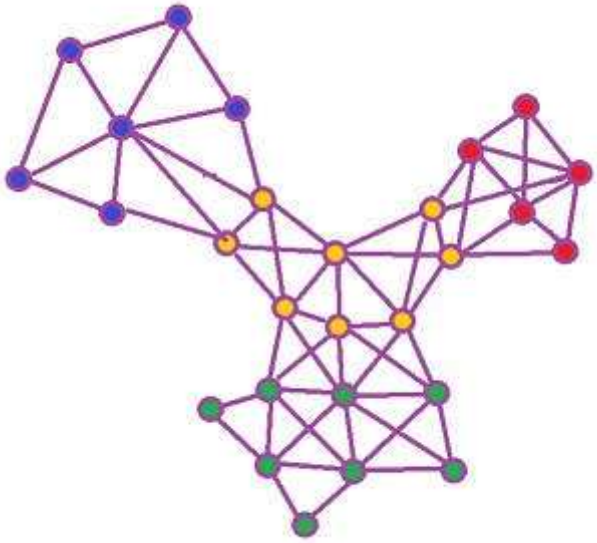
GIRAPHE



Final Report

Name	UNI
Dianya Jiang	dj2459
Vince Pallone	vgp2105
Minh Truong	mt3077
Tongyun Wu	tw2568
Yoki Yuan	yy2738

Outline

1. Introduction
2. Tutorial
3. Language Reference Manual
4. Project Plan
5. Architectural Design
6. Testing
7. Lessons Learned
8. Appendix

1. Introduction

Graphs appear naturally in problems of various areas like chemistry, sociology, and many other disciplines. It intuitively express entities and complex relationships among them. It is especially prevalent in the fields of computer science, math, and data science where graphs are often used to represent data and social networks.

But there does not seem to exist any computer language that provides the essential features in defining and utilizing graphs. In most cases, you need to define a set of vertexes and edges in order to define a graph, which is complicated and not very efficient. We built a language that has many graph-based features which will allow users to effectively define, manipulate, and process their graph data.

GIRAPHE aims to be mathematical language with clean syntax as well as support for graphs, nodes, edges, and lists. The main goal of the language is to provide an easy way to work with graphs and compute algorithms such as depth-first search or Dijkstra's. The same general components as MicroC are used: scanner, parser, ast, semantic check, and code generation. Many of our features come from a vast C library that compiles to LLVM IR, which is then linked to the output of codegen.

2. Tutorial

First, clone the repo from GitHub and switch to the directory

```
$ git clone https://github.com/minhptruong/COMS4115_PLT.git
$ cd COMS4115_PLT
```

Then, create the GIRAPHE to LLVM compiler, “giraphe.native”

```
$ make
```

Next, compile and run any GIRAPHE program with our script

```
$ ./run.sh <file_name>
```

It compiles the C library to LLVM IR, links it with “giraphe.native” output, and generates an executable using CLANG.

After you finish running the program, just run the following to delete all build files

```
$ make clean
```

3. Language Reference Manual

3.1 Lexical Elements

3.1.1 Identifiers

Identifiers are strings used for naming different elements, such as variables, functions, classes. These identifiers are case sensitive, and can involve letters, digits, and underscore ‘_’, but should always start with a letter. These rules are described by the definitions involving regular expressions below:

```
      identifier  :=  (letter) (letter | digit | underscore)*
           digit  :=  ‘0’ - ‘9’
          letter  :=  uppercase_letter | lowercase_letter
uppercase_letter  :=  ‘A’ - ‘Z’
lowercase_letter  :=  ‘a’ - ‘z’
```

3.1.2 Keywords

```
if else while main return int bool float string list hashmap node edge graph null void for break continue
```

3.1.3 Literals

3.1.3.1 String Literals

- string literals are ASCII characters inside double quotation marks. C escape character rules are applied as same.

Escape Sequence	Description
\”	Insert a double quote at this point.
\\	Insert a backslash at this point
\n	Insert a newline at this point
\t	Insert a tab at this point
\r	Insert a carriage return at this point
\b	Insert a backspace at this point
\f	Insert a form feed at this point

3.1.3.2 Integer Literals

- A valid integer literal is a decimal value in the valid integer value range

3.1.3.3 Float Literals

- A valid float literal is a floating point number in the valid integer value range

3.1.3.4 Boolean Literals

- A boolean literal represents a truth value and can have the value ***true*** or ***false***

3.1.4 Delimiters

3.1.4.1 Parentheses and Braces

Parentheses are used to force evaluation of parts of a program in a specific order. They are also used to enclose arguments for a function.

3.1.4.2 Commas

Commas are used to separate function arguments.

3.1.4.3 Brackets

Brackets are used for list initialization

3.1.4.4 Semicolon

A semicolon is used to terminate a sequence of code.

3.1.4.5 Curly Braces

Curly braces are used to enclose function definitions, blocks of code, function definitions. In general, blocks enclosed within curly braces do not need to be terminated with semicolons.

3.1.4.6 Periods

Periods are used for accessing fields of object.

3.1.4.7 Whitespace

Whitespace (unless used in a string literal) is used to separate tokens, but has no special meaning otherwise. List of whitespace characters: spaces, tabs, newlines, vertical tabs, and formfeed characters.

3.2 Data Types

GIRAPHE is statically typed. The types of all variables are known at compile time and cannot be changed.

3.2.1 Primitive Data Types

int (5)	32-bit signed integers that can range from −2,147,483,648 to 2,147,483,647
float (8.7)	Represented by IEEE 754 double-precision 64-bit number
string (“Hello!”)	A set of chars
bool (true)	true or false value
null	null
void	Empty return type

3.2.2 Non-Primitive Data Types

list	A sequence of the same type of data in square braces separated by commas
hashmap	A hashed set of keys and values of any type
node	A vertex represents in (x,y)
edge	A line between two nodes, possibly with weight and/or direction
graph	A set of nodes and edges

3.2.2.1 Declaring lists

- Declare a list literal by specifying a type, followed by brackets enclosing the number of elements in the list, followed by an identifier for the list
- All lists dynamically sized
- The following will declare a list of integers called “myList”

```
list<int> myList;
```

- The following will initialize a list of element size 4

```
myList = [1, 2, 3, 4];
```

3.2.2.2 Accessing and setting list elements

- List elements can be accessed by providing the desired index of the element you wish to access enclosed in brackets next to the identifier
- The following will set the element at index 1 of “myList” to 0

```
myList.set(1, 0);
```

3.2.2.3 List length

- The following will return the length of list “myList”

```
myList.size();
```

3.2.2.4 List add

- To add an element to “myList”

```
myList.add(5);
[1, 2, 3] + [4, 5, 6];
[1, 2, 3] +4;
```

3.2.2.5 List remove

- To remove the element of “myList”

```
myList.remove(5);
[1, 2, 3] - 3;
```

3.2.2.6 List contains

- To determine if specific value i is in “myList”, return true or false

```
myList.contains(i);
```

3.2.3 Nodes

3.2.3.1 Node initialize

- To initialize a node with no edges, use the node function, can be any type

```
node n = node(“this is node”);
node n = node(1>2);
```

3.2.3.2 Node visited flag

- To set and get the value of the visited flag of a specific node, use the

```
n.setVisited(); // set n as visited;
n.isVisited(); // determine if n has been visited
```

3.2.4 Edges

- Edge is built-in as underlying c library data-structure which will not be shown explicitly in our language.

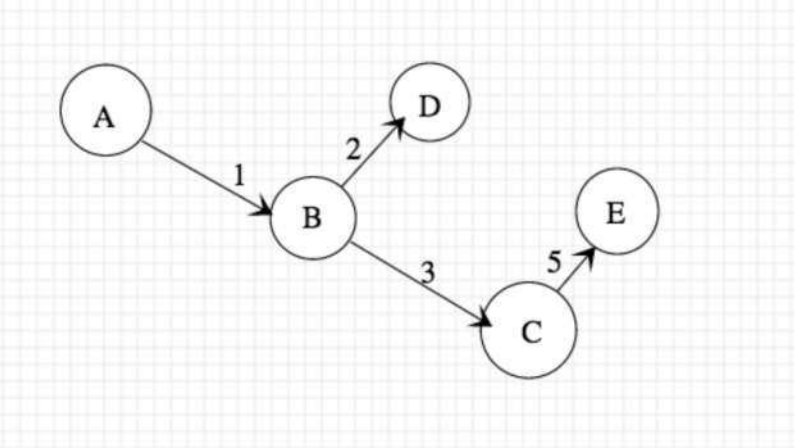
3.2.5 Graphs

3.2.5.1 Graph initialize

- To initialize a graph with no nodes or edges, use the following specific form:

graph g = a->1$b->3$c + b->2$d + c -> 5$e

The above assignment use the "->" symbol to separate source from destination. The number between "->" and "$" symbols is the weight of the edge from the source to destination. By using "+", we can link all the sub graph into a whole graph.



3.2.5.2 get Graph size

- To get size of nodes in the graph:

```
int graphSize = g.size();
```

3.2.5.3 get all nodes of Graph

- To get get all nodes of graph:

```
list<node> nodes = g.getAllNodes();
```

3.2.5.4 set all nodes of the graph as unvisited

- To set all nodes as unvisited:

```
g.setAllUnvisited();
```

This function is very useful in the application of our language, when dealing with cyclic graph, doing dfs, bfs dijkstra and so on.

3.2.5.5 run dfs on the graph

- Run dfs from node a:

  g.dfs(a);

  A very useful build-in function of our language

3.2.5.6 run bfs on the graph

- Run bfs from node a:

  g.bfs(a);

  Similar to dfs, basic graph algorithm

3.2.5.7 determine if graph has a specific node:

- Determine if node a is in graph, return true or false:

  g.hasNode(a);

3.2.5.8 determine if graph has a specific edge:

- Determine if there is an edge between node a and node b in graph:

  g.hasEdge(a, b);

3.2.5.9 add a node to the Graph:

- Add a node c into graph:

  g.addNode(c);

  Add node is enough, no need to add adge.

3.2.5.10 add a edge to the Graph:

- Add an edge from node a to node b with weight 3:

  g.addEdge(a, b, 3);

  If there is already edge between source and destination nodes, we just update the weight. The source and destination node can be new.

3.2.5.10 link Graph

- Link as much graphs as we want as long as they have shared edge or nodes:

  Graph g = g1 + g2 +g3;

  If there is no shared edge or node, just return the first graph.

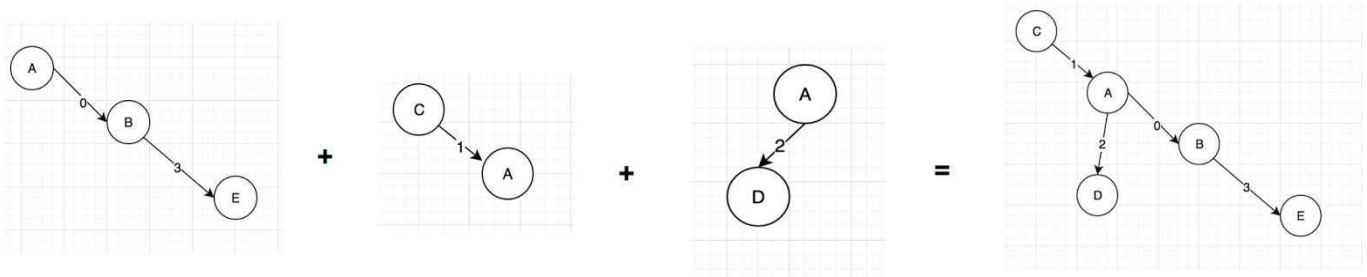
```
node a = node("A");
node b = node("B");
node c = node("C");
node d = node("D");
node e = node("E");

print("Using + to link graph");

print("Link graphs: A -> 0$B ->3$E +  C -> 1$A + A -> 2$D");
print("Shared nodes: A");

graph g = a -> 0$b ->3$e +  c -> 1$a + a -> 2$d;

print(g);
```



3.2.6 Map

3.2.6.1 Declaring a map

- To initialize an empty map

  map<string, int> myMap = init_map(4,4);

3.2.6.2 Adding key and value

- To add the key and value in the map

  put(myMap, “fun”, 4115);

3.2.6.3 Clear map

- To clear all the keys and values in a map

```
clear(myMap);
```

3.2.6.4 Map size

- To get the size of the map

```
size(myMap);
```

3.2.6.5 Map remove

- To remove a value associate with a key

```
remove(myMap, “fun”);
```

3.2.6.6 Get value from map

- To get a value associate with a key

```
get(myMap, “fun”);
```

3.2.6.7 Check key

- Return true if the map contains the key, otherwise return false

```
containsKey(myMap, ”fun”);
```

3.2.6.8 Check value

- Return true if the map contains the value, otherwise return false

```
containsValue(myMap, 4115);
```

3.3 Expressions and Operators

3.3.1 Expressions

Expressions consist of one or more operands with zero or more operators. Innermost expressions are evaluated first, as determined by grouping into parentheses, and operator

precedence helps determine order of evaluation. Expressions are otherwise evaluated left to right.

3.3.2 Operators

3.3.2.1 Basic

The table below presents the language operators (including assignment operators, mathematical operators, logical operators, and comparison operators), descriptions, and associativity rules. Operator precedence is highest at the top and lowest at the bottom of the table.

Operator	Description	Associativity
!	Logical Not	Right-to-left
=	Assignment	
->	Assign direction for graph	
$	Assign weight for graph	
* / %	Multiplication, division, remainder	Left-to-right
+ -	Addition, subtraction	
< <= > >=	Inequality Operators	
== !=	Comparison Operators	
&&	Logical And	
||	Logical Or	

3.4 Control Flow

;	end of line
//	Start of a one line comment
/*	Start of a comment block
*/	End of a comment block

if (expression) {
  ...
}
	Execute the statements between curly braces if expression is true

if (expression) {
  ...
} else {
  ...
}
	Can have an optional else statement. One can also nest the if-else statement.

while (loop condition) {
  ...
}
	The while statement is used to execute a block of code continuously in a loop until the specified condition is no longer met. If the condition is not met upon initially reaching the while loop, the code is never executed

for (initialization; loop condition; increment;)
{
  ...
}
	Iterate until loop condition is met, incrementing each time

continue	Continue to next iteration of the loop
break	Break out of the loop
return	Return signals that what follows it should be the return values

3.5 Functions

3.5.1 Function Definitions

- Function definitions consist of a return type, a function identifier, a set of parameters and their types, and then a block of code to execute when that function is called with the specified parameters. An example of an addition function definition is as follows:

```
int sum(int a, int b)  { return a + b; }
```

3.5.2 Calling Functions

- A function can be called its identifier followed by its params in parentheses. for example:

```
sum(0,100);
```

3.6 Program Structure and Scope

3.6.1 Scope

Our language implements static scope rule. Besides global scope, each function will open a new scope. Inner function can access the variable of the outer scope including the static scope. Declarations made within blocks of an if statement, a while statement, or a function definition are only available for reference within that block.

3.7 Built-in Functions

3.7.1 The print function

The print function can be used to print out strings, integers, and booleans, node, graph, list, hashmap to the command line, because we have define print type for every type. The general structure for calling the print function is as follows:

```
print ( "GIRAPHE" );
print ( 62 );
```

API of Graph(G)		
Name	Function Expression	Description
addNode	G.addNode(n)	Add node to graph
addEdge	G.addEdge(a, b, 3)	Add edge from a to b with weight 3
hasNode	G.hasNode(n)	Check whether node n is in the graph
size	G.size()	Return the number of nodes in G
root	G.root()	Return the root of the graph
setAllUnvisited	G.setAllUnvisited	Set all nodes of the graph as unvisited
dfs	G.dfs(a)	Run dfs on graph
bfs	G.bfs(a)	Run bfs on graph
getAllNodes	G.getAllNodes()	returns the list of nodes in the graph
dijkstra	G.dijkstra(a)	Dijkstra algorithm implementation

API of Node(N)		
Name	Function Expression	Description
setVisited	n.setVisited()	Set node n to be visited
isVisited	n.isVisited()	Determine if node n is visited

4. Project Plan

4.1 Style Guide

Working with multiple editors, languages, and environments made sticking to a style very important. Anytime possibly confusing code is added, comments must follow. We use two spaces for indentation with a space between each expression. Rewriting code is heavily frowned upon, while refactoring when necessary is encouraged. In general, our code follows the style already set in the MicroC compiler. For the GIRAPHE style, follow Java best practices. Between C and OCaml, both camel case and snake case were used.

4.2 Timeline

Feb 8	Proposal
Feb 20	Scanner first draft complete
Feb 22	LRM first draft complete
Feb 27	Parser first draft complete
Mar 6	AST first draft complete
Mar 20	Codegen implement string type
Mar 20	Codegen implement print string function
Mar 27	GIRAPHE Hello World
Apr 4	List C library finished
Apr 7	Hashmap, Edge, Node C library finished
Apr 12	Graph C library finished
Apr 10	Add non-primitive types to scanner, parser, ast
Apr 17	Work out issues between front and back end
Apr 24	Begin linking C library to codegen
May 3	List, Hashmap, Graph, Node,Edge linked
May 4	Queue C library finished and linked
May 5	Graph basic function added, binary operation for list simply defined
May 6	Fix Graph basic function, test cases added, semantic check added
May 7	Minheap C library started, bfs,dfs finished
May 8	Minheap linked, dijkstra C library started.
May 9	Dijkstra linked, filter function started, major bug fixes, test case finished
May 10	Filter function finished, and cleanup

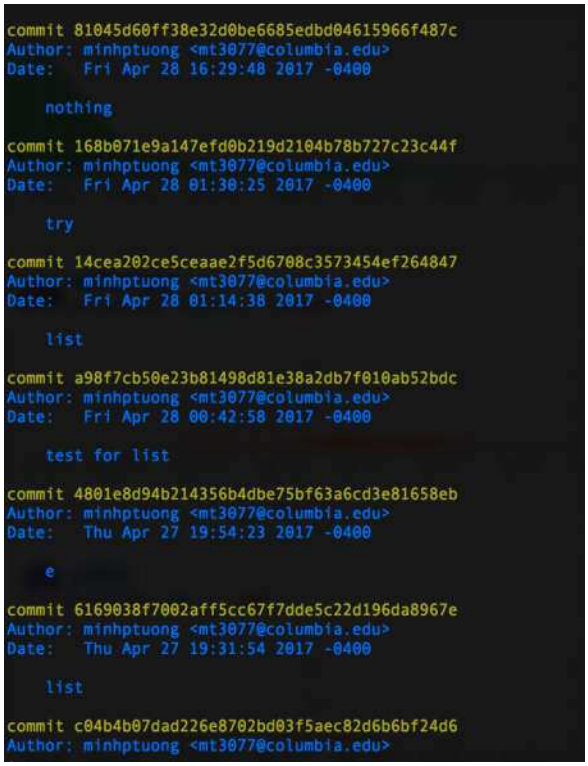
4.3 Roles and Responsibilities

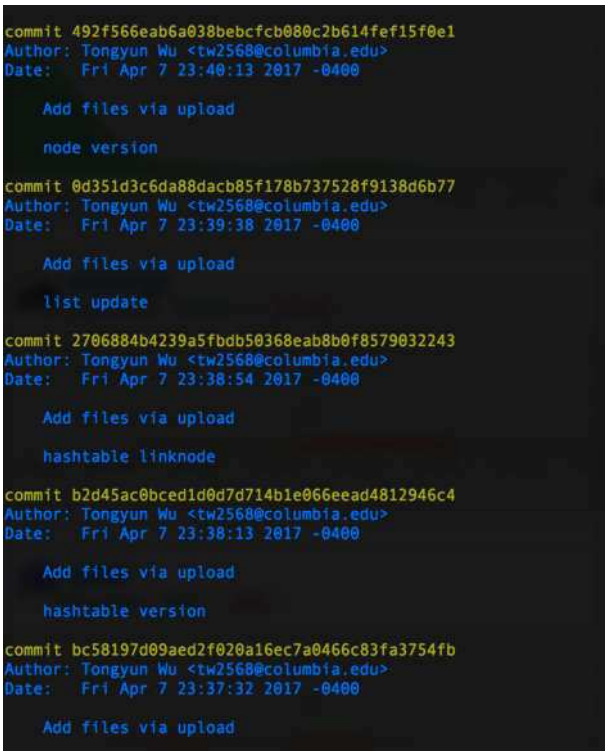
Dianya Jiang:	Project planning, Test case, Scanner, Parser
Minh Truong:	Scanner, Linking C Libraries, Code generation
Tongyun Wu:	AST, Parser, Code generation, Writing C Libraries
Vince Pallone:	Scanner, Linking C Libraries, Code generation
Yoki Yuan:	Semantics, Parser, Sast, Checker, Writing C Library

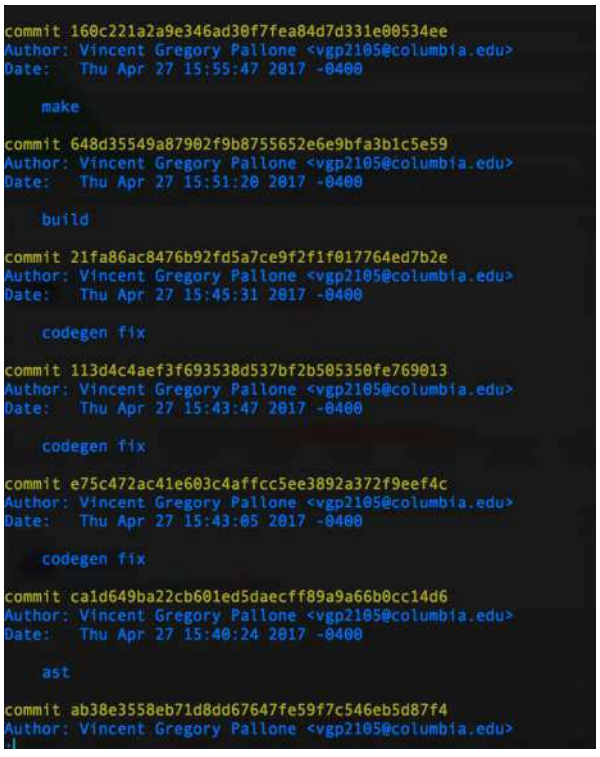
4.4 Development Tools

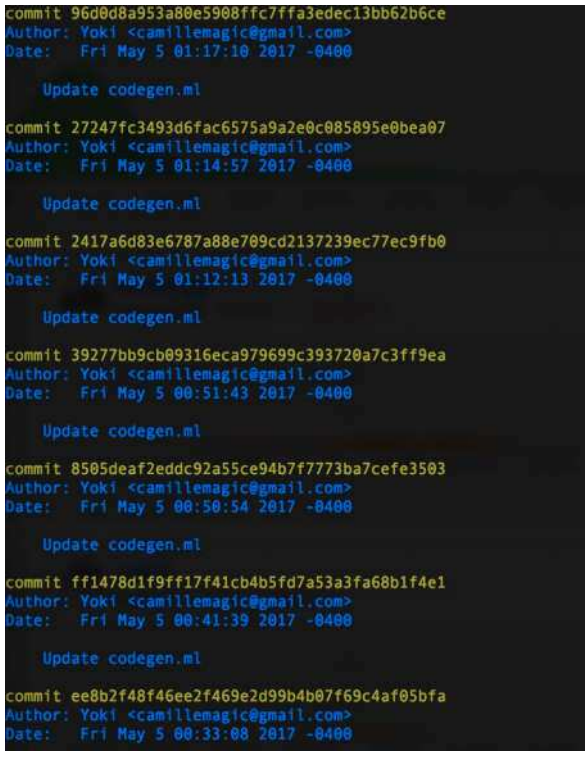
Slack: for team communication
Github: push and save code
Ubuntu: run code and debug

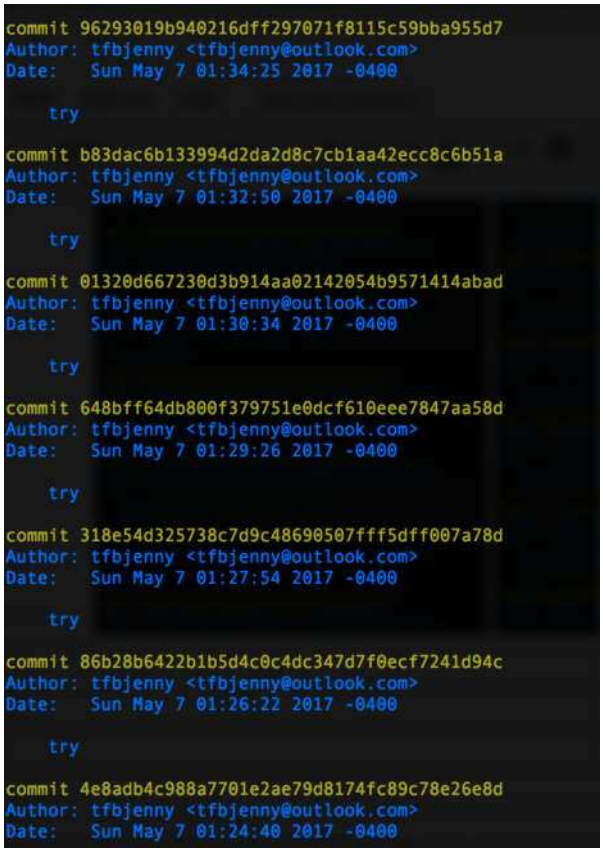
4.5 Commits


commit 81045d60ff38e32d0be6685edbd04615966f487c
Author: minhptuong <mt3077@columbia.edu>
Date: Fri Apr 28 16:29:48 2017 -0400
nothing
commit 168b071e9a147efd0b219d2104b78b727c23c44f
Author: minhptuong <mt3077@columbia.edu>
Date: Fri Apr 28 01:30:25 2017 -0400
try
commit 14cea202ce5ceaae2f5d6708c3573454ef264847
Author: minhptuong <mt3077@columbia.edu>
Date: Fri Apr 28 01:14:38 2017 -0400
list
commit a98f7cb50e23b81498d81e38a2db7f010ab52bdc
Author: minhptuong <mt3077@columbia.edu>
Date: Fri Apr 28 00:42:58 2017 -0400
test for list
commit 4801e8d94b214356b4dbe75bf63a6cd3e81658eb
Author: minhptuong <mt3077@columbia.edu>
Date: Thu Apr 27 19:54:23 2017 -0400
e
commit 6169038f7002aff5cc67f7dde5c22d196da8967e
Author: minhptuong <mt3077@columbia.edu>
Date: Thu Apr 27 19:31:54 2017 -0400
list
commit c04b4b07dad226e8702bd03f5aec82d6b6bf24d6
Author: minhptuong <mt3077@columbia.edu>


commit 492f566eab6a038bebcfcb080c2b614fef15f0e1
Author: Tongyun Wu <tw2568@columbia.edu>
Date: Fri Apr 7 23:40:13 2017 -0400
Add files via upload
node version
commit 0d351d3c6da88dacb85f178b737528f9138d6b77
Author: Tongyun Wu <tw2568@columbia.edu>
Date: Fri Apr 7 23:39:38 2017 -0400
Add files via upload
list update
commit 2706884b4239a5fbdb50368eab8b0f8579032243
Author: Tongyun Wu <tw2568@columbia.edu>
Date: Fri Apr 7 23:38:54 2017 -0400
Add files via upload
hashtable linknode
commit b2d45ac0bced1d0d7d714b1e066eead4812946c4
Author: Tongyun Wu <tw2568@columbia.edu>
Date: Fri Apr 7 23:38:13 2017 -0400
Add files via upload
hashtable version
commit bc58197d09aed2f020a16ec7a0466c83fa3754fb
Author: Tongyun Wu <tw2568@columbia.edu>
Date: Fri Apr 7 23:37:32 2017 -0400
Add files via upload


commit 160c221a2a9e346ad30f7fea84d7d331e00534ee
Author: Vincent Gregory Pallone <vgp2105@columbia.edu>
Date: Thu Apr 27 15:55:47 2017 -0400
make
commit 648d35549a87902f9b8755652e6e9bfa3b1c5e59
Author: Vincent Gregory Pallone <vgp2105@columbia.edu>
Date: Thu Apr 27 15:51:20 2017 -0400
build
commit 21fa86ac8476b92fd5a7ce9f2f1f017764ed7b2e
Author: Vincent Gregory Pallone <vgp2105@columbia.edu>
Date: Thu Apr 27 15:45:31 2017 -0400
codegen fix
commit 113d4c4aef3f693538d537bf2b505350fe769013
Author: Vincent Gregory Pallone <vgp2105@columbia.edu>
Date: Thu Apr 27 15:43:47 2017 -0400
codegen fix
commit e75c472ac41e603c4affcc5ee3892a372f9eef4c
Author: Vincent Gregory Pallone <vgp2105@columbia.edu>
Date: Thu Apr 27 15:43:05 2017 -0400
codegen fix
commit ca1d649ba22cb601ed5daecff89a9a66b0cc14d6
Author: Vincent Gregory Pallone <vgp2105@columbia.edu>
Date: Thu Apr 27 15:40:24 2017 -0400
ast
commit ab38e3558eb71d8dd67647fe59f7c546eb5d87f4
Author: Vincent Gregory Pallone <vgp2105@columbia.edu>

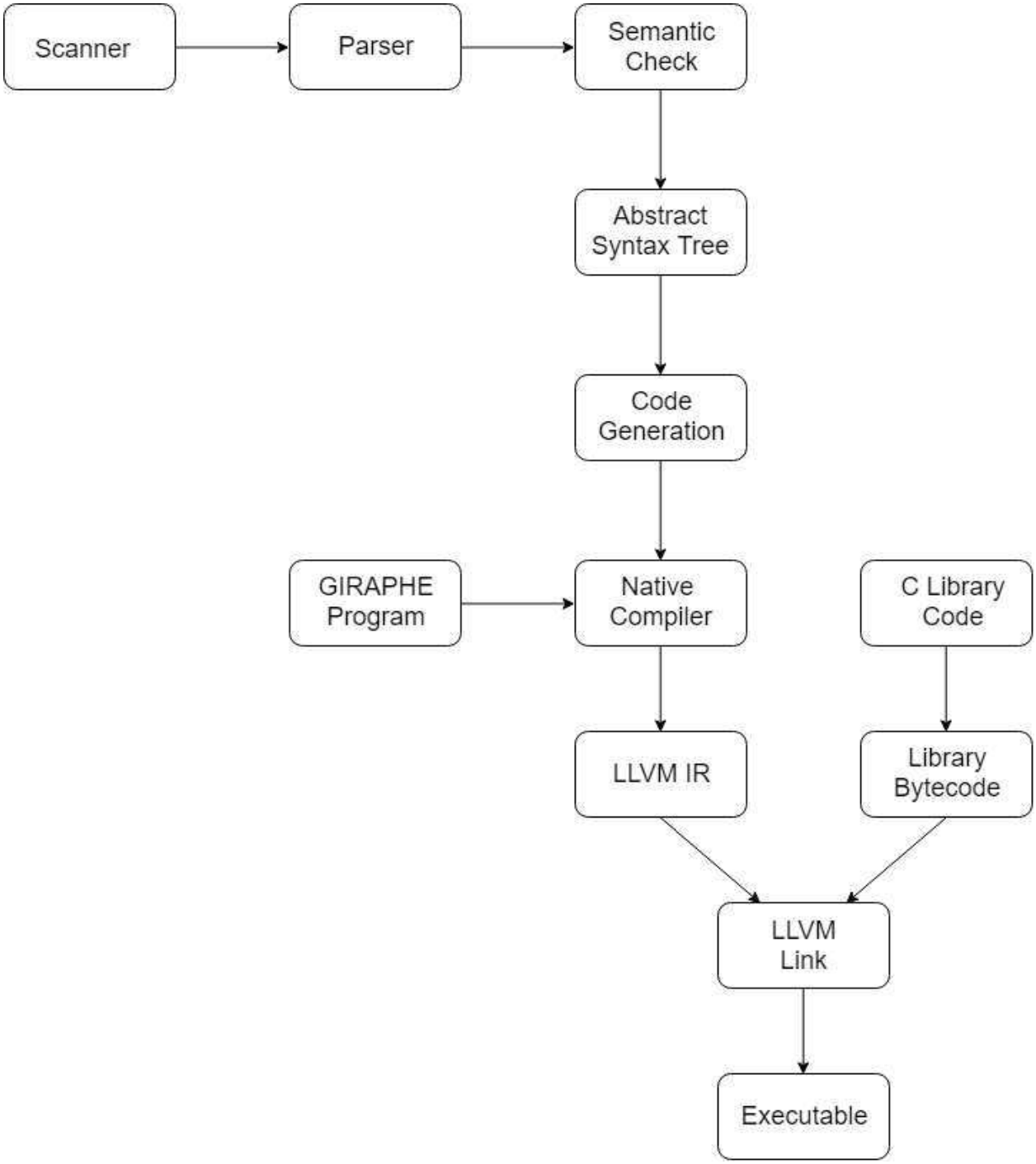

commit 96d0d8a953a80e5908ffc7ffa3edec13bb62b6ce
Author: Yoki <camillemagic@gmail.com>
Date: Fri May 5 01:17:10 2017 -0400
Update codegen.ml
commit 27247fc3493d6fac6575a9a2e0c085895e0bea07
Author: Yoki <camillemagic@gmail.com>
Date: Fri May 5 01:14:57 2017 -0400
Update codegen.ml
commit 2417a6d83e6787a88e709cd2137239ec77ec9fb0
Author: Yoki <camillemagic@gmail.com>
Date: Fri May 5 01:12:13 2017 -0400
Update codegen.ml
commit 39277bb9cb09316eca979699c393720a7c3ff9ea
Author: Yoki <camillemagic@gmail.com>
Date: Fri May 5 00:51:43 2017 -0400
Update codegen.ml
commit 8505deaf2eddc92a55ce94b7f7773ba7cefe3503
Author: Yoki <camillemagic@gmail.com>
Date: Fri May 5 00:50:54 2017 -0400
Update codegen.ml
commit ff1478d1f9ff17f41cb4b5fd7a53a3fa68b1f4e1
Author: Yoki <camillemagic@gmail.com>
Date: Fri May 5 00:41:39 2017 -0400
Update codegen.ml
commit ee8b2f48f46ee2f469e2d99b4b07f69c4af05bfa
Author: Yoki <camillemagic@gmail.com>
Date: Fri May 5 00:33:08 2017 -0400



5. Architecture Design

Let me know if this needs changes, i made it (vince)

AST -> Checker -> SAST-> Code gen (something like this)



5.1 Scanner (scanner.mll)

The scanner takes the GIRAPHE source code and splits it into tokens based on which characters or strings it sees in a particular order. Comments and whitespace are discarded.

Literals, keywords, and identifiers are all read in as the compiler begins identify the structure of the program. The scanner feeds this tokenized input to the parser.

5.2 Parser (parser.mly) and Abstract Syntax Tree (ast.ml)

The Parser uses the tokens generated from the Scanner to construct the Abstract Syntax Tree. The AST has an entry point requiring a list of function declarations and a list of variable declarations. These boil down to the expressions and statements written in GIRAPHE source code. Scanner and Parser are important in GIRAPHE, because as a graph language, we need to design many symbols to represent the relationship between nodes, edges and so on, that is where scanner and parser works. So we design very carefully to make it perfectly match with what we want.

5.3 Semantic Check (semant.ml)

Semantic Check is very important for our language, because we need to check, firstly, basic implementation like return, args, locals and body. For example, we made invalid return type, invalid assignment and so many basic tests, to make sure we handle all the corner cases and raise proper error message.

Another important part is function test, as GIRAPHE is a graph language, unlike other languages like Java, Python which has lots of semantic check format and raise error template, we need to carefully analyze the possible error and continuously update and promote our semantic check. As developing GIRAPHE, we constantly found many errors in semantic check, and we need to make changes for it. As this job is tough and hard, out semantic checker Yoki designed very funny and cute raise error message, to make our language more special.

5.4 Code Generation (codegen.ml)

There is no doubt that code generation is definitely a hard part for us in developing our language. As we have many build-in functions, code generation really plays an important role in design. The input of the code generator is the semantic-checked AST. The output of the code generator is a .ll file in LLVM syntax.

As MicroC is a very good tutorial, we basically borrow the structure of it. For example, Declare the context, linked C library, data types, functions--whether complex or simple. It helps to translate all the functions and programs in the program.

5.5 C Library (lib.c)

Our library contains structs and algorithms that have been implemented in C which have been linked into our language and are used to power many of the data structures and graphing functions in our language. For example, some of the C code that has been linked into our language include various structs that form the underlying infrastructure of our Graph data structure. Additionally various graphing algorithms that are available to our Graph data structures, such as Dijkstra's, are implemented in C and linked into LLVM.

Compilation command in order to link our C library:

ocamlbuild -use-ocamlfind -pkgs llvm,llvm.analysis,llvm.linker,llvm.bitreader,llvm.irreader -cflags -w,+a-4 giraphe.native

clang -emit-llvm -o clibary.bc -c clibrary.c -Wno-varargs

C library plays an important role in GIRAPHE functions, because most of our functions are build-in, to design a workable and efficient C language program seems crucial. For C library function, we can divide them into following several parts:

1. List

In GIRAPHE, list is similar to ArrayList in Java, the size is not fixed. Type need to be defined when initialize.

- Add/Remove element

  GIRAPHE supports add element at the end of the list or remove from any position.

```
list<int> a = [1, 2, 3, 4];
a.add(5)   is  same as  a = a + 5  => [1, 2, 3, 4, 5];
a.remove(6) => return [1, 2, 3, 4];
a.remove(2) => return [1, 3, 4];
a.contains(1) => return true;
a.contains(8) => return false;
```

- Concatenate list
  We also support list concatenate:
  list<int> a = [1, 2];
  list<int> b = [3, 4];
  a + b => [1, 2, 3, 4];

- Get size, get and set element
  a.get(i);
  a.set(0,5);
  a.size();

2. Node & Edge

- Add node, add edge

We can add a node to graph, without needing edge, and we can also add edge to graph while indicates the source and destination node. If there already exists an edge between nodes, we just update the value of edge, or we create an edge between nodes.

3. Graph

- The graph stores nodes and edges inside the graph. It has methods for adding nodes and edges, query if an edge or a node is in the graph. Besides, several graph algorithms are also built into the graph such as DFS, BFS as well as Dijkstra.

4. Hashmap

- The hashmap implements the functionality of a dictionary. It stores key-value pair. It allows fast query and access of element.

5. Minheap

- The minheap is implemented using list as underlying data structure. It can suppport insert data and output min value element of the heap.

6. Queue

- The queue is implemented using two stack. It maintains the FIFO property.

5.6 Cool Feature

- Course Prerequisite Relationship

Topological sort is very useful in our daily life for various problems like scheduling, and distributing computational load. Here we take an example, which aims at benefiting students at Columbia Computer Science.

