES = $(TESTS:%=test-%.mc) $(TESTS:%=test-%.out) \
             $(FAILS:%=fail-%.mc) $(FAILS:%=fail-%.err)

TARFILES = ast.ml sast.ml codegen.ml Makefile _tags smap.ml microcparse.mly \
           README scanner.mll semant.ml testall.sh \
           printbig.c arcade-font.pbm font2c \
           Dockerfile \
           $(TESTFILES:%=tests/%)

microc.tar.gz : $(TARFILES)
cd .. && tar czf microc/microc.tar.gz \
           $(TARFILES:%=microc/%)

```

scanner.mll

1. { open Parser }
2. let letter = ['a'-'z' 'A'-'Z']
3. let digit = ['0'-'9']
4. let punc = [',' ';' ':' '!' '?' '-' '\' '(' ')' '?']
5. let otherChar = ['@' '#' '\$' '%' '^' '|' '^' '&' '*' '~' '[' ']' '{' '}' '\\' '/' '+' '-' '*' '_' '>' '<' '=']
6. let whitespace = [' ' '\t' '\r' '\n']
- 7.
8. rule tokenize = parse
9. (* scoping *)
10. | '[' { LBRACKET }
11. | ']' { RBRACKET }
12. | '{' { LBRACE }
13. | '}' { RBRACE }
14. | '(' { LPAREN }
15. | ')' { RPAREN }

- 16.
17. (* operators *)
 18. | '+' { PLUS }
 19. | '-' { MINUS }
 20. | '*' { TIMES }
 21. | '/' { DIVIDE }
 22. | '=' { ASSIGN }
 23. | "+=" { ADDEQUAL }
 24. | "-=" { MINUSEQUAL }
 25. | "*=" { TIMESEQUAL }
 26. | "/=" { DIVEQUAL }
 27. | ">>" { RSHIFT }
 28. | "<<" { LSHIFT }
 29. | '~' { BITNOT }
 30. | '&' { BITAND }
 31. | '|' { BITOR }
 32. | '^' { XOR }
 33. | "++" { CONCAT }
 34. | "[<]" { ADDHEAD }
 35. | "[>]" { ADDTAIL }
- 36.
- 37.
38. (* statement symbols *)
 39. | ';' { SEMICOLON }
 40. | ':' { PROBCOLON }
 41. | ',' { COMMA }
- 42.
43. (* SMAP types *)
 44. | "list" { LIST }
 45. | "prob" { PROB }
- 46.
47. (* loops *)
 48. | "while" { WHILE }
 49. | "for" { FOR }
- 50.
51. (* primitive types *)
 52. | "int" { INT }
 53. | "char" { CHAR }
 54. | "bool" { BOOL }
 55. | "string" { STRING }

56. | "float" { FLOAT }

57. | "void" { VOID }

58.

59. (* function declaration *)

60. | "fn" { FUNCTION }

61.

62. (* comments *)

63. | "/*" { comment lexbuf }

64. | "//" { lineComment lexbuf }

65.

66. (* whitespace *)

67. | whitespace { tokenize lexbuf }

68.

69. (* comparison operators *)

70. | "==" { COMPEQ }

71. | '<' { COMPLT }

72. | "<=" { COMPLEQ }

73. | ">" { COMPGT }

74. | ">=" { COMPGEQ }

75. | "!=" { COMPNEQ }

76. | "&&" { AND }

77. | "||" { OR }

78. | '!' { NOT }

79.

80. (* Accessor *)

81. (* if we put dot and length together, "length" doesn't have to be a keyword!*)

82. | ".length" { LENGTH }

83. | '#' { OCTOTHORPE }

84.

85. (* control flow *)

86. | "if" { IF }

87. | "elif" { ELIF }

88. | "else" { ELSE }

89. | "switch" { SWITCH }

90. | "case" { CASE }

91. | "default" { DEFAULT }

92. | "break" { BREAK }

93. | "continue" { CONTINUE }

94. | "return" { RETURN }

95.


```

96. (* literals *)
97. | \" \\ ' \n' \"          { CHAR_LIT('\n')}
98. | [\" ] ((letter | digit | punc | [\" ] | otherChar | whitespace ) as c)[\" ] { CHAR_LIT(c) }
    (* char literal *)
99. | [\" ] ((letter | digit | punc | [\" ] | otherChar | whitespace)* as s)[\" ] { STRING_LIT(s) }
    (* string literal *)
100. | [0'-9']+ as lit          { INT_LIT(int_of_string lit) } (* int
    literal *)
101. | [0'-9']* [.] [0'-9']+ as flit          { FLOAT_LIT(float_of_string flit)} (*
    float literal *)
102. | "true"                  { BOOL_LIT(true) }
103. | "false"                 { BOOL_LIT(false) }
104.
105. (* identifiers *)
106. | [a'-z' 'A'-'Z' ' '_' ][a'-z' 'A'-'Z' '0'-'9' ' '_' ]* as var_id { ID(var_id) }
107.
108. (* EOF *)
109. | eof                      { EOF }
110. | _ as ch                  { raise (Failure("illegal character " ^ Char.escaped ch)) }
111.
112. and lineComment =
113.   parse '\n'                { tokenize lexbuf }
114.   | _                       { lineComment lexbuf }
115.
116. and comment =
117.   parse "*/"                { tokenize lexbuf }
118.   | _                       { comment lexbuf }

```

ast.ml

```

type binary_op = Add | Sub | Mul | Div | Concat | CompEq | CompGeq | CompLt | CompLeq
| CompGt | CompNeq | RShift | LShift | BitAnd | BitOr | Xor
type unary_op = BitNot | Not | Bang | Octothorpe | Neg
type assign_op = PlusEqual | MinusEqual | TimesEqual | DivEqual | Equal

type typ = Int | Bool | Float | Void | Char | String | Prob | List
type typ_name = typ list

type bind = typ_name * string

type expr =

```

```

    Binop of expr * binary_op * expr
|   Unop of unary_op * expr
|   Assign of expr * assign_op * expr
|   Int_lit of int
|   Bool_lit of bool
|   Float_lit of float
|   Char_lit of char
|   String_lit of string
|   List_lit of expr list
|   Id of string
|   Cast of typ_name * expr
|   ListElement of string * expr * expr list
|   ListAddHead of expr * expr
|   ListAddTail of expr * expr
|   Length of expr
|   ProbColon of expr * expr
|   FunCall of string * expr list
|   Index of expr
|   Noexpr

type vdecl =
    Vdecl of bind * expr

type stmt =
    Block of stmt list
|   Expr of expr
|   Return of expr
|   If of expr * stmt
|   If_Else of stmt * stmt
|   If_Elif of stmt * stmt * stmt list
|   If_Elif_Else of stmt * stmt * stmt list * stmt
|   For of expr * expr * expr * stmt
|   While of expr * stmt
|   Break
|   Continue
|   Elif of expr * stmt (*only use inside an if statement, never alone!*)

type func_decl = { typ_name : typ_name;
                  fname : string;
                  formals : bind list;
                  locals : vdecl list;
                  body : stmt list;

```

```

    }

type program = vdecl list * func_decl list

(* dummy pretty printer*)
(*          (vars, funcs)                                *)
let string_of_program (_,_) = "someday I'll pretty print the AST! \n"

let string_of_uop = function
  Neg -> "-"
| Not -> "!"
| BitNot -> "~"
| Bang -> "!"
| Octothorpe -> "#"

let string_of_ttyp = function
  Int -> "Int"
| Bool -> "Bool"
| Float -> "Float"
| Char -> "Char"
| String -> "String"
| List -> "List"
| Void -> "Voerid"
| Prob -> "Prob"

let string_of_op = function
  Add -> "+"
| Sub -> "-"
| Mul -> "*"
| Div -> "/"
| Concat -> "++"
| CompEq -> "=="
| CompLt -> "<"
| CompNeq -> "!="
| CompLeq -> "<="
| CompGt -> ">"
| CompGeq -> ">="
| RShift -> ">>"
| LShift -> "<<"
| BitAnd -> "&"
| BitOr -> "|"

```

```

| Xor -> "^"

let string_of_assign = function
  PlusEqual -> "+="
| MinusEqual -> "-="
| TimesEqual -> "*="
| DivEqual -> "/="
| Equal -> "="

(* !NON EXHUASTIVE PATTERN MATCHING! *)
let rec string_of_expr = function
  Int_lit(l) -> string_of_int l
| Float_lit(l) -> string_of_float l (*check floats again*)
| Bool_lit(true) -> "true"
| Bool_lit(false) -> "false"
| Id(s) -> s
| Binop(e1, o, e2) ->
  string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
| Unop(o, e) -> string_of_uop o ^ string_of_expr e
| Assign(v, o, e) -> string_of_expr v ^ " " ^ string_of_assign o ^ " " ^
string_of_expr e
| FunCall(f, e1) ->
  f ^ "(" ^ String.concat ", " (List.map string_of_expr e1) ^ ")"
| Noexpr -> ""
| _ -> "can't print this expr yet"

(* !NON EXHUASTIVE PATTERN MATCHING! *)
let rec string_of_stmt = function
  Block(stmts) ->
  "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
| Expr(expr) -> string_of_expr expr ^ ";\n";
| Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
(*| If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
| If(e, s1, s2) -> "if (" ^ string_of_expr e ^ ")\n" ^
  string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2*)
| For(e1, e2, e3, s) ->
  "for (" ^ string_of_expr e1 ^ " ; " ^ string_of_expr e2 ^ " ; " ^
  string_of_expr e3 ^ ") " ^ string_of_stmt s
| While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s

let string_of_typ_name typ_lst =
List.fold_left (fun acc elt -> acc^elt^" ") "" (List.map string_of_typ typ_lst)

```

parser.mly

1. %{ open Ast % }
- 2.
3. /* TOKENS

*****/
- 4.
5. %token EOF ASSIGN SEMICOLON PROBCOLON COMMA IF ELIF ELSE
6. // scoping
7. %token LBRACKET RBRACKET LBRACE RBRACE LPAREN RPAREN
- 8.
9. // operators
10. %token BITNOT BITAND BITOR XOR CONCAT
11. %token PLUS MINUS TIMES DIVIDE RSHIFT LSHIFT
12. %token ADDEQUAL MINUSEQUAL TIMESEQUAL DIVEQUAL WHILE FOR
13. %token AND OR NOT LENGTH OCTOTHORPE
14. %token ADDHEAD ADDTAIL
- 15.
16. // primitives
17. %token INT CHAR BOOL STRING FLOAT PROB LIST VOID RETURN
- 18.
19. %token SWITCH CASE DEFAULT BREAK CONTINUE FUNCTION
- 20.
21. //comparison
22. %token COMPEQ COMPLT COMPLEQ COMPGT COMPGEQ COMPNEQ
- 23.
24. //literals
25. %token <bool> BOOL_LIT
26. %token <int> INT_LIT
27. %token <char> CHAR_LIT
28. %token <float> FLOAT_LIT
29. %token <string> STRING_LIT
30. //identifier
31. %token <string> ID
- 32.
- 33.

34. /* OPERATOR PRECEDENCE

*****/

35.

36. %left SEMICOLON

37. %right ASSIGN PROBCOLON

38.

39.

40. %left CONCAT

41. %left PLUS MINUS

42. %left TIMES DIVIDE

43.

44. %left ADDEQUAL MINUSEQUAL TIMESEQUAL DIVEQUAL COMPEQ COMPLT
COMPEQ RSHIFT LSHIFT

45. %left COMPGT COMPGEQ COMPNEQ

46. %left BITOR BITAND BITNOT XOR

47. %left OCTOTHORPE

48.

49. %left COMMA LENGTH

50.

51. %nonassoc ELIF

52. %nonassoc ELSE

53. %nonassoc CAST

54. %right NOT

55.

56. %start program

57. %type <Ast.program> program

58.

59. %%

60.

61. /* EXPRESSIONS

*****/

62. expr:

63. /* binary expressions */

64. expr PLUS expr { Binop(\$1, Add, \$3) }

65. | expr MINUS expr { Binop(\$1, Sub, \$3) }

66. | expr TIMES expr { Binop(\$1, Mul, \$3) }

67. | expr DIVIDE expr { Binop(\$1, Div, \$3) }

68. | expr COMPEQ expr { Binop(\$1, CompEq, \$3) }