We put many Computer Science courses in Columbia University as nodes, and use edge to indicate their prerequisite relationship. For example, if you want to take Algorithm II, you should take Algorithm I to make sure you are fully prepared to handle the work. The graph is as follows:

The code is as follows:

```
node a = node("COMS 1004: Introduction to Computer Science and Programming in Java");
node b = node("COMS 4111: Introduction to Databases");
node c = node("COMS 6111: Advanced Database Systems");
node d = node("CSEE 4119: Computer Networks");
node e = node("CSEE 4140: Networking Laboratory");
node f = node("COMS 4118: Operating Systems");
node g = node("COMS 3101: Programming Language Python");
node h = node("CSOR 4231: Analysis of Algorithms I");
node i1 = node("COMS 6232: Analysis of Algorithms II");
node j = node("COMS 4771: Machine Learning");
node k = node("COMS 4772: Advanced Machine Learning");
node l = node("COMS 4995: Deep Learning for Computer Vision");

graph ga = a -> b -> f + b -> c + a -> d -> f + d -> e + h -> f + g -> h -> j -> l + h -> i1 + j -> k;

int i=0;
list<node> n = ga.getAllNodes();
list<node> tmp = n;
list<node> res = n;
int stmp = 0;
ga.setAllUnvisited();
```

```
while(i<ga.size()){
        if(n.get(i).isVisited()){
                i = i + 1;
        }else{
                tmp = ga.dfs(n.get(i));
                stmp = tmp.size();
                while(stmp > 0) {
                        res.add(tmp.get(stmp-1));
                        stmp = stmp - 1;
                }
                i = i + 1;
        }
}

while(i > 0) {
        print(res.get(i-1+ga.size()));
        i=i-1;
}
```

We use dfs to do the topological sort, we can also implement using bfs.

Then we get final output.

node COMS 3101: Programming Language Python
node CSOR 4231: Analysis of Algorithms I
node COMS 6232: Analysis of Algorithms II
node COMS 4771: Machine Learning
node COMS 4772: Advanced Machine Learning
node COMS 4995: Deep Learning for Computer Vision
node COMS 1004: Introduction to Computer Science and Programming in Java
node CSEE 4119: Computer Networks
node CSEE 4140: Networking Laboratory
node COMS 4111: Introduction to Databases
node COMS 6111: Advanced Database Systems
node COMS 4118: Operating Systems

By knowing the prerequisite relationship between courses, students can better schedule their class registration, and choose a field that they want to dig into.

- Find nearest restaurant

This example also aims at benefiting Columbia students. We divide the location around Columbia into many types, for example, restaurants, gyms, libraries, shopping centers and so on. And give a start point and target type destination. For instance, I want to have lunch when class is over, I just input my current location -- Columbia University and type -- Restaurant, it will print a shortest path to reach the nearest restaurant around me.

The code is as follows:

```
node b = node("Vine");
node c = node("Xian Famous Food");
node d = node("Butler Library");
node e = node("WestSide Market");
node f = node("Duane Reade");

graph gh = a -> 1$d ->8$c + a -> 1$d -> 2$b -> 1$e -> 1$f -> 1$c + a ->5$b;
list<node> path = [a];
node cur = a;
int curVal = 0;
print("1 : Restaurant");
print("2 : Unversity Building");
print("3 : Library");
print("4 : SuperMarket");
print("5 : Pharmacy");
print("");
hashmap<int> hmap = { a : 2, b : 1, c : 1, d : 3, e : 4, f : 5};

void filter(node sour, int target) {
   print("-- try to find places of target type --");
   print(target);
   list<node> nodes = gh.getAllNodes();
   int size = nodes.size();
   int i = 0;
   while (i < size) {
      cur = nodes.get(i);
      curVal = hmap.get(cur);
```

```
        if (curVal == target) {
            print("");
            print("--------- have found target ------------");
            print("---------     target      --------------");
            print(cur);
            print("--------- shortest path to target  -----");
            path = gh.dijkstra(sour, cur);
            print(path);
            print("");
            print("");
        }
        i = i + 1;
    }
}
filter(a, 1);
```

The result is as follows:

```
1 : Restaurant
2 : Unversity Building
3 : Library
4 : SuperMarket
5 : Pharmacy

-- try to find places of target type --
1

--------- have found target ------------
---------     target      --------------
node Xian Famous Food
--------- shortest path to target  -----
[node Mudd
node Butler Library
node Vine
node WestSide Market
node Duane Reade
node Xian Famous Food
]
```

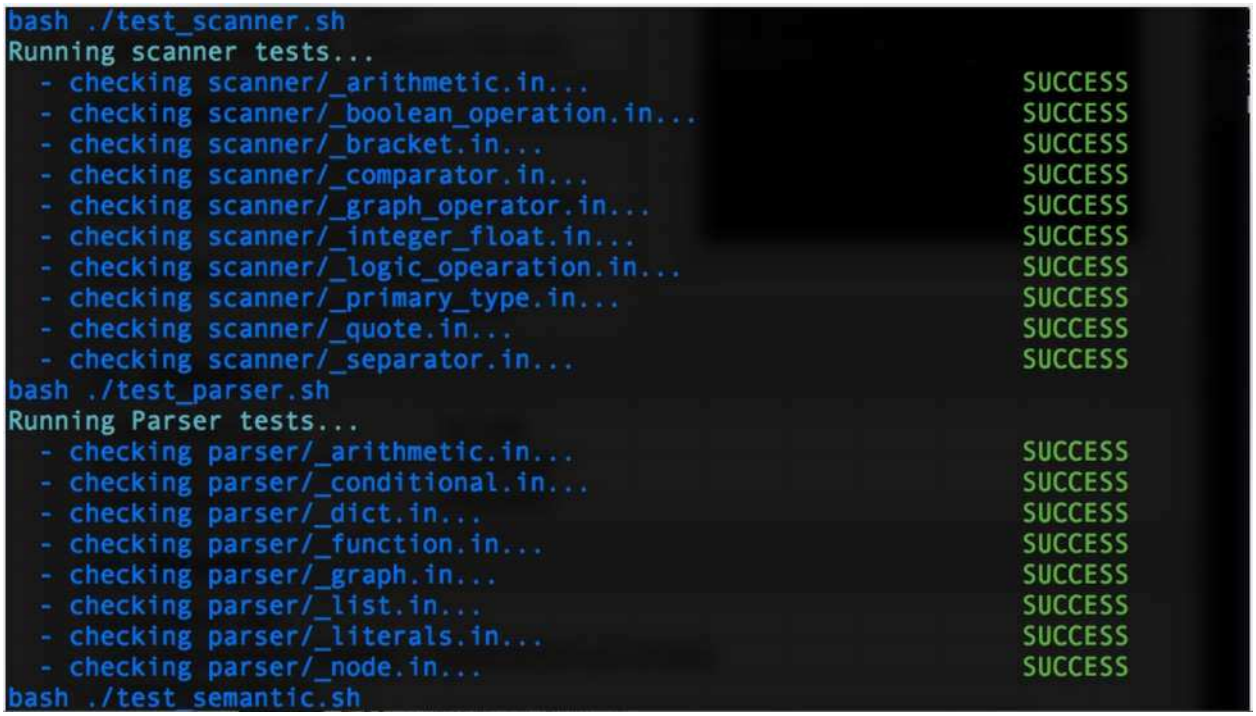
--------- have found target ----------
--------- target ----------
node Vine
--------- shortest path to target -----
[node Mudd
node Butler Library
node Vine
]

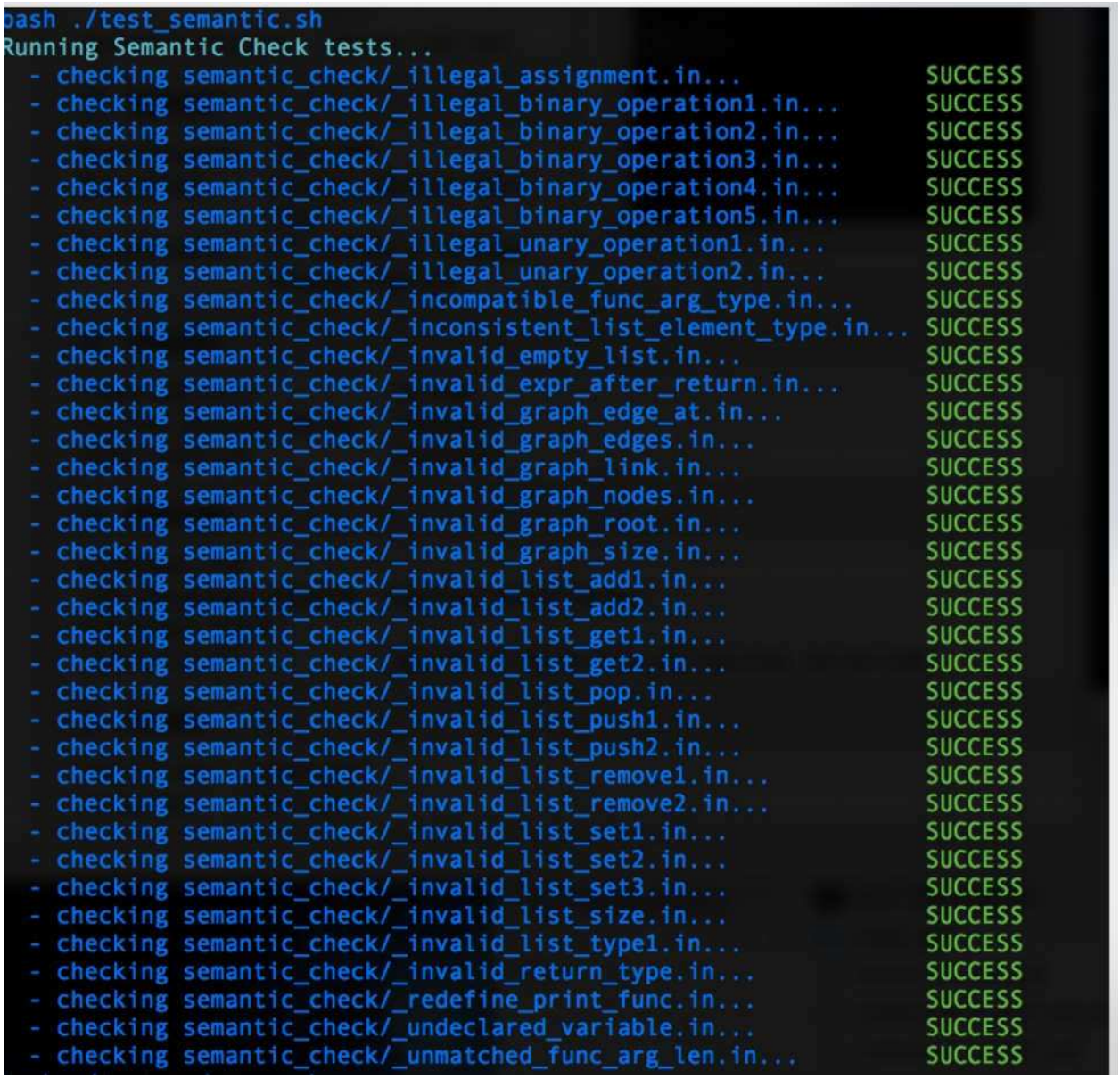
This cool feature is implemented with hashmap and dijkstra algorithm. Every key in hashmap is a type, and value is a list of specific location. When type is set, we just grab all the values, and for each value, we do dijkstra algorithm.

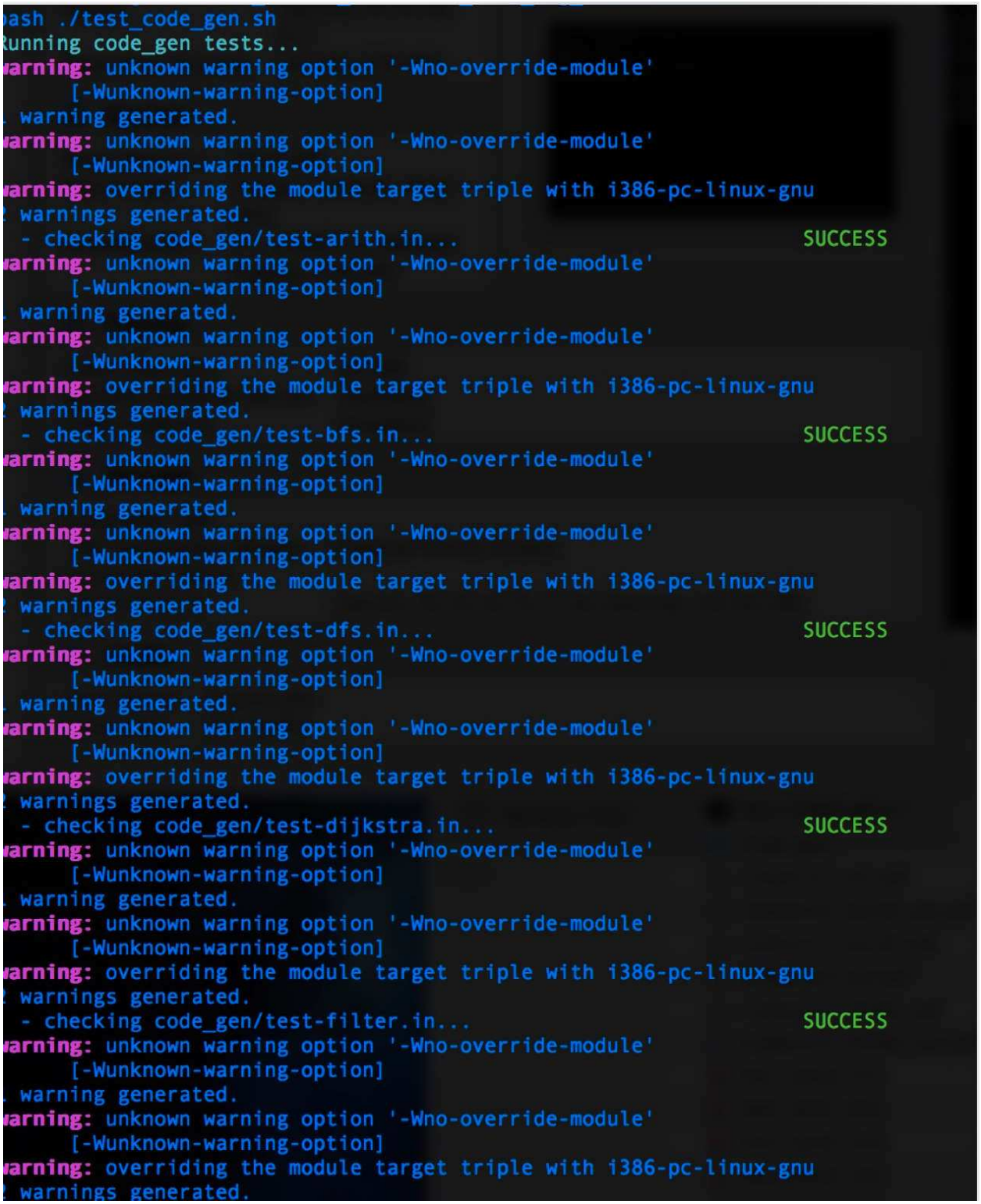
This implementation is of highly use, people can save time buy get the shortest way to reach their targets.

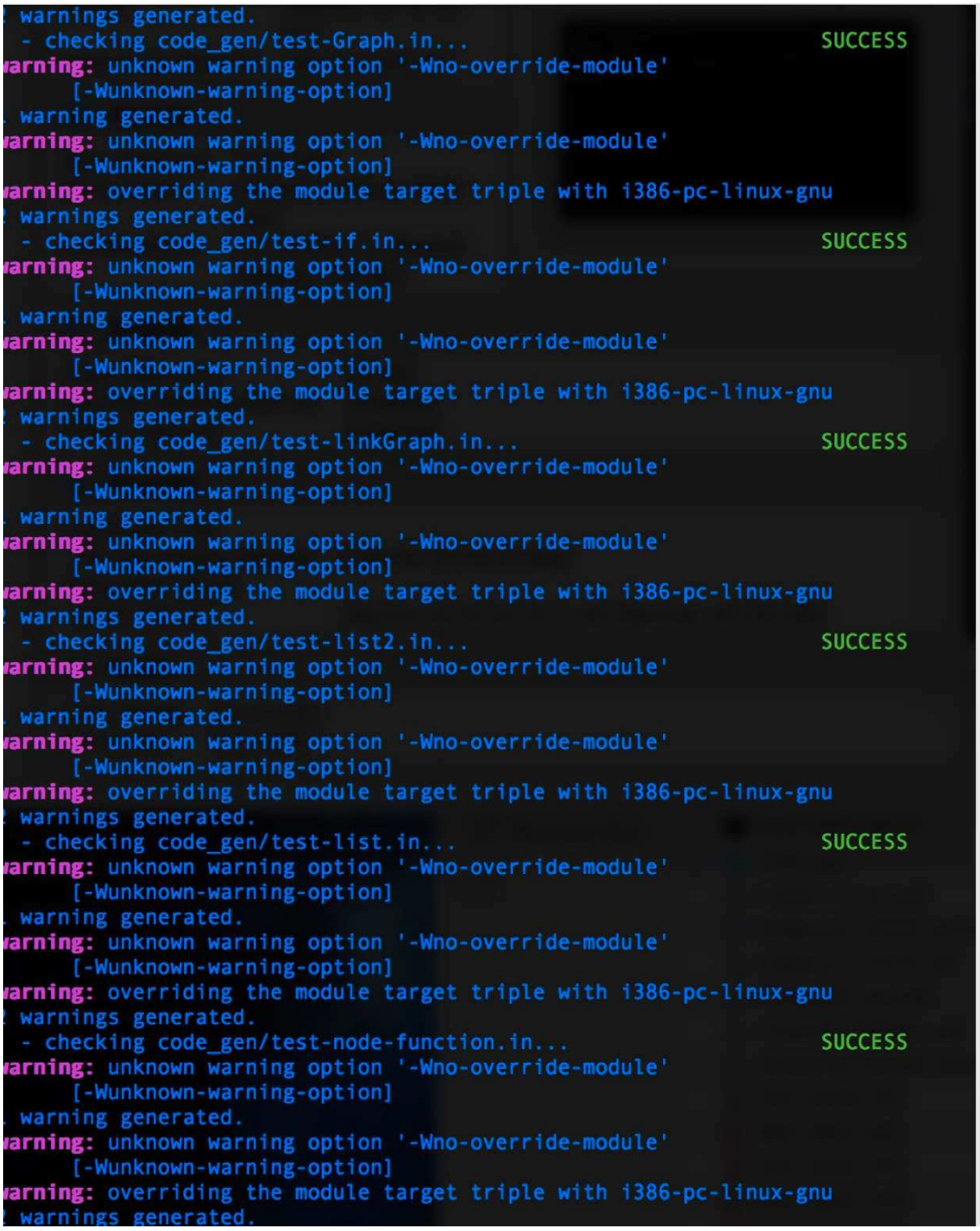
6. Testing

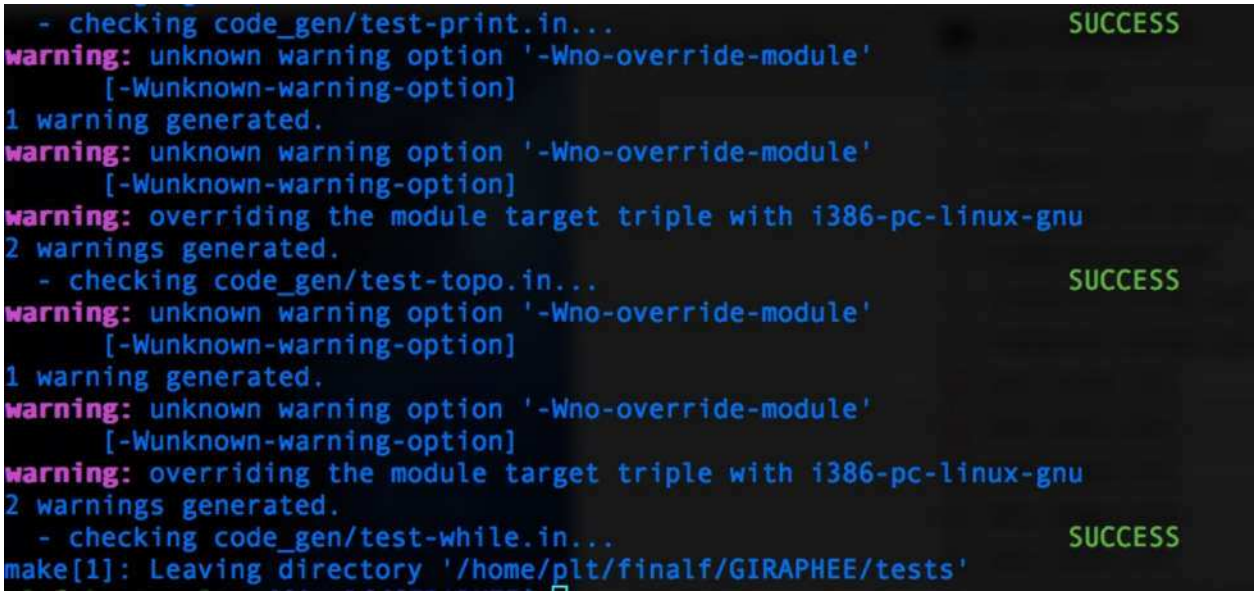
Automation Testing:




bash ./test_semantic.sh
Running Semantic Check tests...
- checking semantic_check/_illegal_assignment.in... SUCCESS
- checking semantic_check/_illegal_binary_operation1.in... SUCCESS
- checking semantic_check/_illegal_binary_operation2.in... SUCCESS
- checking semantic_check/_illegal_binary_operation3.in... SUCCESS
- checking semantic_check/_illegal_binary_operation4.in... SUCCESS
- checking semantic_check/_illegal_binary_operation5.in... SUCCESS
- checking semantic_check/_illegal_unary_operation1.in... SUCCESS
- checking semantic_check/_illegal_unary_operation2.in... SUCCESS
- checking semantic_check/_incompatible_func_arg_type.in... SUCCESS
- checking semantic_check/_inconsistent_list_element_type.in... SUCCESS
- checking semantic_check/_invalid_empty_list.in... SUCCESS
- checking semantic_check/_invalid_expr_after_return.in... SUCCESS
- checking semantic_check/_invalid_graph_edge_at.in... SUCCESS
- checking semantic_check/_invalid_graph_edges.in... SUCCESS
- checking semantic_check/_invalid_graph_link.in... SUCCESS
- checking semantic_check/_invalid_graph_nodes.in... SUCCESS
- checking semantic_check/_invalid_graph_root.in... SUCCESS
- checking semantic_check/_invalid_graph_size.in... SUCCESS
- checking semantic_check/_invalid_list_add1.in... SUCCESS
- checking semantic_check/_invalid_list_add2.in... SUCCESS
- checking semantic_check/_invalid_list_get1.in... SUCCESS
- checking semantic_check/_invalid_list_get2.in... SUCCESS
- checking semantic_check/_invalid_list_pop.in... SUCCESS
- checking semantic_check/_invalid_list_push1.in... SUCCESS
- checking semantic_check/_invalid_list_push2.in... SUCCESS
- checking semantic_check/_invalid_list_remove1.in... SUCCESS
- checking semantic_check/_invalid_list_remove2.in... SUCCESS
- checking semantic_check/_invalid_list_set1.in... SUCCESS
- checking semantic_check/_invalid_list_set2.in... SUCCESS
- checking semantic_check/_invalid_list_set3.in... SUCCESS
- checking semantic_check/_invalid_list_size.in... SUCCESS
- checking semantic_check/_invalid_list_type1.in... SUCCESS
- checking semantic_check/_invalid_return_type.in... SUCCESS
- checking semantic_check/_redefine_print_func.in... SUCCESS
- checking semantic_check/_undeclared_variable.in... SUCCESS
- checking semantic_check/_unmatched_func_arg_len.in... SUCCESS


ash ./test_code_gen.sh
Running code_gen tests...
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
arning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-arith.in... SUCCESS
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
arning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-bfs.in... SUCCESS
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
arning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-dfs.in... SUCCESS
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
arning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-dijkstra.in... SUCCESS
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
arning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-filter.in... SUCCESS
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
arning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
arning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.


warnings generated.
- checking code_gen/test-Graph.in... SUCCESS
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-if.in... SUCCESS
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-linkGraph.in... SUCCESS
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-list2.in... SUCCESS
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-list.in... SUCCESS
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.
- checking code_gen/test-node-function.in... SUCCESS
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning generated.
warning: unknown warning option '-Wno-override-module'
[-Wunknown-warning-option]
warning: overriding the module target triple with i386-pc-linux-gnu
warnings generated.



6.1 Test Graph

Input: test-Graph.in

```
node a = node("A");
node b = node("B");
node c = node("C");
node d = node("D");
node e = node("E");
node f = node("F");

node m = node("M");
node h = node("H");

node x = node("X");
node y = node("Y");
node z = node("Z");


print(" A -> E + A -> B -> D + A -> C ->F ");

graph g = a -> 1$e + a -> 2$b -> 3$d + a -> 4$c -> 5$f;
print(g);

print("Get Graph Size:");
print(g.size());

print("Add Nodes to Graph");
g.addNode(m);
print(g.size());
print(g);

print("Judge whether Nodes exist");
print(g.hasNode(x));
print(g.hasNode(a));
```

```
33  print("Add Edge to Graph, Support new Nodes");
34  g.addEdge(e, f, 5);
35  g.addEdge(a, z, 10);
36  print(g.size());
37  list<node> nodes = g.getAllNodes();
38  print(nodes);
39  print(g);
40
41  print("Judge whether Edges exist between two Nodes");
42  print(g.hasEdge(c,d));
43  print(g.hasEdge(a,b));
```

Output: test-Graph.out

```
 A -> E + A -> B -> D + A -> C ->F
-------------------------------------
Nodes:
node A
node E
node B
node D
node C
node F

Edges:
edgeA ->E: 1
edgeB ->D: 3
edgeA ->B: 2
edgeC ->F: 5
edgeA ->C: 4
-------------------------------------
Get Graph Size:
6
Add Nodes to Graph
7
-------------------------------------
Nodes:
node A
node E
node B
node D
node C
node F
node M
```

```
31
32  Edges:
33  edgeA ->E: 1
34  edgeB ->D: 3
35  edgeA ->B: 2
36  edgeC ->F: 5
37  edgeA ->C: 4
38  ------------------------------------
39  Judge whether Nodes exist
40  false
41  true
42  Add Edge to Graph, Support new Nodes
43  8
44  [node A
45  node E
46  node B
47  node D
48  node C
49  node F
50  node M
51  node Z
52  ]
53  ------------------------------------
54  Nodes:
55  node A
56  node E
57  node B
58  node D
59  node C
60  node F
```

```
node F
node M
node Z

Edges:
edgeA ->E: 1
edgeB ->D: 3
edgeA ->B: 2
edgeC ->F: 5
edgeA ->C: 4
edgeE ->F: 5
edgeA ->Z: 10
---------------------------------------
Judge whether Edges exist between two Nodes
false
true
```

6.2 Test arith

Input: test-Arith.in

```
print(1+2);
print(2-3);
print(5*15);
print(9/3);
print(8.2/4);
print(5%2.3);
print(1.2+1.3);
print(8.2-1.5);

int a = 0;
int foo(int b) {
  int a = 1;
  int bar() {
    return a+b;
  }
  return bar();
}
print(a);
print(foo(3));
```

Output: test-Arith.out

```
3
-1
75
3
2.050000
0.400000
2.500000
6.700000
0
4
```

6.3 Test BFS

Input: test-bfs.in

```
node a = node("a");
node b = node("b");
node c = node("c");
node d = node("d");
node e = node("e");
node f = node("f");
node g = node("g");

print("a -> [1&b -> 1&e -> [4&g -> 1&b, 2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]");

graph gh = a -> 1$b ->1$e -> 4$g ->1$b + a -> 1$b ->1$e ->2$c + a -> 5$c -> 1$g +a -> 5$c -> 1$f->1$c + a -> 3$d->2$c +a -> 3$d -> 3$f;

print("Breath First Search from a");
list<node> res = gh.bfs(a);
print(res);
print("Set all Nodes to Unvisited");
gh.setAllUnvisited();

print("Breath First Search from c");
list<node> ret = gh.bfs(c);
print(ret);
print("Set all Nodes to Unvisited");
gh.setAllUnvisited();
```

Output: test-bfs.out

```
a -> [1&b -> 1&e -> [4&g -> 1&b, 2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]
Breath First Search from a
[node a
node b
node c
node d
node e
node g
node f
]
Set all Nodes to Unvisited
Breath First Search from c
[node c
node g
node f
node b
node e
]
Set all Nodes to Unvisited
```

6.4 Test DFS

Input: test-dfs.in

```
node a = node("a");
node b = node("b");
node c = node("c");
node d = node("d");
node e = node("e");
node f = node("f");
node g = node("g");

print("a -> [1&b -> 1&e -> [4&g -> 1&b, 2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]");

graph gh = a -> 1$b ->1$e -> 4$g ->1$b + a -> 1$b ->1$e ->2$c + a -> 5$c -> 1$g +a -> 5$c -> 1$f->1$c + a -> 3$d->2$c +a -> 3$d -> 3$f;

print("Depth First Search from c");
list<node> res = gh.dfs(c);
print(res);
print("Set all Nodes to Unvisited");
gh.setAllUnvisited();
print("Depth First Search from b");
list<node> ret = gh.dfs(b);
print(ret);
```

Output: test-dfs.out

```
1  a -> [1&b -> 1&e -> [4&g -> 1&b, 2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]
2  Depth First Search from c
3  [node c
4  node f
5  node g
6  node b
7  node e
8  ]
9  Set all Nodes to Unvisited
10 Depth First Search from b
11 [node b
12 node e
13 node c
14 node f
15 node g
16 ]
```

6.5 Test Dijsktra

Input: test-dijsktra.in

Output: test-dijsktra.out

```
node a = node("a");
node b = node("b");
node c = node("c");
node d = node("d");
node e = node("e");
node f = node("f");
node g = node("g");

print("a -> [1&b -> 1&e -> [4&g -> 1&b, 2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]");

graph gh = a -> 1$b ->1$e -> 4$g ->1$b + a -> 1$b ->1$e ->2$c + a -> 5$c -> 1$g +a -> 5$c -> 1$f->1$c + a -> 3$d->2$c +a -> 3$d -> 3$f;

print("From a to f");
gh.dijkstra(a,f);
print("From a to g");
gh.dijkstra(a, g);
```

```
1  a -> [1&b -> 1&e -> [4&g -> 1&b, 2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]
2  From a to f
3  a --> b --> e --> c --> f
4  From a to g
5  a --> b --> e --> c --> g
```

6.6 Test If

Input: test-if.in

```
int a = 0;
if (a < 1) {
        print("small");
}else {
        print("large");
}

float b = 12;
if (b > 10) {
        print(b);
}

bool c = false;
if (!c) {
        print("true");
}
```

Output: test-if.out

```
small
12.000000
true
```

6.6 Test LinkGraph

Input: test-linkGraph.in

```
1   node a = node("A");
2   node b = node("B");
3   node c = node("C");
4   node d = node("D");
5   node e = node("E");
6
7   print("Using + to link graph");
8
9   print("Link graphs: A -> 0$B ->3$E +  C -> 1$A + A -> 2$D");
10  print("Shared nodes: A");
11
12  graph g = a -> 0$b ->3$e +  c -> 1$a + a -> 2$d;
13
14  print(g);
```

Output: test-linkGraph.out

```
1   Using + to link graph
2   Link graphs: A -> 0$B ->3$E +  C -> 1$A + A -> 2$D
3   Shared nodes: A
4   ------------------------------------
5   Nodes:
6   node B
7   node E
8   node A
9   node C
10  node D
11
12  Edges:
13  edgeB ->E: 3
14  edgeA ->B: 0
15  edgeC ->A: 1
16  edgeA ->D: 2
17  ------------------------------------
```

6.7 Test List

Input: test-list.in

```
print("------------------general method for list------------------");

list<int> a = [1, 2, 3];
list<float> b = [1.0, 2.0, 3.0];
list<string> c = ["a", "b", "c"];
list<bool> d = [true, false, true];

print(a);
print(b);
print(c);
print(d);

node n1 = node(7);
node n2 = node(8);
node n3 = node(9);
list<node> nd = [n1, n2, n3];
print(nd);

list<graph> gp = [n1->n2, n2->n3, n3->n1];
print(gp);

print("------------ test add, remove ---------------");

print("remove element not existing a.remove(4), list doesn't change");
a.remove(4);
print(a);
print("adding element using list.add()");
a.add(5);
print(a);
print("remove element using -");
```

```
31  a=a-2;
32  print(a);
33  a=a-1;
34  a=a-5;
35  print("remove list to empty");
36  a=a-3;
37  print(a);
38  print("add element using +");
39  a=a+6;
40  print(a);
41  print("-----------------------------------------");
42
43  print("------------ test size, get, set ---------------");
44  print(b);
45  print("Size:");
46  print(b.size());
47  print("Get first element:");
48  print(b.get(0));
49  print("Set function:");
50  b.set(1,5.0);
51  print(b);
52  print("-----------------------------------------------");
```

Output: test-list.out

```
------------------general method for list---------
[1, 2, 3]
[1.000000, 2.000000, 3.000000]
[a, b, c]
[true, false, true]
[node 7
node 8
node 9
]
[-------------------------------------
Nodes:
node 7
node 8

Edges:
edge7 ->8: 0
-------------------------------------
-------------------------------------
Nodes:
node 8
node 9

Edges:
edge8 ->9: 0
-------------------------------------
-------------------------------------
Nodes:
node 9
node 7

```

```
Edges:
edge9 ->7: 0
-------------------------------------
]
------------ test add, remove ---------------
remove element not existing a.remove(4), list doesn't change
[1, 2, 3]
adding element using list.add()
[1, 2, 3, 5]
remove element using -
[1, 3, 5]
remove list to empty
[]
add element using +
[6]
-----------------------------------------
------------ test size, get, set ---------------
[1.000000, 2.000000, 3.000000]
Size:
3
Get first element:
1.000000
Set function:
[1.000000, 5.000000, 3.000000]
-----------------------------------------------
```

6.8 Test List 2

Input: test-list2.in

```
print("------------------test contains, concatenate------------------");

list<int> a = [1, 2, 3];

print("test whether list contains 2, 4");
print(a.contains(2));
print(a.contains(4));

list<int> b = [4, 5, 6];

list<int> c = a + b;
print("print concatenate [1,2,3] and [4,5,6]");
print(c);


print("------------------------------------------------------------");
```

Output: test-list2.out

```
------------------test contains, concatenate------------------
test whether list contains 2, 4
true
false
print concatenate [1,2,3] and [4,5,6]
[1, 2, 3, 4, 5, 6]
------------------------------------------------------------
```

6.9 Test Node Function

Input: test-node-function.in

```
node a = node(1);
node b = node("GIRAPHE");
node c = node(1>2);
print(a);
print(b);
print(c);

node d = node("hi");
print("Is Node d Visited?");
print(d.isVisited());
print("Set Node d to Visited");
d.setVisited();
print(d.isVisited());

```

Output: test-node-function.out

```
node 1
node GIRAPHE
node false
Is Node d Visited?
false
Set Node d to Visited
true
```

6.10 Test Print

Input: test-print.in

```
print(23);
print(-1.2);
print("Hello, World");
print(true);
print(1>0.2);
print([1, 2, 3]);
print(node(10));
print({"a": 1, "b": 2});

int a = 4115;
float b = 20.17;
string c = "GIRAPHE";

printf("%d ** %.2f ** %s\n", a, b, c);
```

Output: test-print.out

```
23
-1.200000
Hello, World
true
true
[1, 2, 3]
node 10
{a: 1, b: 2}
4115 ** 20.17 ** GIRAPHE
```

6.11 Test While

Input: test-while.in

```
int a = 0;
while (a < 2) {
        print(a);
        a = a + 1;
}


bool c = true;
while (c) {
        print(c);
        c = ! c;
}
```

Output: test-while.out

```
0
1
true
```

6.12 Test Topological Sort

Input: topo.in

```
node a = node("COMS 1004: Introduction to Computer Science and Programming in Java");
node b = node("COMS 4111: Introduction to Databases");
node c = node("COMS 6111: Advanced Database Systems");
node d = node("CSEE 4119: Computer Networks");
node e = node("CSEE 4140: Networking Laboratory");
node f = node("COMS 4118: Operating Systems");
node g = node("COMS 3101: Programming Language Python");
node h = node("CSOR 4231: Analysis of Algorithms I");
node i1 = node("COMS 6232: Analysis of Algorithms II");
node j = node("COMS 4771: Machine Learning");
node k = node("COMS 4772: Advanced Machine Learning");
node l = node("COMS 4995: Deep Learning for Computer Vision");

graph ga = a -> b -> f + b -> c + a -> d -> f + d -> e + h -> f + g -> h -> j -> l + h -> i1 + j -> k;

int i=0;
list<node> n = ga.getAllNodes();
list<node> tmp = n;
list<node> res = n;
int stmp = 0;
ga.setAllUnvisited();

 while(i<ga.size()){
        if(n.get(i).isVisited()){
                i = i + 1;
        }else{
                tmp = ga.dfs(n.get(i));
                stmp = tmp.size();
                while(stmp > 0) {
                        res.add(tmp.get(stmp-1));
                        stmp = stmp - 1;
                }
                i = i + 1;
        }
 }

 while(i > 0) {
        print(res.get(i-1+ga.size()));
        i=i-1;
 }
```

Output: topo.out

```
node COMS 3101: Programming Language Python
node CSOR 4231: Analysis of Algorithms I
node COMS 6232: Analysis of Algorithms II
node COMS 4771: Machine Learning
node COMS 4772: Advanced Machine Learning
node COMS 4995: Deep Learning for Computer Vision
node COMS 1004: Introduction to Computer Science and Programming in Java
node CSEE 4119: Computer Networks
node CSEE 4140: Networking Laboratory
node COMS 4111: Introduction to Databases
node COMS 6111: Advanced Database Systems
node COMS 4118: Operating Systems
```

7. Lessons Learned

7.1 Dianya Jiang

Be sure to start and decide what to do early. Try to make a detailed plan and tight schedule both for me and my teammates. Communication is important. When design a language, it can be really painful, but interesting, keep add new feature, make progress and find new bugs, fix them. Besides, learning how to use Github more efficiently is also crucial, or the all the profiles will be in a mess some time. Last but not least, good teammates are sooooo important, try to find teammates early and find those who are responsible and dedicated, willing to work and really make contributions, or you final week will be really busy and in a mess.

7.2 Minh Truong

It is important to communicate and make important decisions early. Understanding MircroC is important. Going to class would probably be really help since a lot of the material that you need to build your language is pretty dense and trying to understand it without a good base is really tough.

- *Do the difficult things while they are easy and do the great things while they are small. A journey of a thousand miles must begin with a single step*
- *When I let go of what I am, I become what I might be*

7.3 Tongyun Wu

Start Early is the most important thing for the project. Try to make a detailed schedule of the project and meet with TA every week to confirm the progress of the project. Another thing I learned is that the integrated test suite should be set up as early as possible and should be kept updated when new feature is added and implemented. Find all options that can be used to implement specific features and discuss those with TAs or Prof. Edwards, which may save you significant times when you finally figure out that you chose the wrong and painful way.

7.4 Vince Pallone

Start early and work together. Go over the MicroC code and understand everything before making changes. I recommend maintaining a very clean repo with the master branch always working. As you add features, create new branches and merge them upon passing rigorous tests. Testing each part of the compiler as well as specific language features will help identify issues. Do not expect to change one part without seeing errors from another.

7.5 Yoki Yuan

Do more, talk less. Have a gelivable teammate will be wonderful, otherwise will ruin the entire final week to deal with the project and have no time to review other courses.