```

69. | expr COMPLT expr      { Binop($1, CompLt, $3)  }
70. | expr COMPLEQ expr   { Binop($1, CompLeq, $3) }
71. | expr COMPGT expr    { Binop($1, CompGt, $3)  }
72. | expr COMPGEQ expr   { Binop($1, CompEq, $3)  }
73. | expr COMPNEQ expr   { Binop($1, CompNeq, $3) }
74. | expr RSHIFT expr    { Binop($1, RShift, $3)  }
75. | expr LSHIFT expr    { Binop($1, LShift, $3)  }
76. | expr BITAND expr     { Binop($1, BitAnd, $3)  }
77. | expr BITOR expr     { Binop($1, BitOr, $3)   }
78. | expr XOR expr       { Binop($1, Xor, $3)    }
79. /* assignment expressions (special case of binary) */
80. | expr ASSIGN expr     { Assign($1, Equal, $3)  }
81. | expr ADDEQUAL expr  { Assign($1, PlusEqual,$3) }
82. | expr MINUSEQUAL expr { Assign($1, MinusEqual, $3) }
83. | expr TIMESEQUAL expr { Assign($1, TimesEqual, $3) }
84. | expr DIVEQUAL expr  { Assign($1, DivEqual, $3) }
85. /* unary expressions */
86. | MINUS expr          { Unop(Neg, $2)        } /* what is our syntax for
   negative numbers? */
87. | BITNOT expr        { Unop(BitNot, $2)     }
88. | NOT expr           { Unop(Not, $2)       }
89. | expr NOT           { Unop(Bang, $1)      }
90. | expr OCTOTHORPE    { Unop(Octothorpe, $1)  }
91. | LPAREN typ_decl RPAREN expr %prec CAST { Cast($2, $4)      } /* added
   this in for now. Hoping precedence is correct*/
92. /* literal expressions */
93. | INT_LIT            { Int_lit ($1)        }
94. | BOOL_LIT          { Bool_lit($1)        }
95. | STRING_LIT        { String_lit($1)      }
96. | FLOAT_LIT         { Float_lit ($1)      }
97. | CHAR_LIT          { Char_lit($1)        }
98. /* identifiers */
99. | id                 { $1                }
100. /* List expressions */
101. | LBRACKET expr_list_opt RBRACKET      { List_lit(List.rev $2)
   } /* list literal */
102. | id ADDHEAD ASSIGN expr              { ListAddHead($1,$4)  } /* add
   head */
103. | id ADDTAIL ASSIGN expr              { ListAddTail($1,$4)  } /* add
   tail */

```

```

104. | expr CONCAT expr { Binop($1, Concat, $3) }
105. | expr LENGTH { Length($1) } /* also for prob
types */
106. /* prob type expressions */
107. | expr PROBCOLON expr { ProbColon($1,$3) }
108. /*function call expression */
109. | ID LPAREN args_opt RPAREN { FunCall ($1,$3) }
110. /* parentheses around an expression */
111. | LPAREN expr RPAREN { $2 }
112.
113. /* identifier is either an indexed list element or regular text identifier */
114. id:
115. ID { Id($1) }
116. | list_elt { $1 }
117.
118. /* indexing into a list */
119. list_elt: ID index index_list_opt { ListElement($1,$2,$3) } /* indexed elt (for ex
a[0][0][1]) */
120. index: LBRACKET expr RBRACKET { Index($2) } /* a single index
(for ex. [0]) */
121. index_list_opt:
122. { [] } /* 0 or more indices (for ex. [1][2]) */
123. | index index_list_opt { $1::$2 }
124.
125. /* recursive expressions to handle plurals */
126. expr_opt: /* zero or one expressions */
127. { Noexpr }
128. | expr { $1 }
129.
130. expr_list_opt: /* zero or more expressions */
131. { [] }
132. | expr_list { $1 }
133.
134. expr_list: /* one or more expressions */
135. expr { [$1] }
136. | expr_list COMMA expr { $3 :: $1 }
137.
138. args_opt:
139. { [] } /* zero or more arguments */
140. | args_list { List.rev $1 }

```



```

141.
142.     args_list:
143.         expr          { [$1]      } /* one argument      */
144.     | args_list COMMA expr { $3 :: $1  } /* more than one argument */
145.
146.     /* STATEMENTS
*****
*****/
147.
148.     stmt: /* statements end with either a semicolon or a block */
149.         expr SEMICOLON          { Expr($1)      }
150.     | BREAK SEMICOLON          { Break
    }
151.     | CONTINUE SEMICOLON      { Continue
    }
152.     | RETURN expr_opt SEMICOLON {
    Return($2)  }
153.     | if_stmt                  { $1          }
154.     | if_stmt ELSE block       { If_Else($1,$3) }
155.     | if_stmt elif elif_list   { If_Elif($1,$2,$3) }
156.     | if_stmt elif elif_list ELSE block {
    If_Elif_Else($1,$2,$3,$5)}
157.     | FOR LPAREN expr_opt SEMICOLON expr_opt SEMICOLON expr_opt
    RPAREN block      { For($3,$5,$7,$9) }
158.     | WHILE LPAREN expr RPAREN block {
    While($3,$5)  }
159.
160.     if_stmt: IF LPAREN expr RPAREN block {
    If($3,$5)  }
161.     elif:  ELIF LPAREN expr RPAREN block {
    Elif($3,$5)  }
162.     block: LBRACE stmt_list RBRACE {
    Block(List.rev $2)}
163.
164.     stmt_list:
165.         /* nothing */ { [] } /* zero or more statements */
166.     | stmt_list stmt { $2::$1 }
167.
168.     elif_list:
169.         /*nothing*/ { [] } /* zero or more elif statements*/

```

```

170.     | elif elif_list    { $1::$2 }
171.
172.
173.     program: decls EOF { $1 }           /* a program consists of declarations and
the EOF token*/
174.     decls:
175.         /* nothing */ { ([], []) }
176.     | decls vdecl { (($2 :: fst $1), snd $1) } /* global variable declarations */
177.     | decls fdecl { (fst $1, ($2 :: snd $1)) } /* function definitions      */
178.
179.
180.     /* FUNCTION DECLARATIONS
*****
*****/
181.     fdecl: typ_decl ID LPAREN formals_opt RPAREN /* are we going to use the
function keyword??*/
182.     LBRACE vdecl_list stmt_list RBRACE
183.     {{ typ_name = $1;
184.        fname = $2;
185.        formals = List.rev $4;
186.        locals = List.rev $7;
187.        body = List.rev $8;
188.     }}
189.
190.     formals_opt:
191.     /* nothing */ { [] }                /* zero or more formal parameters
*/
192.     | formal_list { $1 }
193.
194.     formal_list:
195.     typ_decl ID { [($1,$2)] } /* formal param consists of type
declaration and identifier */
196.     | formal_list COMMA typ_decl ID { ($3,$4) :: $1 } /* a list of formal params are
separated by commas */
197.
198.     vdecl_list:
199.     /* nothing */ { [] }                /* zero or more variable
declarations */
200.     | vdecl_list vdecl { $2 :: $1 }
201.

```

```

202.     vdecl:                               /* a variable declaration consists of */
203.     typ_decl ID SEMICOLON { Vdecl (($1, $2),Noexpr) } /* a type
      declaration and identifier, */
204.     | typ_decl ID ASSIGN expr SEMICOLON { Vdecl (($1, $2),$4) } /* and
      optional initializer */
205.
206.
207.     /* TYPE DECLARATIONS
      *****
      *****/
208.     typ_decl:                             /* A type decl is composed of a list of types */
209.     | spec_qual_list { $1 }
210.
211.     typ_spec:
212.     INT { Int }
213.     | BOOL { Bool }
214.     | FLOAT { Float }
215.     | VOID { Void }
216.     | CHAR { Char }
217.     | STRING { String }
218.
219.     typ_qual:
220.     | PROB { Prob }
221.     | LIST { List }
222.
223.     spec_qual_list:
224.     | typ_spec spec_qual_list_opt {$1::$2} /* int/bool/float/void/char/string, followed
      by zero or more types */
225.     | typ_qual spec_qual_list {$1::$2} /* list/prob, followed by at least one type */
226.
227.     spec_qual_list_opt:                    /* zero or more types */
228.     | /*nothing*/ { [] }
229.     | spec_qual_list { $1 }

```

sast.ml

```
(* Semantically-checked Abstract Syntax Tree and functions for printing it *)
```

```
open Ast
```

```

type sexpr = typ_name * sx
and sx =
  SBinop of sexpr * binary_op * sexpr
| SUnop of unary_op * sexpr
| SAssign of sexpr * assign_op * sexpr
| SInt_lit of int
| SBool_lit of bool
| SFloat_lit of float
| SChar_lit of char
| SString_lit of string
| SList_lit of sexpr list
| SId of string
| SCast of typ_name * sexpr
| SListElement of sexpr * sexpr * sexpr list
| SListAddHead of sexpr * sexpr
| SListAddTail of sexpr * sexpr
| SLength of sexpr
| SProbColon of sexpr * sexpr
| SFuncall of string * sexpr list
| SIndex of sexpr
| SNoexpr

type svdecl =
  SVdecl of bind * sexpr

type sstmt =
  SBlock of sstmt list
| SExpr of sexpr
| SReturn of sexpr
| SIf of sexpr * sstmt
| SIf_Else of sstmt * sstmt
| SIf_Elif of sstmt * sstmt * sstmt list
| SIf_Elif_Else of sstmt * sstmt * sstmt list * sstmt
| SFor of sexpr * sexpr * sexpr * sstmt
| SWhile of sexpr * sstmt
| SBreak
| SContinue
| SElif of sexpr * sstmt (*only use inside an if statement, never alone!*)

type sfunc_decl = { styp_name : typ_name;

```

```

        sfname : string;
        sformals : bind list;
        slocals : svdecl list;
        sbody : sstmt list;
    }

let rec string_of_sexpr (t, e) =
  "(" ^ string_of_typ (List.hd t) ^ " : " ^ (match e with
    SInt_lit(l) -> string_of_int l
  | SFloat_lit(l) -> string_of_float l      (*check floats again*)
  | SBool_lit(true) -> "true"
  | SBool_lit(false) -> "false"
  | SId(s) -> s
  | SBinop(e1, o, e2) ->
      string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
  | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
  | SAssign(v, o, e) -> string_of_sexpr v ^ " " ^ string_of_assign o ^ " " ^
string_of_sexpr e
  | SFuncall(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
  | SNoexpr -> ""
      ) ^ ")"

let rec string_of_sstmt = function
  SBlock(stmts) ->
    "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "}\n"
  | SExpr(expr) -> string_of_sexpr expr ^ ";\n";
  | SReturn(expr) -> "return " ^ string_of_sexpr expr ^ ";\n";
  (*| SIf(e, s, SBlock([])) ->
    "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s
  | SIf(e, s1, s2) -> "if (" ^ string_of_sexpr e ^ ")\n" ^
    string_of_sstmt s1 ^ "else\n" ^ string_of_sstmt s2*)
  | SFor(e1, e2, e3, s) ->
    "for (" ^ string_of_sexpr e1 ^ " ; " ^ string_of_sexpr e2 ^ " ; " ^
    string_of_sexpr e3 ^ ") " ^ string_of_sstmt s
  | SWhile(e, s) -> "while (" ^ string_of_sexpr e ^ ") " ^ string_of_sstmt s

type sprogram = svdecl list * sfunc_decl list

type sDummy = (sprogram * string) (* Dummy SAST node for throughline*)

```

```

(*****)
(*
type sexpr = typ * sx
and sx =
  SLiteral of int
| SFliteral of string
| SBoolLit of bool
| SId of string
| SBinop of sexpr * binary_op * sexpr
| SUnop of unary_op * sexpr
| SAssign of string * sexpr
| SCall of string * sexpr list
| SNoexpr

type sstmt =
  SBlock of sstmt list
| SExpr of sexpr
| SReturn of sexpr
| SIf of sexpr * sstmt * sstmt
| SFor of sexpr * sexpr * sexpr * sstmt
| SWhile of sexpr * sstmt

type sfunc_decl = {
  styp_name : typ_name;
  sfname : string;
  sformals : bind list;
  slocals : bind list;
  sbody : sstmt list;
}

(*type sprogram = svdecl list * sfunc_decl list (* the start of our real SAST*)*)

(*
type sexpr = typ * sx
and sx =
  SLiteral of int
| SFliteral of string
| SBoolLit of bool
| SId of string
| SBinop of sexpr * op * sexpr

```

```

| SUnop of uop * sexpr
| SAssign of string * sexpr
| SCall of string * sexpr list
| SNoexpr

type sstmt =
  SBlock of sstmt list
| SExpr of sexpr
| SReturn of sexpr
| SIf of sexpr * sstmt * sstmt
| SFor of sexpr * sexpr * sexpr * sstmt
| SWhile of sexpr * sstmt

type sfunc_decl = {
  styp : typ;
  sfname : string;
  sformals : bind list;
  slocals : bind list;
  sbody : sstmt list;
}

type sprogram = bind list * sfunc_decl list

(* Pretty-printing functions *)

let rec string_of_sexpr (t, e) =
  "(" ^ string_of_typ t ^ " : " ^ (match e with
    SLiteral(l) -> string_of_int l
  | SBoolLit(true) -> "true"
  | SBoolLit(false) -> "false"
  | SFliteral(l) -> l
  | SId(s) -> s
  | SBinop(e1, o, e2) ->
    string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
  | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
  | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e
  | SCall(f, el) ->
    f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
  | SNoexpr -> ""
  ) ^ ")"

let rec string_of_sstmt = function

```

```

SBlock(stmts) ->
  "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "}\n"
| SExpr(expr) -> string_of_sexpr expr ^ ";\n";
| SReturn(expr) -> "return " ^ string_of_sexpr expr ^ ";\n";
| SIf(e, s, SBlock([])) ->
  "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s
| SIf(e, s1, s2) -> "if (" ^ string_of_sexpr e ^ ")\n" ^
  string_of_sstmt s1 ^ "else\n" ^ string_of_sstmt s2
| SFor(e1, e2, e3, s) ->
  "for (" ^ string_of_sexpr e1 ^ " ; " ^ string_of_sexpr e2 ^ " ; " ^
  string_of_sexpr e3 ^ ") " ^ string_of_sstmt s
| SWhile(e, s) -> "while (" ^ string_of_sexpr e ^ ") " ^ string_of_sstmt s

let string_of_sfdecl fdecl =
  string_of_typ fdecl.styp ^ " " ^
  fdecl.sfname ^ "(" ^ String.concat ", " (List.map snd fdecl.sformals) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_vdecl fdecl.slocals) ^
  String.concat "" (List.map string_of_sstmt fdecl.sbody) ^
  "}\n"

let string_of_sprogram (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_sfdecl funcs)
*)
*)

(* dummy pretty printer*)
(* (vars, funcs) *)

let string_of_sprogram (_,_) = "someday I'll pretty print the SAST! \n"

```

codegen.ml

```

(* Code generation: translate takes a semantically checked AST and
produces LLVM IR

LLVM tutorial: Make sure to read the OCaml version of the tutorial

http://llvm.org/docs/tutorial/index.html

```


Detailed documentation on the OCaml LLVM library:

<http://llvm.moe/>

<http://llvm.moe/ocaml/>

*)

```
module L = Llvn
```

```
module A = Ast
```

```
open Sast
```

```
module StringMap = Map.Make(String)
```

```
(* translate : Sast.program -> Llvn.module *)
```

```
let translate (globals, functions) =
```

```
  let context = L.global_context () in
```

```
(* Create the LLVM compilation module into which we will generate code *)
```

```
let the_module = L.create_module context "Smap" in
```

```
(*
```

```
For reference - delete before committing!!!
```

```
typedef struct list{
```

```
  void **data;
```

```
  int capacity;
```

```
  int size;
```

```
} list;
```

```
typedef struct prob {
```

```
  list probs;
```

```
  list vals;
```

```
  int length;
```

```
} prob;
```

```
*)
```

```
let i32_t = L.i32_type context (* 32 bit integer *)
```

```
and i8_t = L.i8_type context (* 8 bit integer *)
```

```
and i1_t = L.i1_type context (* 1 bit integer *)
```

```
and str = L.pointer_type (L.i8_type context)
```

```
and float_t = L.double_type context
```

```
and void_t = L.void_type context
```

```
in
```

```
let dummy_t = L.struct_type context [|str; i32_t;|]
```

```

in
let list_t : L.lltype = L.named_struct_type context "list"
in
let prob_t : L.lltype = L.named_struct_type context "prob"
in
(* Return the LLVM type for a MicroC type *)
let ltype_of_typ = function
  A.Int -> i32_t
| A.Char -> i8_t
| A.Bool -> i1_t
| A.Float -> float_t
| A.Void -> void_t
| A.String -> L.pointer_type list_t
| A.Prob -> L.pointer_type prob_t (*need to change to prob_t eventually*)
| A.List -> L.pointer_type list_t
| _ -> void_t (*add in prob, string, and list types later! *)
in

(*make sure to include struct definitions for the llvm struct types named earlier*)
let _ =
  L.struct_set_body list_t
    [| L.pointer_type (L.pointer_type i8_t)
      ; i32_t
      ; i32_t || false;
L.struct_set_body prob_t
  [| list_t
    ; list_t
    ; i32_t || false;
in

(* for reference
SFuncall ("printint", [e])
  -> L.build_call printint_func [| (expr builder e) |]
    "printint" builder
*)
let genGlobalListLit (typlist, exp) n m =
  let defaultZero =
    let zero = L.const_int i32_t 0 in
    L.const_named_struct list_t [|zero; zero; zero|]
  in
  let wholeThing = (match exp with
    (SList_lit(inits))

```

```

-> (match inits with
  ([A.Int], SInt_lit(v)) :: rest -> let one = L.const_int i32_t 1 in
                                   L.const_named_struct list_t [|one; one; one|]
  | _ -> let zero = L.const_int i32_t 0 in
         L.const_named_struct list_t [|zero; zero; zero|])
  | _ -> defaultZero)
  in StringMap.add n (L.define_global n wholeThing the_module) m
  in

(* Create a map of global variables after creating each *)
let global_vars : L.llvalue StringMap.t =
let global_var m (SVdecl((t, n), exp)) =
let simpleType = List.hd t in (*just use head of type list for now *)
match simpleType with
  A.List
  -> genGlobalListLit exp n m
  | _
  -> let init = (match simpleType with (* add more types in this pattern match! *)
                A.Float
                -> (match exp with
                    (_, SFloat_lit v)
                    -> L.const_float (ltype_of_typ simpleType) v
                    | _
                    -> L.const_float (ltype_of_typ simpleType) 0.0)
                | A.Int
                -> (match exp with
                    (_, SInt_lit v)
                    -> L.const_int (ltype_of_typ simpleType) v
                    | _
                    -> L.const_int(ltype_of_typ simpleType) 0)
                (*TODO: change this to actually work with prob type!!!*)
                | A.Prob
                -> L.const_int i32_t 0
                | _
                -> L.const_int i32_t 0)
      in StringMap.add n (L.define_global n init the_module) m
  in
List.fold_left global_var StringMap.empty globals in

(* declare built in C functions *)
let printstr_t: L.lltype = L.function_type i32_t [| L.pointer_type i8_t |] in
let printstr_func : L.llvalue = L.declare_function "printstr" printstr_t the_module
in

```