8. Appendix

```
ast.ml:1:(* Binary Operators *)
ast.ml:2:type binop =
ast.ml:3:  Add         (* + *)
ast.ml:4:| Sub         (* - *)
ast.ml:5:| Mult        (* * *)
ast.ml:6:| Div         (* / *)
ast.ml:7:| Mod         (* % *)
ast.ml:8:| Equal       (* == *)
ast.ml:9:| Neq         (* != *)
ast.ml:10:| Less        (* < *)
ast.ml:11:| Leq         (* <= *)
ast.ml:12:| Greater     (* > *)
ast.ml:13:| Geq         (* >= *)
ast.ml:14:| And         (* and *)
ast.ml:15:| Or          (* or *)
ast.ml:16:
ast.ml:17:(* Unary Operators *)
ast.ml:18:type unop =
ast.ml:19:  Neg         (* - *)
ast.ml:20:| Not         (* not *)
ast.ml:21:
ast.ml:22:(* Numbers int | float *)
ast.ml:23:type num =
ast.ml:24:  Num_Int of int
ast.ml:25:| Num_Float of float
ast.ml:26:
ast.ml:27:(* Variable Type *)
ast.ml:28:type var_type =
ast.ml:29:  Int_t                  (* int *)
ast.ml:30:| Float_t                (* float *)
ast.ml:31:| String_t               (* string *)
ast.ml:32:| Bool_t
ast.ml:33:| Node_t
ast.ml:34:| Graph_t
ast.ml:35:| Dict_Int_t
ast.ml:36:| Dict_Float_t
ast.ml:37:| Dict_String_t
ast.ml:38:| Dict_Node_t
ast.ml:39:| Dict_Graph_t
ast.ml:40:| List_Int_t
ast.ml:41:| List_Float_t
ast.ml:42:| List_String_t
ast.ml:43:| List_Bool_t
ast.ml:44:| List_Node_t
ast.ml:45:| List_Graph_t
ast.ml:46:| Void_t
ast.ml:47:| Null_t
ast.ml:48:
ast.ml:49:(* Type Declaration *)
ast.ml:50:type formal =
ast.ml:51:| Formal of var_type * string   (* int aNum *)
ast.ml:52:
ast.ml:53:type expr =
ast.ml:54:    Num_Lit of num
ast.ml:55:|   Null
ast.ml:56:|   String_Lit of string
ast.ml:57:|   Bool_lit of bool
ast.ml:58:|   Node of expr
ast.ml:59:|   Graph_Link of expr * expr * expr
ast.ml:60:|      Binop of expr * binop * expr
```

```
ast.ml:61:|      Unop of unop * expr
ast.ml:62:|   Id of string
ast.ml:63:|   Assign of string * expr
ast.ml:64:|   Noexpr
ast.ml:65:|   ListP of expr list
ast.ml:66:|   DictP of (expr * expr) list
ast.ml:67:|   Call of string * expr list    (* function call *)
ast.ml:68:|   CallDefault of expr * string * expr list
ast.ml:69:
ast.ml:70:
ast.ml:71:type var_decl =
ast.ml:72:| Local of var_type * string * expr
ast.ml:73:
ast.ml:74:(* Statements *)
ast.ml:75:type stmt =
ast.ml:76:  Expr of expr     (* set foo = bar + 3 *)
ast.ml:77:| Return of expr
ast.ml:78:| For of expr * expr * expr * stmt list
ast.ml:79:| If of expr * stmt list * stmt list
ast.ml:80:| While of expr * stmt list
ast.ml:81:| Var_dec of var_decl
ast.ml:82:| Func of func_decl
ast.ml:83:
ast.ml:84:
ast.ml:85:(* Function Declaration *)
ast.ml:86:and func_decl = {
ast.ml:87:  returnType: var_type;
ast.ml:88:  name: string;
ast.ml:89:  args: formal list;
ast.ml:90:  body: stmt list;
ast.ml:91:}
ast.ml:92:
ast.ml:93:
ast.ml:94:(* Program entry point *)
ast.ml:95:type program = stmt list
ast.ml:96:checker.ml:1:module A = Ast
checker.ml:2:module S = Sast
checker.ml:3:
checker.ml:4:module StringMap = Map.Make(String)
checker.ml:5:let node_num = ref 0
checker.ml:6:
checker.ml:7:let convert_binop = function
checker.ml:8:    A.Add -> S.Add
checker.ml:9:  | A.Sub -> S.Sub
checker.ml:10:  | A.Mult -> S.Mult
checker.ml:11:  | A.Div -> S.Div
checker.ml:12:  | A.Mod -> S.Mod
checker.ml:13:  | A.Equal -> S.Equal
checker.ml:14:  | A.Neq -> S.Neq
checker.ml:15:  | A.Less -> S.Less
checker.ml:16:  | A.Leq -> S.Leq
checker.ml:17:  | A.Greater -> S.Greater
checker.ml:18:  | A.Geq -> S.Geq
checker.ml:19:  | A.And -> S.And
checker.ml:20:  | A.Or -> S.Or
checker.ml:21:
checker.ml:22:let convert_unop = function
checker.ml:23:  A.Neg -> S.Neg
checker.ml:24:| A.Not -> S.Not
checker.ml:25:
```

```
checker.ml:26:let convert_num = function
checker.ml:27:    A.Num_Int(a) -> S.Num_Int(a)
checker.ml:28:  | A.Num_Float(a) -> S.Num_Float(a)
checker.ml:29:
checker.ml:30:let convert_var_type = function
checker.ml:31:    A.Int_t -> S.Int_t
checker.ml:32:  | A.Float_t -> S.Float_t
checker.ml:33:  | A.String_t -> S.String_t
checker.ml:34:  | A.Bool_t -> S.Bool_t
checker.ml:35:  | A.Node_t -> S.Node_t
checker.ml:36:  | A.Graph_t -> S.Graph_t
checker.ml:37:  | A.List_Int_t -> S.List_Int_t
checker.ml:38:  | A.List_Float_t -> S.List_Float_t
checker.ml:39:  | A.List_String_t -> S.List_String_t
checker.ml:40:  | A.List_Node_t -> S.List_Node_t
checker.ml:41:  | A.List_Graph_t -> S.List_Graph_t
checker.ml:42:  | A.List_Bool_t -> S.List_Bool_t
checker.ml:43:  | A.Dict_Int_t -> S.Dict_Int_t
checker.ml:44:  | A.Dict_Float_t -> S.Dict_Float_t
checker.ml:45:  | A.Dict_String_t -> S.Dict_String_t
checker.ml:46:  | A.Dict_Node_t -> S.Dict_Node_t
checker.ml:47:  | A.Dict_Graph_t -> S.Dict_Graph_t
checker.ml:48:  | A.Void_t -> S.Void_t
checker.ml:49:  | A.Null_t -> S.Null_t
checker.ml:50:
checker.ml:51:let rec get_entire_name m aux cur_name =
checker.ml:52:  if (StringMap.mem cur_name m) then
checker.ml:53:    let aux = (StringMap.find cur_name m) ^ "." ^ aux in
checker.ml:54:    (get_entire_name m aux (StringMap.find cur_name m))
checker.ml:55:  else aux
checker.ml:56:
checker.ml:57:let increase_node_num =
checker.ml:58:  let node_num = ref(!node_num) in
checker.ml:59:  !(node_num) - 1
checker.ml:60:
checker.ml:61:let rec convert_expr m = function
checker.ml:62:    A.Num_Lit(a) -> S.Num_Lit(convert_num a)
checker.ml:63:|   A.Null -> S.Null
checker.ml:64:|   A.String_Lit(a) -> S.String_Lit(a)
checker.ml:65:|   A.Bool_lit(a) -> S.Bool_lit(a)
checker.ml:66:|   A.Node(a) -> node_num := (!node_num + 1);
S.Node(!node_num - 1, convert_expr m a)
checker.ml:67:|   A.Graph_Link(a,c,d) -> S.Graph_Link(
checker.ml:68:      convert_expr m a,
checker.ml:69:      convert_expr m c,
checker.ml:70:      (match (c,d) with
checker.ml:71:        | (A.ListP(_), A.ListP(_))
checker.ml:72:        | (A.ListP(_), A.Noexpr)
checker.ml:73:        | (A.ListP(_), A.Null) -> convert_expr m d
checker.ml:74:        | (A.ListP(_), _) -> S.ListP([convert_expr m d])
checker.ml:75:        | _ -> convert_expr m d
checker.ml:76:      ))
checker.ml:77:|   A.Binop(a,b,c) -> S.Binop(convert_expr m a,
convert_binop b, convert_expr m c)
checker.ml:78:|   A.Unop(a,b) -> S.Unop(convert_unop a, convert_expr m b)
checker.ml:79:|   A.Id(a) -> S.Id(a)
checker.ml:80:|   A.Assign(a,b) -> S.Assign(a, convert_expr m b)
checker.ml:81:|   A.Noexpr -> S.Noexpr
checker.ml:82:|   A.ListP(a) -> S.ListP(convert_expr_list m a)
checker.ml:83:|   A.DictP(a) -> S.DictP(convert_dict_list m a)
```

```
checker.ml:84:|   A.Call(a,b) -> S.Call(get_entire_name m a a,
convert_expr_list m b)
checker.ml:85:|   A.CallDefault(a,b,c) -> S.CallDefault(convert_expr m a,
b, convert_expr_list m c)
checker.ml:86:
checker.ml:87:
checker.ml:88:
checker.ml:89:and convert_expr_list m = function
checker.ml:90:    [] -> []
checker.ml:91:  | [x] -> [convert_expr m x]
checker.ml:92:  | _ as l -> (List.map (convert_expr m) l)
checker.ml:93:
checker.ml:94:and convert_dict m = function
checker.ml:95:  (c,d) -> (convert_expr m c, convert_expr m d)
checker.ml:96:
checker.ml:97:and convert_dict_list m = function
checker.ml:98:    [] -> []
checker.ml:99:  | [x] -> [convert_dict m x]
checker.ml:100:  | _ as l -> (List.map (convert_dict m) l)
checker.ml:101:
checker.ml:102:and convert_e m = function
checker.ml:103:    (c,d,e) ->(convert_expr m c, convert_expr m d,
convert_expr m e)
checker.ml:104:
checker.ml:105:and convert_e_list m = function
checker.ml:106:    [] -> []
checker.ml:107:  | [x] -> [convert_e m x]
checker.ml:108:  | _ as l -> (List.map (convert_e m) l)
checker.ml:109:
checker.ml:110:let convert_formal = function
checker.ml:111:  | A.Formal(v, s) -> S.Formal(convert_var_type v, s)
checker.ml:112:
checker.ml:113:let convert_formal_list = function
checker.ml:114:    [] -> []
checker.ml:115:  | [x] -> [convert_formal x]
checker.ml:116:  | _ as l -> (List.map convert_formal l)
checker.ml:117:
checker.ml:118:(* create a main funcition outside of the whole statement
list *)
checker.ml:119:let createMain stmts = A.Func({
checker.ml:120:    A.returnType = A.Int_t;
checker.ml:121:    A.name = "main";
checker.ml:122:    A.args = [];
checker.ml:123:    A.body = stmts;
checker.ml:124:  })
checker.ml:125:
checker.ml:126:let rec get_funcs_from_body_a = function
checker.ml:127:    [] -> []
checker.ml:128:  | A.Func(_) as x::tl -> x :: (get_funcs_from_body_a tl)
checker.ml:129:  | _::tl -> get_funcs_from_body_a tl
checker.ml:130:
checker.ml:131:let rec get_body_from_body_a = function
checker.ml:132:    [] -> []
checker.ml:133:  | A.Func(_)::tl -> get_body_from_body_a tl
checker.ml:134:  | _ as x::tl -> x :: (get_body_from_body_a tl)
checker.ml:135:
checker.ml:136:let rec mapper parent map = function
checker.ml:137:   [] -> map
checker.ml:138: | A.Func{A.name = n; _}::tl ->
checker.ml:139:    mapper parent (StringMap.add n parent map) tl
```

```
checker.ml:140: | _-> map
checker.ml:141:
checker.ml:142:let convert_bfs_insider my_map = function
checker.ml:143:    A.Func{A.name = n; A.body = b; _}->
checker.ml:144:    let curr = get_funcs_from_body_a b in
checker.ml:145:      let my_map = mapper n my_map curr in
checker.ml:146:    (curr,my_map)
checker.ml:147:  | _->([],my_map)
checker.ml:148:
checker.ml:149:let rec bfser m result = function
checker.ml:150:    [] ->(List.rev result, m)
checker.ml:151:  | A.Func{A.returnType = r; A.name = n; A.args = args;
A.body = b} as a ::tl -> let result1 = convert_bfs_insider m a in
checker.ml:152:    let latterlist = tl @ (fst result1) in
checker.ml:153:    let m = (snd result1) in
checker.ml:154:    let addedFunc = A.Func({
checker.ml:155:      A.returnType = r; A.name = n; A.args = args; A.body
= get_body_from_body_a b
checker.ml:156:    }) in
checker.ml:157:    let result = result @ [addedFunc] in
checker.ml:158:     bfser m result latterlist
checker.ml:159:  | _->([], m)
checker.ml:160:
checker.ml:161:(* convert stament in A to C, except those Var_dec and
Func, we will convert them separately *)
checker.ml:162:let rec convert_stmt m = function
checker.ml:163:    A.Expr(a) -> S.Expr(convert_expr m a)
checker.ml:164:  | A.Return(a) -> S.Return(convert_expr m a)
checker.ml:165:  | A.For(e1, e2, e3, stls) -> S.For(convert_expr m e1,
convert_expr m e2, convert_expr m e3, List.map (convert_stmt m) stls)
checker.ml:166:  | A.If(e, stls1, stls2) -> S.If(convert_expr m e,
List.map (convert_stmt m) stls1, List.map (convert_stmt m) stls2)
checker.ml:167:  | A.While(e, stls) -> S.While(convert_expr m e, List.map
(convert_stmt m) stls)
checker.ml:168:  | _ -> S.Expr(S.Noexpr)
checker.ml:169:
checker.ml:170:
checker.ml:171:let rec get_body_from_body_c m = function
checker.ml:172:    [] -> []
checker.ml:173:  | A.Var_dec(A.Local(_, name, v))::tl when v <> A.Noexpr
-> S.Expr(S.Assign(name, convert_expr m v)) :: (get_body_from_body_c m
tl)
checker.ml:174:  | A.Var_dec(A.Local(_, _, v))::tl when v = A.Noexpr ->
(get_body_from_body_c m tl)
checker.ml:175:  | _ as x::tl -> (convert_stmt m x) ::
(get_body_from_body_c m tl)
checker.ml:176:
checker.ml:177:let rec get_local_from_body_c = function
checker.ml:178:    [] -> []
checker.ml:179:  | A.Var_dec(A.Local(typ, name, _))::tl ->
S.Formal(convert_var_type typ, name) :: (get_local_from_body_c tl)
checker.ml:180:  | _::tl -> get_local_from_body_c tl
checker.ml:181:
checker.ml:182:(* convert the horizental level function list in A to C *)
checker.ml:183:let rec convert_func_list_c m = function
checker.ml:184:    [] -> []
checker.ml:185:  | A.Func{A.returnType = r; A.name = n; A.args = a;
A.body = b} :: tl -> {
checker.ml:186:    S.returnType = convert_var_type r;
checker.ml:187:    S.name = get_entire_name m n n;
```

```
checker.ml:188:    S.args = convert_formal_list a;
checker.ml:189:    S.body = get_body_from_body_c m b;
checker.ml:190:    S.locals = get_local_from_body_c b;
checker.ml:191:    S.pname = if n = "main" then "main" else
get_entire_name m (StringMap.find n m) (StringMap.find n m)
checker.ml:192:  } :: (convert_func_list_c m tl)
checker.ml:193:  | _::tl -> convert_func_list_c m tl
checker.ml:194:
checker.ml:195:(* entry point *)
checker.ml:196:let convert stmts =
checker.ml:197:  let funcs = createMain stmts in
checker.ml:198:  let horizen_funcs_m = bfser StringMap.empty [] [funcs]
in
checker.ml:199:  convert_func_list_c (snd horizen_funcs_m) (fst
horizen_funcs_m)
checker.ml:200:codegen.ml:1:(* Code generation: translate takes a
semantically checked AST and
codegen.ml:2:produces LLVM IR
codegen.ml:3:
codegen.ml:4:LLVM tutorial: Make sure to read the OCaml version of the
tutorial
codegen.ml:5:
codegen.ml:6:http://llvm.org/docs/tutorial/index.html
codegen.ml:7:
codegen.ml:8:Detailed documentation on the OCaml LLVM library:
codegen.ml:9:
codegen.ml:10:http://llvm.moe/
codegen.ml:11:http://llvm.moe/ocaml/
codegen.ml:12:
codegen.ml:13:*)
codegen.ml:14:
codegen.ml:15:module L = Llvm
codegen.ml:16:module A = Sast
codegen.ml:17:
codegen.ml:18:module StringMap = Map.Make(String)
codegen.ml:19:
codegen.ml:20:let context = L.global_context ()
codegen.ml:21:let llctx = L.global_context ()
codegen.ml:22:let llmem = L.MemoryBuffer.of_file "lib.bc"
codegen.ml:23:let llm = Llvm_bitreader.parse_bitcode llctx llmem
codegen.ml:24:let the_module = L.create_module context "GIRAPHE"
codegen.ml:25:
codegen.ml:26:let i32_t  = L.i32_type  context
codegen.ml:27:and i8_t   = L.i8_type   context
codegen.ml:28:and i1_t   = L.i1_type   context
codegen.ml:29:and f_t    = L.double_type context
codegen.ml:30:and str_t  = L.pointer_type (L.i8_type context)
codegen.ml:31:and ptr_t  = L.pointer_type (L.i8_type context)
codegen.ml:32:and void_t = L.void_type context
codegen.ml:33:and list_t = L.pointer_type (match L.type_by_name llm
"struct.List" with
codegen.ml:34:    None -> raise (Failure "struct.List not found")
codegen.ml:35:  | Some x -> x)
codegen.ml:36:and hmap_t = L.pointer_type (match L.type_by_name llm
"struct.hashmap_map" with
codegen.ml:37:    None -> raise (Failure "struct.hashmap_map not found")
codegen.ml:38:  | Some x -> x)
codegen.ml:39:and node_t = L.pointer_type (match L.type_by_name llm
"struct.Node" with
codegen.ml:40:    None -> raise (Failure "struct.Node not found")
```

```
codegen.ml:41:  | Some x -> x)
codegen.ml:42:and edge_t = L.pointer_type (match L.type_by_name llm
"struct.Edge" with
codegen.ml:43:    None -> raise (Failure "struct.Edge not found")
codegen.ml:44:  | Some x -> x)
codegen.ml:45:and graph_t = L.pointer_type (match L.type_by_name llm
"struct.Graph" with
codegen.ml:46:    None -> raise (Failure "struct.Graph not found")
codegen.ml:47:  | Some x -> x)
codegen.ml:48:
codegen.ml:49:let int_zero = L.const_int i32_t 0
codegen.ml:50:and float_zero = L.const_float f_t 0.
codegen.ml:51:and bool_false = L.const_int i1_t 0
codegen.ml:52:and bool_true = L.const_int i1_t 1
codegen.ml:53:and const_null = L.const_int i32_t 0
codegen.ml:54:and str_null = L.const_null str_t
codegen.ml:55:and list_null = L.const_null list_t
codegen.ml:56:and hmap_null = L.const_null hmap_t
codegen.ml:57:and node_null = L.const_null node_t
codegen.ml:58:and graph_null = L.const_null graph_t
codegen.ml:59:
codegen.ml:60:let ltype_of_typ = function
codegen.ml:61:    A.Int_t -> i32_t
codegen.ml:62:  | A.Float_t -> f_t
codegen.ml:63:  | A.Bool_t -> i1_t
codegen.ml:64:  | A.String_t -> str_t
codegen.ml:65:  | A.List_Int_t -> list_t
codegen.ml:66:  | A.List_Float_t -> list_t
codegen.ml:67:  | A.List_String_t -> list_t
codegen.ml:68:  | A.List_Node_t -> list_t
codegen.ml:69:  | A.List_Graph_t -> list_t
codegen.ml:70:  | A.List_Bool_t -> list_t
codegen.ml:71:  | A.Dict_Int_t -> hmap_t
codegen.ml:72:  | A.Dict_Float_t -> hmap_t
codegen.ml:73:  | A.Dict_String_t -> hmap_t
codegen.ml:74:  | A.Dict_Node_t -> hmap_t
codegen.ml:75:  | A.Dict_Graph_t -> hmap_t
codegen.ml:76:  | A.Node_t -> node_t
codegen.ml:77:  | A.Edge_t -> edge_t
codegen.ml:78:  | A.Graph_t -> graph_t
codegen.ml:79:  | A.Void_t -> void_t
codegen.ml:80:  | _ -> raise (Failure ("ltype_of_typ error"))
codegen.ml:81:
codegen.ml:82:let lconst_of_typ = function
codegen.ml:83:    A.Int_t -> L.const_int i32_t 0
codegen.ml:84:  | A.Float_t -> L.const_int i32_t 1
codegen.ml:85:  | A.Bool_t -> L.const_int i32_t 2
codegen.ml:86:  | A.String_t -> L.const_int i32_t 3
codegen.ml:87:  | A.Node_t -> L.const_int i32_t 4
codegen.ml:88:  | A.Graph_t -> L.const_int i32_t 5
codegen.ml:89:  | A.Edge_t -> L.const_int i32_t 8
codegen.ml:90:  | _ -> raise (Failure ("lconst_of_typ error"))
codegen.ml:91:
codegen.ml:92:let get_list_typ = function
codegen.ml:93:    A.List_Int_t -> A.Int_t
codegen.ml:94:  | A.List_Float_t -> A.Float_t
codegen.ml:95:  | A.List_String_t -> A.String_t
codegen.ml:96:  | A.List_Node_t -> A.Node_t
codegen.ml:97:  | A.List_Graph_t -> A.Graph_t
codegen.ml:98:  | A.List_Bool_t -> A.Bool_t
```

```
codegen.ml:99:  | _ -> raise (Failure ("get_list_typ error"))
codegen.ml:100:
codegen.ml:101:let get_hmap_typ = function
codegen.ml:102:    A.Dict_Int_t -> A.Int_t
codegen.ml:103:  | A.Dict_Float_t -> A.Float_t
codegen.ml:104:  | A.Dict_String_t -> A.String_t
codegen.ml:105:  | A.Dict_Node_t -> A.Node_t
codegen.ml:106:  | A.Dict_Graph_t -> A.Graph_t
codegen.ml:107:  | _ -> raise (Failure ("get_hmap_typ error"))
codegen.ml:108:
codegen.ml:109:let null_of_typ = function
codegen.ml:110:  | A.String_t -> str_null
codegen.ml:111:  | A.Node_t -> node_null
codegen.ml:112:  | A.Graph_t -> graph_null
codegen.ml:113:  | A.List_Int_t
codegen.ml:114:  | A.List_Float_t
codegen.ml:115:  | A.List_String_t
codegen.ml:116:  | A.List_Node_t
codegen.ml:117:  | A.List_Graph_t
codegen.ml:118:  | A.List_Bool_t -> list_null
codegen.ml:119:  | A.Dict_Int_t
codegen.ml:120:  | A.Dict_Float_t
codegen.ml:121:  | A.Dict_String_t
codegen.ml:122:  | A.Dict_Node_t
codegen.ml:123:  | A.Dict_Graph_t -> hmap_null
codegen.ml:124:  | _ -> raise (Failure ("null type error"))
codegen.ml:125:
codegen.ml:126:let init_val = function
codegen.ml:127:  | A.Int_t as t -> L.const_int (ltype_of_typ t) 0
codegen.ml:128:  | A.Bool_t as t -> L.const_int (ltype_of_typ t) 0
codegen.ml:129:  | A.Float_t as t-> L.const_float (ltype_of_typ t) 0.
codegen.ml:130:  | t-> L.const_null (ltype_of_typ t)
codegen.ml:131:
codegen.ml:132:(* Cast data type *)
codegen.ml:133:let to_int_t  = L.function_type i32_t [| L.pointer_type
i8_t |]
codegen.ml:134:let to_int_f  = L.declare_function "VoidtoInt" to_int_t
the_module
codegen.ml:135:let to_int void_ptr llbuilder =
codegen.ml:136:  let actuals = [| void_ptr |] in
codegen.ml:137:    L.build_call to_int_f actuals "VoidtoInt" llbuilder
codegen.ml:138:
codegen.ml:139:let int_to_float llbuilder v = L.build_sitofp v f_t "tmp"
llbuilder
codegen.ml:140:
codegen.ml:141:let to_float_t  = L.function_type f_t [| L.pointer_type
i8_t |]
codegen.ml:142:let to_float_f  = L.declare_function "VoidtoFloat"
to_float_t the_module
codegen.ml:143:let to_float void_ptr llbuilder =
codegen.ml:144:  let actuals = [| void_ptr |] in
codegen.ml:145:    L.build_call to_float_f actuals "VoidtoFloat"
llbuilder
codegen.ml:146:
codegen.ml:147:let to_bool_t  = L.function_type i1_t [| L.pointer_type
i8_t |]
codegen.ml:148:let to_bool_f  = L.declare_function "VoidtoBool" to_bool_t
the_module
codegen.ml:149:let to_bool void_ptr llbuilder =
codegen.ml:150:  let actuals = [| void_ptr |] in
```

```
codegen.ml:151:    L.build_call to_bool_f actuals "VoidtoBool" llbuilder
codegen.ml:152:
codegen.ml:153:let to_string_t  = L.function_type str_t [| L.pointer_type
i8_t |]
codegen.ml:154:let to_string_f  = L.declare_function "VoidtoString"
to_string_t the_module
codegen.ml:155:let to_string void_ptr llbuilder =
codegen.ml:156:  let actuals = [| void_ptr |] in
codegen.ml:157:    L.build_call to_string_f actuals "VoidtoString"
llbuilder
codegen.ml:158:
codegen.ml:159:let to_node_t  = L.function_type node_t [| L.pointer_type
i8_t |]
codegen.ml:160:let to_node_f  = L.declare_function "VoidtoNode" to_node_t
the_module
codegen.ml:161:let to_node void_ptr llbuilder =
codegen.ml:162:  let actuals = [| void_ptr |] in
codegen.ml:163:    L.build_call to_node_f actuals "VoidtoNode" llbuilder
codegen.ml:164:
codegen.ml:165:let to_graph_t  = L.function_type graph_t [|
L.pointer_type i8_t |]
codegen.ml:166:let to_graph_f  = L.declare_function "VoidtoGraph"
to_graph_t the_module
codegen.ml:167:let to_graph void_ptr llbuilder =
codegen.ml:168:  let actuals = [| void_ptr |] in
codegen.ml:169:    L.build_call to_graph_f actuals "VoidtoGraph"
llbuilder
codegen.ml:170:
codegen.ml:171:let void_to_typ data_ptr llbuilder = function
codegen.ml:172:    A.Int_t -> to_int data_ptr llbuilder
codegen.ml:173:  | A.Float_t -> to_float data_ptr llbuilder
codegen.ml:174:  | A.Bool_t -> to_bool data_ptr llbuilder
codegen.ml:175:  | A.String_t -> to_string data_ptr llbuilder
codegen.ml:176:  | A.Node_t -> to_node data_ptr llbuilder
codegen.ml:177:  | A.Graph_t -> to_graph data_ptr llbuilder
codegen.ml:178:  | _ -> raise (Failure("invalid type error"))
codegen.ml:179:
codegen.ml:180:(* Print functions *)
codegen.ml:181:let printf_t = L.var_arg_function_type i32_t [| str_t |]
codegen.ml:182:let printf_func = L.declare_function "printf" printf_t
the_module
codegen.ml:183:let print_gen llbuilder el =
codegen.ml:184:  L.build_call printf_func (Array.of_list el) "printf"
llbuilder
codegen.ml:185:
codegen.ml:186:let print_bool_t = L.function_type i32_t [| i1_t |]
codegen.ml:187:let print_bool_f = L.declare_function "printBool"
print_bool_t the_module
codegen.ml:188:let print_bool e llbuilder =
codegen.ml:189:  L.build_call print_bool_f [| e |] "print_bool" llbuilder
codegen.ml:190:
codegen.ml:191:let print_list_t  = L.function_type i32_t [| list_t |]
codegen.ml:192:let print_list_f  = L.declare_function "printList"
print_list_t the_module
codegen.ml:193:let print_list l llbuilder =
codegen.ml:194:  L.build_call print_list_f [| l |] "printList" llbuilder
codegen.ml:195:
codegen.ml:196:let print_hmap_t  = L.function_type i32_t [| hmap_t |]
codegen.ml:197:let print_hmap_f  = L.declare_function "hashmap_print"
print_hmap_t the_module
```

```
codegen.ml:198:let print_hmap d llbuilder =
codegen.ml:199:  L.build_call print_hmap_f [| d |] "hashmap_print"
llbuilder
codegen.ml:200:
codegen.ml:201:let print_node_t  = L.function_type i32_t [| node_t |]
codegen.ml:202:let print_node_f  = L.declare_function "printNode"
print_node_t the_module
codegen.ml:203:let print_node node llbuilder =
codegen.ml:204:   L.build_call print_node_f [| node |] "printNode"
llbuilder
codegen.ml:205:
codegen.ml:206:let print_edge_t  = L.function_type i32_t [| edge_t |]
codegen.ml:207:let print_edge_f  = L.declare_function "printEdgeValue"
print_edge_t the_module
codegen.ml:208:let print_edge edge llbuilder =
codegen.ml:209:    L.build_call print_edge_f [| edge |] "printEdge"
llbuilder
codegen.ml:210:
codegen.ml:211:let print_graph_t  = L.function_type i32_t [| graph_t |]
codegen.ml:212:let print_graph_f  = L.declare_function "printGraph"
print_graph_t the_module
codegen.ml:213:let print_graph graph llbuilder =
codegen.ml:214:  L.build_call print_graph_f [| graph |] "printGraph"
llbuilder
codegen.ml:215:
codegen.ml:216:let string_lit_gen s llbuilder =
codegen.ml:217:  L.build_global_stringptr s "str_tmp" llbuilder
codegen.ml:218:
codegen.ml:219:(* List functions *)
codegen.ml:220:let list_init_t  = L.function_type list_t [| i32_t |]
codegen.ml:221:let list_init_f  = L.declare_function "createList"
list_init_t the_module
codegen.ml:222:let list_init typ llbuilder =
codegen.ml:223:  let actuals = [|lconst_of_typ typ|]in (
codegen.ml:224:    L.build_call list_init_f actuals "createList"
llbuilder
codegen.ml:225:  )
codegen.ml:226:
codegen.ml:227:let list_remove_t  = L.var_arg_function_type list_t [|
list_t |]
codegen.ml:228:let list_remove_f  = L.declare_function "removeData"
list_remove_t the_module
codegen.ml:229:let list_remove data l_ptr llbuilder =
codegen.ml:230:  let actuals = [| l_ptr; data|] in
codegen.ml:231:    (L.build_call list_remove_f actuals "removeData"
llbuilder)
codegen.ml:232:
codegen.ml:233:let list_add_t  = L.var_arg_function_type list_t [| list_t
|]
codegen.ml:234:let list_add_f  = L.declare_function "addList" list_add_t
the_module
codegen.ml:235:let list_add data l_ptr llbuilder =
codegen.ml:236:  let actuals = [| l_ptr; data|] in
codegen.ml:237:    (L.build_call list_add_f actuals "addList" llbuilder)
codegen.ml:238:
codegen.ml:239:let list_set_t  = L.var_arg_function_type i32_t [| list_t;
i32_t |]
codegen.ml:240:let list_set_f  = L.declare_function "setList" list_set_t
the_module
codegen.ml:241:let list_set l_ptr index data llbuilder =
```

```
codegen.ml:242:  let actuals = [| l_ptr; index; data |] in
codegen.ml:243:    ignore(L.build_call list_set_f actuals "setList"
llbuilder);
codegen.ml:244:    l_ptr
codegen.ml:245:
codegen.ml:246:let list_size_t  = L.var_arg_function_type i32_t [| list_t
|]
codegen.ml:247:let list_size_f  = L.declare_function "getListSize"
list_size_t the_module
codegen.ml:248:let list_size l_ptr llbuilder =
codegen.ml:249:  let actuals = [| l_ptr |] in
codegen.ml:250:    L.build_call list_size_f actuals "getListSize"
llbuilder
codegen.ml:251:
codegen.ml:252:let list_pop_t  = L.var_arg_function_type (L.pointer_type
i8_t) [| list_t |]
codegen.ml:253:let list_pop_f  = L.declare_function "popList" list_pop_t
the_module
codegen.ml:254:let list_pop l_ptr typ llbuilder =
codegen.ml:255:  let actuals = [| l_ptr |] in
codegen.ml:256:  let data_ptr = L.build_call list_pop_f actuals "popList"
llbuilder in
codegen.ml:257:  void_to_typ data_ptr llbuilder typ
codegen.ml:258:
codegen.ml:259:let list_get_t  = L.var_arg_function_type (L.pointer_type
i8_t) [| list_t; i32_t|]
codegen.ml:260:let list_get_f  = L.declare_function "getList" list_get_t
the_module
codegen.ml:261:let list_get l_ptr index typ llbuilder =
codegen.ml:262:  let actuals = [| l_ptr; index|] in
codegen.ml:263:  let data_ptr = L.build_call list_get_f actuals "getList"
llbuilder in
codegen.ml:264:  void_to_typ data_ptr llbuilder typ
codegen.ml:265:
codegen.ml:266:let list_concat_t  = L.var_arg_function_type list_t [|
list_t; list_t |]
codegen.ml:267:let list_concat_f  = L.declare_function "concatList"
list_concat_t the_module
codegen.ml:268:let list_concat l_ptr1 l_ptr2 llbuilder =
codegen.ml:269:  let actuals = [| l_ptr1; l_ptr2 |] in
codegen.ml:270:    L.build_call list_concat_f actuals "concatList"
llbuilder
codegen.ml:271:
codegen.ml:272:let list_contains_t  = L.var_arg_function_type i1_t [|
list_t |]
codegen.ml:273:let list_contains_f  = L.declare_function "listContains"
list_contains_t the_module
codegen.ml:274:let list_contains l_ptr data llbuilder =
codegen.ml:275:  let actuals = [| l_ptr; data|] in
codegen.ml:276:    (L.build_call list_contains_f actuals "listContains"
llbuilder)
codegen.ml:277:
codegen.ml:278:let cast_float data typ builder =
codegen.ml:279:  if typ == A.Float_t then int_to_float builder data else
data
codegen.ml:280:
codegen.ml:281:let rec list_add_all l_ptr typ llbuilder = function
codegen.ml:282:  | [] -> l_ptr
codegen.ml:283:  | h :: tl -> list_add_all (list_add (cast_float h typ
llbuilder) l_ptr llbuilder) typ llbuilder tl
```

```
codegen.ml:284:
codegen.ml:285:let list_func builder list_ptr params_list expr_tpy =
function
codegen.ml:286:  | "remove" -> (list_remove (List.hd params_list)
list_ptr builder), expr_tpy
codegen.ml:287:  | "add" -> (list_add (List.hd params_list) list_ptr
builder), expr_tpy
codegen.ml:288:  | "get" -> (list_get list_ptr (  List.hd params_list)
(get_list_typ expr_tpy) builder), (get_list_typ expr_tpy)
codegen.ml:289:  | "set" -> (list_set list_ptr (List.hd params_list)
(List.nth params_list 1) builder), expr_tpy
codegen.ml:290:  | "size" -> (list_size list_ptr builder), A.Int_t
codegen.ml:291:  | "pop" -> (list_pop list_ptr (get_list_typ expr_tpy)
builder), (get_list_typ expr_tpy)
codegen.ml:292:  | "push" -> (list_add (List.hd params_list) list_ptr
builder), expr_tpy
codegen.ml:293:  | "contains" -> (list_contains list_ptr (List.hd
params_list) builder), A.Bool_t
codegen.ml:294:  | _ -> raise (Failure ("invalid list function"))
codegen.ml:295:
codegen.ml:296:(* Hashmap functions *)
codegen.ml:297:let hmap_init_t = L.var_arg_function_type hmap_t [| i32_t;
i32_t |]
codegen.ml:298:let hmap_init_f = L.declare_function "hashmap_new"
hmap_init_t the_module
codegen.ml:299:let hmap_init fst_typ snd_typ llbuilder =
codegen.ml:300:    L.build_call hmap_init_f [| fst_typ; snd_typ |]
"hashmap" llbuilder
codegen.ml:301:
codegen.ml:302:let hmap_put_t = L.var_arg_function_type hmap_t [| hmap_t
|]
codegen.ml:303:let hmap_put_f = L.declare_function "hashmap_put"
hmap_put_t the_module
codegen.ml:304:let hmap_put d key v llbuilder =
codegen.ml:305:    let actuals = [| d; key; v |] in
codegen.ml:306:    ignore (L.build_call hmap_put_f actuals "hashmap_put"
llbuilder); d
codegen.ml:307:
codegen.ml:308:let hmap_get_t = L.var_arg_function_type (L.pointer_type
i8_t) [| hmap_t |]
codegen.ml:309:let hmap_get_f = L.declare_function "hashmap_get"
hmap_get_t the_module
codegen.ml:310:let hmap_get hmap_ptr key llbuilder v_typ =
codegen.ml:311:    let actuals = [| hmap_ptr; key |] in
codegen.ml:312:    let data_ptr = L.build_call hmap_get_f actuals
"hashmap_get" llbuilder in
codegen.ml:313:    void_to_typ data_ptr llbuilder v_typ
codegen.ml:314:
codegen.ml:315:let hmap_remove_t = L.var_arg_function_type hmap_t [|
hmap_t |]
codegen.ml:316:let hmap_remove_f = L.declare_function "hashmap_remove"
hmap_remove_t the_module
codegen.ml:317:let hmap_remove hmap_ptr key llbuilder =
codegen.ml:318:    let actuals = [| hmap_ptr; key |] in
codegen.ml:319:    L.build_call hmap_remove_f actuals "hashmap_remove"
llbuilder
codegen.ml:320:
codegen.ml:321:let hmap_size_t = L.var_arg_function_type i32_t [| hmap_t
|]
```

```
codegen.ml:322:let hmap_size_f = L.declare_function "hashmap_length"
hmap_size_t the_module
codegen.ml:323:let hmap_size hmap_ptr llbuilder =
codegen.ml:324:    let actuals = [| hmap_ptr |] in
codegen.ml:325:    L.build_call hmap_size_f actuals "hashmap_length"
llbuilder
codegen.ml:326:
codegen.ml:327:let hmap_keys_t = L.var_arg_function_type list_t [| hmap_t
|]
codegen.ml:328:let hmap_keys_f = L.declare_function "hashmap_keys"
hmap_keys_t the_module
codegen.ml:329:let hmap_keys hmap_ptr llbuilder =
codegen.ml:330:    let actuals = [| hmap_ptr |] in
codegen.ml:331:    L.build_call hmap_keys_f actuals "hashmap_keys"
llbuilder
codegen.ml:332:
codegen.ml:333:let hmap_key_typ_t = L.var_arg_function_type i32_t [|
hmap_t |]
codegen.ml:334:let hmap_key_typ_f = L.declare_function "hashmap_keytype"
hmap_key_typ_t the_module
codegen.ml:335:let hmap_key_typ hmap_ptr llbuilder =
codegen.ml:336:    let actuals = [| hmap_ptr |] in
codegen.ml:337:    L.build_call hmap_key_typ_f actuals "hashmap_keytype"
llbuilder
codegen.ml:338:
codegen.ml:339:let hmap_has_key_t = L.var_arg_function_type i1_t [|
hmap_t |]
codegen.ml:340:let hmap_has_key_f = L.declare_function "hashmap_haskey"
hmap_has_key_t the_module
codegen.ml:341:let hmap_has_key hmap_ptr key llbuilder =
codegen.ml:342:    let actuals = [| hmap_ptr; key |] in
codegen.ml:343:    L.build_call hmap_has_key_f actuals "hashmap_haskey"
llbuilder
codegen.ml:344:
codegen.ml:345:let rec hmap_put_all hmap_ptr llbuilder = function
codegen.ml:346:  | [] -> hmap_ptr
codegen.ml:347:  | hd :: tl -> ignore(hmap_put hmap_ptr (fst hd) (snd hd)
llbuilder); hmap_put_all hmap_ptr llbuilder tl
codegen.ml:348:
codegen.ml:349:let hmap_func builder hmap_ptr params_list v_typ =
function