```

let corresponding_char_t: L.lltype = L.function_type i32_t [| i32_t |] in
let corresponding_char_func : L.llvalue = L.declare_function "corresponding_char"
corresponding_char_t the_module in

let printchar_t : L.lltype = L.function_type i32_t [| i8_t |] in
let printchar_func : L.llvalue = L.declare_function "printchar" printchar_t
the_module in

let print_list_char_t : L.lltype = L.function_type i32_t [| L.pointer_type list_t |]
in
let print_list_char_func : L.llvalue = L.declare_function "print_list_char"
print_list_char_t the_module in

let printb_t: L.lltype = L.function_type i32_t [| i1_t |] in
let printb_func : L.llvalue = L.declare_function "printb" printb_t the_module in

let stringLength_t: L.lltype = L.function_type i32_t [| L.pointer_type list_t |] in
let stringLength_func: L.llvalue = L.declare_function "stringLength" stringLength_t
the_module in

let stringReverse_t: L.lltype = L.function_type i32_t [| L.pointer_type list_t |] in
let stringReverse_func: L.llvalue = L.declare_function "stringReverse"
stringReverse_t the_module in

let absolute_t: L.lltype = L.function_type i32_t [| i32_t |] in
let absolute_func : L.llvalue = L.declare_function "absolute" absolute_t the_module
in

let key_test_t: L.lltype = L.function_type i32_t [| i32_t;i32_t |] in
let key_test_func : L.llvalue = L.declare_function "key_test" key_test_t the_module
in

let characterLocation_t: L.lltype = L.function_type i32_t [| L.pointer_type
list_t;L.pointer_type list_t |] in
let characterLocation_func : L.llvalue = L.declare_function "characterLocation"
characterLocation_t the_module in

let isCompDivisible_t: L.lltype = L.function_type i32_t [| i32_t |] in
let isCompDivisible_func : L.llvalue = L.declare_function "isCompDivisible"
isCompDivisible_t the_module in

let divisible_t: L.lltype = L.function_type i32_t [| i32_t;i32_t |] in

```

```

let divisible_func : L.llvalue = L.declare_function "divisible" divisible_t
the_module in

let power_t : L.lltype = L.function_type i32_t [| i32_t;i32_t |] in
let power_func : L.llvalue = L.declare_function "power" power_t the_module in

let ceilFloat_t : L.lltype = L.function_type i32_t [| float_t |] in
let ceilFloat_func : L.llvalue = L.declare_function "ceilFloat" ceilFloat_t
the_module in

let ascii_t : L.lltype = L.function_type i32_t [| L.pointer_type list_t |] in
let ascii_func : L.llvalue = L.declare_function "ascii" ascii_t the_module in

let int_to_char_t : L.lltype = L.function_type (L.pointer_type list_t) [| i32_t|] in
let int_to_char_func : L.llvalue = L.declare_function "int_to_char" int_to_char_t
the_module in

let corresponding_int_t : L.lltype = L.function_type i32_t [| L.pointer_type
list_t;i32_t |] in
let corresponding_int_func : L.llvalue = L.declare_function "corresponding_int"
corresponding_int_t the_module in

let testMakeStruct_t : L.lltype = L.function_type dummy_t [| L.pointer_type i8_t;i32_t
|] in
let testMakeStruct_func : L.llvalue = L.declare_function "testMakeStruct"
testMakeStruct_t the_module in

let printint_t : L.lltype = L.function_type i32_t [| i32_t |] in
let printint_func : L.llvalue = L.declare_function "printint" printint_t the_module
in

let printf_t : L.lltype = L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
let printf_func : L.llvalue = L.declare_function "printf" printf_t the_module in

let push_front_t : L.lltype = L.var_arg_function_type i32_t [|L.pointer_type list_t;
L.pointer_type i8_t|] in
let push_front_func : L.llvalue = L.declare_function "push_front" push_front_t
the_module in

let push_back_t : L.lltype = L.var_arg_function_type i32_t [|L.pointer_type list_t;
L.pointer_type i8_t|] in

```

```

let push_back_func : L.llvalue = L.declare_function "push_back" push_back_t
the_module in

let init_list_t : L.lltype = L.var_arg_function_type i32_t [|L.pointer_type list_t|]
in
let init_list_func : L.llvalue = L.declare_function "init_list" init_list_t
the_module in

let init_prob_t : L.lltype = L.var_arg_function_type i32_t [|L.pointer_type prob_t;
L.pointer_type list_t;
L.pointer_type list_t|]
in
let init_prob_func : L.llvalue = L.declare_function "init_prob" init_prob_t
the_module in

let peek_t : L.lltype = L.var_arg_function_type (L.pointer_type i8_t)
[|L.pointer_type prob_t|] in
let peek_func : L.llvalue = L.declare_function "peek" peek_t the_module in
let list_length_t : L.lltype = L.var_arg_function_type i32_t [|L.pointer_type
list_t|] in
let list_length_func : L.llvalue = L.declare_function "list_length" list_length_t
the_module in

let modulus_operation_t : L.lltype = L.var_arg_function_type i32_t [|i32_t; i32_t|] in
let modulus_operation_func : L.llvalue = L.declare_function "modulus_operation"
modulus_operation_t the_module in

let get_length_t : L.lltype = L.var_arg_function_type i32_t [|L.pointer_type prob_t|]
in
let get_length_func : L.llvalue = L.declare_function "get_length" get_length_t
the_module in

let get_at_t : L.lltype = L.var_arg_function_type (L.pointer_type i8_t)
[|L.pointer_type list_t; i32_t|] in
let get_at_func : L.llvalue = L.declare_function "get_at" get_at_t the_module in

let set_at_t : L.lltype = L.var_arg_function_type (L.pointer_type list_t)
[|L.pointer_type list_t; i32_t; L.pointer_type i8_t|] in
let set_at_func : L.llvalue = L.declare_function "set_at" set_at_t the_module in

let print_list_int_t : L.lltype = L.var_arg_function_type i32_t [|L.pointer_type
list_t|] in

```

```

let print_list_int_func : L.llvalue = L.declare_function "print_list_int"
print_list_int_t the_module in

let print_list_float_t : L.lltype = L.var_arg_function_type i32_t [|L.pointer_type
list_t|] in
let print_list_float_func : L.llvalue = L.declare_function "print_list_float"
print_list_float_t the_module in

let get_vals_t : L.lltype = L.var_arg_function_type (L.pointer_type list_t)
[|L.pointer_type prob_t|] in
let get_vals_func : L.llvalue = L.declare_function "get_vals" get_vals_t the_module
in

let get_probs_t : L.lltype = L.var_arg_function_type (L.pointer_type list_t)
[|L.pointer_type prob_t|] in
let get_probs_func : L.llvalue = L.declare_function "get_probs" get_probs_t
the_module in

let print_prob_int_debug_t : L.lltype = L.var_arg_function_type i32_t
[|L.pointer_type prob_t|] in
let print_prob_int_debug_func : L.llvalue = L.declare_function "print_prob_int_debug"
print_prob_int_debug_t the_module in

let bad_add_head_t : L.lltype = L.var_arg_function_type i32_t [| L.pointer_type
list_t; L.pointer_type i8_t |] in
let bad_add_head_func : L.llvalue = L.declare_function "bad_add_head" bad_add_head_t
the_module in

let very_bad_get_head_t : L.lltype = L.var_arg_function_type i32_t [| L.pointer_type
list_t|] in
let very_bad_get_head_func : L.llvalue = L.declare_function "very_bad_get_head"
very_bad_get_head_t the_module in

(* Define each function (arguments and return type) so we can
call it even before we've created its body *)
let function_decls : (L.llvalue * sfunc_decl) StringMap.t =
let function_decl m fdecl =
let name = fdecl.sfname
and formal_types = Array.of_list (List.map (fun (t,_) -> ltype_of_ttyp (List.hd t))
fdecl.sformals) in
let simpleReturnType = List.hd fdecl.styp_name in
let ftype = L.function_type (ltype_of_ttyp simpleReturnType) formal_types in

```

```

StringMap.add name (L.define_function name ftype the_module, fdecl) m in
List.fold_left function_decl StringMap.empty functions in
(* ... *)
(* Fill in the body of the given function *)
let build_function_body fdecl =
  let (the_function, _) =
    StringMap.find fdecl.sfname function_decls in
  let builder =
    L.builder_at_end context (L.entry_block the_function) in
  let test_str =
    L.build_global_stringptr "test test test!\n" "test" builder in
  let newLine_str =
    L.build_global_stringptr "\n" "newLine" builder in
  let int_format_str =
    L.build_global_stringptr "%d" "fmt" builder
  and float_format_str =
    L.build_global_stringptr "%f" "fmt" builder in

let local_vars =
  let add_formal m (t, n) p = (* add formal parameters to scope *)
    L.set_value_name n p;
  let simpleType = List.hd t in
  let local = L.build_alloca (ltype_of_typ simpleType) n builder in
  ignore (L.build_store p local builder);
  StringMap.add n local m
  and add_local m (SVdecl((t, n), exp)) = (* add locals to scope *)
  let simpleType2 = List.hd t in
  let local_var = L.build_alloca (ltype_of_typ simpleType2) n builder in
  StringMap.add n local_var m in
  let formals = List.fold_left2 add_formal StringMap.empty fdecl.sformals
(Array.to_list (L.params the_function)) in
  List.fold_left add_local formals fdecl.slocals in

(* Return the value for a variable or formal argument. Check local names first, then
global names *)
let lookup n = try StringMap.find n local_vars with
  Not_found -> StringMap.find n global_vars in

let rec expr builder ((theTyp, e) : sexpr) =
  (*
allocInitElt helper function
Description:

```



```

Allocate and initialize a list elt, then cast to ( char * )
For ex: int * tmp = malloc(sizeof(int));
*tmp = 5;
char * tmp2 = ( char* ) tmp;
After this step, ready to pass vals to the push_front function!
*)
let allocInitElt v =
  (* save type of element *)
  let vtype = fst v in
  (* 5 *)
  let value = expr builder v in
  (* int * tmp *)
  let tmp = match (vtype) with
    [A.Int] -> L.build_alloc (L.pointer_type i32_t) "tmp" builder
  | [A.Float] -> L.build_alloc (L.pointer_type float_t) "tmp" builder
  | [A.Char] -> L.build_alloc (L.pointer_type i8_t) "tmp" builder
  | [A.Bool] -> raise (Failure("fill in for bool"))
  | [A.Prob] -> raise (Failure("fill in for prob someday... >.<\"\"))

  in
  (* malloc(sizeof(int)) *)
  let heapAddr = match (vtype) with
    [A.Int] -> L.build_malloc i32_t "tmpAddr" builder
  | [A.Float] -> L.build_malloc float_t "tmpAddr" builder
  | [A.Char] -> L.build_malloc i8_t "tmpAddr" builder
  | [A.Bool] -> raise (Failure("fill in for bool"))
  | [A.Prob] -> raise (Failure("fill in for prob someday... >.<\"\"))

  in
  (* tmp = malloc(sizeof(int)); *)
  let tmp' = ignore (L.build_store heapAddr tmp builder); tmp in
  (* *tmp *)
  let tmp'' = L.build_load tmp' "tmpWithVal" builder in
  (* *tmp = 5 *)
  let tmp''' = ignore (L.build_store value tmp'' builder); tmp'' in
  (* char * tmp2 = ( char* ) tmp; *)
  let asChar = L.build_bitcast tmp''' (L.pointer_type i8_t) "asChar" builder
  in asChar
  in
  (* multidimensional indexing for lists helper function *)
  let rec deRef ((typs,i)::rest) handle =
    let elt = L.build_call get_at_func [| handle; expr builder (typs,i)|]
      "get_at" builder in
    match typs with

```