codegen.ml:350:  | "get" -> (hmap_get hmap_ptr (List.hd params_list)
builder (get_hmap_typ v_typ)), (get_hmap_typ v_typ)
codegen.ml:351:  | "put" -> (hmap_put hmap_ptr (List.hd params_list)
(List.nth params_list 1) builder), v_typ
codegen.ml:352:  | "remove" -> (hmap_remove hmap_ptr (List.hd
params_list) builder), v_typ
codegen.ml:353:  | "size" -> (hmap_size hmap_ptr builder), A.Int_t
codegen.ml:354:  | "keys" -> (hmap_keys hmap_ptr builder), A.List_Null_t
codegen.ml:355:  | "has" -> (hmap_has_key hmap_ptr (List.hd params_list)
builder), A.Bool_t
codegen.ml:356:  | _ as name -> raise (Failure ("invalid hashmap call" ^
name))
codegen.ml:357:
codegen.ml:358:(* Node functions *)
codegen.ml:359:let create_node_t  = L.var_arg_function_type node_t [|
i32_t; i32_t |]
codegen.ml:360:let create_node_f  = L.declare_function "createNode"
create_node_t the_module
codegen.ml:361:let create_node (id, typ, nval) llbuilder =
```

```
codegen.ml:362:  let actuals = [| id; lconst_of_typ typ; nval |] in
codegen.ml:363:    L.build_call create_node_f actuals "node" llbuilder
codegen.ml:364:
codegen.ml:365:let node_get_value_t = L.function_type ptr_t [| node_t;
i32_t |]
codegen.ml:366:let node_get_value_f = L.declare_function "nodeGetValue"
node_get_value_t the_module
codegen.ml:367:let node_get_value node typ llbuilder =
codegen.ml:368:  let actuals = [| node; lconst_of_typ typ |] in
codegen.ml:369:  let ret = L.build_call node_get_value_f actuals
"nodeValue" llbuilder in
codegen.ml:370:  (   match typ with
codegen.ml:371:    | A.Int_t -> to_int ret llbuilder
codegen.ml:372:    | A.Float_t -> to_float ret llbuilder
codegen.ml:373:    | A.Bool_t -> to_bool ret llbuilder
codegen.ml:374:    | A.String_t -> to_string ret llbuilder
codegen.ml:375:    | _ -> raise (Failure("invalid node value"))
codegen.ml:376:  )
codegen.ml:377:
codegen.ml:378:let set_visited_t  = L.function_type i1_t [| node_t |]
codegen.ml:379:  let set_visited_f  = L.declare_function "setVisited"
set_visited_t the_module
codegen.ml:380:  let set_visited node llbuilder =
codegen.ml:381:    L.build_call set_visited_f [| node |] "setVisited"
llbuilder
codegen.ml:382:
codegen.ml:383:let is_visited_t  = L.function_type i1_t [| node_t |]
codegen.ml:384:  let is_visited_f  = L.declare_function "isVisited"
is_visited_t the_module
codegen.ml:385:  let is_visited node llbuilder =
codegen.ml:386:    L.build_call is_visited_f [| node |] "isVisited"
llbuilder
codegen.ml:387:
codegen.ml:388:let node_func builder node_ptr = function
codegen.ml:389: | "setVisited" -> set_visited node_ptr builder, A.Bool_t
codegen.ml:390: | "isVisited" -> is_visited node_ptr builder, A.Bool_t
codegen.ml:391: | _ as name -> raise (Failure ("node function error" ^
name))
codegen.ml:392:
codegen.ml:393:(* Edge functions *)
codegen.ml:394:let edge_get_value_t = L.function_type ptr_t [| edge_t;
i32_t |]
codegen.ml:395:let edge_get_value_f = L.declare_function "edgeGetValue"
edge_get_value_t the_module
codegen.ml:396:let edge_get_value edge typ llbuilder =
codegen.ml:397:  let actuals = [| edge; lconst_of_typ typ |] in
codegen.ml:398:  let ret = L.build_call edge_get_value_f actuals
"edgeValue" llbuilder in
codegen.ml:399:  (   match typ with
codegen.ml:400:    | A.Int_t -> to_int ret llbuilder
codegen.ml:401:    | A.Float_t -> to_float ret llbuilder
codegen.ml:402:    | A.Bool_t -> to_bool ret llbuilder
codegen.ml:403:    | A.String_t -> to_string ret llbuilder
codegen.ml:404:    | _ -> raise (Failure("invalid edge value"))
codegen.ml:405:  )
codegen.ml:406:
codegen.ml:407:(* Graph functions *)
codegen.ml:408:let create_graph_t  = L.function_type graph_t [| |]
codegen.ml:409:let create_graph_f  = L.declare_function "createGraph"
create_graph_t the_module
```

```
codegen.ml:410:let create_graph llbuilder =
codegen.ml:411:  L.build_call create_graph_f [| |] "graph" llbuilder
codegen.ml:412:
codegen.ml:413:let graph_num_of_nodes_t  = L.function_type i32_t [|
graph_t |]
codegen.ml:414:let graph_num_of_nodes_f  = L.declare_function
"graphNumOfNodes" graph_num_of_nodes_t the_module
codegen.ml:415:let graph_num_of_nodes g llbuilder =
codegen.ml:416:  L.build_call graph_num_of_nodes_f [| g |]
"graphNodeSize" llbuilder
codegen.ml:417:
codegen.ml:418:let graph_num_of_edges_t  = L.function_type i32_t [|
graph_t |]
codegen.ml:419:let graph_num_of_edges_f  = L.declare_function
"graphNumOfEdges" graph_num_of_edges_t the_module
codegen.ml:420:let graph_num_of_edges g llbuilder =
codegen.ml:421:  L.build_call graph_num_of_edges_f [| g |]
"graphEdgeSize" llbuilder
codegen.ml:422:
codegen.ml:423:let copy_graph_t  = L.function_type graph_t [| graph_t |]
codegen.ml:424:let copy_graph_f  = L.declare_function "copyGraph"
copy_graph_t the_module
codegen.ml:425:let copy_graph g llbuilder =
codegen.ml:426:  L.build_call copy_graph_f [| g |] "graph" llbuilder
codegen.ml:427:
codegen.ml:428:let merge_graph_t = L.function_type graph_t [| graph_t;
graph_t |]
codegen.ml:429:let merge_graph_f = L.declare_function "mergeGraph"
merge_graph_t the_module
codegen.ml:430:let merge_graph g1 g2 llbuilder =
codegen.ml:431:  L.build_call merge_graph_f [| g1; g2 |] "graph"
llbuilder
codegen.ml:432:
codegen.ml:433:let graph_get_root_t  = L.function_type node_t [| graph_t
|]
codegen.ml:434:let graph_get_root_f  = L.declare_function "graphGetRoot"
graph_get_root_t the_module
codegen.ml:435:let graph_get_root g llbuilder =
codegen.ml:436:  L.build_call graph_get_root_f [| g |] "rootNode"
llbuilder
codegen.ml:437:
codegen.ml:438:let graph_set_root_t  = L.function_type graph_t [|
graph_t; node_t |]
codegen.ml:439:let graph_set_root_f  = L.declare_function "graphSetRoot"
graph_set_root_t the_module
codegen.ml:440:let graph_set_root graph node llbuilder = (
codegen.ml:441:      ignore(L.build_call graph_set_root_f [| graph; node
|] "setRootRes" llbuilder);
codegen.ml:442:      graph
codegen.ml:443:  )
codegen.ml:444:
codegen.ml:445:let graph_add_list_t = L.function_type i32_t [| graph_t;
i32_t; list_t; list_t |]
codegen.ml:446:let graph_add_list_f = L.declare_function "graphAddList"
graph_add_list_t the_module
codegen.ml:447:let graph_add_list graph vals (edges, etyp) llbuilder =
codegen.ml:448:  let edges = (
codegen.ml:449:    match etyp with
codegen.ml:450:    | A.List_Int_t | A.List_Float_t | A.List_String_t
```

```
codegen.ml:451:    | A.List_Node_t | A.List_Graph_t | A.List_Bool_t ->
edges
codegen.ml:452:    | _ -> list_null
codegen.ml:453:  ) in
codegen.ml:454:  L.build_call graph_add_list_f [| graph; vals; edges |]
"graphAddList" llbuilder
codegen.ml:455:
codegen.ml:456:let graph_add_node_t = L.function_type i32_t [| graph_t;
node_t |]
codegen.ml:457:let graph_add_node_f = L.declare_function "graphAddNode"
graph_add_node_t the_module
codegen.ml:458:let graph_add_node graph node llbuilder =
codegen.ml:459:  L.build_call graph_add_node_f [| graph; node |]
"addNodeRes" llbuilder
codegen.ml:460:
codegen.ml:461:let graph_add_edge_t = L.function_type i32_t
codegen.ml:462:  [| graph_t; node_t; node_t; i32_t; i32_t; f_t; i1_t;
str_t |]
codegen.ml:463:let graph_add_edge_f = L.declare_function "graphAddEdge"
graph_add_edge_t the_module
codegen.ml:464:let graph_add_edge graph (sour, dest) (typ, vals)
llbuilder =
codegen.ml:465:  let actuals = [| graph; sour; dest; int_zero; int_zero;
float_zero; bool_false; str_null |] in
codegen.ml:466:  let actuals_r = [| graph; dest; sour; int_zero;
int_zero; float_zero; bool_false; str_null |] in
codegen.ml:467:  let (typ_val, loc) = (match typ with
codegen.ml:468:    | A.Int_t -> (0, 4)
codegen.ml:469:    | A.Float_t -> (1, 5)
codegen.ml:470:    | A.Bool_t -> (2, 6)
codegen.ml:471:    | A.String_t -> (3, 7)
codegen.ml:472:    | A.Void_t | A.Null_t -> (-1, 4)
codegen.ml:473:    | _ -> raise (Failure "invalid edge value")
codegen.ml:474:  ) in (
codegen.ml:475:    ignore( actuals.(3) <- (L.const_int i32_t typ_val) );
codegen.ml:476:    ignore( actuals_r.(3) <- (L.const_int i32_t typ_val)
);
codegen.ml:477:    ignore( actuals.(loc) <- vals );
codegen.ml:478:    ignore( actuals_r.(loc) <- vals );
codegen.ml:479:    L.build_call graph_add_edge_f actuals
"addRightEdgeRes" llbuilder
codegen.ml:480:  )
codegen.ml:481:
codegen.ml:482:let graph_edge_exist_t = L.function_type i1_t [| graph_t;
node_t; node_t |]
codegen.ml:483:let graph_edge_exist_f = L.declare_function
"graphEdgeExist" graph_edge_exist_t the_module
codegen.ml:484:let graph_edge_exist graph sour dest llbuilder =
codegen.ml:485:  L.build_call graph_edge_exist_f [| graph; sour; dest |]
"graphEdgeExist" llbuilder
codegen.ml:486:
codegen.ml:487:let graph_get_edge_t = L.function_type edge_t [| graph_t;
node_t; node_t |]
codegen.ml:488:let graph_get_edge_f = L.declare_function "graphGetEdge"
graph_get_edge_t the_module
codegen.ml:489:let graph_get_edge graph sour dest llbuilder =
codegen.ml:490:  L.build_call graph_get_edge_f [| graph; sour; dest |]
"edgeValue" llbuilder
codegen.ml:491:
```

```
codegen.ml:492:let graph_get_child_nodes_t  = L.function_type list_t [|
graph_t; node_t |]
codegen.ml:493:let graph_get_child_nodes_f  = L.declare_function
"graphGetChildNodes" graph_get_child_nodes_t the_module
codegen.ml:494:let graph_get_child_nodes graph root llbuilder =
codegen.ml:495:  L.build_call graph_get_child_nodes_f [| graph; root |]
"childNodes" llbuilder
codegen.ml:496:
codegen.ml:497:let graph_get_all_nodes_t  = L.function_type list_t [|
graph_t |]
codegen.ml:498:let graph_get_all_nodes_f  = L.declare_function
"graphGetAllNodes" graph_get_all_nodes_t the_module
codegen.ml:499:let graph_get_all_nodes graph llbuilder =
codegen.ml:500:  L.build_call graph_get_all_nodes_f [| graph |]
"nodesList" llbuilder
codegen.ml:501:
codegen.ml:502:let graph_remove_node_t = L.function_type list_t [|
graph_t; node_t |]
codegen.ml:503:let graph_remove_node_f = L.declare_function
"graphRemoveNode" graph_remove_node_t the_module
codegen.ml:504:let graph_remove_node graph node llbuilder =
codegen.ml:505:  L.build_call graph_remove_node_f [| graph; node |]
"listOfSubGraphs" llbuilder
codegen.ml:506:
codegen.ml:507:let graph_sub_graph_t = L.function_type list_t [| graph_t;
graph_t |]
codegen.ml:508:let graph_sub_graph_f = L.declare_function "subGraph"
graph_sub_graph_t the_module
codegen.ml:509:let graph_sub_graph g1 g2 llbuilder =
codegen.ml:510:  L.build_call graph_sub_graph_f [| g1; g2 |]
"listOfSubGraphs" llbuilder
codegen.ml:511:
codegen.ml:512:let set_allunvisited_t = L.function_type i1_t [|graph_t|]
codegen.ml:513:let set_allunvisited_f = L.declare_function
"setAllUnvisited" set_allunvisited_t the_module
codegen.ml:514:let set_allunvisited g llbuilder =
codegen.ml:515:  L.build_call set_allunvisited_f [|g|] "setAllUnvisited"
llbuilder
codegen.ml:516:
codegen.ml:517:let dfs_t = L.function_type list_t [| graph_t; node_t |]
codegen.ml:518:let dfs_f = L.declare_function "dfs" dfs_t the_module
codegen.ml:519:let dfs g sour llbuilder =
codegen.ml:520:  L.build_call dfs_f [|g; sour |] "dfs" llbuilder
codegen.ml:521:
codegen.ml:522:
codegen.ml:523:let bfs_t = L.function_type list_t [| graph_t; node_t |]
codegen.ml:524:let bfs_f = L.declare_function "bfs" bfs_t the_module
codegen.ml:525:let bfs g sour llbuilder =
codegen.ml:526:  L.build_call bfs_f [|g; sour |] "bfs" llbuilder
codegen.ml:527:
codegen.ml:528:let graph_contains_node_t = L.function_type i1_t [|
graph_t; node_t |]
codegen.ml:529:let graph_contains_node_f = L.declare_function
"containsNode" graph_contains_node_t the_module
codegen.ml:530:let graph_contains_node graph node llbuilder =
codegen.ml:531:  L.build_call graph_contains_node_f [| graph; node |]
"containsNode" llbuilder
codegen.ml:532:
codegen.ml:533:let graph_add_edgeP_t = L.var_arg_function_type i32_t [|
graph_t; node_t; node_t; i32_t|]
```

```
codegen.ml:534:let graph_add_edgeP_f = L.declare_function
"graphAddEdgeP" graph_add_edgeP_t the_module
codegen.ml:535:let graph_add_edgeP g n1 n2 typ data llbuilder =
codegen.ml:536:  L.build_call graph_add_edgeP_f [| g; n1; n2; typ; data
|] "graphAddEdge" llbuilder
codegen.ml:537:
codegen.ml:538:let dijkstra_t = L.function_type list_t [| graph_t;
node_t; node_t |]
codegen.ml:539:let dijkstra_f = L.declare_function "dijkstra" dijkstra_t
the_module
codegen.ml:540:let dijkstra graph n1 n2 llbuilder =
codegen.ml:541:  L.build_call dijkstra_f [| graph; n1; n2 |] "dijkstra"
llbuilder
codegen.ml:542:
codegen.ml:543:let graph_call_default_main llbuilder gh params_list
fname=
codegen.ml:544:  let param_list = (List.map (fun e -> fst(e))
params_list) in
codegen.ml:545:  match fname with
codegen.ml:546:  | "root" -> graph_get_root gh llbuilder , A.Node_t
codegen.ml:547:  | "size" -> graph_num_of_nodes gh llbuilder, A.Int_t
codegen.ml:548:  | "getAllNodes" -> graph_get_all_nodes gh llbuilder,
A.List_Node_t
codegen.ml:549:  | "setAllUnvisited" -> set_allunvisited gh llbuilder,
A.Bool_t
codegen.ml:550:  | "dfs" -> dfs gh (List.hd param_list) llbuilder,
A.List_Node_t
codegen.ml:551:  | "bfs" -> bfs gh (List.hd param_list) llbuilder,
A.List_Node_t
codegen.ml:552:  | "hasNode" -> graph_contains_node gh (List.hd
param_list) llbuilder, A.Bool_t
codegen.ml:553:  | "hasEdge" -> graph_edge_exist gh (List.hd param_list)
(List.nth param_list 1) llbuilder, A.Bool_t
codegen.ml:554:  | "addNode" -> graph_add_node gh (List.hd param_list)
llbuilder, A.Int_t
codegen.ml:555:  | "addEdge" -> graph_add_edge gh (List.hd param_list,
List.nth param_list 1) (snd (List.nth params_list 2), fst (List.nth
params_list 2)) llbuilder, A.Int_t
codegen.ml:556:  | "dijkstra" -> dijkstra gh (List.hd param_list)
(List.nth param_list 1) llbuilder, A.List_Node_t
codegen.ml:557:  | _ as name -> raise (Failure("graph function error " ^
name ))
codegen.ml:558:
codegen.ml:559:let context_funcs_vars = Hashtbl.create 50
codegen.ml:560:let print_hashtbl tb =
codegen.ml:561:  print_endline (Hashtbl.fold (fun k _ m -> (k^", "^m)) tb
"")
codegen.ml:562:
codegen.ml:563:(* Codegen entry *)
codegen.ml:564:let translate program =
codegen.ml:565:  (* Define each function (arguments and return type) so
we can call it *)
codegen.ml:566:  let function_decls =
codegen.ml:567:    let function_decl m fdecl =
codegen.ml:568:      let name = fdecl.A.name
codegen.ml:569:      and formal_types =
codegen.ml:570:         Array.of_list (List.map (fun (A.Formal(t, _)) ->
ltype_of_typ t) fdecl.A.args)
codegen.ml:571:      in
```

```
codegen.ml:572:      let ftype = L.var_arg_function_type (ltype_of_typ
fdecl.A.returnType) formal_types in
codegen.ml:573:      StringMap.add name (L.define_function name ftype
the_module, fdecl) m in
codegen.ml:574:    List.fold_left function_decl StringMap.empty program
in
codegen.ml:575:
codegen.ml:576:  (* Fill in the body of the given function *)
codegen.ml:577:  let build_function_body fdecl =
codegen.ml:578:    let get_var_name fname n = (fname ^ "." ^ n) in
codegen.ml:579:    let (the_function, _) = StringMap.find fdecl.A.name
function_decls in
codegen.ml:580:    (* let bb = L.append_block context "entry"
the_function in *)
codegen.ml:581:    let builder = L.builder_at_end context (L.entry_block
the_function) in
codegen.ml:582:
codegen.ml:583:    (* Construct the function's "locals": formal arguments
and locally
codegen.ml:584:       declared variables.  Allocate each on the stack,
initialize their
codegen.ml:585:       value, if appropriate, and remember their values in
the "locals" map *)
codegen.ml:586:    let _ =
codegen.ml:587:      let add_to_context locals =
codegen.ml:588:        ignore(Hashtbl.add context_funcs_vars fdecl.A.name
locals);
codegen.ml:589:        locals
codegen.ml:590:      in
codegen.ml:591:      let add_formal m (A.Formal(t, n)) p =
codegen.ml:592:        let n' = get_var_name fdecl.A.name n in
codegen.ml:593:        let local = L.define_global n' (init_val t)
the_module in
codegen.ml:594:          if L.is_null p then () else ignore
(L.build_store p local builder);
codegen.ml:595:         StringMap.add n' (local, t) m
codegen.ml:596:      in
codegen.ml:597:
codegen.ml:598:      let add_local m (A.Formal(t, n)) =
codegen.ml:599:        let n' = get_var_name fdecl.A.name n in
codegen.ml:600:       let local_var = L.define_global n' (init_val t)
the_module in
codegen.ml:601:        StringMap.add n' (local_var, t) m
codegen.ml:602:      in
codegen.ml:603:
codegen.ml:604:      let formals = List.fold_left2 add_formal
StringMap.empty fdecl.A.args
codegen.ml:605:          (Array.to_list (L.params the_function)) in
codegen.ml:606:      add_to_context (List.fold_left add_local formals
fdecl.A.locals)
codegen.ml:607:    in
codegen.ml:608:
codegen.ml:609:    (* Return the value for a variable or formal argument
*)
codegen.ml:610:    (* let lookup n = StringMap.find n local_vars
codegen.ml:611:    in *)
codegen.ml:612:    let lookup n =
codegen.ml:613:      let get_parent_func_name fname =
codegen.ml:614:        let (_, fdecl) = StringMap.find fname
function_decls in
```

```
codegen.ml:615:        fdecl.A.pname
codegen.ml:616:      in
codegen.ml:617:      let rec aux n fname = (
codegen.ml:618:        try StringMap.find (get_var_name fname n)
(Hashtbl.find context_funcs_vars fname)
codegen.ml:619:        with Not_found -> (
codegen.ml:620:          if fname = "main" then
codegen.ml:621:            (raise (Failure("variable not found")))
codegen.ml:622:          else
codegen.ml:623:            (aux n (get_parent_func_name fname))
codegen.ml:624:        )
codegen.ml:625:      ) in
codegen.ml:626:      aux n fdecl.A.name
codegen.ml:627:    in
codegen.ml:628:
codegen.ml:629:    (* Construct code for an expression; return its value
*)
codegen.ml:630:    let handle_binop e1 op e2 dtype llbuilder =
codegen.ml:631:      (* Generate llvalues from e1 and e2 *)
codegen.ml:632:
codegen.ml:633:      let float_ops op e1 e2 =
codegen.ml:634:        match op with
codegen.ml:635:          A.Add     -> L.build_fadd e1 e2 "flt_addtmp"
llbuilder
codegen.ml:636:        | A.Sub     -> L.build_fsub e1 e2 "flt_subtmp"
llbuilder
codegen.ml:637:        | A.Mult    -> L.build_fmul e1 e2 "flt_multmp"
llbuilder
codegen.ml:638:        | A.Div     -> L.build_fdiv e1 e2 "flt_divtmp"
llbuilder
codegen.ml:639:        | A.Mod     -> L.build_frem e1 e2 "flt_sremtmp"
llbuilder
codegen.ml:640:        | A.Equal   -> L.build_fcmp L.Fcmp.Oeq e1 e2
"flt_eqtmp" llbuilder
codegen.ml:641:        | A.Neq     -> L.build_fcmp L.Fcmp.One e1 e2
"flt_neqtmp" llbuilder
codegen.ml:642:        | A.Less    -> L.build_fcmp L.Fcmp.Ult e1 e2
"flt_lesstmp" llbuilder
codegen.ml:643:        | A.Leq     -> L.build_fcmp L.Fcmp.Ole e1 e2
"flt_leqtmp" llbuilder
codegen.ml:644:        | A.Greater -> L.build_fcmp L.Fcmp.Ogt e1 e2
"flt_sgttmp" llbuilder
codegen.ml:645:        | A.Geq     -> L.build_fcmp L.Fcmp.Oge e1 e2
"flt_sgetmp" llbuilder
codegen.ml:646:        | _ -> raise (Failure("invalid binary operation"))
codegen.ml:647:      in
codegen.ml:648:
codegen.ml:649:      (* chars are considered ints, so they will use
int_ops as well*)
codegen.ml:650:      let int_ops op e1 e2 =
codegen.ml:651:        match op with
codegen.ml:652:          A.Add     -> L.build_add e1 e2 "addtmp"
llbuilder
codegen.ml:653:        | A.Sub     -> L.build_sub e1 e2 "subtmp"
llbuilder
codegen.ml:654:        | A.Mult    -> L.build_mul e1 e2 "multmp"
llbuilder
codegen.ml:655:        | A.Div     -> L.build_sdiv e1 e2 "divtmp"
llbuilder
```

```
codegen.ml:656:        | A.Mod     -> L.build_srem e1 e2 "sremtmp"
llbuilder
codegen.ml:657:        | A.And     -> L.build_and e1 e2 "andtmp"
llbuilder
codegen.ml:658:        | A.Or      -> L.build_or  e1 e2 "ortmp" llbuilder
codegen.ml:659:        | A.Equal   -> L.build_icmp L.Icmp.Eq e1 e2
"eqtmp" llbuilder
codegen.ml:660:        | A.Neq     -> L.build_icmp L.Icmp.Ne e1 e2
"neqtmp" llbuilder
codegen.ml:661:        | A.Less    -> L.build_icmp L.Icmp.Slt e1 e2
"lesstmp" llbuilder
codegen.ml:662:        | A.Leq     -> L.build_icmp L.Icmp.Sle e1 e2
"leqtmp" llbuilder
codegen.ml:663:        | A.Greater -> L.build_icmp L.Icmp.Sgt e1 e2
"sgttmp" llbuilder
codegen.ml:664:        | A.Geq     -> L.build_icmp L.Icmp.Sge e1 e2
"sgetmp" llbuilder
codegen.ml:665:        | _ -> raise (Failure("invalid binary operation"))
codegen.ml:666:        in
codegen.ml:667:      let type_handler d = match d with
codegen.ml:668:        | A.Float_t -> float_ops op e1 e2
codegen.ml:669:        | A.Bool_t
codegen.ml:670:        | A.Int_t -> int_ops op e1 e2
codegen.ml:671:        | _ -> raise (Failure("invalid type error"))
codegen.ml:672:      in (type_handler dtype,
codegen.ml:673:        match op with
codegen.ml:674:        | A.Add | A.Sub | A.Mult | A.Div | A.Mod -> dtype
codegen.ml:675:        | _ -> A.Bool_t
codegen.ml:676:      )
codegen.ml:677:    in
codegen.ml:678:
codegen.ml:679:    let rec expr builder = function
codegen.ml:680:      A.Num_Lit(A.Num_Int i)   -> (L.const_int i32_t i,
A.Int_t)
codegen.ml:681:      | A.Num_Lit(A.Num_Float f) -> (L.const_float f_t f,
A.Float_t)
codegen.ml:682:      | A.Bool_lit b -> (L.const_int i1_t (if b then 1
else 0), A.Bool_t)
codegen.ml:683:      | A.String_Lit s -> (string_lit_gen s builder,
A.String_t)
codegen.ml:684:      | A.Noexpr -> (L.const_int i32_t 0, A.Void_t)
codegen.ml:685:      | A.Null -> (const_null, A.Null_t)
codegen.ml:686:      | A.Id s ->
codegen.ml:687:          let (var, typ) = lookup s in
codegen.ml:688:          (L.build_load var s builder, typ)
codegen.ml:689:      | A.Node(id, e) ->
codegen.ml:690:          let (nval, typ) = expr builder e in
codegen.ml:691:          (create_node (L.const_int i32_t id, typ, nval)
builder, A.Node_t)
codegen.ml:692:      | A.ListP(ls) ->
codegen.ml:693:          let from_expr_typ_to_list_typ = function
codegen.ml:694:              A.Int_t -> A.List_Int_t
codegen.ml:695:            | A.Float_t -> A.List_Float_t
codegen.ml:696:            | A.String_t -> A.List_String_t
codegen.ml:697:            | A.Node_t -> A.List_Node_t
codegen.ml:698:            | A.Graph_t -> A.List_Graph_t
codegen.ml:699:            | A.Bool_t -> A.List_Bool_t
codegen.ml:700:            | _ -> A.List_Int_t
codegen.ml:701:          in
codegen.ml:702:
```

```
codegen.ml:703:          let rec check_float_typ = function
codegen.ml:704:            [] -> A.Int_t
codegen.ml:705:          | hd::ls -> if (snd(expr builder hd)) ==
A.Float_t
codegen.ml:706:                      then A.Float_t else check_float_typ
ls in
codegen.ml:707:
codegen.ml:708:          let rec check_graph_typ = function
codegen.ml:709:              [] -> A.Node_t
codegen.ml:710:            | hd::ls -> if (snd(expr builder hd)) ==
A.Graph_t
codegen.ml:711:                        then A.Graph_t else
check_graph_typ ls in
codegen.ml:712:
codegen.ml:713:          let list_typ = snd (expr builder (List.hd ls))
in
codegen.ml:714:          let list_typ = if list_typ == A.Int_t
codegen.ml:715:            then check_float_typ ls else list_typ in
codegen.ml:716:          let list_typ = if list_typ == A.Node_t
codegen.ml:717:            then check_graph_typ ls else list_typ in
codegen.ml:718:
codegen.ml:719:          let list_conversion el =
codegen.ml:720:            let (e_val, e_typ) = expr builder el in
codegen.ml:721:            (   match e_typ with
codegen.ml:722:              | A.Node_t when list_typ = A.Graph_t -> (
codegen.ml:723:                  let gh = create_graph builder in (
codegen.ml:724:                      ignore(graph_add_node gh e_val
builder);
codegen.ml:725:                      (gh, A.Graph_t)
codegen.ml:726:                  )
codegen.ml:727:                )
codegen.ml:728:              | _ -> (e_val, e_typ)
codegen.ml:729:            )
codegen.ml:730:          in
codegen.ml:731:
codegen.ml:732:          let list_ptr_t = (list_init list_typ builder,
codegen.ml:733:            from_expr_typ_to_list_typ list_typ) in
codegen.ml:734:              list_add_all (fst list_ptr_t) list_typ
builder (
codegen.ml:735:                List.map fst (List.map list_conversion
ls)), (snd list_ptr_t)
codegen.ml:736:
codegen.ml:737:      | A.DictP(expr_list) ->
codegen.ml:738:          let from_type_to_hmap_typ = function
codegen.ml:739:              A.Int_t -> A.Dict_Int_t
codegen.ml:740:            | A.String_t -> A.Dict_String_t
codegen.ml:741:            | A.Node_t -> A.Dict_Node_t
codegen.ml:742:            | A.Float_t -> A.Dict_Float_t
codegen.ml:743:            | A.Graph_t -> A.Dict_Graph_t
codegen.ml:744:            | _ -> raise (Failure "hmap type error")
codegen.ml:745:          in
codegen.ml:746:          let first_expr_kv = List.hd expr_list in
codegen.ml:747:          (* get type of key and value *)
codegen.ml:748:          let typ1 = lconst_of_typ (snd (expr builder (fst
first_expr_kv))) in
codegen.ml:749:          let typ2 = lconst_of_typ (snd (expr builder (snd
first_expr_kv))) in
codegen.ml:750:          let return_typ = from_type_to_hmap_typ (snd
(expr builder (snd first_expr_kv))) in
codegen.ml:751:          let hmap_ptr = hmap_init typ1 typ2 builder in
```

```
codegen.ml:752:          ignore(hmap_put_all hmap_ptr builder
codegen.ml:753:                (List.map (fun (key, v) -> fst(expr
builder key), fst(expr builder v)) expr_list), return_typ);
codegen.ml:754:                (hmap_ptr, return_typ)
codegen.ml:755:
codegen.ml:756:       | A.Graph_Link(left, right, edges) ->       (*  *)
codegen.ml:757:          let (ln, ln_type) = expr builder left in
codegen.ml:758:          let (rn, rn_type) = expr builder right in
codegen.ml:759:          let (el, el_type) = expr builder edges in (
codegen.ml:760:            match (ln_type, rn_type, el_type) with
codegen.ml:761:            | (A.Node_t, A.Null_t, _) -> (
codegen.ml:762:                let gh = create_graph builder in (
codegen.ml:763:                    ignore(graph_add_node gh ln builder);
codegen.ml:764:                    (gh, A.Graph_t)
codegen.ml:765:                )
codegen.ml:766:              )
codegen.ml:767:            | (A.Node_t, A.Node_t, _) -> (
codegen.ml:768:                let gh = create_graph builder in (
codegen.ml:769:                    ignore(graph_add_node gh ln builder);
(* Also set the root *)
codegen.ml:770:                    ignore(graph_add_node gh rn builder);
codegen.ml:771:                    ignore(graph_add_edge gh (ln, rn)
(el_type, el) builder);
codegen.ml:772:                    (gh, A.Graph_t)
codegen.ml:773:                )
codegen.ml:774:              )
codegen.ml:775:            | (A.Node_t, A.Graph_t, _) -> (
codegen.ml:776:                let gh = copy_graph rn builder in
codegen.ml:777:                let rt = graph_get_root rn builder in (
codegen.ml:778:                    ignore(graph_add_node gh ln builder);
codegen.ml:779:                    ignore(graph_set_root gh ln builder);
codegen.ml:780:                    ignore(graph_add_edge gh (ln, rt)
(el_type, el) builder);
codegen.ml:781:                    (gh, A.Graph_t)
codegen.ml:782:                )
codegen.ml:783:              )
codegen.ml:784:            | (A.Node_t, A.List_Graph_t, _)
codegen.ml:785:            | (A.Node_t, A.List_Node_t, _) -> (
codegen.ml:786:                let gh = create_graph builder in (
codegen.ml:787:                  ignore(graph_add_node gh ln builder); (*
Also set the root *)
codegen.ml:788:                  ignore(graph_add_list gh rn (el,
el_type) builder);
codegen.ml:789:                  (gh, A.Graph_t)
codegen.ml:790:                )
codegen.ml:791:              )
codegen.ml:792:            | _ -> raise (Failure "graph link error")
codegen.ml:793:          )
codegen.ml:794:       | A.Binop (e1, op, e2) ->
codegen.ml:795:         let (e1', t1) = expr builder e1
codegen.ml:796:         and (e2', t2) = expr builder e2 in
codegen.ml:797:         (* Handle Automatic Binop Type Converstion *)
codegen.ml:798:         (match (t1, t2) with
codegen.ml:799:           | (A.List_Int_t, A.List_Int_t)
codegen.ml:800:           | (A.List_Float_t, A.List_Float_t)
codegen.ml:801:           | (A.List_Bool_t, A.List_Bool_t)
codegen.ml:802:           | (A.List_String_t, A.List_String_t)
codegen.ml:803:           | (A.List_Node_t, A.List_Node_t)
codegen.ml:804:           | (A.List_Graph_t, A.List_Graph_t) -> (
codegen.ml:805:                match op with
```

```
codegen.ml:806:                | A.Add -> (list_concat e1' e2' builder,
t1)
codegen.ml:807:                | _ -> raise (Failure("invalid graph
operation"))
codegen.ml:808:              )
codegen.ml:809:    | (A.List_Int_t, A.Int_t)
codegen.ml:810:    | (A.List_Float_t, A.Float_t)
codegen.ml:811:    | (A.List_Bool_t, A.Bool_t)
codegen.ml:812:    | (A.List_String_t, A.String_t)
codegen.ml:813:    | (A.List_Node_t, A.Node_t)
codegen.ml:814:    | (A.List_Graph_t, A.Graph_t)-> (
codegen.ml:815:        match op with
codegen.ml:816:        | A.Add -> (list_add e2' e1' builder, t1)
codegen.ml:817:        | A.Sub -> (list_remove e2' e1' builder, t1)
codegen.ml:818:        | _ -> raise (Failure (""))
codegen.ml:819:    )
codegen.ml:820:          | ( A.Graph_t, A.Graph_t) -> (
codegen.ml:821:                match op with
codegen.ml:822:                | A.Add -> (merge_graph e1' e2' builder,
A.Graph_t)
codegen.ml:823:                | A.Sub -> (graph_sub_graph e1' e2'
builder, A.List_Graph_t)
codegen.ml:824:                | _ -> raise (Failure("invalid graph
operation"))
codegen.ml:825:              )
codegen.ml:826:           | ( A.Graph_t, A.Node_t ) -> (  (* *)
codegen.ml:827:                match  op with
codegen.ml:828:                | A.Sub -> (graph_remove_node e1' e2'
builder, A.List_Graph_t)
codegen.ml:829:                | _ -> raise (Failure("invalid
operation"))
codegen.ml:830:            )
codegen.ml:831:          | ( _, A.Null_t ) -> (
codegen.ml:832:                  match op with
codegen.ml:833:                | A.Equal -> (L.build_is_null e1' "isNull"
builder, A.Bool_t)
codegen.ml:834:                | A.Neq -> (L.build_is_not_null e1'
"isNull" builder, A.Bool_t)
codegen.ml:835:                | _ -> raise (Failure("invalid
operation"))
codegen.ml:836:            )
codegen.ml:837:          | ( A.Null_t, _ ) -> (
codegen.ml:838:                  match op with
codegen.ml:839:                | A.Equal -> (L.build_is_null e2'
"isNotNull" builder, A.Bool_t)
codegen.ml:840:                | A.Neq -> (L.build_is_not_null e2'
"isNotNull" builder, A.Bool_t)
codegen.ml:841:                | _ -> raise (Failure("invalid
operation"))
codegen.ml:842:            )
codegen.ml:843:          | ( t1, t2) when t1 = t2 -> handle_binop e1' op
e2' t1 builder
codegen.ml:844:          | ( A.Int_t, A.Float_t) ->
codegen.ml:845:              handle_binop (int_to_float builder e1') op
e2' A.Float_t builder
codegen.ml:846:          | ( A.Float_t, A.Int_t ) ->
codegen.ml:847:              handle_binop e1' op (int_to_float builder
e2') A.Float_t builder
codegen.ml:848:          | _ -> raise (Failure ("invalid operation"))
codegen.ml:849:        )
```

```
codegen.ml:850:      | A.Unop(op, e) ->
codegen.ml:851:         let (e', typ) = expr builder e in
codegen.ml:852:         ((match op with
codegen.ml:853:           A.Neg     -> if typ = A.Int_t then L.build_neg
else L.build_fneg
codegen.ml:854:         | A.Not     -> L.build_not) e' "tmp" builder,
typ)
codegen.ml:855:      | A.Assign (s, e) ->
codegen.ml:856:         let (e', etyp) = expr builder e in
codegen.ml:857:         let (var, typ) = lookup s in
codegen.ml:858:         (( match (etyp, typ) with
codegen.ml:859:           | (t1, t2) when t1 = t2 -> ignore
(L.build_store e' var builder); e'
codegen.ml:860:           | (A.List_Null_t, _) -> ignore (L.build_store
e' var builder); e'
codegen.ml:861:           | (A.Null_t, _) -> ignore (L.build_store
(null_of_typ typ) var builder); (null_of_typ typ)
codegen.ml:862:           | (A.Int_t, A.Float_t) -> let e' =
(int_to_float builder e') in ignore (L.build_store e' var builder); e'
codegen.ml:863:           | (A.List_Node_t, _) -> ignore (L.build_store
e' var builder); e'
codegen.ml:864:           | _ -> raise (Failure("assign type error"))
codegen.ml:865:         ), typ)
codegen.ml:866:      | A.Call ("print", el) ->
codegen.ml:867:         let print_expr e =
codegen.ml:868:           let (eval, etyp) = expr builder e in (
codegen.ml:869:             match etyp with
codegen.ml:870:             | A.Int_t -> ignore(print_gen builder
[(string_lit_gen "%d\n" builder); eval])
codegen.ml:871:             | A.Null_t -> ignore(print_gen builder
[(string_lit_gen "null\n" builder)])
codegen.ml:872:             | A.Bool_t -> ignore(print_bool eval
builder)
codegen.ml:873:             | A.Float_t -> ignore(print_gen builder
[(string_lit_gen "%f\n" builder); eval])
codegen.ml:874:             | A.String_t -> ignore(print_gen builder
[(string_lit_gen "%s\n" builder); eval])
codegen.ml:875:             | A.Node_t -> ignore(print_node eval
builder)
codegen.ml:876:             | A.Edge_t -> ignore(print_edge eval
builder)
codegen.ml:877:             | A.Dict_Int_t | A.Dict_Float_t |
A.Dict_String_t | A.Dict_Node_t
codegen.ml:878:             | A.Dict_Graph_t -> ignore(print_hmap eval
builder)
codegen.ml:879:             | A.List_Int_t -> ignore(print_list eval
builder)
codegen.ml:880:             | A.List_Float_t -> ignore(print_list eval
builder)
codegen.ml:881:             | A.List_Bool_t -> ignore(print_list eval
builder)
codegen.ml:882:             | A.List_String_t -> ignore(print_list eval
builder)
codegen.ml:883:             | A.List_Node_t -> ignore(print_list eval
builder)
codegen.ml:884:             | A.List_Graph_t -> ignore(print_list eval
builder)
codegen.ml:885:             | A.Graph_t -> ignore(print_graph eval
builder)
codegen.ml:886:             | _ -> raise (Failure("invalid print type"))
```

```
codegen.ml:887:          ) in List.iter print_expr el; (L.const_int i32_t
0, A.Void_t)
codegen.ml:888:      | A.Call ("printf", el) ->
codegen.ml:889:          (print_gen builder (List.map
codegen.ml:890:            (fun e -> (let (eval, _) = expr builder e in
eval))
codegen.ml:891:             el), A.Void_t)
codegen.ml:892:      | A.Call (f, act) ->
codegen.ml:893:         let (fdef, fdecl) = StringMap.find f
function_decls in
codegen.ml:894:         let actuals = List.rev (List.map
codegen.ml:895:           (fun e -> (let (eval, _) = expr builder e in
eval)) (List.rev act)) in
codegen.ml:896:         let result = (match fdecl.A.returnType with
A.Void_t -> ""
codegen.ml:897:                                                    | _ ->
f ^ "_result") in
codegen.ml:898:          (L.build_call fdef (Array.of_list actuals) result
builder, fdecl.A.returnType)
codegen.ml:899:
codegen.ml:900:      (* default get operator of hmap *)
codegen.ml:901:      | A.CallDefault(val_name, default_func_name,
params_list) ->
codegen.ml:902:        (* get caller tpye *)
codegen.ml:903:        let (id_val, expr_tpy) = (expr builder val_name)
in
codegen.ml:904:        let assign_func_by_typ builder = function
codegen.ml:905:          (* deal with list *)
codegen.ml:906:          | A.List_Int_t | A.List_Float_t |
A.List_String_t | A.List_Bool_t
codegen.ml:907:          | A.List_Node_t | A.List_Graph_t ->
codegen.ml:908:              list_func builder id_val (List.map (fun e ->
fst (expr builder e)) params_list) expr_tpy default_func_name
codegen.ml:909:          | A.Dict_Int_t | A.Dict_Float_t |
A.Dict_String_t | A.Dict_Node_t | A.Dict_Graph_t ->
codegen.ml:910:              hmap_func builder id_val (List.map (fun e ->
fst (expr builder e)) params_list) expr_tpy default_func_name
codegen.ml:911:          | A.Graph_t ->
codegen.ml:912:              graph_call_default_main builder id_val
(List.map (fun e -> (expr builder e)) params_list) default_func_name
codegen.ml:913:    | A.Node_t ->
codegen.ml:914:              node_func builder id_val default_func_name
codegen.ml:915:          | _ -> raise (Failure ("invalid function"))
codegen.ml:916:          in
codegen.ml:917:            assign_func_by_typ builder expr_tpy
codegen.ml:918:
codegen.ml:919:(*       | _ -> (L.const_int i32_t 0, A.Void_t)
codegen.ml:920: *)    in
codegen.ml:921:
codegen.ml:922:    (* Invoke "f builder" if the current block doesn't
already
codegen.ml:923:       have a terminal (e.g., a branch). *)
codegen.ml:924:    let add_terminal builder f =
codegen.ml:925:      match L.block_terminator (L.insertion_block builder)
with
codegen.ml:926:        Some _ -> ()
codegen.ml:927:      | None -> ignore (f builder) in
codegen.ml:928:
codegen.ml:929:    (* Build the code for the given statement; return the
builder for
```

```
codegen.ml:930:       the statement's successor *)
codegen.ml:931:    let rec stmt builder = function
codegen.ml:932:      | A.Expr e -> ignore (expr builder e); builder
codegen.ml:933:      | A.Return e ->
codegen.ml:934:          ignore (
codegen.ml:935:            let (ev, et) = expr builder e in
codegen.ml:936:            match (fdecl.A.returnType, et) with
codegen.ml:937:                  (A.Void_t, _) -> L.build_ret_void
builder
codegen.ml:938:                | (t1, t2) when t1 = t2 -> L.build_ret ev
builder
codegen.ml:939:            | (A.Float_t, A.Int_t) -> L.build_ret
(int_to_float builder ev) builder
codegen.ml:940:            | (t1, A.Null_t) -> L.build_ret (init_val t1)
builder
codegen.ml:941:            | _ -> raise (Failure("return type error"))
codegen.ml:942:          ); builder
codegen.ml:943:      | A.If (predicate, then_stmt, else_stmt) ->
codegen.ml:944:         let (bool_val, _) = expr builder predicate in
codegen.ml:945:        let merge_bb = L.append_block context "merge"
the_function in
codegen.ml:946:
codegen.ml:947:        let then_bb = L.append_block context "then"
the_function in
codegen.ml:948:        add_terminal (
codegen.ml:949:            List.fold_left stmt (L.builder_at_end context
then_bb) then_stmt
codegen.ml:950:          ) (L.build_br merge_bb);
codegen.ml:951:
codegen.ml:952:        let else_bb = L.append_block context "else"
the_function in
codegen.ml:953:         add_terminal (
codegen.ml:954:            List.fold_left stmt (L.builder_at_end context
else_bb) else_stmt
codegen.ml:955:          ) (L.build_br merge_bb);
codegen.ml:956:
codegen.ml:957:        ignore (L.build_cond_br bool_val then_bb else_bb
builder);
codegen.ml:958:        L.builder_at_end context merge_bb
codegen.ml:959:
codegen.ml:960:      | A.While (predicate, body) ->
codegen.ml:961:          let pred_bb = L.append_block context "while"
the_function in
codegen.ml:962:          ignore (L.build_br pred_bb builder);
codegen.ml:963:
codegen.ml:964:          let body_bb = L.append_block context
"while_body" the_function in
codegen.ml:965:          add_terminal (
codegen.ml:966:              List.fold_left stmt (L.builder_at_end
context body_bb) body
codegen.ml:967:            ) (L.build_br pred_bb);
codegen.ml:968:
codegen.ml:969:          let pred_builder = L.builder_at_end context
pred_bb in
codegen.ml:970:          let (bool_val, _) = expr pred_builder predicate
in
codegen.ml:971:
codegen.ml:972:          let merge_bb = L.append_block context "merge"
the_function in
```

```
codegen.ml:973:          ignore (L.build_cond_br bool_val body_bb
merge_bb pred_builder);
codegen.ml:974:          L.builder_at_end context merge_bb
codegen.ml:975:
codegen.ml:976:      | A.For (e1, e2, e3, body) -> List.fold_left stmt
builder
codegen.ml:977:         ( [A.Expr e1 ; A.While (e2, body @ [A.Expr e3]) ]
)
codegen.ml:978:    in
codegen.ml:979:
codegen.ml:980:    (* Build the code for each statement in the function
*)
codegen.ml:981:    let builder = List.fold_left stmt builder fdecl.A.body
in
codegen.ml:982:
codegen.ml:983:    (* Add a return if the last block falls off the end *)
codegen.ml:984:    add_terminal builder (match fdecl.A.returnType with
codegen.ml:985:        A.Void_t -> L.build_ret_void
codegen.ml:986:      | A.Int_t as t -> L.build_ret (L.const_int
(ltype_of_typ t) 0)
codegen.ml:987:      | A.Bool_t as t -> L.build_ret (L.const_int
(ltype_of_typ t) 0)
codegen.ml:988:      | A.Float_t as t-> L.build_ret (L.const_float
(ltype_of_typ t) 0.)
codegen.ml:989:      | t-> L.build_ret (L.const_null (ltype_of_typ t))
codegen.ml:990:    )
codegen.ml:991:  in
codegen.ml:992:
codegen.ml:993:  List.iter build_function_body (List.rev program);
codegen.ml:994:
codegen.ml:995:  the_module
codegen.ml:996:giraphe.ml:1:(* Top-level of the MicroC compiler: scan &
parse the input,
giraphe.ml:2:   check the resulting AST, generate LLVM IR, and dump the
module *)
giraphe.ml:3:
giraphe.ml:4:type action = LLVM_IR | Compile (* | AST *)
giraphe.ml:5:
giraphe.ml:6:let _ =
giraphe.ml:7:  let action = if Array.length Sys.argv > 1 then
giraphe.ml:8:    List.assoc Sys.argv.(1) [
giraphe.ml:9:                      ("-l", LLVM_IR);  (* Generate LLVM,
don't check *)
giraphe.ml:10:                     ("-c", Compile) ] (* Generate, check
LLVM IR *)
giraphe.ml:11:  else Compile in
giraphe.ml:12:  let lexbuf = Lexing.from_channel stdin in
giraphe.ml:13:  let ast = Parser.program Scanner.token lexbuf in
giraphe.ml:14:  let sast = Checker.convert ast in
giraphe.ml:15:  Semant.check sast;
giraphe.ml:16:  match action with
giraphe.ml:17:  | LLVM_IR -> print_string (Llvm.string_of_llmodule
(Codegen.translate sast))
giraphe.ml:18:  | Compile -> let m = Codegen.translate sast in
giraphe.ml:19:    Llvm_analysis.assert_valid_module m;
giraphe.ml:20:    print_string (Llvm.string_of_llmodule m)
giraphe.ml:21:giraphe.sh:1:if [ $# -eq 1 ]
giraphe.sh:2:then
giraphe.sh:3:    ./giraphe.native < $1 > a.ll
giraphe.sh:4:else
```

```
giraphe.sh:5:    ./giraphe.native $1 < $2 > a.ll
giraphe.sh:6:fi
giraphe.sh:7:
giraphe.sh:8:clang -Wno-override-module lib.bc a.ll -o $1.exe
giraphe.sh:9:./$1.exe
giraphe.sh:10:rm a.ll
giraphe.sh:11:rm ./$1.exe
giraphe.sh:12:
giraphe.sh:13:# /usr/local/opt/llvm38/bin/clang-3.8
giraphe.sh:14:lib.c:1:#include <stdio.h>
lib.c:2:#include <stdlib.h>
lib.c:3:#include <limits.h>
lib.c:4:#include <stdarg.h>
lib.c:5:#include "lib.h"
lib.c:6:#include <stdint.h>
lib.c:7:#include <string.h>
lib.c:8:#include <stdbool.h>
lib.c:9:
lib.c:10:///////casting stuff
lib.c:11:
lib.c:12:int32_t VoidtoInt(void* add){
lib.c:13:  return *((int32_t*) add);
lib.c:14:}
lib.c:15:
lib.c:16:double VoidtoFloat(void* add){
lib.c:17:  return *((double*) add);
lib.c:18:}
lib.c:19:
lib.c:20:bool VoidtoBool(void* add){
lib.c:21:  return *((bool*) add);
lib.c:22:}
lib.c:23:
lib.c:24:char* VoidtoString(void* add){
lib.c:25:  return (char*) add;
lib.c:26:}
lib.c:27:
lib.c:28:struct Node* VoidtoNode(void* add){
lib.c:29:  return (struct Node*) add;
lib.c:30:}
lib.c:31:
lib.c:32:struct Graph* VoidtoGraph(void* add){
lib.c:33:  return (struct Graph*) add;
lib.c:34:}
lib.c:35:
lib.c:36:struct Edge* VoidtoEdge (void* add){
lib.c:37:  return (struct Edge*) add;
lib.c:38:}
lib.c:39:
lib.c:40:void* InttoVoid(int32_t val){
lib.c:41:  int* tmp = (int*)malloc(sizeof(int));
lib.c:42:  *tmp = val;
lib.c:43:  return (void*) tmp;
lib.c:44:}
lib.c:45:
lib.c:46:void* FloattoVoid(double val){
lib.c:47:  double* tmp = (double*)malloc(sizeof(double));
lib.c:48:  *tmp = val;
lib.c:49:  return (void*) tmp;
lib.c:50:}
lib.c:51:
```

```
lib.c:52:void* BooltoVoid(bool val){
lib.c:53:  bool* tmp = (bool*)malloc(sizeof(bool));
lib.c:54:  *tmp = val;
lib.c:55:  return (void*) tmp;
lib.c:56:}
lib.c:57:
lib.c:58:void* StringtoVoid(char* val){
lib.c:59:  return (void*) val;
lib.c:60:}
lib.c:61:
lib.c:62:void* NodetoVoid(struct Node* val){
lib.c:63:  return (void*) val;
lib.c:64:}
lib.c:65:
lib.c:66:void* GraphtoVoid(struct Graph* val){
lib.c:67:  return (void*) val;
lib.c:68:}
lib.c:69:
lib.c:70:void* EdgetoVoid(struct Edge* val){
lib.c:71:  return (void*) val;
lib.c:72:
lib.c:73:}
lib.c:74:
lib.c:75:bool isInt(int32_t d){
lib.c:76:  return (d==INT);
lib.c:77:};
lib.c:78:
lib.c:79:bool isFloat(int32_t d){return (d==INT);};
lib.c:80:bool isBool(int32_t d){return (d==BOOL);};
lib.c:81:bool isString(int32_t d){return (d==STRING);};
lib.c:82:bool isNode(int32_t d){return (d==NODE);};
lib.c:83:bool isEdge(int32_t d) {return (d ==EDGE);};
lib.c:84:bool isGraph(int32_t d){return (d==GRAPH);};
lib.c:85:
lib.c:86:
lib.c:87:
lib.c:88:
lib.c:89:
lib.c:90:void printBool(bool a) {
lib.c:91:  if(a){
lib.c:92:        printf("%s\n", "true") ;
lib.c:93:  }else{
lib.c:94:        printf("%s\n", "false");
lib.c:95:  }
lib.c:96:  return;
lib.c:97:}
lib.c:98:/**********************************
lib.c:99:  List Methods
lib.c:100:**********************************/
lib.c:101:
lib.c:102:bool boolCompare(bool target, bool cur) {
lib.c:103:       return target == cur;
lib.c:104: }
lib.c:105:
lib.c:106: bool stringCompare(char* target, char* cur) {
lib.c:107:       return strcmp(target, cur) == 0;
lib.c:108: }
lib.c:109:
lib.c:110: bool intCompare(int target, int cur) {
lib.c:111:       return target == cur;
```

```
lib.c:112: }
lib.c:113:
lib.c:114: bool floatCompare(double target, double cur) {
lib.c:115:       return target == cur;
lib.c:116: }
lib.c:117:
lib.c:118: bool nodeCompare(struct Node* target, struct Node* cur) {
lib.c:119:       return target->id == cur->id;
lib.c:120: }
lib.c:121:
lib.c:122: bool edgeCompare(struct Edge* target, struct Edge* cur){
lib.c:123:       return (target->sour->id == cur->sour->id) && (target-
>dest->id == cur->dest->id);
lib.c:124: }
lib.c:125:
lib.c:126:struct List* createList(int32_t list_type) {
lib.c:127:
lib.c:128: struct List* list = (struct List*) malloc(sizeof(struct
List));
lib.c:129: list->list_type = list_type;
lib.c:130: list->index = 0;
lib.c:131: list->size = 1;
lib.c:132: list->stored_data = (void**)malloc(list->size *
sizeof(void*));
lib.c:133:
lib.c:134: return list;
lib.c:135:}
lib.c:136:
lib.c:137:int rangeHelper(int size_of_list, int index){
lib.c:138: if(size_of_list <= 0 || index >= size_of_list || index < 0){
lib.c:139:
lib.c:140:       printf("Cannot access out of range!\n");
lib.c:141:
lib.c:142:       exit(1);
lib.c:143: }else {
lib.c:144:            if (index < 0){
lib.c:145:            index += size_of_list;
lib.c:146:       }
lib.c:147: }
lib.c:148: return index;
lib.c:149:}
lib.c:150:
lib.c:151:struct List* addListHelper(struct List* list, void* data ){
lib.c:152:
lib.c:153: if (list->index >= list->size){
lib.c:154:
lib.c:155:       list->size = list->size * 2;
lib.c:156:
lib.c:157:       list->stored_data = (void**) realloc(list->stored_data,
list->size * sizeof(void*));
lib.c:158: }
lib.c:159: *(list->stored_data + list->index) = data;
lib.c:160:
lib.c:161: list->index =  list->index+1;
lib.c:162:
lib.c:163: return list;
lib.c:164:}
lib.c:165:
lib.c:166:struct List* concatList(struct List* left_list, struct List*
right_list){
```

```
lib.c:167:
lib.c:168: int index = right_list->index;
lib.c:169:
lib.c:170: int i = 0;
lib.c:171:
lib.c:172: while(i<index){
lib.c:173:
lib.c:174:       left_list = addListHelper(left_list, *(right_list-
>stored_data+i++));
lib.c:175:
lib.c:176: }
lib.c:177: return left_list;
lib.c:178:}
lib.c:179:
lib.c:180:
lib.c:181:struct List* addList(struct List* list, ... ) {
lib.c:182:
lib.c:183: if (list == NULL) {
lib.c:184:       printf("List has not been initialized. \n");
lib.c:185:       exit(1);
lib.c:186:       return NULL;
lib.c:187: }else{
lib.c:188:
lib.c:189: va_list data_list;
lib.c:190: va_start(data_list, 1);
lib.c:191: void * data;
lib.c:192:
lib.c:193: switch (list->list_type) {
lib.c:194:
lib.c:195:       case INT:
lib.c:196:            data = InttoVoid(va_arg(data_list, int));
lib.c:197:            break;
lib.c:198:
lib.c:199:       case FLOAT:
lib.c:200:            data = FloattoVoid(va_arg(data_list, double));
lib.c:201:            break;
lib.c:202:
lib.c:203:       case BOOL:
lib.c:204:            data = BooltoVoid(va_arg(data_list, bool));
lib.c:205:            break;
lib.c:206:
lib.c:207:       case STRING:
lib.c:208:            data = StringtoVoid(va_arg(data_list, char*));
lib.c:209:            break;
lib.c:210:
lib.c:211:       case NODE:
lib.c:212:            data = NodetoVoid(va_arg(data_list, struct
Node*));
lib.c:213:            break;
lib.c:214:
lib.c:215:       case EDGE:
lib.c:216:            data =EdgetoVoid(va_arg(data_list, struct Edge*));
lib.c:217:            break;
lib.c:218:
lib.c:219:       case GRAPH:
lib.c:220:            data = GraphtoVoid(va_arg(data_list, struct
Graph*));
lib.c:221:            break;
lib.c:222:
lib.c:223:       default:
```

```
lib.c:224:            break;
lib.c:225: }
lib.c:226:  va_end(data_list);
lib.c:227:  return addListHelper(list, data);
lib.c:228: }
lib.c:229:
lib.c:230:}
lib.c:231:
lib.c:232:void* getList(struct List* list, int position){
lib.c:233:
lib.c:234: if (list != NULL) {
lib.c:235:
lib.c:236:       position = rangeHelper(list->index, position);
lib.c:237:       void* ret =  *(list->stored_data + position);
lib.c:238:       return ret;
lib.c:239:
lib.c:240: }else{
lib.c:241:
lib.c:242:       printf("Must initialize list \n");
lib.c:243:       exit(1);
lib.c:244:
lib.c:245: }
lib.c:246:
lib.c:247:}
lib.c:248:
lib.c:249:void* popList(struct List* list){
lib.c:250: if (list != NULL && list->size >= 1) {
lib.c:251:
lib.c:252:       void* add = *(list->stored_data + list->index-1);
lib.c:253:       list->index--;
lib.c:254:
lib.c:255:       return add;
lib.c:256: }else{
lib.c:257:       printf("There are no elements or NULL list\n");
lib.c:258: }
lib.c:259:
lib.c:260:}
lib.c:261:
lib.c:262:int32_t setList(struct List* list, int index, ...){
lib.c:263: if (list != NULL) {
lib.c:264:
lib.c:265:       index = rangeHelper(list->index, index);
lib.c:266:       va_list data_list;
lib.c:267:       va_start(data_list, 1);
lib.c:268:       void * data;
lib.c:269:
lib.c:270:       switch (list->list_type) {
lib.c:271:            case INT:
lib.c:272:                  data = InttoVoid(va_arg(data_list, int));
lib.c:273:                  break;
lib.c:274:
lib.c:275:            case FLOAT:
lib.c:276:                  data = FloattoVoid(va_arg(data_list,
double));
lib.c:277:                  break;
lib.c:278:
lib.c:279:            case BOOL:
lib.c:280:                  data = BooltoVoid(va_arg(data_list, bool));
lib.c:281:                  break;
lib.c:282:
```

```
lib.c:283:            case STRING:
lib.c:284:                  data = StringtoVoid(va_arg(data_list,
char*));
lib.c:285:                  break;
lib.c:286:
lib.c:287:            case NODE:
lib.c:288:                  data = NodetoVoid(va_arg(data_list, struct
Node*));
lib.c:289:                  break;
lib.c:290:            case EDGE:
lib.c:291:                  data = EdgetoVoid(va_arg(data_list, struct
Edge*));
lib.c:292:                  break;
lib.c:293:
lib.c:294:            case GRAPH:
lib.c:295:                  data = GraphtoVoid(va_arg(data_list, struct
Graph*));
lib.c:296:                  break;
lib.c:297:
lib.c:298:            default:
lib.c:299:                  break;
lib.c:300:       }
lib.c:301:       *(list->stored_data + index) = data;
lib.c:302:       }
lib.c:303: else{
lib.c:304:
lib.c:305:       printf("Null list. \n");
lib.c:306:
lib.c:307:       exit(1);
lib.c:308: }
lib.c:309:
lib.c:310: return 0;
lib.c:311:}
lib.c:312:
lib.c:313:int getListSize(struct List* list){
lib.c:314: if (list == NULL) {
lib.c:315:
lib.c:316:       printf("List is NULL\n");
lib.c:317:
lib.c:318:       exit(1);
lib.c:319: }
lib.c:320: int size = list->index;
lib.c:321: return size;
lib.c:322:}
lib.c:323:
lib.c:324:int32_t removeList(struct List* list, int index){
lib.c:325:
lib.c:326: if (list != NULL) {
lib.c:327:
lib.c:328: index = rangeHelper(list->index, index);
lib.c:329:
lib.c:330: for(int i=index; i < list->index; i++){
lib.c:331:
lib.c:332:       *(list->stored_data + i) = *(list->stored_data + i+1);
lib.c:333:
lib.c:334: }
lib.c:335:
lib.c:336: list->index--;
lib.c:337:
lib.c:338: }else{
```

```
lib.c:339:       printf("List is NULL. \n");
lib.c:340:
lib.c:341:       exit(1);
lib.c:342:
lib.c:343: }
lib.c:344:
lib.c:345: return 0;
lib.c:346:}
lib.c:347:
lib.c:348:
lib.c:349:struct List* removeData(struct List* list, ...) {
lib.c:350: if (list == NULL) {
lib.c:351:       return list;
lib.c:352: } else if (list->index == 0) {
lib.c:353:       return list;
lib.c:354: } else {
lib.c:355:       int p = 0;
lib.c:356:       int index = list->index;
lib.c:357:       va_list data_list;
lib.c:358:       va_start(data_list, 1);
lib.c:359:       switch (list->list_type) {
lib.c:360:    case INT:
lib.c:361:      ;
lib.c:362:      int tmp = va_arg(data_list, int);
lib.c:363:      while (p < index) {
lib.c:364:        if (intCompare(tmp, *((int *)(*(list->stored_data +
p))))) {
lib.c:365:
lib.c:366:            removeList(list, p);
lib.c:367:            p--;
lib.c:368:            index--;
lib.c:369:       }
lib.c:370:
lib.c:371:        p++;
lib.c:372:      }
lib.c:373:      break;
lib.c:374:
lib.c:375:    case FLOAT:
lib.c:376:      ;
lib.c:377:      double tmpF = va_arg(data_list, double);
lib.c:378:      while (p < index) {
lib.c:379:        if (floatCompare(tmpF, *((double *)(*(list->stored_data
+ p))))) {
lib.c:380:            removeList(list, p);
lib.c:381:            p--;
lib.c:382:            index--;
lib.c:383:       }
lib.c:384:        p++;
lib.c:385:      }
lib.c:386:      break;
lib.c:387:
lib.c:388:    case BOOL:
lib.c:389:      ;
lib.c:390:      bool tmpB = va_arg(data_list, bool);
lib.c:391:      while (p < index) {
lib.c:392:        if (boolCompare(tmpB, *((bool *)(*(list->stored_data +
p))))) {
lib.c:393:            removeList(list, p);
lib.c:394:            p--;
lib.c:395:            index--;
```

```
lib.c:396:       }
lib.c:397:        p++;
lib.c:398:      }
lib.c:399:      break;
lib.c:400:
lib.c:401:    case STRING:
lib.c:402: ;
lib.c:403:      char * tmpC = va_arg(data_list, char*);
lib.c:404:      while (p < index) {
lib.c:405:        if (stringCompare(tmpC, ((char *)(*(list->stored_data +
p))))) {
lib.c:406:            removeList(list, p);
lib.c:407:            p--;
lib.c:408:            index--;
lib.c:409:       }
lib.c:410:        p++;
lib.c:411:      }
lib.c:412:      break;
lib.c:413:
lib.c:414:    case NODE:
lib.c:415: ;
lib.c:416:      struct Node * tmpN = va_arg(data_list, struct Node *);
lib.c:417:      while (p < index) {
lib.c:418:        if (nodeCompare(tmpN, ((struct Node *)(*(list-
>stored_data + p))))) {
lib.c:419:            removeList(list, p);
lib.c:420:            p--;
lib.c:421:            index--;
lib.c:422:       }
lib.c:423:        p++;
lib.c:424:      }
lib.c:425:      break;
lib.c:426:
lib.c:427:
lib.c:428:     case EDGE:
lib.c:429: ;
lib.c:430:      struct Edge * tmpE = va_arg(data_list, struct Edge *);
lib.c:431:      while (p < index) {
lib.c:432:        if (edgeCompare(tmpE, ((struct Edge *)(*(list-
>stored_data + p))))) {
lib.c:433:            removeList(list, p);
lib.c:434:            p--;
lib.c:435:            index--;
lib.c:436:       }
lib.c:437:        p++;
lib.c:438:      }
lib.c:439:      break;
lib.c:440:
lib.c:441:    // case GRAPH:
lib.c:442:    //   while (p < index) {
lib.c:443:    //     if (graphCompare(va_arg(data_list, struct Graph *),
((struct Graph *)(*(list->stored_data + p))))) {
lib.c:444: //         result = true;
lib.c:445: //         break;
lib.c:446: //    }
lib.c:447:    //     p++;
lib.c:448:    //   }
lib.c:449:    //   break;
lib.c:450:
lib.c:451:    default:
```

```
lib.c:452:      printf("Unsupported List Type!\n");
lib.c:453:   exit(1);
lib.c:454:    }
lib.c:455: va_end(data_list);
lib.c:456: return list;
lib.c:457: }
lib.c:458:}
lib.c:459:
lib.c:460:bool listContains(struct List *list, ...) {
lib.c:461:  if (list == NULL) {
lib.c:462:    return false;
lib.c:463:  } else if (list->index == 0) {
lib.c:464:    return false;
lib.c:465:  } else {
lib.c:466:    int p = 0;
lib.c:467: void *data;
lib.c:468:    int index = list->index;
lib.c:469:    va_list data_list;
lib.c:470:    va_start(data_list, 1);
lib.c:471:    bool result = false;
lib.c:472:    switch (list->list_type) {
lib.c:473:    case INT:
lib.c:474:      ;
lib.c:475:      int tmp = va_arg(data_list, int);
lib.c:476:      while (p < index) {
lib.c:477:        if (intCompare(tmp, *((int *)(*(list->stored_data +
p))))) {
lib.c:478:             result = true;
lib.c:479:             break;
lib.c:480:       }
lib.c:481:        p++;
lib.c:482:      }
lib.c:483:      break;
lib.c:484:
lib.c:485:    case FLOAT:
lib.c:486:      ;
lib.c:487:      double tmpF = va_arg(data_list, double);
lib.c:488:      while (p < index) {
lib.c:489:        if (floatCompare(tmpF, *((double *)(*(list->stored_data
+ p))))) {
lib.c:490:             result = true;
lib.c:491:             break;
lib.c:492:       }
lib.c:493:        p++;
lib.c:494:      }
lib.c:495:      break;
lib.c:496:
lib.c:497:    case BOOL:
lib.c:498:      ;
lib.c:499:      bool tmpB = va_arg(data_list, bool);
lib.c:500:      while (p < index) {
lib.c:501:        if (boolCompare(tmpB, *((bool *)(*(list->stored_data +
p))))) {
lib.c:502:             result = true;
lib.c:503:             break;
lib.c:504:       }
lib.c:505:        p++;
lib.c:506:      }
lib.c:507:      break;
lib.c:508:
```

```
lib.c:509:    case STRING:
lib.c:510: ;
lib.c:511:      char * tmpC = va_arg(data_list, char*);
lib.c:512:      while (p < index) {
lib.c:513:        if (stringCompare(tmpC, ((char *)(*(list->stored_data +
p))))) {
lib.c:514:             result = 1;
lib.c:515:             break;
lib.c:516:       }
lib.c:517:        p++;
lib.c:518:      }
lib.c:519:      break;
lib.c:520:
lib.c:521:    case NODE:
lib.c:522: ;
lib.c:523:      struct Node * tmpN = va_arg(data_list, struct Node *);
lib.c:524:      while (p < index) {
lib.c:525:        if (nodeCompare(tmpN, ((struct Node *)(*(list-
>stored_data + p))))) {
lib.c:526:             result = 1;
lib.c:527:             break;
lib.c:528:       }
lib.c:529:        p++;
lib.c:530:      }
lib.c:531:      break;
lib.c:532:
lib.c:533:      case EDGE:
lib.c:534: ;
lib.c:535:      struct Edge * tmpE = va_arg(data_list, struct Edge *);
lib.c:536:      while (p < index) {
lib.c:537:        if (edgeCompare(tmpE, ((struct Edge *)(*(list-
>stored_data + p))))) {
lib.c:538:             result = 1;
lib.c:539:             break;
lib.c:540:       }
lib.c:541:        p++;
lib.c:542:      }
lib.c:543:      break;
lib.c:544:
lib.c:545:
lib.c:546:
lib.c:547:    default:
lib.c:548:      printf("Unsupported List Type!\n");
lib.c:549:   exit(1);
lib.c:550:    }
lib.c:551:    va_end(data_list);
lib.c:552: return result;
lib.c:553:  }
lib.c:554:}
lib.c:555:
lib.c:556:int32_t printList(struct List * list){
lib.c:557: if (list == NULL) {
lib.c:558:       printf("(null)\n");
lib.c:559:       return 0;
lib.c:560: }
lib.c:561: int index = list->index - 1;
lib.c:562: if (index < 0) {
lib.c:563:       printf("[]\n");
lib.c:564:       return 0;
lib.c:565: }
```

```
lib.c:566: int p = 0;
lib.c:567: printf("[");
lib.c:568: switch (list->list_type) {
lib.c:569:       case INT:
lib.c:570:            while(p < index){
lib.c:571:                 printf("%d, ", *((int*)(*(list->stored_data +
p))));
lib.c:572:                 p++;
lib.c:573:            }
lib.c:574:            printf("%d", *((int*)(*(list->stored_data + p))));
lib.c:575:            break;
lib.c:576:
lib.c:577:       case FLOAT:
lib.c:578:            while(p < index){
lib.c:579:                 printf("%f, ", *((double*)(*(list-
>stored_data + p))));
lib.c:580:                 p++;
lib.c:581:            }
lib.c:582:            printf("%f", *((double*)(*(list->stored_data +
p))));
lib.c:583:            break;
lib.c:584:
lib.c:585:       case BOOL:
lib.c:586:            while(p < index){
lib.c:587:                 printf("%s, ", *((bool*)(*(list->stored_data
+ p))) ? "true" : "false");
lib.c:588:                 p++;
lib.c:589:            }
lib.c:590:            printf("%s", *((bool*)(*(list->stored_data + p)))
? "true" : "false");
lib.c:591:            break;
lib.c:592:
lib.c:593:       case STRING:
lib.c:594:            while(p < index){
lib.c:595:                 printf("%s, ", ((char*)(*(list->stored_data +
p))));
lib.c:596:                 p++;
lib.c:597:            }
lib.c:598:            printf("%s", ((char*)(*(list->stored_data + p))));
lib.c:599:            break;
lib.c:600:
lib.c:601:       case NODE:
lib.c:602:            while(p < index){
lib.c:603:                 printNode((struct Node*)(*(list->stored_data
+ p)));
lib.c:604:                 p++;
lib.c:605:            }
lib.c:606:            printNode((struct Node*)(*(list->stored_data +
p)));
lib.c:607:            break;
lib.c:608:
lib.c:609:       case GRAPH:
lib.c:610:            while(p < index){
lib.c:611:                 printGraph((struct Graph*)(*(list-
>stored_data + p)));
lib.c:612:                 p++;
lib.c:613:            }
lib.c:614:            printGraph((struct Graph*)(*(list->stored_data +
p)));
lib.c:615:            break;
```

```
lib.c:616:
lib.c:617:       case EDGE:
lib.c:618:            while(p < index){
lib.c:619:                 printEdge((struct Edge*)(*(list->stored_data
+ p)));
lib.c:620:                 p++;
lib.c:621:            }
lib.c:622:            printEdge((struct Edge*)(*(list->stored_data +
p)));
lib.c:623:            break;
lib.c:624:
lib.c:625:
lib.c:626:       default:
lib.c:627:            printf("Unsupported List Type!\n");
lib.c:628:            return 1;
lib.c:629: }
lib.c:630: printf("]\n");
lib.c:631: return 0;
lib.c:632:
lib.c:633:}
lib.c:634:/***********************************
lib.c:635: Hashmap
lib.c:636:***********************************/
lib.c:637:
lib.c:638:struct hashmap_map* hashmap_new(int32_t keytyp,int32_t
valuetyp) {
lib.c:639: struct hashmap_map* m = (struct hashmap_map*)
malloc(sizeof(struct hashmap_map));
lib.c:640: m->data = (struct hashmap_element*) calloc(INITIAL_SIZE,
sizeof(struct hashmap_element));
lib.c:641: m->keytype = keytyp;
lib.c:642: m->valuetype = valuetyp;
lib.c:643: m->table_size = INITIAL_SIZE;
lib.c:644: m->size = 0;
lib.c:645: return m;
lib.c:646:}
lib.c:647:
lib.c:648:
lib.c:649:static unsigned long crc32_tab[] = {
lib.c:650:      0x00000000L, 0x77073096L, 0xee0e612cL, 0x990951baL,
0x076dc419L,
lib.c:651:      0x706af48fL, 0xe963a535L, 0x9e6495a3L, 0x0edb8832L,
0x79dcb8a4L,
lib.c:652:      0xe0d5e91eL, 0x97d2d988L, 0x09b64c2bL, 0x7eb17cbdL,
0xe7b82d07L,
lib.c:653:      0x90bf1d91L, 0x1db71064L, 0x6ab020f2L, 0xf3b97148L,
0x84be41deL,
lib.c:654:      0x1adad47dL, 0x6ddde4ebL, 0xf4d4b551L, 0x83d385c7L,
0x136c9856L,
lib.c:655:      0x646ba8c0L, 0xfd62f97aL, 0x8a65c9ecL, 0x14015c4fL,
0x63066cd9L,
lib.c:656:      0xfa0f3d63L, 0x8d080df5L, 0x3b6e20c8L, 0x4c69105eL,
0xd56041e4L,
lib.c:657:      0xa2677172L, 0x3c03e4d1L, 0x4b04d447L, 0xd20d85fdL,
0xa50ab56bL,
lib.c:658:      0x35b5a8faL, 0x42b2986cL, 0xdbbbc9d6L, 0xacbcf940L,
0x32d86ce3L,
lib.c:659:      0x45df5c75L, 0xdcd60dcfL, 0xabd13d59L, 0x26d930acL,
0x51de003aL,
```

```