```

    [A.Int]
    -> let eltAsPtr = L.build_bitcast elt (L.pointer_type i32_t) "eltAsPtr"
builder in
    L.build_load eltAsPtr "eltDeref" builder
  | [A.Float]
    -> let eltAsPtr = L.build_bitcast elt (L.pointer_type float_t) "eltAsPtr"
builder in
    L.build_load eltAsPtr "eltDeref" builder
  | [A.Char]
    -> let eltAsPtr = L.build_bitcast elt (L.pointer_type i8_t) "eltAsPtr" builder
in
    L.build_load eltAsPtr "eltDeref" builder
  | [A.String]
    -> let eltAsPtr = L.build_bitcast elt (L.pointer_type list_t) "eltAsPtr"
builder in
    L.build_load eltAsPtr "eltDeref" builder
  | A.List::t
    ->
    if (List.length rest) = 0
    then L.build_bitcast elt (L.pointer_type list_t) "eltAsPtr" builder
    else deRef rest (L.build_bitcast elt (L.pointer_type list_t) "eltAsPtr"
builder)
  | [A.Bool]
    -> raise(Failure("fill in for bool"))
  | A.Prob::t1
    -> raise(Failure("fill in for prob someday... >.<\\"))
in
match e with
SInt_lit i
-> L.const_int i32_t i
| SIndex i
-> expr builder i
| SString_lit s
-> let len = String.length s in
let rec explode i m str acc =
  if i = m then acc
  else let c = (String.get str i) in
c::(explode (i+1) m str acc) in
let chars = explode 0 len s [] in
let toCharNode c = ([A.Char],SChar_lit c) in
let asList = ([A.List;A.Char], SList_lit(List.map toCharNode chars)) in
(* L.build_global_stringptr s "the_str" builder *)

```

```

    expr builder asList
  | SBool_lit b
  -> L.const_int i1_t (if b then 1 else 0)
  | SFloat_lit l
  -> L.const_float float_t l
  | SChar_lit l
  -> L.const_int i8_t (Char.code l)
  | SNoexpr
  -> L.const_int i32_t 0
  | SId s
  -> L.build_load (lookup s) s builder
  | SLength(p)
  -> let p' = expr builder p in
      (match List.hd (fst p) with
       | A.Prob
       -> L.build_call get_length_func [| p' |]
          "get_length" builder
       | A.List
       -> L.build_call list_length_func [| p' |]
          "list_length" builder
       | A.String
       -> L.build_call list_length_func [| p' |]
          "list_length" builder
       | _
       -> raise (Failure("semant should have caught this misuse of .length!
"^(A.string_of_ttyp_name (fst p)))) )
  | SListElement (listId,index,rest)
  -> let eltType = fst index in
      (match eltType with
       | A.List::tl
       -> let theList = expr builder listId in
          deRef (index::rest) theList
       | _
       -> let theList = expr builder listId in
          let elt = L.build_call get_at_func [| theList; expr builder index|]
              "get_at" builder in
          let eltAsPtr = match eltType with
              [A.Int] -> L.build_bitcast elt (L.pointer_type i32_t) "eltAsPtr"
builder
              | [A.Float] -> L.build_bitcast elt (L.pointer_type float_t)
"eltAsPtr" builder

```

```

| [A.Char] -> L.build_bitcast elt (L.pointer_type i8_t)
"eltAsPtr" builder
| [A.Bool] -> raise(Failure("fill in for bool"))
| A.Prob::t1 -> raise(Failure("fill in for prob someday...
>.<\""))

    in
    let eltDeRef = L.build_load eltAsPtr "eltDeref" builder in
    eltDeRef)
| SListAddHead (e1,e2)
-> let e1' = expr builder e1
    and toAdd = match fst e2 with
                (A.List::t)
                -> expr builder e2
                | _
                -> allocInitElt e2
    in let _ = L.build_call push_front_func [| e1'; toAdd |]
        "push_front" builder in e1'
| SListAddTail (e1,e2)
-> let e1' = expr builder e1
    and toAdd = match fst e2 with
                (A.List::t)
                -> expr builder e2
                | _
                -> allocInitElt e2
    in let _ = L.build_call push_back_func [| e1'; toAdd |]
        "push_back" builder in e1'
| SList_lit elts
-> (match theTyp with
    [A.List;A.List;u]
    -> (*create the sublists*)
    let sublists = List.map (expr builder) elts in
    (* cast the list pointers to char pointers*)
    let toChar l = L.build_bitcast l (L.pointer_type i8_t) "toChar" builder in
    let charPtrs = List.map toChar sublists in
    let addElt lst c =
        L.build_call push_back_func [| lst; c |]
        "push_back" builder
    in
    (* allocate list struct *)
    let aList = L.build_alloc (L.pointer_type list_t) "theList" in
    (* get pointer to list struct *)
    let anAllocatedList = L.build_malloc list_t "listAddr" builder in

```

```

(* initialize the list with list_init *)
let _ = L.build_call init_list_func [| anAllocatedList|]
"init_list" builder in
let _ = List.map (addElt anAllocatedList) charPtrs
(* return the pointer to the now-intialized list*)
in anAllocatedList
| [A.List;u] (*simple list of one type - (no lists of lists here)*)
-> (match elts with
  [] (*CASE: EMPTY LIST *)
  ->
    (* get pointer to list struct *)
    let anAllocatedList = L.build_malloc list_t "listAddr" builder in
    (* initialize the list with list_init *)
    let _ = L.build_call init_list_func [| anAllocatedList|]
    "init_list" builder
    in anAllocatedList
  | (elt::rest) (*CASE: NON-EMPTY LIST *)
  ->
    let addElt lst c =
      L.build_call push_back_func [| lst; c |]
      "push_back" builder
    in
    (* allocate list struct *)
    let aList = L.build_alloca (L.pointer_type list_t) "theList" in
    (* get pointer to list struct *)
    let anAllocatedList = L.build_malloc list_t "listAddr" builder in
    (* initialize the list with list_init *)
    let _ = L.build_call init_list_func [| anAllocatedList|]
    "init_list" builder in
    (* add any starting elts to the list *)
    let values = List.map allocInitElt (elt::rest)
    in
    let _ = List.map (addElt anAllocatedList) values
    (* return the pointer to the now-intialized list*)
    in anAllocatedList)
|_
-> raise (Failure "Only support simple lists at the moment >.<\"")
| SProbColon (lhs,rhs)
-> let lhs' = expr builder lhs
and rhs' = expr builder rhs in
(* get pointer to prob struct *)
let anAllocatedProb = L.build_malloc prob_t "probAddr" builder in

```

```

(* initialize the prob with prob_init *)
let _ = L.build_call init_prob_func [| anAllocatedProb; lhs'; rhs' |]
"init_prob" builder in
(* let vs = L.build_call get_vals_func [| anAllocatedProb |]
"get_vals" builder in
let _ = L.build_call print_list_int_func [| (vs) |]
"print_list_int" builder in *)
anAllocatedProb
| SUnop (op, ((t, _) as e))
-> let e' = expr builder e in (match op with
  A.Neg when t = [A.Float] -> L.build_fneg e' "tmp" builder
  | A.Neg
-> L.build_neg e' "tmp" builder
  | A.BitNot
-> L.build_not e' "tmp" builder
  | A.Bang
-> let cPtr = L.build_call peek_func [| e' |] "peek" builder in
  (* let vs = L.build_call get_vals_func [| e' |] "get_vals" builder in
  let cPtr = L.build_call get_at_func [|vs; L.const_int i32_t 0|] "get_at"
builder in *)
  let eltAsPtr = match theTyp with
    [A.Int] -> L.build_bitcast cPtr (L.pointer_type i32_t)
"eltAsPtr" builder
    | [A.Float] -> L.build_bitcast cPtr (L.pointer_type float_t)
"eltAsPtr" builder
    | [A.Char] -> L.build_bitcast cPtr (L.pointer_type i8_t)
"eltAsPtr" builder
    | [A.Bool] -> L.build_bitcast cPtr (L.pointer_type i1_t)
"eltAsPtr" builder
    | [A.List] -> L.build_bitcast cPtr (L.pointer_type list_t)
"eltAsPtr" builder
  in
  let eltDeRef = L.build_load eltAsPtr "eltDeref" builder in
  eltDeRef
  | A.Octothorpe
-> L.build_call get_probs_func [| e' |] "get_probs" builder
  | A.Not
-> L.build_not e' "tmp" builder)
| SBinop (([Float], _) as e1, op, e2) ->
let e1' = expr builder e1
and e2' = expr builder e2 in
(match op with

```

```

    A.Add      -> L.build_fadd
  | A.Sub      -> L.build_fsub
  | A.Mul      -> L.build_fmud
  | A.Div      -> L.build_fdiv
  | A.CompNeq  -> L.build_fcmp L.Fcmp.One
  | A.CompLt   -> L.build_fcmp L.Fcmp.Olt
  | A.CompLeq  -> L.build_fcmp L.Fcmp.Ole
  | A.CompGt   -> L.build_fcmp L.Fcmp.Ogt
  | A.CompGeq  -> L.build_fcmp L.Fcmp.Oge
  | A.BitAnd | A.BitOr ->
      raise (Failure "internal error: semant should have rejected and/or on float")
) e1' e2' "tmp" builder
  | SBinop (e1, op, e2) ->
  let e1' = expr builder e1
  and e2' = expr builder e2 in
  (match op with
    A.Add      -> L.build_add
  | A.Sub      -> L.build_sub
  | A.Mul      -> L.build_mul
  | A.Div      -> L.build_sdiv
  | A.BitAnd   -> L.build_and
  | A.BitOr    -> L.build_or
  | A.CompNeq  -> L.build_icmp L.Icmp.Ne
  | A.CompLt   -> L.build_icmp L.Icmp.Slt
  | A.CompLeq  -> L.build_icmp L.Icmp.Sle
  | A.CompGt   -> L.build_icmp L.Icmp.Sgt
  | A.CompGeq  -> L.build_icmp L.Icmp.Sge
  | A.CompEq   -> L.build_icmp L.Icmp.Eq
  | A.RShift   -> L.build_ashr
  | A.LShift   -> L.build_lshr
  | A.Xor      -> L.build_xor
) e1' e2' "tmp" builder
  | SAssign (e1, op, e2)
-> (match e1 with
    (_, SId nm)
    -> let e1' = lookup nm
        and e2' = expr builder e2 in
        (match op with (*PlusEqual | MinusEqual | TimesEqual | DivEqual | Equal*)
          PlusEqual -> raise (Failure("special assignments need binops to be able
to work"))
          | MinusEqual -> raise (Failure("special assignments need binops to be able
to work"))

```