lib.c:660:      0xc8d75180L, 0xbfd06116L, 0x21b4f4b5L, 0x56b3c423L,
0xcfba9599L,
lib.c:661:      0xb8bda50fL, 0x2802b89eL, 0x5f058808L, 0xc60cd9b2L,
0xb10be924L,
lib.c:662:      0x2f6f7c87L, 0x58684c11L, 0xc1611dabL, 0xb6662d3dL,
0x76dc4190L,
lib.c:663:      0x01db7106L, 0x98d220bcL, 0xefd5102aL, 0x71b18589L,
0x06b6b51fL,
lib.c:664:      0x9fbfe4a5L, 0xe8b8d433L, 0x7807c9a2L, 0x0f00f934L,
0x9609a88eL,
lib.c:665:      0xe10e9818L, 0x7f6a0dbbL, 0x086d3d2dL, 0x91646c97L,
0xe6635c01L,
lib.c:666:      0x6b6b51f4L, 0x1c6c6162L, 0x856530d8L, 0xf262004eL,
0x6c0695edL,
lib.c:667:      0x1b01a57bL, 0x8208f4c1L, 0xf50fc457L, 0x65b0d9c6L,
0x12b7e950L,
lib.c:668:      0x8bbeb8eaL, 0xfcb9887cL, 0x62dd1ddfL, 0x15da2d49L,
0x8cd37cf3L,
lib.c:669:      0xfbd44c65L, 0x4db26158L, 0x3ab551ceL, 0xa3bc0074L,
0xd4bb30e2L,
lib.c:670:      0x4adfa541L, 0x3dd895d7L, 0xa4d1c46dL, 0xd3d6f4fbL,
0x4369e96aL,
lib.c:671:      0x346ed9fcL, 0xad678846L, 0xda60b8d0L, 0x44042d73L,
0x33031de5L,
lib.c:672:      0xaa0a4c5fL, 0xdd0d7cc9L, 0x5005713cL, 0x270241aaL,
0xbe0b1010L,
lib.c:673:      0xc90c2086L, 0x5768b525L, 0x206f85b3L, 0xb966d409L,
0xce61e49fL,
lib.c:674:      0x5edef90eL, 0x29d9c998L, 0xb0d09822L, 0xc7d7a8b4L,
0x59b33d17L,
lib.c:675:      0x2eb40d81L, 0xb7bd5c3bL, 0xc0ba6cadL, 0xedb88320L,
0x9abfb3b6L,
lib.c:676:      0x03b6e20cL, 0x74b1d29aL, 0xead54739L, 0x9dd277afL,
0x04db2615L,
lib.c:677:      0x73dc1683L, 0xe3630b12L, 0x94643b84L, 0x0d6d6a3eL,
0x7a6a5aa8L,
lib.c:678:      0xe40ecf0bL, 0x9309ff9dL, 0x0a00ae27L, 0x7d079eb1L,
0xf00f9344L,
lib.c:679:      0x8708a3d2L, 0x1e01f268L, 0x6906c2feL, 0xf762575dL,
0x806567cbL,
lib.c:680:      0x196c3671L, 0x6e6b06e7L, 0xfed41b76L, 0x89d32be0L,
0x10da7a5aL,
lib.c:681:      0x67dd4accL, 0xf9b9df6fL, 0x8ebeeff9L, 0x17b7be43L,
0x60b08ed5L,
lib.c:682:      0xd6d6a3e8L, 0xa1d1937eL, 0x38d8c2c4L, 0x4fdff252L,
0xd1bb67f1L,
lib.c:683:      0xa6bc5767L, 0x3fb506ddL, 0x48b2364bL, 0xd80d2bdaL,
0xaf0a1b4cL,
lib.c:684:      0x36034af6L, 0x41047a60L, 0xdf60efc3L, 0xa867df55L,
0x316e8eefL,
lib.c:685:      0x4669be79L, 0xcb61b38cL, 0xbc66831aL, 0x256fd2a0L,
0x5268e236L,
lib.c:686:      0xcc0c7795L, 0xbb0b4703L, 0x220216b9L, 0x5505262fL,
0xc5ba3bbeL,
lib.c:687:      0xb2bd0b28L, 0x2bb45a92L, 0x5cb36a04L, 0xc2d7ffa7L,
0xb5d0cf31L,
lib.c:688:      0x2cd99e8bL, 0x5bdeae1dL, 0x9b64c2b0L, 0xec63f226L,
0x756aa39cL,
lib.c:689:      0x026d930aL, 0x9c0906a9L, 0xeb0e363fL, 0x72076785L,
0x05005713L,
```

```
lib.c:690:      0x95bf4a82L, 0xe2b87a14L, 0x7bb12baeL, 0x0cb61b38L,
0x92d28e9bL,
lib.c:691:      0xe5d5be0dL, 0x7cdcefb7L, 0x0bdbdf21L, 0x86d3d2d4L,
0xf1d4e242L,
lib.c:692:      0x68ddb3f8L, 0x1fda836eL, 0x81be16cdL, 0xf6b9265bL,
0x6fb077e1L,
lib.c:693:      0x18b74777L, 0x88085ae6L, 0xff0f6a70L, 0x66063bcaL,
0x11010b5cL,
lib.c:694:      0x8f659effL, 0xf862ae69L, 0x616bffd3L, 0x166ccf45L,
0xa00ae278L,
lib.c:695:      0xd70dd2eeL, 0x4e048354L, 0x3903b3c2L, 0xa7672661L,
0xd06016f7L,
lib.c:696:      0x4969474dL, 0x3e6e77dbL, 0xaed16a4aL, 0xd9d65adcL,
0x40df0b66L,
lib.c:697:      0x37d83bf0L, 0xa9bcae53L, 0xdebb9ec5L, 0x47b2cf7fL,
0x30b5ffe9L,
lib.c:698:      0xbdbdf21cL, 0xcabac28aL, 0x53b39330L, 0x24b4a3a6L,
0xbad03605L,
lib.c:699:      0xcdd70693L, 0x54de5729L, 0x23d967bfL, 0xb3667a2eL,
0xc4614ab8L,
lib.c:700:      0x5d681b02L, 0x2a6f2b94L, 0xb40bbe37L, 0xc30c8ea1L,
0x5a05df1bL,
lib.c:701:      0x2d02ef8dL
lib.c:702:   };
lib.c:703:
lib.c:704:/* Return a 32-bit CRC of the contents of the buffer. */
lib.c:705:
lib.c:706:unsigned long crc32(const unsigned char *s, unsigned int len)
lib.c:707:{
lib.c:708:  unsigned int i;
lib.c:709:  unsigned long crc32val;
lib.c:710:
lib.c:711:  crc32val = 0;
lib.c:712:  for (i = 0;  i < len;  i ++)
lib.c:713:    {
lib.c:714:      crc32val =
lib.c:715: crc32_tab[(crc32val ^ s[i]) & 0xff] ^
lib.c:716:   (crc32val >> 8);
lib.c:717:    }
lib.c:718:  return crc32val;
lib.c:719:}
lib.c:720:
lib.c:721:/*
lib.c:722: * Hashing function for a string
lib.c:723: */
lib.c:724:unsigned int hashmap_hash_int(struct hashmap_map * m, char*
keystring){
lib.c:725:
lib.c:726:    unsigned long key = crc32((unsigned char*)(keystring),
strlen(keystring));
lib.c:727:
lib.c:728: /* Robert Jenkins' 32 bit Mix Function */
lib.c:729: key += (key << 12);
lib.c:730: key ^= (key >> 22);
lib.c:731: key += (key << 4);
lib.c:732: key ^= (key >> 9);
lib.c:733: key += (key << 10);
lib.c:734: key ^= (key >> 2);
lib.c:735: key += (key << 7);
lib.c:736: key ^= (key >> 12);
```

```
lib.c:737:
lib.c:738: /* Knuth's Multiplicative Method */
lib.c:739: key = (key >> 3) * 2654435761;
lib.c:740:
lib.c:741: return key % m->table_size;
lib.c:742:}
lib.c:743:
lib.c:744:/*
lib.c:745: * Return the integer of the location in data
lib.c:746: * to store the point to the item, or MAP_FULL.
lib.c:747: */
lib.c:748:int hashmap_hash(struct hashmap_map* m, char* key){
lib.c:749: int curr;
lib.c:750: int i;
lib.c:751:
lib.c:752: /* If full, return immediately */
lib.c:753: if(m->size >= (m->table_size/2)) return MAP_FULL;
lib.c:754:
lib.c:755: /* Find the best index */
lib.c:756: curr = hashmap_hash_int(m, key);
lib.c:757:
lib.c:758: /* Linear probing */
lib.c:759: for(i = 0; i< MAX_CHAIN_LENGTH; i++){
lib.c:760:       if(m->data[curr].in_use == 0)
lib.c:761:             return curr;
lib.c:762:
lib.c:763:       if(m->data[curr].in_use == 1 && (strcmp(m-
>data[curr].key,key)==0))
lib.c:764:             return curr;
lib.c:765:
lib.c:766:       curr = (curr + 1) % m->table_size;
lib.c:767: }
lib.c:768:
lib.c:769: return MAP_FULL;
lib.c:770:}
lib.c:771:
lib.c:772:/*
lib.c:773: * Doubles the size of the hashmap, and rehashes all the
elements
lib.c:774: */
lib.c:775:int hashmap_rehash(struct hashmap_map *m){
lib.c:776: int i;
lib.c:777: int old_size;
lib.c:778: struct hashmap_element* curr;
lib.c:779:
lib.c:780: /* Setup the new elements */
lib.c:781: struct hashmap_element* temp = (struct hashmap_element *)
lib.c:782:       calloc(2 * m->table_size, sizeof(struct
hashmap_element));
lib.c:783: if(!temp) return MAP_OMEM;
lib.c:784:
lib.c:785: /* Update the array */
lib.c:786: curr = m->data;
lib.c:787: m->data = temp;
lib.c:788:
lib.c:789: /* Update the size */
lib.c:790: old_size = m->table_size;
lib.c:791: m->table_size = 2 * m->table_size;
lib.c:792: m->size = 0;
lib.c:793:
```

```
lib.c:794: /* Rehash the elements */
lib.c:795: for(i = 0; i < old_size; i++){
lib.c:796:        int status;
lib.c:797:
lib.c:798:        if (curr[i].in_use == 0)
lib.c:799:            continue;
lib.c:800:
lib.c:801:       hashmap_put(m, curr[i].key, curr[i].data);
lib.c:802: }
lib.c:803:
lib.c:804: free(curr);
lib.c:805:
lib.c:806: return MAP_OK;
lib.c:807:}
lib.c:808:
lib.c:809:/*
lib.c:810: * Add a pointer to the hashmap with some key
lib.c:811: */
lib.c:812:struct hashmap_map* hashmap_put(struct hashmap_map* m,...){
lib.c:813: int index;
lib.c:814: void* data1;
lib.c:815: void* data2;
lib.c:816: char* key;
lib.c:817: va_list ap;
lib.c:818: va_start(ap, 2);
lib.c:819:
lib.c:820: switch (m->keytype) {
lib.c:821:       case INT:
lib.c:822:            data1 = InttoVoid(va_arg(ap, int));
lib.c:823:            key = malloc(16);
lib.c:824:            snprintf(key, 16, "%d", VoidtoInt(data1));
lib.c:825:            break;
lib.c:826:
lib.c:827:       case STRING:
lib.c:828:            data1 = StringtoVoid(va_arg(ap, char*));
lib.c:829:            key = VoidtoString(data1);
lib.c:830:            //printf("%s\n",key);
lib.c:831:            break;
lib.c:832:
lib.c:833:       case NODE:
lib.c:834:            data1 = NodetoVoid(va_arg(ap, struct Node*));
lib.c:835:            key = malloc(16);
lib.c:836:            snprintf(key, 16, "%d", VoidtoNode(data1)->id);
lib.c:837:            //printf("%s\n",key);
lib.c:838:            break;
lib.c:839:
lib.c:840:       default:
lib.c:841:            break;
lib.c:842: }
lib.c:843:
lib.c:844: switch (m->valuetype) {
lib.c:845:       case INT:
lib.c:846:            data2 = InttoVoid(va_arg(ap, int));
lib.c:847:            break;
lib.c:848:
lib.c:849:       case FLOAT:
lib.c:850:            data2 = FloattoVoid(va_arg(ap, double));
lib.c:851:            break;
lib.c:852:
lib.c:853:       case BOOL:
```

```
lib.c:854:            data2 = BooltoVoid(va_arg(ap, double));
lib.c:855:            break;
lib.c:856:
lib.c:857:      case STRING:
lib.c:858:            data2 = StringtoVoid(va_arg(ap, char*));
lib.c:859:            break;
lib.c:860:
lib.c:861:      case NODE:
lib.c:862:            data2 = NodetoVoid(va_arg(ap, struct Node*));
lib.c:863:            break;
lib.c:864:
lib.c:865:      case GRAPH:
lib.c:866:            data2 = GraphtoVoid(va_arg(ap, struct Graph*));
lib.c:867:            break;
lib.c:868:
lib.c:869:      default:
lib.c:870:            break;
lib.c:871: }
lib.c:872:
lib.c:873: va_end(ap);
lib.c:874:
lib.c:875: /* Find a place to put our value */
lib.c:876: index = hashmap_hash(m, key);
lib.c:877: while(index == MAP_FULL){
lib.c:878:      if (hashmap_rehash(m) == MAP_OMEM) {
lib.c:879:            printf("Error! Hashmap out of Memory\n");
lib.c:880:            exit(1);
lib.c:881:      }
lib.c:882:      index = hashmap_hash(m, key);
lib.c:883: }
lib.c:884:
lib.c:885: /* Set the data */
lib.c:886: m->data[index].data[0] = data1;
lib.c:887: m->data[index].data[1] = data2;
lib.c:888: m->data[index].key = key;
lib.c:889: m->data[index].in_use = 1;
lib.c:890: m->size++;
lib.c:891:
lib.c:892: return m;
lib.c:893:}
lib.c:894:
lib.c:895:/*
lib.c:896: * Get your pointer out of the hashmap with a key
lib.c:897: */
lib.c:898:// int hashmap_get(map_t in, char* key, any_t *arg){
lib.c:899:void* hashmap_get(struct hashmap_map* m,...){
lib.c:900: int curr;
lib.c:901: int i;
lib.c:902: char* key;
lib.c:903: va_list ap;
lib.c:904: va_start(ap, 1);
lib.c:905:
lib.c:906: switch (m->keytype) {
lib.c:907:      case INT:
lib.c:908:            key = malloc(16);
lib.c:909:            snprintf(key, 16, "%d", va_arg(ap, int));
lib.c:910:            break;
lib.c:911:
lib.c:912:      case STRING:
lib.c:913:            key = va_arg(ap, char*);
```

```
lib.c:914:             break;
lib.c:915:
lib.c:916:       case NODE:
lib.c:917:             key = malloc(16);
lib.c:918:             snprintf(key, 16, "%d", va_arg(ap, struct Node*)-
>id);
lib.c:919:             break;
lib.c:920:
lib.c:921:       default:
lib.c:922:             break;
lib.c:923: }
lib.c:924: va_end(ap);
lib.c:925:
lib.c:926: /* Find data location */
lib.c:927: curr = hashmap_hash_int(m, key);
lib.c:928:
lib.c:929: /* Linear probing, if necessary */
lib.c:930: for(i = 0; i<MAX_CHAIN_LENGTH; i++){
lib.c:931:
lib.c:932:        int in_use = m->data[curr].in_use;
lib.c:933:        if (in_use == 1){
lib.c:934:            if (strcmp(m->data[curr].key,key)==0){
lib.c:935:                // *arg = (m->data[curr].data);
lib.c:936:                // return MAP_OK;
lib.c:937:                return m->data[curr].data[1];
lib.c:938:            }
lib.c:939:       }
lib.c:940:
lib.c:941:       curr = (curr + 1) % m->table_size;
lib.c:942: }
lib.c:943: printf("Error! Hashmap_Get:Key not Exist!\n");
lib.c:944: exit(1);
lib.c:945:}
lib.c:946:
lib.c:947:/*
lib.c:948: * Iterate the function parameter over each element in the
hashmap.  The
lib.c:949: * additional any_t argument is passed to the function as its
first
lib.c:950: * argument and the hashmap element is the second.
lib.c:951: */
lib.c:952:int hashmap_iterate(struct hashmap_map* m, Func f) {
lib.c:953:
lib.c:954: /* On empty hashmap, return immediately */
lib.c:955: if (hashmap_length(m) <= 0)
lib.c:956:       return MAP_MISSING;
lib.c:957:
lib.c:958: /* Linear probing */
lib.c:959: for(int i = 0; i< m->table_size; i++)
lib.c:960:       if(m->data[i].in_use != 0) {
lib.c:961:             int status = f(m->data[i].key, m->data[i].data[0],
m->data[i].data[1]);
lib.c:962:             if (status != MAP_OK) {
lib.c:963:                   return status;
lib.c:964:             }
lib.c:965:       }
lib.c:966:
lib.c:967:    return MAP_OK;
lib.c:968:}
lib.c:969:
```

```
lib.c:970:int hashmap_printhelper(char* key, int32_t type, void* value){
lib.c:971: switch (type) {
lib.c:972:       case INT:
lib.c:973:            printf("%s: %d",key, VoidtoInt(value));
lib.c:974:            break;
lib.c:975:
lib.c:976:       case FLOAT:
lib.c:977:            printf("%s: %f",key, VoidtoFloat(value));
lib.c:978:            break;
lib.c:979:
lib.c:980:       case BOOL:
lib.c:981:            printf("%s: %d",key, VoidtoBool(value));
lib.c:982:            break;
lib.c:983:
lib.c:984:       case STRING:
lib.c:985:            printf("%s: %s",key, VoidtoString(value));
lib.c:986:            break;
lib.c:987:
lib.c:988:       case NODE:
lib.c:989:            printf("%s: ",key);
lib.c:990:            printNode(VoidtoNode(value));
lib.c:991:            break;
lib.c:992:
lib.c:993:       case GRAPH:
lib.c:994:            printf("%s: ",key);
lib.c:995:            printGraph(VoidtoGraph(value));
lib.c:996:            break;
lib.c:997:
lib.c:998:       default:
lib.c:999:            break;
lib.c:1000:}
lib.c:1001:return 0;
lib.c:1002:}
lib.c:1003:
lib.c:1004:int hashmap_print(struct hashmap_map* m){
lib.c:1005:if (m == NULL) {
lib.c:1006:      printf("(null)\n");
lib.c:1007:      return 0;
lib.c:1008:}
lib.c:1009:printf("{");
lib.c:1010:for(int i = 0, c=0; i< m->table_size; i++)
lib.c:1011:      if(m->data[i].in_use != 0) {
lib.c:1012:            hashmap_printhelper(m->data[i].key, m->valuetype,
m->data[i].data[1]);
lib.c:1013:            c++;
lib.c:1014:            if (c < m->size) printf(", ");
lib.c:1015:      }
lib.c:1016:printf("}\n");
lib.c:1017:return 0;
lib.c:1018:}
lib.c:1019:
lib.c:1020:bool hashmap_haskey(struct hashmap_map* m,...){
lib.c:1021:int curr;
lib.c:1022:int i;
lib.c:1023:char* key;
lib.c:1024:va_list ap;
lib.c:1025:va_start(ap, 1);
lib.c:1026:
lib.c:1027:switch (m->keytype) {
lib.c:1028:      case INT:
```

```
lib.c:1029:            key = malloc(16);
lib.c:1030:            snprintf(key, 16, "%d", va_arg(ap, int));
lib.c:1031:            break;
lib.c:1032:
lib.c:1033:      case STRING:
lib.c:1034:            key = va_arg(ap, char*);
lib.c:1035:            break;
lib.c:1036:
lib.c:1037:      case NODE:
lib.c:1038:            key = malloc(16);
lib.c:1039:            snprintf(key, 16, "%d", va_arg(ap, struct Node*)-
>id);
lib.c:1040:            break;
lib.c:1041:
lib.c:1042:      default:
lib.c:1043:            break;
lib.c:1044:}
lib.c:1045:va_end(ap);
lib.c:1046:/* Find data location */
lib.c:1047:curr = hashmap_hash_int(m, key);
lib.c:1048:/* Linear probing, if necessary */
lib.c:1049:for(i = 0; i<MAX_CHAIN_LENGTH; i++){
lib.c:1050:        int in_use = m->data[curr].in_use;
lib.c:1051:        if (in_use == 1){
lib.c:1052:            if (strcmp(m->data[curr].key,key)==0){
lib.c:1053:                // *arg = (m->data[curr].data);
lib.c:1054:                // return MAP_OK;
lib.c:1055:                return 1;
lib.c:1056:            }
lib.c:1057:      }
lib.c:1058:      curr = (curr + 1) % m->table_size;
lib.c:1059:}
lib.c:1060:return 0;
lib.c:1061:}
lib.c:1062:
lib.c:1063:struct List* hashmap_keys(struct hashmap_map* m){
lib.c:1064:if (hashmap_length(m) <= 0){
lib.c:1065:      printf("Error! hashmap_getkey: No keys!\n");
lib.c:1066:      exit(1);
lib.c:1067:}
lib.c:1068:struct List* dataset = createList(m->keytype);
lib.c:1069:for(int i = 0; i< m->table_size; i++){
lib.c:1070:      if(m->data[i].in_use != 0) {
lib.c:1071:            switch (m->keytype) {
lib.c:1072:                  case INT:
lib.c:1073:                        addList(dataset, VoidtoInt(m-
>data[i].data[0]));
lib.c:1074:                        break;
lib.c:1075:
lib.c:1076:                  case STRING:
lib.c:1077:                        addList(dataset, VoidtoString(m-
>data[i].data[0]));
lib.c:1078:                        break;
lib.c:1079:
lib.c:1080:                  case NODE:
lib.c:1081:                        addList(dataset, VoidtoNode(m-
>data[i].data[0]));
lib.c:1082:                        break;
lib.c:1083:
lib.c:1084:                  default:
```

```
lib.c:1085:                         break;
lib.c:1086:}
lib.c:1087:      }
lib.c:1088:}
lib.c:1089:return dataset;
lib.c:1090:}
lib.c:1091:
lib.c:1092:// /*
lib.c:1093://  * Remove an element with that key from the map
lib.c:1094://  */
lib.c:1095:struct hashmap_map* hashmap_remove(struct hashmap_map* m,...){
lib.c:1096:int i;
lib.c:1097:int curr;
lib.c:1098:char* key;
lib.c:1099:va_list ap;
lib.c:1100:va_start(ap, 1);
lib.c:1101:
lib.c:1102:switch (m->keytype) {
lib.c:1103:      case INT:
lib.c:1104:           key = malloc(16);
lib.c:1105:           snprintf(key, 16, "%d", va_arg(ap, int));
lib.c:1106:           break;
lib.c:1107:
lib.c:1108:      case STRING:
lib.c:1109:           key = va_arg(ap, char*);
lib.c:1110:           break;
lib.c:1111:
lib.c:1112:      case NODE:
lib.c:1113:           key = malloc(16);
lib.c:1114:           snprintf(key, 16, "%d", va_arg(ap, struct Node*)-
>id);
lib.c:1115:           break;
lib.c:1116:
lib.c:1117:      default:
lib.c:1118:           break;
lib.c:1119:}
lib.c:1120:va_end(ap);
lib.c:1121:
lib.c:1122:/* Find key */
lib.c:1123:curr = hashmap_hash_int(m, key);
lib.c:1124:
lib.c:1125:/* Linear probing, if necessary */
lib.c:1126:for(i = 0; i<MAX_CHAIN_LENGTH; i++){
lib.c:1127:
lib.c:1128:        int in_use = m->data[curr].in_use;
lib.c:1129:        if (in_use == 1){
lib.c:1130:            if (strcmp(m->data[curr].key,key)==0){
lib.c:1131:                /* Blank out the fields */
lib.c:1132:                m->data[curr].in_use = 0;
lib.c:1133:                m->data[curr].data[0] = NULL;
lib.c:1134:                m->data[curr].data[1] = NULL;
lib.c:1135:                m->data[curr].key = NULL;
lib.c:1136:
lib.c:1137:                /* Reduce the size */
lib.c:1138:                m->size--;
lib.c:1139:                return m;
lib.c:1140:            }
lib.c:1141:      }
lib.c:1142:      curr = (curr + 1) % m->table_size;
lib.c:1143:}
```

```
lib.c:1144:printf("Error! hashmap_remove: Missing data!\n");
lib.c:1145:exit(1);
lib.c:1146:}
lib.c:1147:
lib.c:1148:
lib.c:1149:int hashmap_length(struct hashmap_map* m){
lib.c:1150:if(m != NULL) return m->size;
lib.c:1151:else return 0;
lib.c:1152:}
lib.c:1153:
lib.c:1154:int32_t hashmap_keytype(struct hashmap_map* m){
lib.c:1155:return m->keytype;
lib.c:1156:}
lib.c:1157:
lib.c:1158:int32_t hashmap_valuetype(struct hashmap_map* m){
lib.c:1159:return m->valuetype;
lib.c:1160:}
lib.c:1161:
lib.c:1162:
lib.c:1163:
lib.c:1164:/**********************************
lib.c:1165:Minheap Methods
lib.c:1166:**********************************/
lib.c:1167:
lib.c:1168:struct minHeap* initList(int32_t type) {
lib.c:1169:  struct minHeap *heap = malloc(sizeof(heap));
lib.c:1170:  if (heap != NULL) {
lib.c:1171:    heap->type = type;
lib.c:1172:    heap->array = createList(EDGE);
lib.c:1173:  }
lib.c:1174:  return heap;
lib.c:1175:}
lib.c:1176:
lib.c:1177:void swap(struct List* list, int index1, int index2){
lib.c:1178:    struct Edge* data1 = (struct Edge*) getList(list, index1);
lib.c:1179:    struct Edge* data2 = (struct Edge*) getList(list, index2);
lib.c:1180:
lib.c:1181:    setList(list, index1, data2);
lib.c:1182:    setList(list, index2, data1);
lib.c:1183:}
lib.c:1184:
lib.c:1185:int eCompare(struct minHeap* hp, struct Edge* lchild, struct
Edge* rchild) {
lib.c:1186:    switch(hp->type) {
lib.c:1187:        case INT:
lib.c:1188:          return lchild->a > rchild->a ? 1 : 0;
lib.c:1189:          break;
lib.c:1190:        case FLOAT:
lib.c:1191:          return lchild->b > rchild->b ? 1 : 0;
lib.c:1192:          break;
lib.c:1193:        case BOOL:
lib.c:1194:          return lchild->c > rchild->c ? 1 : 0;
lib.c:1195:          break;
lib.c:1196:        default:
lib.c:1197:          printf("[Error] Unsupported type for edge compare
!\n");
lib.c:1198:          exit(1);
lib.c:1199:          break;
lib.c:1200:    }
lib.c:1201:}
```

```
lib.c:1202:
lib.c:1203:void heapify(struct minHeap* hp, int size){
lib.c:1204:if (size <= 1) {
lib.c:1205:      return;
lib.c:1206:}
lib.c:1207:    int i = (size - 1) /2;
lib.c:1208:    struct Edge* lchild = NULL;
lib.c:1209:    struct Edge* rchild = NULL;
lib.c:1210:    struct Edge* cur = NULL;
lib.c:1211:    //int largest = i;
lib.c:1212:    while (i >= 0) {
lib.c:1213:        cur = (struct Edge*) getList(hp->array, i);
lib.c:1214:        if (2 * i + 1 < size) {
lib.c:1215:            lchild = (struct Edge*) getList(hp->array, 2 * i +
1);
lib.c:1216:        }
lib.c:1217:        if (2 * i + 2 < size) {
lib.c:1218:            rchild = (struct Edge*) getList(hp->array, 2 * i +
2);
lib.c:1219:        }
lib.c:1220:        if (rchild != NULL && lchild != NULL) {
lib.c:1221:            if (eCompare(hp, lchild, rchild) > 0) {
lib.c:1222:                if (eCompare(hp, cur, rchild) > 0) {
lib.c:1223:                    swap(hp->array, i, 2 * i + 2);
lib.c:1224:                }
lib.c:1225:            } else {
lib.c:1226:                if (eCompare(hp, cur, lchild) > 0) {
lib.c:1227:                    swap(hp->array, i, 2 * i + 1);
lib.c:1228:                }
lib.c:1229:            }
lib.c:1230:        } else if (rchild == NULL && lchild != NULL){
lib.c:1231:            if (eCompare(hp, cur, lchild) > 0) {
lib.c:1232:                swap(hp->array, i, 2 * i + 1);
lib.c:1233:            }
lib.c:1234:        }
lib.c:1235:        i--;
lib.c:1236:    }
lib.c:1237:}
lib.c:1238:
lib.c:1239:void insertData(struct minHeap* hp, struct Edge* data){
lib.c:1240:// printf("************\n");
lib.c:1241:// printList(hp->array);
lib.c:1242:// printf("************\n");
lib.c:1243:    if(getListSize(hp->array) > 0){
lib.c:1244:      addList(hp->array, data);
lib.c:1245://   printf("---------------\n");
lib.c:1246://   printList(hp->array);
lib.c:1247://   printf("---------------\n");
lib.c:1248:      int size = getListSize(hp->array);
lib.c:1249:      heapify(hp, size);
lib.c:1250:    } else {
lib.c:1251:      addList(hp->array, data);
lib.c:1252:    }
lib.c:1253:// printf("---------------\n");
lib.c:1254://   printList(hp->array);
lib.c:1255://   printf("---------------\n");
lib.c:1256:}
lib.c:1257:
lib.c:1258:struct Edge* getMinValue(struct minHeap* hp){
lib.c:1259://printf("start getMinValue\n");
```

```
lib.c:1260:    if(getListSize(hp->array) > 0){
lib.c:1261:      struct Edge* data = (struct Edge*) getList(hp->array,0);
lib.c:1262:      int size = getListSize(hp->array);
lib.c:1263:      swap(hp->array, 0, size - 1);
lib.c:1264:      popList(hp->array);
lib.c:1265:  //printf("HelloPOP\n");
lib.c:1266:  //printf("%d\n", size - 1);
lib.c:1267:      heapify(hp,size - 1);
lib.c:1268:  //printf("HelloHEAP\n");
lib.c:1269:      return data;
lib.c:1270:    } else {
lib.c:1271:      printf("Cannot get value from empty minHeap");
lib.c:1272:      return NULL;
lib.c:1273:    }
lib.c:1274:}
lib.c:1275:
lib.c:1276:int32_t printHeap(struct minHeap* hp){
lib.c:1277:    printList(hp->array);
lib.c:1278:    //printf("printHeap finished");
lib.c:1279:    return 0;
lib.c:1280:}
lib.c:1281:
lib.c:1282:void decreasePriority(struct minHeap* hp, struct Edge* e) {
lib.c:1283://printf("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n");
lib.c:1284://printEdge(e);
lib.c:1285://printHeap(hp);
lib.c:1286:bool flag = true;
lib.c:1287://int size = getListSize(hp->array);
lib.c:1288:for (int i = 0; i < getListSize(hp->array); i++) {
lib.c:1289:      struct Edge* elem = (struct Edge*) getList(hp->array,
i);
lib.c:1290:      if (edgeCompare(e, elem)) {
lib.c:1291:                setList(hp->array, i, e);
lib.c:1292:                flag = false;
lib.c:1293:                break;
lib.c:1294:      }
lib.c:1295:}
lib.c:1296://printHeap(hp);
lib.c:1297://printf("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\n");
lib.c:1298:heapify(hp, getListSize(hp->array));
lib.c:1299://printHeap(hp);
lib.c:1300://printf("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n");
lib.c:1301:}
lib.c:1302:
lib.c:1303:/**********************************
lib.c:1304:Queue Methods
lib.c:1305:**********************************/
lib.c:1306:struct Queue* createQueue(int32_t type) {
lib.c:1307:struct Queue* new = (struct Queue*) malloc(sizeof(struct
Queue));
lib.c:1308:new->lst = createList(type);
lib.c:1309:return new;
lib.c:1310:}
lib.c:1311:
lib.c:1312:struct Queue* pushBack(struct Queue* q, ...) {
lib.c:1313:va_list ap;
lib.c:1314:va_start(ap, 1);
lib.c:1315:void* data;
lib.c:1316:switch (q->lst->list_type) {
lib.c:1317:      case INT:
```

```
lib.c:1318:            addList(q->lst, va_arg(ap, int));
lib.c:1319:            break;
lib.c:1320:
lib.c:1321:      case FLOAT:
lib.c:1322:            addList(q->lst, va_arg(ap, double));
lib.c:1323:            break;
lib.c:1324:
lib.c:1325:      case BOOL:
lib.c:1326:            addList(q->lst, va_arg(ap, bool));
lib.c:1327:            break;
lib.c:1328:
lib.c:1329:      case STRING:
lib.c:1330:            addList(q->lst, va_arg(ap, char*));
lib.c:1331:            break;
lib.c:1332:
lib.c:1333:      case NODE:
lib.c:1334:            addList(q->lst, va_arg(ap, struct Node*));
lib.c:1335:            break;
lib.c:1336:      case EDGE:
lib.c:1337:            addList(q->lst, va_arg(ap, struct Edge*));
lib.c:1338:            break;
lib.c:1339:
lib.c:1340:      case GRAPH:
lib.c:1341:            addList(q->lst, va_arg(ap, struct Graph*));
lib.c:1342:            break;
lib.c:1343:
lib.c:1344:      default:
lib.c:1345:            break;
lib.c:1346:}
lib.c:1347:  va_end(ap);
lib.c:1348:  return q;
lib.c:1349:}
lib.c:1350:
lib.c:1351:void* popFront(struct Queue* q) {
lib.c:1352:if (q == NULL) {
lib.c:1353:      printf("[Error] popFront() - queue doesn't exist. \n");
lib.c:1354:      exit(1);
lib.c:1355:} else if (q->lst->index -1 < 0) {
lib.c:1356:      printf("Error! Nothing Can be poped T.T\n");
lib.c:1357:      exit(1);
lib.c:1358:}
lib.c:1359:void* data = getList(q->lst, 0);
lib.c:1360:removeList(q->lst, 0);
lib.c:1361:return data;
lib.c:1362:}
lib.c:1363:
lib.c:1364:int getQueueSize(struct Queue* q) {
lib.c:1365:return getListSize(q->lst);
lib.c:1366:}
lib.c:1367:
lib.c:1368:int32_t printQueue(struct Queue* q) {
lib.c:1369:return printList(q->lst);
lib.c:1370:}
lib.c:1371:
lib.c:1372:/**********************************
lib.c:1373:Node Methods
lib.c:1374:**********************************/
lib.c:1375:
lib.c:1376:struct Node* createNode(int32_t id, int32_t type, ...) {
lib.c:1377:
```

```
lib.c:1378:struct Node* node = (struct Node*) malloc(sizeof(struct
Node));
lib.c:1379:node->id = id;
lib.c:1380:node->type = type;
lib.c:1381:node->visited = false;
lib.c:1382:va_list ap;
lib.c:1383:va_start(ap, 1);
lib.c:1384:
lib.c:1385:switch (type) {
lib.c:1386:      case INT:
lib.c:1387:           node->a = va_arg(ap, int);  break;
lib.c:1388:      case FLOAT:
lib.c:1389:           node->b = va_arg(ap, double);     break;
lib.c:1390:      case BOOL:
lib.c:1391:           node->c = va_arg(ap, bool); break;
lib.c:1392:      case STRING:
lib.c:1393:           node->d = va_arg(ap, char*);break;
lib.c:1394:      default:
lib.c:1395:           break;
lib.c:1396:}
lib.c:1397:  va_end(ap);
lib.c:1398:  return node;
lib.c:1399:}
lib.c:1400:
lib.c:1401:bool setVisited(struct Node* node) {
lib.c:1402:node->visited = true;
lib.c:1403:return 1;
lib.c:1404:}
lib.c:1405:
lib.c:1406:bool isVisited(struct Node* node) {
lib.c:1407:return node->visited;
lib.c:1408:}
lib.c:1409:
lib.c:1410:void* nodeGetValue(struct Node* node, int32_t type) {
lib.c:1411:      if (node == NULL) {
lib.c:1412:           printf("[Error] Node doesn't exist!\n");
lib.c:1413:           exit(1);
lib.c:1414:      }
lib.c:1415:      void* res;
lib.c:1416:      switch (type) {
lib.c:1417:           case INT:
lib.c:1418:                if (node->type == INT)
lib.c:1419:                     res = InttoVoid(node->a);
lib.c:1420:                else if (node->type == FLOAT)
lib.c:1421:                     res = InttoVoid((int)node->b);
lib.c:1422:                else if (node->type == BOOL)
lib.c:1423:                     res = InttoVoid( node->c ? 1 : 0 );
lib.c:1424:                else {
lib.c:1425:                     res = InttoVoid(0);
lib.c:1426:                }
lib.c:1427:                break;
lib.c:1428:           case FLOAT:
lib.c:1429:                if (node->type == INT)
lib.c:1430:                     res = FloattoVoid((double)node->a);
lib.c:1431:                else if (node->type == FLOAT)
lib.c:1432:                     res = FloattoVoid(node->b);
lib.c:1433:                else if (node->type == BOOL)
lib.c:1434:                     res = FloattoVoid( node->c ? 1 : 0 );
lib.c:1435:                else {
lib.c:1436:                     res = FloattoVoid(0);
```

```
lib.c:1437:                 }
lib.c:1438:                 break;
lib.c:1439:           case BOOL:
lib.c:1440:                 if (node->type == INT)
lib.c:1441:                       res = BooltoVoid(node->a != 0);
lib.c:1442:                 else if (node->type == FLOAT)
lib.c:1443:                       res = BooltoVoid(node->b != 0);
lib.c:1444:                 else if (node->type == BOOL)
lib.c:1445:                       res = BooltoVoid(node->c);
lib.c:1446:                 else {
lib.c:1447:                       res = BooltoVoid(false);
lib.c:1448:                 }
lib.c:1449:                 break;
lib.c:1450:           case STRING:
lib.c:1451:                 if (node->type == STRING)
lib.c:1452:                       res = StringtoVoid(node->d);
lib.c:1453:                 else{
lib.c:1454:                       res = StringtoVoid("");
lib.c:1455:                 }
lib.c:1456:                 break;
lib.c:1457:           default:
lib.c:1458:                 printf("[Error] Edge Value Type Error!\n");
lib.c:1459:                 exit(1);
lib.c:1460:                 break;
lib.c:1461:      }
lib.c:1462:      return res;
lib.c:1463:}
lib.c:1464:
lib.c:1465:int32_t printNode(struct Node * node) {
lib.c:1466:if (node == NULL) {
lib.c:1467:      printf("(null)\n");
lib.c:1468:      return 0;
lib.c:1469:}
lib.c:1470:switch (node->type) {
lib.c:1471:      case 0:
lib.c:1472:            printf("node %d\n", node->a);
lib.c:1473:            break;
lib.c:1474:      case 1:
lib.c:1475:            printf("node %f\n", node->b);
lib.c:1476:            break;
lib.c:1477:      case 2:
lib.c:1478:            printf("node %s\n", node->c ? "true" : "false");
lib.c:1479:            break;
lib.c:1480:      case 3:
lib.c:1481:            printf("node %s\n", node->d);
lib.c:1482:            break;
lib.c:1483:      default:
lib.c:1484:            printf("node%3d\n", node->id);
lib.c:1485:            break;
lib.c:1486:}
lib.c:1487:return 0;
lib.c:1488:}
lib.c:1489:
lib.c:1490:
lib.c:1491:/**********************************
lib.c:1492:Edge Methods
lib.c:1493:**********************************/
lib.c:1494:
lib.c:1495:struct Edge createEdge(
lib.c:1496:struct Node* sour,
```

```
lib.c:1497:struct Node* dest,
lib.c:1498:int32_t type,
lib.c:1499:int32_t a,
lib.c:1500:double b,
lib.c:1501:bool c,
lib.c:1502:char* d
lib.c:1503:) {
lib.c:1504:
lib.c:1505:struct Edge * newE = malloc(sizeof(struct Edge));
lib.c:1506:newE->sour = sour;
lib.c:1507:newE->dest = dest;
lib.c:1508:newE->type = type;
lib.c:1509:newE->a = a;
lib.c:1510:newE->b = b;
lib.c:1511:newE->c = c;
lib.c:1512:newE->d = d;
lib.c:1513:return (*newE);
lib.c:1514:}
lib.c:1515:
lib.c:1516:struct Edge* createEdgeP(