```

    | TimesEqual -> raise (Failure("special assignments need binops to be able
to work"))
    | DivEqual -> raise (Failure("special assignments need binops to be able to
work"))
    | Equal -> ignore(L.build_store e2' e1' builder); e1')
  | (ty, SListElement (listId, index, rest))
  -> match rest with
    []
    -> let theList = expr builder listId
        and i = expr builder index
        and e2' = allocInitElt e2 in
        let _ = L.build_call set_at_func [| theList; i; e2'|] "set_at" builder
        in theList
    | _
    -> let theList = expr builder listId in

        let lastIndex = List.hd (List.rev rest)
        and allbutLastIndex = index :: (List.rev (List.tl (List.rev rest))) in
        let listToIndexInto = deRef allbutLastIndex theList
        and i = expr builder lastIndex
        and e2' = allocInitElt e2 in
        let _ = L.build_call set_at_func [| listToIndexInto; i; e2'|] "set_at"
builder
        in listToIndexInto )
  | SFuncall ("printint", [e])
  -> L.build_call printint_func [| (expr builder e) |]
    "printint" builder
  | SFuncall ("stringLength", [e]) ->
    L.build_call stringLength_func [| (expr builder e) |]
    "stringLength" builder
  | SFuncall ("stringReverse", [e]) ->
    L.build_call stringReverse_func [| (expr builder e) |]
    "stringReverse" builder
  | SFuncall ("corresponding_char", [e]) ->
    L.build_call corresponding_char_func [| (expr builder e) |]
    "corresponding_char" builder
  | SFuncall ("corresponding_int", [e;e2]) ->
    L.build_call corresponding_int_func [| (expr builder e);(expr builder e2) |]
    "corresponding_int" builder
  | SFuncall ("modulus_operation", [e;e2]) ->
    L.build_call modulus_operation_func [| (expr builder e);(expr builder e2) |]
    "modulus_operation" builder

```



```

| SFuncall ("key_test", [e;e2]) ->
L.build_call key_test_func [| (expr builder e);(expr builder e2) |]
"key_test" builder

| SFuncall ("absolute", [e]) ->
L.build_call absolute_func [| (expr builder e) |]
"absolute" builder

| SFuncall ("ceilFloat", [e]) ->
L.build_call ceilFloat_func [| (expr builder e) |]
"ceilFloat" builder

| SFuncall ("int_to_char", [e]) ->
L.build_call int_to_char_func [| (expr builder e) |]
"int_to_char" builder

| SFuncall ("characterLocation", [e;e2]) ->
L.build_call characterLocation_func [| (expr builder e); (expr builder e2) |]
"characterLocation" builder

| SFuncall ("isCompDivisible", [e]) ->
L.build_call isCompDivisible_func [| (expr builder e) |]
"isCompDivisible" builder

| SFuncall ("divisible", [e;e2]) ->
L.build_call divisible_func [| (expr builder e); (expr builder e2) |]
"divisible" builder

| SFuncall ("power", [e;e2]) ->
L.build_call power_func [| (expr builder e); (expr builder e2) |]
"power" builder

| SFuncall ("ascii", [e]) ->
L.build_call ascii_func [| (expr builder e) |]
"ascii" builder

| SFuncall ("very_bad_get_head", [e])
-> let arg = (expr builder e) in
    L.build_call very_bad_get_head_func [| arg |]
    "very_bad_get_head" builder

| SFuncall ("testMakeStruct",[(_,theSEExpr) as arg])
-> (match theSEExpr with
    SInt_lit _
    -> L.build_call testMakeStruct_func [| test_str; expr builder arg |]
        "testMakeStruct" builder
    | _
    -> L.build_call printb_func [| (expr builder arg) |]
        "printb" builder)

| SFuncall ("println",arg)
-> let _ = expr builder (theTyp,SFuncall("print",arg)) in

```

```

L.build_call printstr_func [| newLine_str |] "printstr" builder
| SFuncall ("print",[typeList,_) as arg])
-> (match typeList with
  [A.Int]
  -> L.build_call printint_func [| (expr builder arg) |]
      "printint" builder
  | [A.Char] ->
      L.build_call printchar_func [| (expr builder arg) |]
          "printchar" builder
  | [A.Bool]
  -> L.build_call printb_func [| (expr builder arg) |]
      "printb" builder
  | [A.Float]
  -> L.build_call printf_func [| float_format_str ; (expr builder arg) |]
      "printf" builder
  | [A.String]
  -> L.build_call print_list_char_func [| (expr builder arg) |]
      "print_list_char" builder
  | [A.List; A.Int]
  -> L.build_call print_list_int_func [| (expr builder arg) |]
      "print_list_int" builder
  | [A.List; A.Float]
  -> L.build_call print_list_float_func [| (expr builder arg) |]
      "print_list_float" builder
  | [A.List; A.Char]
  -> L.build_call print_list_char_func [| (expr builder arg) |]
      "print_list_char" builder
  | _
  -> raise(Failure("printing type " ^ A.string_of_typ_name typeList ^ "not yet
supported")))
| SFuncall ("print_list_int", [e]) ->
    L.build_call print_list_int_func [| (expr builder e) |]
        "print_list_int" builder
| SFuncall (f, args) ->
    let (fdef, fdecl) = StringMap.find f function_decls in
    let llargs = List.rev (List.map (expr builder) (List.rev args)) in
    let result = (match (List.hd fdecl.styp_name) with
      A.Void -> ""
    | _ -> f ^ "_result") in
    L.build_call fdef (Array.of_list llargs) result builder
in

```

```

let add_terminal builder f =
  match L.block_terminator (L.insertion_block builder) with
  Some _ -> ()
  | None -> ignore (f builder) in

let rec stmt builder = function
  SBlock s1 -> List.fold_left stmt builder s1
  | SExpr e -> ignore(expr builder e); builder
  | SReturn e -> ignore(match (List.hd fdecl.styp_name) with
                        (* Special "return nothing" instr *)
                        A.Void -> L.build_ret_void builder
                        (* Build return statement *)
                        | _ -> L.build_ret (expr builder e) builder
  ); builder
  | SWhile (predicate, body) ->
    let pred_bb = L.append_block context "while" the_function in
    ignore(L.build_br pred_bb builder);

    let body_bb = L.append_block context "while_body" the_function in
    add_terminal (stmt (L.builder_at_end context body_bb) body)
      (L.build_br pred_bb);

    let pred_builder = L.builder_at_end context pred_bb in
    let bool_val = expr pred_builder predicate in

    let merge_bb = L.append_block context "merge" the_function in
    ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
    L.builder_at_end context merge_bb

    (* Implement for loops as while loops *)
    | SFor (e1, e2, e3, body) -> stmt builder
      ( SBlock [SExpr e1 ; SWhile (e2, SBlock [body ; SExpr e3]) ] )

    | SIf (p, b) ->
      let pred_bb = L.append_block context "if" the_function in
      ignore(L.build_br pred_bb builder);

      let body_bb = L.append_block context "if_body" the_function in
      let merge_bb = L.append_block context "merge" the_function in
      add_terminal (stmt (L.builder_at_end context body_bb) b)
        (L.build_br merge_bb);

```

```

    let pred_builder = L.builder_at_end context pred_bb in
    let bool_val = expr pred_builder p in

    ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
    L.builder_at_end context merge_bb

| Sif_Else (p, b) ->
    let if_node = (match p with Sif(lhs, rhs) -> (lhs, rhs)
                  | _ -> raise(Failure("If_Else Sif block matching error"))) in

    let pred_bb = L.append_block context "if" the_function in
    ignore(L.build_br pred_bb builder);

    let body_bb = L.append_block context "if_body" the_function in
    let else_bb = L.append_block context "else_body" the_function in
    let merge_bb = L.append_block context "merge" the_function in
    add_terminal (stmt (L.builder_at_end context body_bb) (snd if_node))
                 (L.build_br merge_bb);

    add_terminal (stmt (L.builder_at_end context else_bb) b)
                 (L.build_br merge_bb);

    let pred_builder = L.builder_at_end context pred_bb in
    let bool_val = expr pred_builder (fst if_node) in
    ignore(L.build_cond_br bool_val body_bb else_bb pred_builder);
    L.builder_at_end context merge_bb

| Sif_Elif(if_stmt, fst_elif, elif_lst) ->
    let if_node = (match if_stmt with
                  Sif(lhs, rhs) -> (lhs, rhs)
                  | _ -> raise(Failure("If_Elif Sif block matching error"))) in
    let elif_node = (match fst_elif with
                    SElif(lhs, rhs) -> (lhs, rhs)
                    | _ -> raise(Failure("If_Elif SElif block matching error"))) in

    let pred_bb = L.append_block context "if" the_function in
    ignore(L.build_br pred_bb builder);

    let body_bb = L.append_block context "if_body" the_function in

    let elif_pred_bb = L.append_block context "elif" the_function in

```

```

    let elif_bb = L.append_block context "elif_body" the_function in

    let merge_bb = L.append_block context "merge" the_function in

    (* if stmt body *)
    add_terminal (stmt (L.builder_at_end context body_bb) (snd if_node))
      (L.build_br merge_bb);
    (* elif conditional *)
    let pred_builder = L.builder_at_end context elif_pred_bb in
    let elif_val = expr pred_builder (fst elif_node) in
    ignore(L.build_cond_br elif_val elif_bb merge_bb pred_builder);
    (* elif stmt body *)
    add_terminal (stmt (L.builder_at_end context elif_bb) (snd elif_node))
      (L.build_br merge_bb);

    (* let rec elif_builder elifs = function
    | s :: s ->
    | [] -> None *)

    (* if conditional *)
    let pred_builder = L.builder_at_end context pred_bb in
    let bool_val = expr pred_builder (fst if_node) in
    ignore(L.build_cond_br bool_val body_bb elif_pred_bb pred_builder);

    L.builder_at_end context merge_bb

    (* SIf (predicate, then_stmt, else_stmt) ->
    let bool_val = expr builder predicate in
    let merge_bb = L.append_block context "merge" the_function in
    let build_br_merge = L.build_br merge_bb in (* partial function *)
    let then_bb = L.append_block context "then" the_function in
    add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
      build_br_merge;
    let else_bb = L.append_block context "else" the_function in
    add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
      build_br_merge;
    ignore(L.build_cond_br bool_val then_bb else_bb builder);
    L.builder_at_end context merge_bb *)

    in

    (* Build the code for each statement in the function *)

```

```

let builder = stmt builder (SBlock fdecl.sbody) in
(* Add a return if the last block falls off the end *)
add_terminal_builder (match (List.hd fdecl.styp_name) with
A.Void -> L.build_ret_void
| A.Float -> L.build_ret (L.const_float float_t 0.0)
| t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
(* ... *)

in
List.iter build_function_body functions;
the_module

(*(* <= un comment this line to see the old microc semantic analyzer with nice
colors*)
(* translate : Sast.program -> Llvm.module *)
let translate (globals, functions) =
let context = L.global_context () in
(* Create the LLVM compilation module into which
we will generate code *)
let the_module = L.create_module context "MicroC" in
(* Get types from the context *)
let i32_t = L.i32_type context
and i8_t = L.i8_type context
and i1_t = L.i1_type context
and float_t = L.double_type context
and void_t = L.void_type context in
(* Return the LLVM type for a MicroC type *)
let ltype_of_typ = function
A.Int -> i32_t
| A.Bool -> i1_t
| A.Float -> float_t
| A.Void -> void_t
in
(* Create a map of global variables after creating each *)
let global_vars : L.llvalue StringMap.t =
let global_var m (t, n) =
let init = match t with
A.Float -> L.const_float (ltype_of_typ t) 0.0
| _ -> L.const_int (ltype_of_typ t) 0
in StringMap.add n (L.define_global n init the_module) m in

```