lib.c:1517:struct Node* sour,
lib.c:1518:struct Node* dest,
lib.c:1519:int32_t type,
lib.c:1520:int32_t a,
lib.c:1521:double b,
lib.c:1522:bool c,
lib.c:1523:char* d
lib.c:1524:) {
lib.c:1525:// struct Edge e = {sour, dest, type, a, b, c, d};
lib.c:1526:// return e;
lib.c:1527:struct Edge * newE = malloc(sizeof(struct Edge));
lib.c:1528:newE->sour = sour;
lib.c:1529:newE->dest = dest;
lib.c:1530:newE->type = type;
lib.c:1531:newE->a = a;
lib.c:1532:newE->b = b;
lib.c:1533:newE->c = c;
lib.c:1534:newE->d = d;
lib.c:1535:return newE;
lib.c:1536:}
lib.c:1537:
lib.c:1538:void* edgeGetValue(struct Edge* edge, int32_t type) {
lib.c:1539:if (edge == NULL) {
lib.c:1540:      printf("[Error] Edge doesn't exist!\n");
lib.c:1541:      exit(1);
lib.c:1542:}
lib.c:1543:void* res;
lib.c:1544:switch (type) {
lib.c:1545:      case INT:
lib.c:1546:            if (edge->type == INT)
lib.c:1547:                  res = InttoVoid(edge->a);
lib.c:1548:            else if (edge->type == FLOAT)
lib.c:1549:                  res = InttoVoid((int)edge->b);
lib.c:1550:            else if (edge->type == BOOL)
lib.c:1551:                  res = InttoVoid( edge->c ? 1 : 0 );
lib.c:1552:            else {
lib.c:1553:                  res = InttoVoid(0);
lib.c:1554:            }
lib.c:1555:            break;
lib.c:1556:      case FLOAT:
```

```
lib.c:1557:           if (edge->type == INT)
lib.c:1558:                 res = FloattoVoid((double)edge->a);
lib.c:1559:           else if (edge->type == FLOAT)
lib.c:1560:                 res = FloattoVoid(edge->b);
lib.c:1561:           else if (edge->type == BOOL)
lib.c:1562:                 res = FloattoVoid( edge->c ? 1 : 0 );
lib.c:1563:           else {
lib.c:1564:                 res = FloattoVoid(0);
lib.c:1565:           }
lib.c:1566:           break;
lib.c:1567:     case BOOL:
lib.c:1568:           if (edge->type == INT)
lib.c:1569:                 res = BooltoVoid(edge->a != 0);
lib.c:1570:           else if (edge->type == FLOAT)
lib.c:1571:                 res = BooltoVoid(edge->b != 0);
lib.c:1572:           else if (edge->type == BOOL)
lib.c:1573:                 res = BooltoVoid(edge->c);
lib.c:1574:           else {
lib.c:1575:                 res = BooltoVoid(false);
lib.c:1576:           }
lib.c:1577:           break;
lib.c:1578:     case STRING:
lib.c:1579:           if (edge->type == STRING)
lib.c:1580:                 res = StringtoVoid(edge->d);
lib.c:1581:           else {
lib.c:1582:                 res = StringtoVoid("");
lib.c:1583:           }
lib.c:1584:           break;
lib.c:1585:     default:
lib.c:1586:           printf("[Error] Edge Value Type Error!\n");
lib.c:1587:           exit(1);
lib.c:1588:           break;
lib.c:1589:}
lib.c:1590:return res;
lib.c:1591:}
lib.c:1592:
lib.c:1593:int32_t printEdge(struct Edge * edge) {
lib.c:1594:if (edge == NULL) {
lib.c:1595:     printf("(null)\n");
lib.c:1596:     return 0;
lib.c:1597:}
lib.c:1598:switch (edge->sour->type) {
lib.c:1599:     // case 0:
lib.c:1600:     //    printf("edge %3d->%d\n$%3d", edge->sour->id, edge-
>a, edge->dest->id);
lib.c:1601:     //    break;
lib.c:1602:     // case 1:
lib.c:1603:     //    printf("edge %3d->%f\n$%3d", edge->sour->id, edge-
>b, edge->dest->id);
lib.c:1604:     //    break;
lib.c:1605:     // case 2:
lib.c:1606:     //    printf("edge %3d->%s\n$%3d", edge->sour->id, edge-
>c ? "true" : "false", edge->dest->id);
lib.c:1607:     //    break;
lib.c:1608:     // case 3:
lib.c:1609:     //    printf("edge %3d->%s\n$%3d", edge->sour->id, edge-
>d, edge->dest->id);
lib.c:1610:     //    break;
lib.c:1611:     // default:
```

```
lib.c:1612:      //    printf("edge %3d->node%3d\n", edge->sour->id,
edge->dest->id);
lib.c:1613:      //    break;
lib.c:1614:      case 0:
lib.c:1615:            printf("edge %d ->%d: %d\n", edge->sour->a, edge-
>dest->a, edge->a);
lib.c:1616:            break;
lib.c:1617:      case 1:
lib.c:1618:            printf("edge %f ->%f: %d\n", edge->sour->b, edge-
>dest->b, edge->a);
lib.c:1619:            break;
lib.c:1620:      case 2:
lib.c:1621:            printf("edge %s ->%s: %d\n", edge->sour->c? "true"
: "false", edge->dest->c? "true" : "false", edge->a);
lib.c:1622:            break;
lib.c:1623:      case 3:
lib.c:1624:            printf("edge %s ->%s: %d\n", edge->sour->d, edge-
>dest->d, edge->a);
lib.c:1625:            break;
lib.c:1626:      default:
lib.c:1627:            printf("edge %3d ->%3d: 0\n", edge->sour->id,
edge->dest->id);
lib.c:1628:            break;
lib.c:1629:}
lib.c:1630:return 0;
lib.c:1631:}
lib.c:1632:
lib.c:1633:int32_t printEdgeValue(struct Edge * edge) {
lib.c:1634:if (edge == NULL) {
lib.c:1635:      printf("(null)\n");
lib.c:1636:      return 0;
lib.c:1637:}
lib.c:1638:switch (edge->type) {
lib.c:1639:      case 0:
lib.c:1640:            printf("%d\n", edge->a);
lib.c:1641:            break;
lib.c:1642:      case 1:
lib.c:1643:            printf("%f\n", edge->b);
lib.c:1644:            break;
lib.c:1645:      case 2:
lib.c:1646:            printf("%s\n", edge->c ? "true" : "false");
lib.c:1647:            break;
lib.c:1648:      case 3:
lib.c:1649:            printf("%s\n", edge->d);
lib.c:1650:            break;
lib.c:1651:      default:
lib.c:1652:            printf("[Error] Unknown Edge Value Type!\n");
lib.c:1653:            exit(1);
lib.c:1654:            break;
lib.c:1655:}
lib.c:1656:return 0;
lib.c:1657:}
lib.c:1658:
lib.c:1659:/**********************************
lib.c:1660:Graph Methods
lib.c:1661:**********************************/
lib.c:1662:
lib.c:1663:int32_t graphAddEdgeHelper(struct Graph* g, struct Edge e) {
lib.c:1664:if (g == NULL) exit(1);
lib.c:1665:int i;
```

```
lib.c:1666:for (i=0; i < g->en; i++) {
lib.c:1667:      if (g->edges[i].sour == e.sour && g->edges[i].dest ==
e.dest) {
lib.c:1668:            g->edges[i] = e;
lib.c:1669:            return 0;
lib.c:1670:      }
lib.c:1671:}
lib.c:1672:g->edges[i] = e;
lib.c:1673:g->en ++;
lib.c:1674:return 0;
lib.c:1675:}
lib.c:1676:
lib.c:1677:int32_t graphAddEdgeP( struct Graph* g, struct Node* sour,
struct Node* dest, int32_t type, ...) {
lib.c:1678:if (g == NULL) {
lib.c:1679:      printf("[Error] Graph doesn't exist!\n");
lib.c:1680:      exit(1);
lib.c:1681:}
lib.c:1682:if (sour == dest) return 0;
lib.c:1683:if (g->en + 1 >= g->en_len) {
lib.c:1684:      printf("[Error] # Graph Edges reach the limit!\n");
lib.c:1685:      exit(1);
lib.c:1686:}
lib.c:1687:if (graphAddNode(g, sour) > 0) exit(1);
lib.c:1688:if (graphAddNode(g, dest) > 0) exit(1);
lib.c:1689:
lib.c:1690:// Assign the Edge Value
lib.c:1691:struct Edge e = createEdge(sour, dest, type, 0, 0, 0, NULL);
lib.c:1692:va_list ap;
lib.c:1693:va_start(ap, 1);
lib.c:1694:void* tmp = va_arg(ap, void*);
lib.c:1695:switch (type) {
lib.c:1696:      case INT:
lib.c:1697:            e.a = *((int*)tmp);    break;
lib.c:1698:      case FLOAT:
lib.c:1699:            e.b = *((double*)tmp); break;
lib.c:1700:      case BOOL:
lib.c:1701:            e.c = *((bool*)tmp);   break;
lib.c:1702:      case STRING:
lib.c:1703:            e.d = ((char*)tmp);    break;
lib.c:1704:      default:
lib.c:1705:            break;
lib.c:1706:}
lib.c:1707:  va_end(ap);
lib.c:1708:
lib.c:1709:int i;
lib.c:1710:// Edges already exist in the graph
lib.c:1711:for (i=0; i<g->en; i++) {
lib.c:1712:      if (g->edges[i].sour == sour && g->edges[i].dest ==
dest) {
lib.c:1713:            g->edges[i] = e;
lib.c:1714:            return 0;
lib.c:1715:      }
lib.c:1716:}
lib.c:1717:g->edges[i] = e;
lib.c:1718:g->en++;
lib.c:1719:return 0;
lib.c:1720:}
lib.c:1721:
lib.c:1722:int32_t graphAddEdge(
```

```
lib.c:1723:struct Graph* g,
lib.c:1724:struct Node* sour,
lib.c:1725:struct Node* dest,
lib.c:1726:int32_t type,
lib.c:1727:int32_t a,
lib.c:1728:double b,
lib.c:1729:bool c,
lib.c:1730:char* d
lib.c:1731:) {
lib.c:1732:if (g == NULL) {
lib.c:1733:      printf("[Error] Graph doesn't exist!\n");
lib.c:1734:      exit(1);
lib.c:1735:}
lib.c:1736:if (sour == dest) return 0;
lib.c:1737:if (g->en + 1 >= g->en_len) {
lib.c:1738:      printf("[Error] # Graph Edges reach the limit!\n");
lib.c:1739:      exit(1);
lib.c:1740:}
lib.c:1741:if (graphAddNode(g, sour) > 0) exit(1);
lib.c:1742:if (graphAddNode(g, dest) > 0) exit(1);
lib.c:1743:int i;
lib.c:1744:// Edges already exist in the graph
lib.c:1745:for (i=0; i<g->en; i++) {
lib.c:1746:      if (g->edges[i].sour == sour && g->edges[i].dest ==
dest) {
lib.c:1747:            g->edges[i].type = type;
lib.c:1748:            g->edges[i].a = a;
lib.c:1749:            g->edges[i].b = b;
lib.c:1750:            g->edges[i].c = c;
lib.c:1751:            g->edges[i].d = d;
lib.c:1752:            return 0;
lib.c:1753:      }
lib.c:1754:}
lib.c:1755:struct Edge e = createEdge(sour, dest, type, a, b, c, d);
lib.c:1756:g->edges[i] = e;
lib.c:1757:g->en++;
lib.c:1758:return 0;
lib.c:1759:}
lib.c:1760:
lib.c:1761:bool graphEdgeExist(struct Graph* g, struct Node* sour, struct
Node* dest) {
lib.c:1762:if (g == NULL) {
lib.c:1763:      printf("[Error] Graph doesn't exist!\n");
lib.c:1764:      exit(1);
lib.c:1765:}
lib.c:1766:int i;
lib.c:1767:for (i=0; i<g->en; i++) {
lib.c:1768:      if (g->edges[i].sour == sour && g->edges[i].dest ==
dest) {
lib.c:1769:            return true;
lib.c:1770:      }
lib.c:1771:}
lib.c:1772:return false;
lib.c:1773:}
lib.c:1774:struct Edge* graphGetEdge(struct Graph* g, struct Node* sour,
struct Node* dest) {
lib.c:1775:if (g == NULL) {
lib.c:1776:      printf("[Error] Graph doesn't exist!\n");
lib.c:1777:      exit(1);
lib.c:1778:}
```

```
lib.c:1779:int i;
lib.c:1780:for (i=0; i<g->en; i++) {
lib.c:1781:      if (g->edges[i].sour == sour && g->edges[i].dest ==
dest) {
lib.c:1782:            return &g->edges[i];
lib.c:1783:      }
lib.c:1784:}
lib.c:1785:return NULL;
lib.c:1786:}
lib.c:1787:
lib.c:1788:/*
lib.c:1789:Split the graph into a list of graphs, in which all graphs are
connected.
lib.c:1790: */
lib.c:1791:struct List* splitGraph(struct Graph * gh) {
lib.c:1792:struct List* l = createList(GRAPH);
lib.c:1793:if (gh == NULL) return l;
lib.c:1794:
lib.c:1795:gh = copyGraph(gh);
lib.c:1796:struct Node* root = gh->root;
lib.c:1797:struct Graph* gh_tmp = NULL;
lib.c:1798:int vn = gh->vn, en = gh->en, max_vn = gh->vn, max_en = gh-
>en;
lib.c:1799:int i, j, k;
lib.c:1800:struct List* queue = createList(NODE);
lib.c:1801:struct Node* node = NULL, *node_tmp = NULL;
lib.c:1802:
lib.c:1803:while (vn > 0) {
lib.c:1804:      gh_tmp = createGraph();
lib.c:1805:      for (i=0; i<max_vn; i++) {
lib.c:1806:            if (gh->nodes[i] != NULL) break;
lib.c:1807:      }
lib.c:1808:      addList(queue, gh->nodes[i]);
lib.c:1809:      while (getListSize(queue) > 0) {
lib.c:1810:            node = (struct Node*) getList(queue, 0);
lib.c:1811:            removeList(queue, 0);
lib.c:1812:            graphAddNode(gh_tmp, node);
lib.c:1813:            for (k=0; k<max_vn; k++) {
lib.c:1814:                  if (gh->nodes[k] == node) {
lib.c:1815:                        gh->nodes[k] = NULL;
lib.c:1816:                        vn--;
lib.c:1817:                        break;
lib.c:1818:                  }
lib.c:1819:            }
lib.c:1820:            if (k == max_vn) continue;
lib.c:1821:            for (j=0; j<max_en; j++) {
lib.c:1822:                  if (gh->edges[j].type != -9 && gh-
>edges[j].sour == node) {
lib.c:1823:                        node_tmp = gh->edges[j].dest;
lib.c:1824:                  } else if (gh->edges[j].type != -9 && gh-
>edges[j].dest == node) {
lib.c:1825:                        node_tmp = gh->edges[j].sour;
lib.c:1826:                  } else {
lib.c:1827:                        node_tmp = NULL;
lib.c:1828:                  }
lib.c:1829:                  if (node_tmp == NULL ) continue;
lib.c:1830:                  addList(queue, node_tmp);
lib.c:1831:                  graphAddEdgeHelper(gh_tmp, gh->edges[j]);
lib.c:1832:                  gh->edges[j].type = -9;
lib.c:1833:            }
```

```
lib.c:1834:      }
lib.c:1835:      // Adjust the root to the original one
lib.c:1836:      bool hasRoot = false;
lib.c:1837:      for (i=0; i<gh_tmp->vn; i++) {
lib.c:1838:           if (gh_tmp->nodes[i] == root) {
lib.c:1839:                gh_tmp -> root = root;
lib.c:1840:                hasRoot = true;
lib.c:1841:                break;
lib.c:1842:           }
lib.c:1843:      }
lib.c:1844:      // Make sure the subgrpah with original root is the
first in the list
lib.c:1845:      if (hasRoot && getListSize(l) > 0) {
lib.c:1846:           addList(l, (struct Graph*)getList(l, 0));
lib.c:1847:           setList(l, 0, gh_tmp);
lib.c:1848:      } else {
lib.c:1849:           addList(l, gh_tmp);
lib.c:1850:      }
lib.c:1851:}
lib.c:1852:free(gh);
lib.c:1853:return l;
lib.c:1854:}
lib.c:1855:
lib.c:1856:struct Graph* createGraph() {
lib.c:1857:struct Graph* g = (struct Graph*) malloc( sizeof(struct Graph)
);
lib.c:1858:g->vn = 0;
lib.c:1859:g->en = 0;
lib.c:1860:g->vn_len = 32;
lib.c:1861:g->en_len = 128;
lib.c:1862:g->root = NULL;
lib.c:1863:g->nodes = (struct Node**) malloc( sizeof(struct Node*) * 16
);
lib.c:1864:g->edges = (struct Edge*) malloc( sizeof(struct Edge) * 64 );
lib.c:1865:return g;
lib.c:1866:}
lib.c:1867:struct Graph* copyGraph(struct Graph* a) {
lib.c:1868:if (a == NULL) return NULL;
lib.c:1869:struct Graph* g = (struct Graph*) malloc( sizeof(struct Graph)
);
lib.c:1870:memcpy(g, a, sizeof(struct Graph));
lib.c:1871:g->nodes = (struct Node**) malloc( sizeof(struct Node*) * a-
>vn_len );
lib.c:1872:g->edges = (struct Edge*) malloc( sizeof(struct Edge) * a-
>en_len );
lib.c:1873:int i;
lib.c:1874:for (i=0; i<a->vn; i++) {
lib.c:1875:      g->nodes[i] = a->nodes[i];
lib.c:1876:}
lib.c:1877:struct Edge* tmp;
lib.c:1878:for (i=0; i<a->en; i++) {
lib.c:1879:      tmp = (struct Edge*) malloc( sizeof(struct Edge) );
lib.c:1880:      memcpy(tmp, &a->edges[i], sizeof(struct Edge));
lib.c:1881:      g->edges[i] = *tmp;
lib.c:1882:}
lib.c:1883:return g;
lib.c:1884:}
lib.c:1885:
lib.c:1886:struct Graph* mergeGraph(struct Graph* a, struct Graph* b) {
lib.c:1887:if (b == NULL) return copyGraph(a);
```

```
lib.c:1888:if (a == NULL) return copyGraph(b);
lib.c:1889:
lib.c:1890:struct Graph* gh = copyGraph(a);
lib.c:1891:// Check whether two graph have shared nodes
lib.c:1892:int i; int j;
lib.c:1893:bool hasShared = false;
lib.c:1894:for (i=0; i < a->vn; i++) {
lib.c:1895:      for (j=0; j < b->vn; j++) {
lib.c:1896:            if (a->nodes[i] == b->nodes[j]) {
lib.c:1897:                  hasShared = true;
lib.c:1898:                  break;
lib.c:1899:            }
lib.c:1900:      }
lib.c:1901:      if (hasShared) break;
lib.c:1902:}
lib.c:1903:if (!hasShared) return gh; // Return the copy of graph a
lib.c:1904:
lib.c:1905:for (i=0; i< b->vn; i++) {
lib.c:1906:      graphAddNode(gh, b->nodes[i]);
lib.c:1907:}
lib.c:1908:for (i=0; i< b->en; i++) {
lib.c:1909:      graphAddEdgeHelper(gh, b->edges[i]);
lib.c:1910:}
lib.c:1911:return gh;
lib.c:1912:}
lib.c:1913:struct Node* graphGetRoot(struct Graph* g) {
lib.c:1914:if (g == NULL) {
lib.c:1915:      printf("[Error] Graph doesn't exist!\n");
lib.c:1916:      exit(1);
lib.c:1917:}
lib.c:1918:return g->root;
lib.c:1919:}
lib.c:1920:int32_t graphSetRoot(struct Graph* g, struct Node * root) {
lib.c:1921:if (g == NULL) {
lib.c:1922:      printf("[Error] Graph doesn't exist!\n");
lib.c:1923:      exit(1);
lib.c:1924:}
lib.c:1925:if (root == NULL) {
lib.c:1926:      printf("[Error] Root node doesn't exist!\n");
lib.c:1927:      exit(1);
lib.c:1928:}
lib.c:1929:int i;
lib.c:1930:for (i=0; i<g->vn; i++) {
lib.c:1931:      if (g->nodes[i] == root) {
lib.c:1932:            g->root = root;
lib.c:1933:            return 0;
lib.c:1934:      }
lib.c:1935:}
lib.c:1936:printf("[Error] Root doesn't exist in the graph!\n");
lib.c:1937:exit(1);
lib.c:1938:}
lib.c:1939:
lib.c:1940:struct List* subGraph(struct Graph* a, struct Graph* b) {
lib.c:1941:if (a == NULL) {
lib.c:1942:      printf("[Error] Graph doesn't exist!\n");
lib.c:1943:      exit(1);
lib.c:1944:}
lib.c:1945:struct Graph* gh = copyGraph(a);
lib.c:1946:if (b == NULL || b->en <= 0) {
lib.c:1947:      struct List* l = createList(GRAPH);
```

```
lib.c:1948:      addList(l, gh);
lib.c:1949:      return l;
lib.c:1950:}
lib.c:1951:int i, j, k;
lib.c:1952:for (i = 0; i < b->en; i++) {
lib.c:1953:      struct Edge e = b->edges[i];
lib.c:1954:      for (j = 0; j < gh->en; j++) {
lib.c:1955:           if (gh->edges[j].sour == e.sour && gh-
>edges[j].dest == e.dest) {
lib.c:1956:                gh->edges[j] = gh->edges[gh->en-1];
lib.c:1957:                gh->en --;
lib.c:1958:                break;
lib.c:1959:           }
lib.c:1960:      }
lib.c:1961:}
lib.c:1962:return splitGraph(gh);
lib.c:1963:}
lib.c:1964:
lib.c:1965:int32_t graphAddList(struct Graph* g, int direction, struct
List * l, struct List * edges) {
lib.c:1966:if (g == NULL || g->root == NULL || l == NULL) {
lib.c:1967:      printf("[Error] Graph or List doesn't exist!\n");
lib.c:1968:      exit(1);
lib.c:1969:}
lib.c:1970:int i, j, lsize = l->index, rsize = edges == NULL ? 0 : edges-
>index;
lib.c:1971:if (lsize != rsize && rsize > 1) {
lib.c:1972:      printf("[Error] Edge List Not Compatible with Node/Graph
List!\n");
lib.c:1973:      exit(1);
lib.c:1974:}
lib.c:1975:for (i=0; i<lsize; i++) {
lib.c:1976:      struct Node * node = NULL;
lib.c:1977:      if (l->list_type == GRAPH) {
lib.c:1978:           // Merge the graph
lib.c:1979:           struct Graph * gh_tmp = (struct Graph*)l-
>stored_data[i];
lib.c:1980:           node = gh_tmp->root;
lib.c:1981:           for (j=0; j< gh_tmp->vn; j++) {
lib.c:1982:                graphAddNode(g, gh_tmp->nodes[j]);
lib.c:1983:           }
lib.c:1984:           for (j=0; j< gh_tmp->en; j++) {
lib.c:1985:                graphAddEdgeHelper(g, gh_tmp->edges[j]);
lib.c:1986:           }
lib.c:1987:      } else if (l->list_type == NODE) {
lib.c:1988:           node = (struct Node*)l->stored_data[i];
lib.c:1989:      } else {
lib.c:1990:           printf("[Error] GraphAddList List Type doesn't
supported!!\n");
lib.c:1991:           exit(1);
lib.c:1992:      }
lib.c:1993:
lib.c:1994:      if (node == NULL) continue;
lib.c:1995:      if (edges != NULL && edges->index > 0) {
lib.c:1996:           int edgePos = edges->index == 1 ? 0 : i;
lib.c:1997:           switch (direction) {
lib.c:1998:                case RIGHT_LINK:
lib.c:1999:                     graphAddEdgeP(g, g->root, node, edges-
>list_type, edges->stored_data[edgePos]); break;
lib.c:2000:                case LEFT_LINK:
```

```
lib.c:2001:                         graphAddEdgeP(g, node, g->root, edges-
>list_type, edges->stored_data[edgePos]); break;
lib.c:2002:                 case DOUBLE_LINK:
lib.c:2003:                         graphAddEdgeP(g, g->root, node, edges-
>list_type, edges->stored_data[edgePos]);
lib.c:2004:                         graphAddEdgeP(g, node, g->root, edges-
>list_type, edges->stored_data[edgePos]);
lib.c:2005:                         break;
lib.c:2006:                 default:
lib.c:2007:                         break;
lib.c:2008:           }
lib.c:2009:      } else {
lib.c:2010:           switch (direction) {
lib.c:2011:                 case RIGHT_LINK:
lib.c:2012:                         graphAddEdgeP(g, g->root, node, -1,
NULL); break;
lib.c:2013:                 case LEFT_LINK:
lib.c:2014:                         graphAddEdgeP(g, node, g->root, -1,
NULL); break;
lib.c:2015:                 case DOUBLE_LINK:
lib.c:2016:                         graphAddEdgeP(g, g->root, node, -1,
NULL);
lib.c:2017:                         graphAddEdgeP(g, node, g->root, -1,
NULL);
lib.c:2018:                         break;
lib.c:2019:                 default:
lib.c:2020:                         break;
lib.c:2021:           }
lib.c:2022:      }
lib.c:2023:}
lib.c:2024:return 0;
lib.c:2025:}
lib.c:2026:
lib.c:2027:int32_t graphAddNode(struct Graph* g, struct Node * node) {
lib.c:2028:if (g == NULL) {
lib.c:2029:      printf("[Error] Graph doesn't exist!\n");
lib.c:2030:      exit(1);
lib.c:2031:}
lib.c:2032:if (g->vn + 1 >= g->vn_len) {
lib.c:2033:      printf("[Warning] # Graph Nodes reach the limit!\n");
lib.c:2034:      exit(1);
lib.c:2035:}
lib.c:2036:int i;
lib.c:2037:// Nodes already exist in the graph
lib.c:2038:for (i=0; i<g->vn; i++) {
lib.c:2039:      if (g->nodes[i] == node) return 0;
lib.c:2040:}
lib.c:2041:// Update the root if the graph is empty
lib.c:2042:if (g->root == NULL) {
lib.c:2043:      g->root = node;
lib.c:2044:}
lib.c:2045:g->nodes[i] = node;
lib.c:2046:g->vn++;
lib.c:2047:return 0;
lib.c:2048:}
lib.c:2049:
lib.c:2050:struct List* graphRemoveNode(struct Graph* gh, struct Node *
node) {
lib.c:2051:if (gh == NULL) {
lib.c:2052:      printf("[Error] Graph doesn't exist!\n");
```

```
lib.c:2053:      exit(1);
lib.c:2054:}
lib.c:2055:gh = copyGraph(gh);
lib.c:2056:int i, j;
lib.c:2057:// Remove Node
lib.c:2058:for (i=0; i<gh->vn; i++) {
lib.c:2059:      if (gh->nodes[i] == node) {
lib.c:2060:            for (j=i; j<gh->vn-1; j++) {
lib.c:2061:                  gh->nodes[j] = gh->nodes[j+1];
lib.c:2062:            }
lib.c:2063:            gh->nodes[j] = NULL;
lib.c:2064:            gh->vn--;
lib.c:2065:      }
lib.c:2066:}
lib.c:2067:if (gh->root == node) {
lib.c:2068:      gh->root = gh->vn == 0 ? NULL : gh->nodes[0];
lib.c:2069:}
lib.c:2070:// Remove Edges
lib.c:2071:for (i=0, j=gh->en-1; i<=j;) {
lib.c:2072:      if (gh->edges[i].sour == node || gh->edges[i].dest ==
node) {
lib.c:2073:            gh->edges[i] = gh->edges[j];
lib.c:2074:            gh->en--;
lib.c:2075:            j--;
lib.c:2076:      } else {
lib.c:2077:            i++;
lib.c:2078:      }
lib.c:2079:}
lib.c:2080:return splitGraph(gh);
lib.c:2081:}
lib.c:2082:
lib.c:2083:struct List* graphGetAllNodes(struct Graph* g) {
lib.c:2084:if (g == NULL) {
lib.c:2085:      printf("[Error] Graph doesn't exist!\n");
lib.c:2086:      exit(1);
lib.c:2087:}
lib.c:2088:struct List* ret = createList(NODE);
lib.c:2089:int i;
lib.c:2090:for (i=0; i < g->vn; i++) {
lib.c:2091:      addList(ret, g->nodes[i]);
lib.c:2092:}
lib.c:2093:return ret;
lib.c:2094:}
lib.c:2095:
lib.c:2096:int32_t graphNumOfNodes(struct Graph* g) {
lib.c:2097:if (g == NULL) {
lib.c:2098:      printf("[Error] Graph doesn't exist!\n");
lib.c:2099:      exit(1);
lib.c:2100:}
lib.c:2101:return g->vn;
lib.c:2102:}
lib.c:2103:
lib.c:2104:int32_t graphNumOfEdges(struct Graph* g) {
lib.c:2105:if (g == NULL) {
lib.c:2106:      printf("[Error] Graph doesn't exist!\n");
lib.c:2107:      exit(1);
lib.c:2108:}
lib.c:2109:return g->en;
lib.c:2110:}
lib.c:2111:
```

```
lib.c:2112:struct List* graphGetChildNodes(struct Graph* g, struct Node*
rt) {
lib.c:2113:if (g == NULL) {
lib.c:2114:      printf("[Error] Graph doesn't exist!\n");
lib.c:2115:      exit(1);
lib.c:2116:}
lib.c:2117:struct List* children = createList(NODE);
lib.c:2118:if (rt == NULL) return children;
lib.c:2119:int i;
lib.c:2120:for (i=0; i< g->en; i++) {
lib.c:2121:      if (g->edges[i].sour == rt) {
lib.c:2122:            addList(children, g->edges[i].dest);
lib.c:2123:      }
lib.c:2124:}
lib.c:2125:return children;
lib.c:2126:}
lib.c:2127:
lib.c:2128:bool containsNode(struct Graph* g, struct Node* n) {
lib.c:2129:if (g == NULL) {
lib.c:2130:      printf("[Error] Graph doesn't exist!\n");
lib.c:2131:      exit(1);
lib.c:2132:}
lib.c:2133:int i;
lib.c:2134:for (i = 0; i < g->vn; i++) {
lib.c:2135:      if (g->nodes[i] == n) {
lib.c:2136:            return true;
lib.c:2137:      }
lib.c:2138:}
lib.c:2139:return false;
lib.c:2140:}
lib.c:2141:
lib.c:2142:int32_t printGraph(struct Graph* g) {
lib.c:2143:if (g == NULL) {
lib.c:2144:      printf("(null)\n");
lib.c:2145:      return 0;
lib.c:2146:}
lib.c:2147:printf("----------------------------------------\n");
lib.c:2148:int i;
lib.c:2149:for (i=0; i<g->vn; i++) {
lib.c:2150:      printNode(g->nodes[i]);
lib.c:2151:}
lib.c:2152:for (i=0; i<g->en; i++) {
lib.c:2153:      printEdge(&g->edges[i]);
lib.c:2154:}
lib.c:2155:printf("----------------------------------------\n");
lib.c:2156:return 0;
lib.c:2157:}
lib.c:2158:
lib.c:2159:bool setAllUnvisited(struct Graph* g) {
lib.c:2160:if (g == NULL) {
lib.c:2161:      printf("[Error] Graph doesn't exist!\n");
lib.c:2162:      exit(1);
lib.c:2163:}
lib.c:2164:int i;
lib.c:2165:for (i = 0; i < g->vn; i++) {
lib.c:2166:      g->nodes[i]->visited = false;
lib.c:2167:}
lib.c:2168:return true;
lib.c:2169:}
lib.c:2170:
```

```
lib.c:2171:
lib.c:2172:struct List* dfs(struct Graph* g, struct Node* n) {
lib.c:2173:bool flag = true;
lib.c:2174:struct List* path = createList(NODE);
lib.c:2175:if (g == NULL) {
lib.c:2176:      printf("[Error] Graph doesn't exist!\n");
lib.c:2177:      exit(1);
lib.c:2178:} else if (!containsNode(g, n)) {
lib.c:2179:      printf("[Error] Graph doesn't contain source node!\n");
lib.c:2180:      exit(1);
lib.c:2181:} else {
lib.c:2182:      //printf("------------------------- DFS BEGIN ---------
----------------\n");
lib.c:2183:      //setAllUnvisited(g);
lib.c:2184:      struct List* lst = createList(NODE);
lib.c:2185:      addList(lst, n);
lib.c:2186:      while (getListSize(lst) != 0) {
lib.c:2187:            struct Node* tmp = (struct Node*) popList(lst);
lib.c:2188:            if (tmp->visited == true) {
lib.c:2189:                  flag = false;
lib.c:2190:                  continue;
lib.c:2191:            } else {
lib.c:2192:                  tmp->visited = true;
lib.c:2193:                  addList(path, tmp);
lib.c:2194:                  struct List* childs = graphGetChildNodes(g,
tmp);
lib.c:2195:                  lst = concatList(lst, childs);
lib.c:2196:            }
lib.c:2197:      }
lib.c:2198:      //printf("------------------------- DFS END -----------
-------------\n");
lib.c:2199:}
lib.c:2200:// if (!flag) {
lib.c:2201://    printf("graph has cycle in it\n");
lib.c:2202:// }
lib.c:2203:// return flag;
lib.c:2204://printList(path);
lib.c:2205:return path;
lib.c:2206:}
lib.c:2207:
lib.c:2208:struct List* bfs(struct Graph* g, struct Node* n) {
lib.c:2209:bool flag = true;
lib.c:2210:struct List* path = createList(NODE);
lib.c:2211:if (g == NULL) {
lib.c:2212:      printf("[Error] Graph doesn't exist!\n");
lib.c:2213:      exit(1);
lib.c:2214:} else if (!containsNode(g, n)) {
lib.c:2215:      printf("[Error] Graph doesn't contain source node!\n");
lib.c:2216:      exit(1);
lib.c:2217:} else {
lib.c:2218:      //printf("------------------------- BFS BEGIN ---------
----------------\n");
lib.c:2219:      //setAllUnvisited(g);
lib.c:2220:      struct Queue* q = createQueue(NODE);
lib.c:2221:      pushBack(q, n);
lib.c:2222:      while (getQueueSize(q) != 0) {
lib.c:2223:            struct Node* tmp = (struct Node*) popFront(q);
lib.c:2224:            if (tmp->visited) {
lib.c:2225:                  flag = false;
lib.c:2226:                  continue;
```

```
lib.c:2227:            } else {
lib.c:2228:                  tmp->visited = true;
lib.c:2229:                  //printNode(tmp);
lib.c:2230:                  addList(path, tmp);
lib.c:2231:                  struct List* childs = graphGetChildNodes(g,
tmp);
lib.c:2232:                  concatList(q->lst, childs);
lib.c:2233:            }
lib.c:2234:      }
lib.c:2235:      //printf("------------------------- BFS END -----------
-------------\n");
lib.c:2236:}
lib.c:2237:// if (!flag) {
lib.c:2238://    printf("graph has cycle in it\n");
lib.c:2239:// }
lib.c:2240:// return flag;
lib.c:2241:return path;
lib.c:2242:}
lib.c:2243:
lib.c:2244:struct List* dijkstra(struct Graph* g, struct Node* sour,
struct Node* dest) {
lib.c:2245:setAllUnvisited(g);
lib.c:2246:struct hashmap_map* dist = hashmap_new(NODE, INT);
lib.c:2247:struct hashmap_map* prev = hashmap_new(NODE, NODE);
lib.c:2248:
lib.c:2249:hashmap_put(dist, sour, 0);
lib.c:2250:
lib.c:2251:struct List* set = graphGetAllNodes(g);
lib.c:2252:int graphSize = graphNumOfNodes(g);
lib.c:2253:struct minHeap* minH = initList(INT);
lib.c:2254:// printHeap(minH);
lib.c:2255:for (int i = 0; i < graphSize; i++) {
lib.c:2256:      struct Node* v = (struct Node*) getList(set, i);
lib.c:2257:      if (v->id != sour->id) {
lib.c:2258:            hashmap_put(dist, v, 10000);
lib.c:2259:            hashmap_put(prev, v, NULL);
lib.c:2260:      }
lib.c:2261:      //printf("%d\n", (*(int*)hashmap_get(dist, v)));
lib.c:2262:      int tmp = (*(int*)hashmap_get(dist, v));
lib.c:2263:      struct Edge* eg = createEdgeP(sour, v, 0, tmp, 0.0, 0,
NULL);
lib.c:2264:      insertData(minH, eg);
lib.c:2265:}
lib.c:2266:while (getListSize(minH->array) > 0) {
lib.c:2267:      //printf("WhileStart----------------------------------
---\n");
lib.c:2268:      //printHeap(minH);
lib.c:2269:      //printf("1--------------------------------------\n");
lib.c:2270:      struct Edge* uEdge = getMinValue(minH);
lib.c:2271:      //printHeap(minH);
lib.c:2272:      //printf("2--------------------------------------\n");
lib.c:2273:      struct Node* u = uEdge->dest;
lib.c:2274:      setVisited(u);
lib.c:2275:      if (u->id == dest->id) {
lib.c:2276:            break;
lib.c:2277:      }
lib.c:2278:      struct List* ngbrs = graphGetChildNodes(g, u);
lib.c:2279:      int ngbrsSize = getListSize(ngbrs);
lib.c:2280:
lib.c:2281:      for (int i = 0; i < ngbrsSize; i++) {
```

```
lib.c:2282:            //printf("I: %d, ngbrSize: %d\n", i, ngbrsSize);
lib.c:2283:            struct Node* v = getList(ngbrs, i);
lib.c:2284:            if (isVisited(v)) {
lib.c:2285:                  continue;
lib.c:2286:            }
lib.c:2287:            int alt = graphGetEdge(g, u, v)->a +
(*(int*)hashmap_get(dist, u));
lib.c:2288:            //printf("weight:
**************************\n");
lib.c:2289:            //printf("alt: %d, dist of v : %d", alt,
*(int*)(hashmap_get(dist, v)));
lib.c:2290:          //printHeap(minH);
lib.c:2291:            if (alt < (*(int*)(hashmap_get(dist, v)))) {
lib.c:2292:                  //printf("update\n");
lib.c:2293:                  // hashmap_print(dist);
lib.c:2294:                  // hashmap_print(prev);
lib.c:2295:                  hashmap_remove(dist, v);
lib.c:2296:                  hashmap_remove(prev, v);
lib.c:2297:                  hashmap_put(dist, v, alt);
lib.c:2298:                  hashmap_put(prev, v, u);
lib.c:2299:                  // hashmap_print(dist);
lib.c:2300:                  // hashmap_print(prev);
lib.c:2301:                  struct Edge* newE = createEdgeP(sour, v, INT,
alt, 0.0, 0, NULL);
lib.c:2302:                  decreasePriority(minH, newE);
lib.c:2303:            }
lib.c:2304:      }
lib.c:2305:}
lib.c:2306://printGraph(g);
lib.c:2307://hashmap_print(prev);
lib.c:2308://hashmap_print(dist);
lib.c:2309:struct List* path = createList(NODE);
lib.c:2310:addList(path, dest);
lib.c:2311:struct Node* paren = (struct Node*)hashmap_get(prev, dest);
lib.c:2312:while (paren->id != sour->id) {
lib.c:2313:      addList(path, paren);
lib.c:2314:      paren = (struct Node*)hashmap_get(prev, paren);
lib.c:2315:}
lib.c:2316:addList(path, sour);
lib.c:2317:int pathSize = getListSize(path);
lib.c:2318:struct List* path2 = createList(NODE);
lib.c:2319:int i;
lib.c:2320:for (i = pathSize - 1; i >= 0; i--) {
lib.c:2321:      struct Node* tmp = (struct Node*) popList(path);
lib.c:2322:      addList(path2, tmp);