```

List.fold_left global_var StringMap.empty globals in
let printf_t : L.lltype =
  L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
let printf_func : L.llvalue =
  L.declare_function "printf" printf_t the_module in
let printbig_t : L.lltype =
  L.function_type i32_t [| i32_t |] in
let printbig_func : L.llvalue =
  L.declare_function "printbig" printbig_t the_module in
(* Define each function (arguments and return type) so we can
   call it even before we've created its body *)
let function_decls : (L.llvalue * sfunc_decl) StringMap.t =
  let function_decl m fdecl =
    let name = fdecl.sfname
      and formal_types =
Array.of_list (List.map (fun (t, _) -> ltype_of_typ t) fdecl.sformals)
      in let ftype = L.function_type (ltype_of_typ fdecl.styp) formal_types in
    StringMap.add name (L.define_function name ftype the_module, fdecl) m in
  List.fold_left function_decl StringMap.empty functions in
(* Fill in the body of the given function *)
let build_function_body fdecl =
  let (the_function, _) = StringMap.find fdecl.sfname function_decls in
  let builder = L.builder_at_end context (L.entry_block the_function) in
  let int_format_str = L.build_global_stringptr "%d\n" "fmt" builder
    and float_format_str = L.build_global_stringptr "%g\n" "fmt" builder in
  (* Construct the function's "locals": formal arguments and locally
   declared variables. Allocate each on the stack, initialize their
   value, if appropriate, and remember their values in the "locals" map *)
  let local_vars =
    let add_formal m (t, n) p =
      L.set_value_name n p;
    let local = L.build_alloca (ltype_of_typ t) n builder in
      ignore (L.build_store p local builder);
    StringMap.add n local m
    (* Allocate space for any locally declared variables and add the
     * resulting registers to our map *)
    and add_local m (t, n) =
let local_var = L.build_alloca (ltype_of_typ t) n builder
in StringMap.add n local_var m
      in
let formals = List.fold_left2 add_formal StringMap.empty fdecl.sformals
  (Array.to_list (L.params the_function)) in

```

```

    List.fold_left add_local formals fdecl.slocals
  in
    (* Return the value for a variable or formal argument.
       Check local names first, then global names *)
    let lookup n = try StringMap.find n local_vars
                   with Not_found -> StringMap.find n global_vars
    in
      (* Construct code for an expression; return its value *)
    *)

```

list.c

```

#include <stdlib.h>
#include <stdio.h>
#include "list.h"

int init_list(list *l) {
    l->capacity = INITIAL_LIST_CAPACITY;
    l->size = 0;

    l->data = (char **) malloc(INITIAL_LIST_CAPACITY * sizeof(char *));
    if (!l->data)
        return -1; // failure

    return 0; // success
}

//return length of list
int list_length (list * l){
    return l->size;
}

// checks if list is at capacity
// ret 1 if needs resizing, ret 0 if doesn't
int check_resizing(list *l) {
    if (l->size == l->capacity) {return 1;} else {return 0;}
}

// ret 1 if empty
// ret 0 if non-empty
int check_empty(list *l) {

```



```

    if (l->size == 0) {return 1;} else {return 0;}
}

// ret 0 if successful
// ret -1 on failure
int resize(list *l) {
    l->data = (char **) realloc(l->data, l->capacity * 2 * sizeof(char *));
    if (l->data == NULL)
        return -1;
    l->capacity *= 2;
    return 0;
}

void push_back(list *l, char *item) {
    if (check_resizing(l))
        resize(l);
    // set item
    l->data[l->size] = item;
    l->size++;
}

void push_front(list *l, char *item) {
    if (check_resizing(l))
        resize(l);

    // shift everything over by 1
    for (int i=l->size; i > 0; i--) {
        l->data[i] = l->data[i-1];
    }

    l->data[0] = item;
    l->size++;
}

// ret 0 on success
// ret -1 on failure
// WARNING: WILL CAUSE MEMORY LEAKS! DATA DOES NOT GET FREE'D
int del_back(list *l) {
    if (check_empty(l))
        return -1;

    l->size--;
}

```

```

    return 0;
}

// ret 0 on success
// ret -1 on failure
// WARNING: WILL CAUSE MEMORY LEAKS! DATA DOES NOT GET FREE'D
int del_front(list *l) {
    if (check_empty(l))
        return -1;

    // shift over data
    for (int i=0; i < (l->size - 1); i++)
        l->data[i] = l->data[i+1];

    l->size--;
    return 0;
}

// ret 0 on success
// ret -1 on failure
// WARNING: WILL CAUSE MEMORY LEAKS! DATA DOES NOT GET FREE'D
int del_at(list *l, int i) {
    if (check_empty(l))
        return -1;

    // shift over data
    for (int j=i; j < (l->size - 1); j++)
        l->data[j] = l->data[j+1];

    l->size--;
    return 0;
}

char *get_back(list *l) {
    if (l->size == 0)
        return NULL; // list is empty

    // offset data pointer by size of array
    return l->data[l->size - 1];
}

char *get_front(list *l) {

```

```

    if (l->size == 0)
        return NULL; // list is empty

    // get index 0, ie. first element
    return l->data[0];
}

char *get_at(list *l, int i) {
    if (check_empty(l))
        return NULL;

    // check if index is within bounds of array
    if (i < l->size && i >= 0)
        return l->data[i];

    return NULL;
}

list* set_at(list *l, int i, char* val) {
    if (check_empty(l))
        return NULL;

    // check if index is within bounds of array
    if (i < l->size && i >= 0){
        l->data[i] = val;
        return l;
    }
    return NULL;
}

void print_list_int(list *l) {
    printf("[");
    for (int i=0; i<(l->size)-1; i++){
        printf("%d, ", *(int *)l->data[i]);
    }
    printf("%d]", *(int *)l->data[(l->size)-1]);
}

void print_list_float(list *l) {
    printf("[");
    for (int i=0; i<(l->size)-1; i++){
        printf("%f, ", *(double *)l->data[i]);
    }
}

```

```

    }
    printf("%f]", *(double*)l->data[(l->size)-1]);
}

void print_list_char(list *l) {
    for (int i=0; i<(l->size)-1; i++){
        printf("%c", *l->data[i]);
    }
    printf("%c", *l->data[(l->size)-1]);
}

#ifdef BUILD_TEST
int main() {
    list l;
    init_list(&l);

    int a = 0;
    int *p = &a;

    for (int i=0; i<10; i++) {
        p = (int *) malloc(sizeof(int));
        *p = a;
        printf("pushing: %d\n", *p);
        push_back(&l, (char *) p);
        a++;
    }

    del_front(&l);
    del_front(&l);
    del_front(&l);
    printf("%d\n", *((int *) get_front(&l)));

    return 0;
}
#endif

```

prob.c

```

// implementation of the probability data structure
// WARNING: data does not get freed --> memory leaks galore
// NOTE: underlying probability buckets have 5 points of precision
//      --> this value is chosen to suit most needs and is used to have

```

```

//      --> proper rounding and so sum(probs) == 1.0
//      --> allows to thousandth percent: .001% or .00001

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include "list.h"
#include "prob.h"

double precision(double num, int precision) {
    return floor(pow(10,precision)*num)/pow(10,precision);
}

// return -1 on error
// return 0 on success
int init_prob(prob *p, list *gprobs, list *gdata) {
    // min len of both lists
    int minlen = min(gprobs->size, gdata->size);

    // malloc both arrays in prob --> these are copies
    p->probs.data = (char **) malloc(sizeof(char *) * minlen);
    if (p->probs.data == NULL)
        return -1;
    p->vals.data = (char **) malloc(sizeof(char *) * minlen);
    if (p->vals.data == NULL)
        return -1;

    // set size and capacity to minlen
    p->probs.size = minlen; p->vals.size = minlen;
    p->probs.capacity = minlen; p->vals.capacity = minlen;

    // store this in struct so easy to pull out later
    p->length = minlen;

    // zip and copy
    for (int i = 0; i < minlen; i++) {
        p->probs.data[i] = gprobs->data[i];
        p->vals.data[i] = gdata->data[i];
    }

    // check for negative values
    for (int i = 0; i < minlen; i++) {

```

```

        if (prob_at(p, i) < 0)
            return -1;
    }

    // normalize
    normalize(p);

    // seed rng
    srand48(time(NULL));

    return 0;
}

void normalize(prob *p) {
    // cannot normalize 0 vector
    if (check_nonzero(p) == 0)
        return;

    // calculate magnitude
    double sum = 0;
    for (int i=0; i < p->length; i++) {
        sum += prob_at(p, i);
    }

    // divide each element by sum and round
    for (int i=0; i < p->length; i++)
        prob_at(p, i) = precision(prob_at(p, i)/sum, PROB_PRECISION);

    // last element will be automatically calculated to ensure sum(n) == 1.0
    // WARNING: slight error is allowed
    double sum_of_probs = 0;
    for (int i=0; i < p->length-1; i++)
        sum_of_probs += prob_at(p, i);
    prob_at(p, (p->length - 1)) = (1.0 - sum_of_probs);
}

// checks that probability bucket vector is non-zero
// ret 0 if zero
// ret 1 if non-zero
int check_nonzero(prob *p) {
    for (int i = 0; i < p->length; i++) {
        if (prob_at(p, i) != 0)

```

```

        return 1;
    }
    return 0;
}

list *get_probs(prob *p) { return &p->probs; }

list *get_vals(prob *p) { return &p->vals; }

int get_length(prob *p) { return p->length; }

// returns a value based on the probability distribution
char *peek(prob *p) {
    double rnd = precision(drand48(), PROB_PRECISION);

#ifdef debug
    // printf("rnd: %f\n", rnd);
#endif

    // given: [.25, .5, .25]
    // intervals: -->edge case w/ 0 : [0, .25], (.25, .75], (.75, 1.0]
    // which interval is rnd in?

    for (int i = 0; i < p->length; i++) {
        double min_interval = 0;
        double max_interval = 0;

        // for min_interval
        for (int j=0; j < i; j++)
            min_interval += prob_at(p, j);

        // for max_interval
        for (int j=0; j < (i+1); j++)
            max_interval += prob_at(p, j);

#ifdef debug
        // printf("i=%d\n%f, %f]\n\n", i, min_interval, max_interval);
#endif

        // edge case for i==0
        if (i==0) {
            if (rnd >= min_interval && rnd <= max_interval)

```

```

        return p->vals.data[i];
    }
    else{
        if (rnd > min_interval && rnd <= max_interval)
            return p->vals.data[i];
    }
}

// error
fprintf(stderr, "probability peek error: no matching value found.\n");
int val_0 = *(int*)p->vals.data[0];
double probs_0 = *(double*)p->probs.data[0];
fprintf(stderr, "Rand is %f. Length is %d. \nprobs[0] is %f. vals[0] is %d.",rnd,p-
>length, probs_0, val_0);
return NULL;
}

int print_prob_int_debug(prob* p){
    printf("Prob's length is %d\n",p->length);
    for(int i =0;i<3;i++){
        printf("probs[%d] = %f\n",i, *(double*)p->probs.data[i]);
        printf("vals[%d] = %d\n",i, *(int*)p->vals.data[i]);
    }
    return 0;
}

// changes the probabilities of the target argument
void add_probs(prob *target, prob *p2) {
    int minlen = min(target->length, p2->length);
    for (int i = 0; i < minlen; i++)
        prob_at(target, i) = prob_at(target, i) + prob_at(p2, i);

    normalize(target);
}

// changes the probabilities of the target argument
void sub_probs(prob *target, prob *p2) {
    int minlen = min(target->length, p2->length);
    for (int i = 0; i < minlen; i++)
        prob_at(target, i) = prob_at(target, i) - prob_at(p2, i);

    normalize(target);
}

```



```

}

// changes the probabilities of the target argument
void times_probs(prob *target, prob *p2) {
    int minlen = min(target->length, p2->length);
    for (int i = 0; i < minlen; i++)
        prob_at(target, i) = prob_at(target, i) * prob_at(p2, i);

    normalize(target);
}

// changes the probabilities of the target argument
void div_probs(prob *target, prob *p2) {
    int minlen = min(target->length, p2->length);
    for (int i = 0; i < minlen; i++)
        prob_at(target, i) = prob_at(target, i) / prob_at(p2, i);

    normalize(target);
}

// TODO: return pointer to malloced list of the values that
// satisfy the condition
char *get_values_by_range(prob *p, int min, int max) {
    return NULL;
}

#ifdef BUILD_TEST
int main() {
    list l1;
    init_list(&l1);
    list l2;
    init_list(&l2);

    prob prb;
    int a=5;
    int *p;
    double *d;

    puts("probabilities");

    d = (double *) malloc(sizeof(double));
    *d = .25;
}

```

```

printf("pushing: %f\n", *d);
push_back(&l1, (char *) d);

d = (double *) malloc(sizeof(double));
*d = .5;
printf("pushing: %f\n", *d);
push_back(&l1, (char *) d);

d = (double *) malloc(sizeof(double));
*d = .25;
printf("pushing: %f\n", *d);
push_back(&l1, (char *) d);

puts("values");

for (int i=0; i<3; i++) {
    p = (int *) malloc(sizeof(int));
    *p = a;
    printf("pushing: %d\n", *p);
    push_back(&l2, (char *) p);
    a++;
}

init_prob(&prb, &l1, &l2);
// print_list_int(&prb.probs);
// print_list_int(&prb.vals);

if(check_nonzero(&prb))
    printf("non zero!\n");

int x; int y; int z;
x = y = z = 0;
for(int i=0; i< 100000; i++) {
    int num = *(int *) peek(&prb);
    if (num == 5)
        x++;
    else if (num == 6)
        y++;
    else if (num == 7)
        z++;
}

```

```

printf("probs: ");
print_probs(&prb);

printf("values: ");
print_list_int(&prb.vals);

printf("results: 5: %d // 6: %d // 7: %d\n", x, y, z);

// printf("%d\n", *(int *) peek(&prb));

return 0;
}
#endif

```

polymorphicPrint.c

```

#include <stdio.h>
#include <stdlib.h>
#include "list.h"
// void printstr(const char *str)
// {
//     puts(str);
// }
int all[17576][3];
int first = 0;
int l = 0;

void printb(int x)
{
    if(x){
        printf("true");
    }
    else{
        printf("false");
    }
}

```

```

void printint(int x)
{
    printf("%d\n", x);
}
void printstr(char* x)
{
    printf("%s", x);
}

```

```

//Methods for enigma
int corresponding_int(list* x, int index) {
    char* current_elem = get_at(x,index);
    int ascii_val = (int) (*current_elem);
    return ascii_val - 65;
}

void corresponding_char(int number) {
    char element = (number + 65);
    printf("%c", element);
}

char current_char(int number) {
    char element = (number + 65);
    return element;
}

void initialize_potential_settings() {
    for(int i=0;i<26;i++){
        for(int j=0;j<26;j++){
            for(int k=0;k<26;k++){
                all[l][0] = i;
                all[l][1] = j;
                all[l][2] = k;
                l++;
            }
        }
    }
}

int key_test(int number, int index) {
    if(first==0) {
        initialize_potential_settings();
        first=first+1;
    }

    return all[number][index];
}

int modulus_operation(int x, int y) {
    return x%y;
}

//Methods defined in LRM
int stringLength(list* x) {
    return x->size;
}

int absolute(int x) {
    if (x < 0) {
        return -x;
    }
    return x;
}

void stringReverse(list* x) {
    int len = stringLength(x);

```

```

    for (int j = len-1; j > -1; j--) { //Not efficient but my code with recursion was
failing idk why :-(
        char* temp = get_at(x,j);
        char res = *(temp);
        printf("%c", res);
    }
}
int ascii(list* x) {
    char* current_elem = (get_at(x,0));
    int ascii_val = (int) *current_elem;

    return ascii_val;
}
list* int_to_char(int x) {

    char* z = malloc(1);
    *z = x;
    list *str = malloc(sizeof(struct list));
    push_back(str,z);
    return str;
}
int ceilFloat(double x) {

    int y = (int) x;
    float z = x - y;
    if(z >= 0.5) {
        return y+1;
    }
    return y;
}
int characterLocation(list* x, list* y) {
    int length_x = stringLength(x);
    char* first_element;
    char* second_element = get_at(y,0);
    for(int i=0; i < length_x;i++) {
        first_element = get_at(x,i);
        if(*first_element == *second_element) {
            return i;
        }
    }
    return -1;
}
int isCompDivisible(int x) {
    if(x%2 == 0) {
        return 1;
    }
    return 0;
}
int divisible(int x, int y) {
    if(x < y) {
        return 0;
    }
    if(x*y == 0) {
        return 1;
    }
}

```

```

    return 0;
}
int power(int x, int y) { //Exponentiation by squaring
    int squared = 1;
    for (;;)
    {
        if (y & 1)
            squared *= x;
        y >>= 1;
        if (!y)
            break;
        x *= x;
    }
    return squared;
}
void printchar(char c){
    printf("%c",c);
}
#ifdef BUILD_TEST
int main()
{
    printstr("42");
    return 0;
}
#endif

```

testall.sh

```

#!/bin/sh

# Regression testing script for SMAP
# Step through a list of files
# Compile, run, and check the output of each expected-to-work test
# Compile and check the error of each expected-to-fail test

# Path to the LLVM interpreter
LLI="lli"
#LLI="/usr/local/opt/llvm/bin/lli"

# Path to the LLVM compiler
LLC="llc"

# Path to the C compiler
CC="cc"

# Path to the microc compiler. Usually "./microc.native"

```

```

# Try "_build/microc.native" if ocamlbuild was unable to create a symbolic link.
SMAP="./smap.native"
#SMAP="_build/smap.native"

# Set time limit for all operations
ulimit -t 30

globallog=testall.log
rm -f $globallog
error=0
globalerror=0

keep=0

Usage() {
    echo "Usage: testall.sh [options] [.smap files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}

SignalError() {
    if [ $error -eq 0 ] ; then
        echo "FAILED"
        error=1
    fi
    echo " $1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile. Differences, if any, written to difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
        SignalError "$1 differs"
        echo "FAILED $1 differs from $2" 1>&2
    }
}

# Run <args>
# Report the command, run it, and report any errors

```

```

Run() {
    echo $* 1>&2
    eval $* || {
        SignalError "$1 failed on $*"
        return 1
    }
}

# RunFail <args>
# Report the command, run it, and expect an error
RunFail() {
    echo $* 1>&2
    eval $* && {
        SignalError "failed: $* did not report an error"
        return 1
    }
    return 0
}

Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\\/\\\/\\\/
                s/.mc//'\`
    reffile=`echo $1 | sed 's/.mc$//'\`
    basedir=`echo $1 | sed 's/\/[^\/]*$//'\`/."

    echo -n "$basename..."

    echo 1>&2
    echo "##### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.ll ${basename}.s ${basename}.exe
${basename}.out" &&
    Run "$SMAP" "$1" ">" "${basename}.ll" &&
    Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">" "${basename}.s" &&
    Run "$CC" "-o" "${basename}.exe" "${basename}.s" "polymorphicPrint.o" "prob.o"
"list.o" "testMakeStruct.o" "-lm" "-lncurses" &&
    Run "./${basename}.exe" > "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

```



```

# Report the status and clean up the generated files

if [ $error -eq 0 ] ; then
if [ $keep -eq 0 ] ; then
    rm -f $generatedfiles
fi
echo "OK"
echo "##### SUCCESS" 1>&2
else
echo "##### FAILED" 1>&2
globalerror=$error
fi
}

CheckFail() {
    error=0
    basename=`echo $1 | sed 's/.*\\\/\\\/
                s/.mc//'\`
    reffile=`echo $1 | sed 's/.mc$//'\`
    basedir=""`echo $1 | sed 's/\/[^\/]*$//'\`/."

    echo -n "$basename..."

    echo 1>&2
    echo "##### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
    RunFail "$SMAP" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
    Compare ${basename}.err ${reffile}.err ${basename}.diff

# Report the status and clean up the generated files

if [ $error -eq 0 ] ; then
if [ $keep -eq 0 ] ; then
    rm -f $generatedfiles
fi
echo "OK"
echo "##### SUCCESS" 1>&2
else
echo "##### FAILED" 1>&2

```

```

    globalerror=$error
    fi
}

while getopts kdpsh c; do
    case $c in
        k) # Keep intermediate files
            keep=1
            ;;
        h) # Help
            Usage
            ;;
    esac
done

shift `expr $OPTIND - 1`

LLIFail() {
    echo "Could not find the LLVM interpreter \"$LLI\"."
    echo "Check your LLVM installation and/or modify the LLI variable in testall.sh"
    exit 1
}

which "$LLI" >> $globallog || LLIFail

if [ $# -ge 1 ]
then
    files=$@
else
    files="test/test-*.smap test/fail-*.smap"
fi

for file in $files
do
    case $file in
        *test-*)
            Check $file 2>> $globallog
            ;;
        *fail-*)
            CheckFail $file 2>> $globallog
            ;;
    *)

```

```
    echo "unknown file type $file"
    globalerror=1
    ;;
esac
done

exit $globalerror
```