lib.c:2323:}
lib.c:2324:
lib.c:2325:setAllUnvisited(g);
lib.c:2326:return path2;
lib.c:2327:}
lib.c:2328:
lib.c:2329:// int main() {
lib.c:2330://    struct Queue* q = createQueue(INT);
lib.c:2331://    pushBack(q, 1);
lib.c:2332://    pushBack(q, 2);
lib.c:2333://    pushBack(q, 3);
lib.c:2334://    printQueue(q);
lib.c:2335://    popFront(q);
lib.c:2336://    printQueue(q);
lib.c:2337://    popFront(q);
```

```
lib.c:2338://    printQueue(q);
lib.c:2339://    pushBack(q, 4);
lib.c:2340://    printQueue(q);
lib.c:2341:// }
lib.c:2342:
lib.c:2343:// int main() {
lib.c:2344://    printf("%f", (float)1 );
lib.c:2345:// }
lib.c:2346:
lib.c:2347:// int main(){
lib.c:2348:
lib.c:2349://          struct minHeap* mp = initList(INT);
lib.c:2350://         struct Node* sour = createNode(1, 0, 12);
lib.c:2351://         struct Node* dest = createNode(2, 0, 3);
lib.c:2352://     //printNode(sour);
lib.c:2353://     //printNode(dest);
lib.c:2354://         struct Node* s = createNode(3, 0, 6);
lib.c:2355://         struct Node* d = createNode(4, 0, 4);
lib.c:2356://         struct Edge e = createEdge(sour, dest, 0, 6, 0.0,
0, NULL);
lib.c:2357://         struct Edge edg = createEdge(s, d, 0, 3, 0.0, 0,
NULL);
lib.c:2358://        struct Edge e2 = createEdge(s, dest, 0, 2, 0.0, 0,
NULL);
lib.c:2359://        struct Edge e3 = createEdge(sour, d, 0, 1, 0.0, 0,
NULL);
lib.c:2360://         struct Edge* e_ptr = &(e);
lib.c:2361://         struct Edge* e_ptr2 = &(edg);
lib.c:2362://        //printEdge(e_ptr);
lib.c:2363://    //printEdge(e_ptr2);
lib.c:2364://         insertData(mp, e_ptr);
lib.c:2365://         int size = getListSize(mp->array);
lib.c:2366://         printf("size: %d \n",size);
lib.c:2367://         printHeap(mp);
lib.c:2368://    printf("------ \n");
lib.c:2369://         insertData(mp, e_ptr2);
lib.c:2370://         int size2 = getListSize(mp->array);
lib.c:2371://         printf("size: %d \n",size2);
lib.c:2372://    printHeap(mp);
lib.c:2373://     printf("------ \n");
lib.c:2374://         insertData(mp, &e2);
lib.c:2375://         int size3 = getListSize(mp->array);
lib.c:2376://         printf("size: %d \n",size3);
lib.c:2377://    printHeap(mp);
lib.c:2378://     printf("------ \n");
lib.c:2379://         insertData(mp, &e3);
lib.c:2380://         int size4 = getListSize(mp->array);
lib.c:2381://         printf("size: %d \n",size4);
lib.c:2382://    printHeap(mp);
lib.c:2383:
lib.c:2384://    printf("------ \n");
lib.c:2385://    struct Edge* data = getMinValue(mp);
lib.c:2386://    printEdge(data);
lib.c:2387:
lib.c:2388:// }
lib.c:2389:lib.h:1:#include <stdio.h>
lib.h:2:#include <stdlib.h>
lib.h:3:#include <stdint.h>
lib.h:4:#include <string.h>
lib.h:5:#include <stdbool.h>
```

```
lib.h:6:
lib.h:7:#ifndef _LIB_H_
lib.h:8:#define _LIB_H_
lib.h:9:
lib.h:10:void printBool(bool a);
lib.h:11:
lib.h:12:/************************************
lib.h:13:  Type Declaration
lib.h:14:************************************/
lib.h:15:
lib.h:16:#define INT 0
lib.h:17:#define FLOAT 1
lib.h:18:#define BOOL 2
lib.h:19:#define STRING 3
lib.h:20:#define NODE 4
lib.h:21:#define GRAPH 5
lib.h:22:#define LIST 6
lib.h:23:#define DICT 7
lib.h:24:#define EDGE 8
lib.h:25:
lib.h:26:#define RIGHT_LINK 0
lib.h:27:#define LEFT_LINK 1
lib.h:28:#define DOUBLE_LINK 2
lib.h:29:
lib.h:30:#define MAP_MISSING -3
lib.h:31:#define MAP_FULL -2
lib.h:32:#define MAP_OMEM -1
lib.h:33:#define MAP_OK 0
lib.h:34:
lib.h:35:#define INITIAL_SIZE (256)
lib.h:36:#define MAX_CHAIN_LENGTH (8)
lib.h:37:
lib.h:38:struct List {
lib.h:39:
lib.h:40:  int32_t index;
lib.h:41:  int32_t size;
lib.h:42:  void* *stored_data;
lib.h:43:  int32_t list_type;
lib.h:44:};
lib.h:45:
lib.h:46:
lib.h:47:struct Node {
lib.h:48:  int32_t id;
lib.h:49:  int32_t type;
lib.h:50:  int32_t a;
lib.h:51:  double b;
lib.h:52:  bool c;
lib.h:53:  char* d;
lib.h:54:  bool visited;
lib.h:55:};
lib.h:56:
lib.h:57:struct Edge {
lib.h:58:  struct Node* sour;
lib.h:59:  struct Node* dest;
lib.h:60:  int32_t type;
lib.h:61:  int32_t a;
lib.h:62:  double b;
lib.h:63:  bool c;
lib.h:64:  char* d;
lib.h:65:};
```

```
lib.h:66:
lib.h:67:struct Graph {
lib.h:68:  int32_t vn;
lib.h:69:  int32_t en;
lib.h:70:  int32_t vn_len;
lib.h:71:  int32_t en_len;
lib.h:72:  struct Node* root;
lib.h:73:  struct Node** nodes;
lib.h:74:  struct Edge* edges;
lib.h:75:};
lib.h:76:
lib.h:77:struct Queue {
lib.h:78:  struct List* lst;
lib.h:79:};
lib.h:80:
lib.h:81:struct minHeap {
lib.h:82:    //int size;
lib.h:83:    int32_t type;
lib.h:84:    struct List* array;
lib.h:85:};
lib.h:86:
lib.h:87:struct hashmap_element{
lib.h:88:  char* key;
lib.h:89:  int in_use;
lib.h:90:  void* data[2];
lib.h:91:};
lib.h:92:
lib.h:93:struct hashmap_map{
lib.h:94:  int table_size;
lib.h:95:  int size;
lib.h:96:  int32_t keytype;
lib.h:97:  int32_t valuetype;
lib.h:98:  struct hashmap_element *data;
lib.h:99:};
lib.h:100:
lib.h:101:/************************************
lib.h:102: Hashmap Methods
lib.h:103:************************************/
lib.h:104:
lib.h:105:typedef int (*Func)(void*, void*, void*);
lib.h:106:
lib.h:107:extern struct hashmap_map* hashmap_new(int32_t keytyp,int32_t
valuetyp);
lib.h:108:extern int hashmap_iterate(struct hashmap_map* m, Func f);
lib.h:109:extern int hashmap_print(struct hashmap_map* m);
lib.h:110:extern bool hashmap_haskey(struct hashmap_map* m,...);
lib.h:111:extern struct List* hashmap_keys(struct hashmap_map* m);
lib.h:112:extern struct hashmap_map* hashmap_put(struct hashmap_map*
m,...);
lib.h:113:extern void* hashmap_get(struct hashmap_map* m,...);
lib.h:114:extern struct hashmap_map* hashmap_remove(struct hashmap_map*
m,...);
lib.h:115:extern void hashmap_free(struct hashmap_map* m);
lib.h:116:
lib.h:117:extern int hashmap_length(struct hashmap_map* m);
lib.h:118:extern int32_t hashmap_keytype(struct hashmap_map* m);
lib.h:119:extern int32_t hashmap_valuetype(struct hashmap_map* m);
lib.h:120:
lib.h:121:/************************************
lib.h:122: Heap Methods
```

```
lib.h:123:***********************************/
lib.h:124:
lib.h:125:struct minHeap* initList(int32_t type);
lib.h:126:void swap(struct List* list, int index1, int index2);
lib.h:127:int eCompare(struct minHeap* hp, struct Edge* lchild, struct
Edge* rchild);
lib.h:128:void heapify(struct minHeap* hp, int size);
lib.h:129:void insertData(struct minHeap* hp, struct Edge* data);
lib.h:130:struct Edge* getMinValue(struct minHeap* hp);
lib.h:131:int32_t printHeap(struct minHeap* hp);
lib.h:132:void decreasePriority(struct minHeap* hp, struct Edge* e);
lib.h:133:
lib.h:134:/***********************************
lib.h:135: cast methods
lib.h:136:***********************************/
lib.h:137:
lib.h:138:int32_t VoidtoInt(void* add);
lib.h:139:double VoidtoFloat(void* add);
lib.h:140:bool VoidtoBool(void* add);
lib.h:141:char* VoidtoString(void* add);
lib.h:142:struct Node* VoidtoNode(void* add);
lib.h:143:struct Graph* VoidtoGraph(void* add);
lib.h:144:struct Edge* VoidtoEdge (void* add);
lib.h:145:
lib.h:146:void* InttoVoid(int32_t val);
lib.h:147:void* FloattoVoid(double val);
lib.h:148:void* BooltoVoid(bool val);
lib.h:149:void* StringtoVoid(char* val);
lib.h:150:void* NodetoVoid(struct Node* val);
lib.h:151:void* GraphtoVoid(struct Graph* val);
lib.h:152:void* EdgetoVoid(struct Edge* val);
lib.h:153:
lib.h:154:bool isInt(int32_t d);
lib.h:155:bool isFloat(int32_t d);
lib.h:156:bool isBool(int32_t d);
lib.h:157:bool isString(int32_t d);
lib.h:158:bool isNode(int32_t d);
lib.h:159:bool isEdge(int32_t d);
lib.h:160:bool isGraph(int32_t d);
lib.h:161:
lib.h:162:/***********************************
lib.h:163: List
lib.h:164:***********************************/
lib.h:165:
lib.h:166:int32_t rangeHelper(int size, int index);
lib.h:167:struct List* createList( int32_t type);
lib.h:168:struct List* addListHelper( struct List * list, void* addData);
lib.h:169:struct List* concatList(struct List* list1, struct List*
list2);
lib.h:170:struct List* pushList(struct List* list, ...);
lib.h:171:struct List* addList(struct List* list, ...);
lib.h:172:void* getList(struct List* list, int index);
lib.h:173:void* popList(struct List* list);
lib.h:174:int32_t setList(struct List* list, int index, ...);
lib.h:175:int getListSize(struct List* list);
lib.h:176:int32_t removeList(struct List* list, int index);
lib.h:177:int32_t printList(struct List * list);
lib.h:178:bool listContains(struct List *list, ...);
lib.h:179:struct List* removeData(struct List* list, ...);
lib.h:180:
```

```
lib.h:181:/***********************************
lib.h:182: Queue Methods
lib.h:183:***********************************/
lib.h:184:struct Queue* createQueue(int32_t type);
lib.h:185:struct Queue* pushBack(struct Queue* q, ...);
lib.h:186:void* popFront(struct Queue* q);
lib.h:187:int getQueueSize(struct Queue* q);
lib.h:188:int32_t printQueue(struct Queue* q);
lib.h:189:/***********************************
lib.h:190: Node Methods
lib.h:191:***********************************/
lib.h:192:
lib.h:193:struct Node* createNode(int32_t id, int32_t type, ...);
lib.h:194:
lib.h:195:void* nodeGetValue(struct Node* node, int32_t type);
lib.h:196:int32_t printNode(struct Node * node);
lib.h:197:bool setVisited(struct Node* node);
lib.h:198:bool isVisited(struct Node* node);
lib.h:199:/***********************************
lib.h:200: Edge Methods
lib.h:201:***********************************/
lib.h:202:
lib.h:203:struct Edge createEdge(
lib.h:204: struct Node* sour,
lib.h:205: struct Node* dest,
lib.h:206: int32_t type,
lib.h:207: int32_t a,
lib.h:208: double b,
lib.h:209: bool c,
lib.h:210: char* d
lib.h:211:);
lib.h:212:
lib.h:213:int32_t printEdge(struct Edge * edge);
lib.h:214:int32_t printEdgeValue(struct Edge * edge);
lib.h:215:void* edgeGetValue(struct Edge* edge, int32_t type);
lib.h:216:
lib.h:217:/***********************************
lib.h:218: Graph Methods
lib.h:219:***********************************/
lib.h:220:
lib.h:221:struct Graph* createGraph();
lib.h:222:struct Graph* copyGraph(struct Graph* a);
lib.h:223:struct Graph* mergeGraph(struct Graph* a, struct Graph* b);
lib.h:224:struct List* subGraph(struct Graph* a, struct Graph* b);
lib.h:225:struct Node* graphGetRoot(struct Graph* g);
lib.h:226:int32_t graphSetRoot(struct Graph* g, struct Node * root);
lib.h:227:int32_t graphAddList(struct Graph* g, int direction, struct
List * l, struct List * edges);
lib.h:228:int32_t graphAddNode(struct Graph* g, struct Node * node);
lib.h:229:struct List* graphGetAllNodes(struct Graph* g);
lib.h:230:struct List* graphRemoveNode(struct Graph* g, struct Node *
node);
lib.h:231:int32_t graphAddEdgeP( struct Graph* g, struct Node* sour,
struct Node* dest, int32_t type, ...);
lib.h:232:int32_t graphAddEdge(
lib.h:233: struct Graph* g,
lib.h:234: struct Node* sour,
lib.h:235: struct Node* dest,
lib.h:236: int32_t type,
lib.h:237: int32_t a,
```

```
lib.h:238: double b,
lib.h:239: bool c,
lib.h:240: char* d
lib.h:241:);
lib.h:242:bool graphEdgeExist(struct Graph* g, struct Node* sour, struct
Node* dest);
lib.h:243:struct Edge* graphGetEdge(struct Graph* g, struct Node* sour,
struct Node* dest);
lib.h:244:int32_t graphNumOfNodes(struct Graph* g);
lib.h:245:int32_t graphNumOfEdges(struct Graph* g);
lib.h:246:struct List* graphGetChildNodes(struct Graph* g, struct Node*
rt);
lib.h:247:int32_t printGraph(struct Graph* g);
lib.h:248:bool setAllUnvisited(struct Graph* g);
lib.h:249:struct List* dfs(struct Graph* g, struct Node* n);
lib.h:250:struct List* bfs(struct Graph* g, struct Node* n);
lib.h:251:bool containsNode(struct Graph* g, struct Node* n);
lib.h:252:struct List* dijkstra(struct Graph* g, struct Node* sour,
struct Node* dest);
lib.h:253:#endif /* #ifndef _LIB_H_ */
lib.h:254:Makefile:1:# Giraphe: Makefile
Makefile:2:
Makefile:3:default: giraphe.native
Makefile:4:
Makefile:5:all: clean giraphe.native
Makefile:6:
Makefile:7:test: clean giraphe.native
Makefile:8:cd tests; make
Makefile:9:
Makefile:10:.PHONY : giraphe.native
Makefile:11:giraphe.native :
Makefile:12:     ocamlbuild -use-ocamlfind -pkgs
llvm,llvm.analysis,llvm.linker,llvm.bitreader,llvm.irreader -cflags -
w,+a-4 \
Makefile:13:           giraphe.native;
Makefile:14:     clang -emit-llvm -o lib.bc -c lib.c -Wno-varargs
Makefile:15:
Makefile:16:OBJS = checker.cmx sast.cmx ast.cmx codegen.cmx parser.cmx
scanner.cmx semant.cmx giraphe.cmx parserize.cmx
Makefile:17:
Makefile:18:giraphe: $(OBJS)
Makefile:19:     ocamlfind ocamlopt -linkpkg -package llvm -package
llvm.analysis $(OBJS) -o giraphe
Makefile:20:
Makefile:21:scanner.ml : scanner.mll
Makefile:22:     ocamllex scanner.mll
Makefile:23:
Makefile:24:parser.ml parser.mli : parser.mly
Makefile:25:     ocamlyacc parser.mly
Makefile:26:
Makefile:27:%.cmo : %.ml
Makefile:28:     ocamlc -c $<
Makefile:29:
Makefile:30:%.cmi : %.mli
Makefile:31:     ocamlc -c $<
Makefile:32:
Makefile:33:%.cmx : %.ml
Makefile:34:     ocamlfind ocamlopt -c -package llvm $<
Makefile:35:
Makefile:36:.PHONY : clean
```

```
Makefile:37:clean :
Makefile:38:	ocamlbuild -clean
Makefile:39:	rm -f lib.bc
Makefile:40:	rm -f *.cmx *.cmi *.cmo *.cmx *.o
Makefile:41:	rm -f giraphe parser.ml parser.mli scanner.ml *.cmo
*.cmi
Makefile:42:
Makefile:43:ast.cmo :
Makefile:44:ast.cmx :
Makefile:45:sast.cmo : ast.cmo
Makefile:46:sast.cmx : ast.cmx
Makefile:47:codegen.cmo : sast.cmo
Makefile:48:codegen.cmx : sast.cmx
Makefile:49:giraphe.cmo : semant.cmo scanner.cmo parser.cmi codegen.cmo
ast.cmo
Makefile:50:giraphe.cmx : semant.cmx scanner.cmx parser.cmx codegen.cmx
ast.cmx
Makefile:51:parser.cmo : ast.cmo parser.cmi
Makefile:52:parser.cmx : ast.cmx parser.cmi
Makefile:53:scanner.cmo : parser.cmi
Makefile:54:scanner.cmx : parser.cmx
Makefile:55:semant.cmo : ast.cmo
Makefile:56:semant.cmx : ast.cmx
Makefile:57:parser.cmi : ast.cmo
Makefile:58:parserize.cmx : ast.cmo
Makefile:59:parserize_cast.ml:1:open Cast
parserize_cast.ml:2:open Printf
parserize_cast.ml:3:
parserize_cast.ml:4:(* Unary operators *)
parserize_cast.ml:5:let txt_of_unop = function
parserize_cast.ml:6:  | Not -> "Not"
parserize_cast.ml:7:  | Neg -> "Sub"
parserize_cast.ml:8:
parserize_cast.ml:9:(* Binary operators *)
parserize_cast.ml:10:let txt_of_binop = function
parserize_cast.ml:11:  (* Arithmetic *)
parserize_cast.ml:12:  | Add -> "Add"
parserize_cast.ml:13:  | Sub -> "Sub"
parserize_cast.ml:14:  | Mult -> "Mult"
parserize_cast.ml:15:  | Div -> "Div"
parserize_cast.ml:16:  | Mod -> "Mod"
parserize_cast.ml:17:  (* Boolean *)
parserize_cast.ml:18:  | Or -> "Or"
parserize_cast.ml:19:  | And -> "And"
parserize_cast.ml:20:  | Equal -> "Equal"
parserize_cast.ml:21:  | Neq -> "Neq"
parserize_cast.ml:22:  | Less -> "Less"
parserize_cast.ml:23:  | Leq -> "Leq"
parserize_cast.ml:24:  | Greater -> "Greater"
parserize_cast.ml:25:  | Geq -> "Geq"
parserize_cast.ml:26:
parserize_cast.ml:27:let txt_of_var_type = function
parserize_cast.ml:28:  | Void_t -> "void"
parserize_cast.ml:29:  | Null_t -> "null"
parserize_cast.ml:30:  | Int_t -> "int"
parserize_cast.ml:31:  | Float_t -> "float"
parserize_cast.ml:32:  | String_t -> "string"
parserize_cast.ml:33:  | Bool_t -> "bool"
parserize_cast.ml:34:  | Node_t -> "node"
parserize_cast.ml:35:  | Graph_t -> "graph"
```

```
parserize_cast.ml:36:  | Dict_Int_t -> "dict<int>"
parserize_cast.ml:37:  | Dict_Float_t -> "dict<float>"
parserize_cast.ml:38:  | Dict_String_t -> "dict<string>"
parserize_cast.ml:39:  | Dict_Node_t -> "dict<node>"
parserize_cast.ml:40:  | Dict_Graph_t -> "dict<graph>"
parserize_cast.ml:41:  | List_Int_t -> "list<int>"
parserize_cast.ml:42:  | List_Float_t -> "list<float>"
parserize_cast.ml:43:  | List_String_t -> "list<string>"
parserize_cast.ml:44:  | List_Node_t -> "list<node>"
parserize_cast.ml:45:  | List_Graph_t -> "list<graph>"
parserize_cast.ml:46:
parserize_cast.ml:47:let txt_of_formal = function
parserize_cast.ml:48:| Formal(vtype, name) -> sprintf "Formal(%s, %s)"
(txt_of_var_type vtype) name
parserize_cast.ml:49:
parserize_cast.ml:50:
parserize_cast.ml:51:let txt_of_formal_list formals =
parserize_cast.ml:52:  let rec aux acc = function
parserize_cast.ml:53:    | [] -> sprintf "%s" (String.concat ", "
(List.rev acc))
parserize_cast.ml:54:    | fml :: tl -> aux (txt_of_formal fml :: acc) tl
parserize_cast.ml:55:  in aux [] formals
parserize_cast.ml:56:
parserize_cast.ml:57:let txt_of_num = function
parserize_cast.ml:58:  | Num_Int(x) -> string_of_int x
parserize_cast.ml:59:  | Num_Float(x) -> string_of_float x
parserize_cast.ml:60:
parserize_cast.ml:61:(* Expressions *)
parserize_cast.ml:62:let rec txt_of_expr = function
parserize_cast.ml:63:  | Num_Lit(x) -> sprintf "Num_Lit(%s)" (txt_of_num
x)
parserize_cast.ml:64:  | Bool_lit(x) -> sprintf "Bool_lit(%s)"
(string_of_bool x)
parserize_cast.ml:65:  | String_Lit(x) -> sprintf "String_Lit(%s)" x
parserize_cast.ml:66:  | Null -> sprintf "Null"
parserize_cast.ml:67:  | Node(node_num, x) -> sprintf "Node(%s, %s)"
(string_of_int node_num) (txt_of_expr x)
parserize_cast.ml:68:  | Unop(op, e) -> sprintf "Unop(%s, %s)"
(txt_of_unop op) (txt_of_expr e)
parserize_cast.ml:69:  | Binop(e1, op, e2) -> sprintf "Binop(%s, %s, %s)"
parserize_cast.ml:70:      (txt_of_expr e1) (txt_of_binop op)
(txt_of_expr e2)
parserize_cast.ml:71:  | Graph_Link(e1, op1, e2, e3) -> sprintf
"Graph_Link(%s, %s, %s, WithEdge, %s)"
parserize_cast.ml:72:      (txt_of_expr e1) "RLink" (txt_of_expr e2)
(txt_of_expr e3)
parserize_cast.ml:73:  | Id(x) -> sprintf "Id(%s)" x
parserize_cast.ml:74:  | Assign(e1, e2) -> sprintf "Assign(%s, %s)" e1
(txt_of_expr e2)
parserize_cast.ml:75:  | Noexpr -> sprintf "Noexpression"
parserize_cast.ml:76:  | ListP(l) -> sprintf "List(%s)" (txt_of_list l)
parserize_cast.ml:77:  | DictP(d) -> sprintf "Dict(%s)" (txt_of_dict d)
parserize_cast.ml:78:  | Call(f, args) -> sprintf "Call(%s, [%s])" (f)
(txt_of_list args)
parserize_cast.ml:79:  | CallDefault(e, f, args) -> sprintf
"CallDefault(%s, %s, [%s])" (txt_of_expr e) f (txt_of_list args)
parserize_cast.ml:80:
parserize_cast.ml:81:(*Variable Declaration*)
parserize_cast.ml:82:and txt_of_var_decl = function
```

```
parserize_cast.ml:83:  | Local(var, name, e1) -> sprintf "Local(%s, %s,
%s)"
parserize_cast.ml:84:    (txt_of_var_type var) name (txt_of_expr e1)
parserize_cast.ml:85:
parserize_cast.ml:86:(* Lists *)
parserize_cast.ml:87:and txt_of_list = function
parserize_cast.ml:88:  | [] -> ""
parserize_cast.ml:89:  | [x] -> txt_of_expr x
parserize_cast.ml:90:  | _ as l -> String.concat ", " (List.map
txt_of_expr l)
parserize_cast.ml:91:
parserize_cast.ml:92:(* Dict *)
parserize_cast.ml:93:and txt_of_dict_key_value = function
parserize_cast.ml:94:  | (key, value) -> sprintf "key:%s,value:%s"
(txt_of_expr key) (txt_of_expr value)
parserize_cast.ml:95:
parserize_cast.ml:96:and txt_of_dict = function
parserize_cast.ml:97:  | [] -> ""
parserize_cast.ml:98:  | [x] -> txt_of_dict_key_value x
parserize_cast.ml:99:  | _ as d -> String.concat ", " (List.map
txt_of_dict_key_value d)
parserize_cast.ml:100:
parserize_cast.ml:101:(* Functions Declaration *)
parserize_cast.ml:102:and txt_of_func_decl f =
parserize_cast.ml:103:  sprintf "returnType(%s) name(%s) args(%s)
body{%s} locals{%s} parent(%s)"
parserize_cast.ml:104:    (txt_of_var_type f.returnType) f.name
(txt_of_formal_list f.args) (txt_of_stmts f.body) (txt_of_formal_list
f.locals) f.pname
parserize_cast.ml:105:
parserize_cast.ml:106:(* Statements *)
parserize_cast.ml:107:and txt_of_stmt = function
parserize_cast.ml:108:  | Expr(expr) -> sprintf "Expr(%s);" (txt_of_expr
expr)
parserize_cast.ml:109:  | Return(expr) -> sprintf "Return(%s);"
(txt_of_expr expr)
parserize_cast.ml:110:  | For(e1,e2,e3,s) -> sprintf "For(%s;%s;%s){%s}"
parserize_cast.ml:111:    (txt_of_expr e1) (txt_of_expr e2) (txt_of_expr
e3) (txt_of_stmts s)
parserize_cast.ml:112:  | If(e1,s1,s2) -> sprintf "If(%s){%s} Else{%s}"
parserize_cast.ml:113:    (txt_of_expr e1) (txt_of_stmts s1)
(txt_of_stmts s2)
parserize_cast.ml:114:  | While(e1, s) -> sprintf "While(%s){%s}"
parserize_cast.ml:115:    (txt_of_expr e1) (txt_of_stmts s)
parserize_cast.ml:116:
parserize_cast.ml:117:and txt_of_stmts = function
parserize_cast.ml:118:  | [] -> ""
parserize_cast.ml:119:  | [x] -> txt_of_stmt x
parserize_cast.ml:120:  | _ as s -> String.concat ", " (List.map
txt_of_stmt s)
parserize_cast.ml:121:
parserize_cast.ml:122:and txt_of_funcs funcs =
parserize_cast.ml:123:  let rec aux acc = function
parserize_cast.ml:124:      | [] -> sprintf "%s" (String.concat "\n"
(List.rev acc))
parserize_cast.ml:125:      | f :: tl -> aux (txt_of_func_decl f :: acc)
tl
parserize_cast.ml:126:  in aux [] funcs
parserize_cast.ml:127:
parserize_cast.ml:128:(* Program entry point *)
```

```
parserize_cast.ml:129:let _ =
parserize_cast.ml:130:  let lexbuf = Lexing.from_channel stdin in
parserize_cast.ml:131:  let program = Organizer.convert (Parser.program
Scanner.token lexbuf) in
parserize_cast.ml:132:  let result = txt_of_funcs program in
parserize_cast.ml:133:  print_endline result
parserize_cast.ml:134:parser.mly:1:%{ open Ast %}
parser.mly:2:
parser.mly:3:/* Arithmetic Operators */
parser.mly:4:%token PLUS MINUS TIMES DIVIDE MOD
parser.mly:5:
parser.mly:6:/* Separator */
parser.mly:7:%token SEMICOLON COMMA ASSIGN COLON DOT
parser.mly:8:
parser.mly:9:/* Relational Operators */
parser.mly:10:%token GT GEQ LT LEQ EQ NEQ
parser.mly:11:
parser.mly:12:/* Logical Operators & Keywords*/
parser.mly:13:%token AND OR NOT IF ELSE FOR WHILE BREAK CONTINUE RETURN
parser.mly:14:
parser.mly:15:/* Graph operator */
parser.mly:16:%token RIGHTLINK AMPERSAND  /* */
parser.mly:17:
parser.mly:18:/* Primary Type */
parser.mly:19:%token INT FLOAT STRING BOOL NODE EDGE GRAPH LIST DICT NULL
VOID
parser.mly:20:
parser.mly:21:/* Quote */
parser.mly:22:%token QUOTE
parser.mly:23:
parser.mly:24:/* Bracket */
parser.mly:25:%token LBRACKET RBRACKET LBRACE RBRACE LPAREN RPAREN
parser.mly:26:/* EOF */
parser.mly:27:%token EOF
parser.mly:28:
parser.mly:29:/* Identifiers */
parser.mly:30:%token <string> ID
parser.mly:31:
parser.mly:32:/* Literals */
parser.mly:33:%token <int> INT_LITERAL
parser.mly:34:%token <string> STRING_LITERAL
parser.mly:35:%token <float> FLOAT_LITERAL
parser.mly:36:%token <bool> BOOL_LITERAL
parser.mly:37:
parser.mly:38:/* Order */
parser.mly:39:%left COMMA
parser.mly:40:%right ASSIGN
parser.mly:41:%left AND OR
parser.mly:42:%left EQ NEQ
parser.mly:43:%left GT LT GEQ LEQ
parser.mly:44:%left PLUS MINUS
parser.mly:45:%left TIMES DIVIDE MOD
parser.mly:46:%right NOT
parser.mly:47:%right RIGHTLINK AMPERSAND  /* */
parser.mly:48:%right LPAREN
parser.mly:49:%left  RPAREN
parser.mly:50:%left COLON
parser.mly:51:%left DOT
parser.mly:52:
parser.mly:53:%start program
```

```
parser.mly:54:%type < Ast.program> program
parser.mly:55:
parser.mly:56:%%
parser.mly:57:
parser.mly:58:/* Program flow */
parser.mly:59:program:
parser.mly:60:| stmt_list EOF                          { List.rev $1 }
parser.mly:61:
parser.mly:62:stmt_list:
parser.mly:63:| /* nothing */                         { [] }
parser.mly:64:| stmt_list stmt                        { $2 :: $1 }
parser.mly:65:
parser.mly:66:stmt:
parser.mly:67:| expr SEMICOLON                       { Expr($1) }
parser.mly:68:| func_decl                             { Func($1) }
parser.mly:69:| RETURN SEMICOLON                  { Return(Noexpr) }
parser.mly:70:| RETURN expr SEMICOLON                { Return($2) }
parser.mly:71:| FOR LPAREN for_expr SEMICOLON expr SEMICOLON for_expr
RPAREN LBRACE stmt_list RBRACE
parser.mly:72:  {For($3, $5, $7, List.rev $10)}
parser.mly:73:| IF LPAREN expr RPAREN LBRACE stmt_list RBRACE ELSE LBRACE
stmt_list RBRACE
parser.mly:74:  {If($3,List.rev $6,List.rev $10)}
parser.mly:75:| IF LPAREN expr RPAREN LBRACE stmt_list RBRACE
parser.mly:76:  {If($3,List.rev $6,[])}
parser.mly:77:| WHILE LPAREN expr RPAREN LBRACE stmt_list RBRACE
parser.mly:78:  {While($3, List.rev $6)}
parser.mly:79:| var_decl SEMICOLON                    { Var_dec($1)}
parser.mly:80:
parser.mly:81:var_decl:
parser.mly:82:| var_type ID              { Local($1, $2, Noexpr) }
parser.mly:83:| var_type ID ASSIGN expr  { Local($1, $2, $4) }
parser.mly:84:
parser.mly:85:var_type:
parser.mly:86:| VOID                  {Void_t}
parser.mly:87:| NULL                  {Null_t}
parser.mly:88:| INT                                      {Int_t}
parser.mly:89:| FLOAT                                  {Float_t}
parser.mly:90:| STRING
      {String_t}
parser.mly:91:| BOOL {Bool_t}
parser.mly:92:| NODE {Node_t}
parser.mly:93:| GRAPH {Graph_t}
parser.mly:94:| DICT LT INT GT {Dict_Int_t}
parser.mly:95:| DICT LT FLOAT GT {Dict_Float_t}
parser.mly:96:| DICT LT STRING GT {Dict_String_t}
parser.mly:97:| DICT LT NODE GT {Dict_Node_t}
parser.mly:98:| DICT LT GRAPH GT {Dict_Graph_t}
parser.mly:99:| LIST LT INT GT {List_Int_t}
parser.mly:100:| LIST LT FLOAT GT {List_Float_t}
parser.mly:101:| LIST LT STRING GT {List_String_t}
parser.mly:102:| LIST LT BOOL GT {List_Bool_t}
parser.mly:103:| LIST LT NODE GT {List_Node_t}
parser.mly:104:| LIST LT GRAPH GT {List_Graph_t}
parser.mly:105:
parser.mly:106:formal_list:
parser.mly:107:| /* nothing */              { [] }
parser.mly:108:| formal                     { [$1] }
parser.mly:109:| formal_list COMMA formal { $3 :: $1 }
parser.mly:110:
```

```
parser.mly:111:
parser.mly:112:formal:
parser.mly:113:| var_type ID           { Formal($1, $2) }
parser.mly:114:
parser.mly:115:func_decl:
parser.mly:116:| var_type ID LPAREN formal_list RPAREN LBRACE stmt_list
RBRACE {
parser.mly:117:  {
parser.mly:118:    returnType = $1;
parser.mly:119:    name = $2;
parser.mly:120:    args = List.rev $4;
parser.mly:121:    body = List.rev $7;
parser.mly:122:  }
parser.mly:123:}
parser.mly:124:
parser.mly:125:
parser.mly:126:/* For loop decl*/
parser.mly:127:for_expr:
parser.mly:128:| /* nothing */                          { Noexpr }
parser.mly:129:| expr                                   { $1 }
parser.mly:130:
parser.mly:131:expr:
parser.mly:132:  literals {$1}
parser.mly:133:| NULL                             { Null }
parser.mly:134:| arith_ops                        { $1 }
parser.mly:135:| graph_ops                        { $1 }
parser.mly:136:| NODE LPAREN expr RPAREN { Node($3) }
parser.mly:137:| ID                                                  { Id($1)
}
parser.mly:138:| ID ASSIGN expr                                     {
Assign($1, $3) }
parser.mly:139:| LBRACKET list RBRACKET                 { ListP(List.rev
$2) }
parser.mly:140:| LBRACE  dict RBRACE    { DictP(List.rev $2) }
parser.mly:141:| LPAREN expr RPAREN     { $2 }
parser.mly:142:| ID LPAREN list RPAREN             { Call($1, List.rev
$3) }
parser.mly:143:| INT LPAREN list RPAREN              { Call("int",
List.rev $3) }
parser.mly:144:| FLOAT LPAREN list RPAREN              { Call("float",
List.rev $3) }
parser.mly:145:| BOOL LPAREN list RPAREN              { Call("bool",
List.rev $3) }
parser.mly:146:| STRING LPAREN list RPAREN              { Call("string",
List.rev $3) }
parser.mly:147:| expr DOT ID LPAREN list RPAREN   {CallDefault($1, $3,
List.rev $5)}
parser.mly:148:
parser.mly:149:
parser.mly:150:/* Lists */
parser.mly:151:list:
parser.mly:152:| /* nothing */                          { [] }
parser.mly:153:| expr                                   { [$1] }
parser.mly:154:| list COMMA expr                     { $3 :: $1 }
parser.mly:155:
parser.mly:156:dict_key_value:
parser.mly:157:| expr COLON expr { ($1, $3) }
parser.mly:158:
parser.mly:159:/* hashmap */
parser.mly:160:dict:
```

```
parser.mly:161:| /* nothing */                    { [] }
parser.mly:162:| dict_key_value
{ [$1] }
parser.mly:163:| dict COMMA dict_key_value              {$3 :: $1}
parser.mly:164:
parser.mly:165:arith_ops:
parser.mly:166:| expr PLUS         expr                          {
Binop($1, Add,   $3) }
parser.mly:167:| expr MINUS        expr                          {
Binop($1, Sub,   $3) }
parser.mly:168:| expr TIMES        expr                          {
Binop($1, Mult,  $3) }
parser.mly:169:| expr DIVIDE       expr                          {
Binop($1, Div,   $3) }
parser.mly:170:| expr EQ        expr                        { Binop($1,
Equal, $3) }
parser.mly:171:| expr NEQ     expr                           { Binop($1,
Neq,   $3) }
parser.mly:172:| expr LT      expr                           { Binop($1,
Less,  $3) }
parser.mly:173:| expr LEQ expr                           { Binop($1, Leq,
$3) }
parser.mly:174:| expr GT     expr                            { Binop($1,
Greater,  $3) }
parser.mly:175:| expr GEQ expr                           { Binop($1, Geq,
$3) }
parser.mly:176:| expr AND          expr                          {
Binop($1, And,   $3) }
parser.mly:177:| expr MOD          expr                          {
Binop($1, Mod,   $3) }
parser.mly:178:| expr OR     expr                { Binop($1, Or,    $3) }
parser.mly:179:| NOT  expr                                        {
Unop (Not,   $2) }
parser.mly:180:| MINUS expr                                       {
Unop (Neg, $2) }
parser.mly:181:
parser.mly:182:
parser.mly:183:graph_ops:
parser.mly:184:| expr RIGHTLINK expr                 { Graph_Link($1,
$3, Null) }
parser.mly:185:| expr RIGHTLINK expr AMPERSAND expr  { Graph_Link($1,
$5, $3) }
parser.mly:186:
parser.mly:187:
parser.mly:188:literals:
parser.mly:189:  INT_LITERAL         {Num_Lit( Num_Int($1) )}
parser.mly:190:| FLOAT_LITERAL       {Num_Lit( Num_Float($1) )}
parser.mly:191:| STRING_LITERAL     {String_Lit($1) }
parser.mly:192:| BOOL_LITERAL       {Bool_lit($1) }
parser.mly:193:sast.ml:1:(* Binary Operators *)
sast.ml:2:type binop =
sast.ml:3:  Add         (* + *)
sast.ml:4:| Sub         (* - *)
sast.ml:5:| Mult        (* * *)
sast.ml:6:| Div         (* / *)
sast.ml:7:| Mod         (* % *)
sast.ml:8:| Equal       (* == *)
sast.ml:9:| Neq         (* != *)
sast.ml:10:| Less        (* < *)
sast.ml:11:| Leq         (* <= *)
```

```
sast.ml:12:| Greater     (* > *)
sast.ml:13:| Geq         (* >= *)
sast.ml:14:| And         (* and *)
sast.ml:15:| Or          (* or *)
sast.ml:16:
sast.ml:17:(* Unary Operators *)
sast.ml:18:type unop =
sast.ml:19:  Neg         (* - *)
sast.ml:20:| Not         (* not *)
sast.ml:21:
sast.ml:22:(* Numbers int | float *)
sast.ml:23:type num =
sast.ml:24:  Num_Int of int
sast.ml:25:| Num_Float of float
sast.ml:26:
sast.ml:27:(* Variable Type *)
sast.ml:28:type var_type =
sast.ml:29:  Int_t                   (* int *)
sast.ml:30:| Float_t                 (* float *)
sast.ml:31:| String_t                (* string *)
sast.ml:32:| Bool_t
sast.ml:33:| Node_t
sast.ml:34:| Edge_t
sast.ml:35:| Graph_t
sast.ml:36:| Dict_Int_t
sast.ml:37:| Dict_Float_t
sast.ml:38:| Dict_String_t
sast.ml:39:| Dict_Node_t
sast.ml:40:| Dict_Graph_t
sast.ml:41:| List_Int_t
sast.ml:42:| List_Float_t
sast.ml:43:| List_Bool_t
sast.ml:44:| List_String_t
sast.ml:45:| List_Node_t
sast.ml:46:| List_Graph_t
sast.ml:47:| List_Null_t
sast.ml:48:| Void_t
sast.ml:49:| Null_t
sast.ml:50:
sast.ml:51:(* Type Declaration *)
sast.ml:52:type formal =
sast.ml:53:| Formal of var_type * string   (* int aNum *)
sast.ml:54:
sast.ml:55:type expr =
sast.ml:56:    Num_Lit of num
sast.ml:57:|   Null
sast.ml:58:|   String_Lit of string
sast.ml:59:|   Bool_lit of bool
sast.ml:60:|   Node of int * expr
sast.ml:61:|   Graph_Link of expr * expr * expr
sast.ml:62:|   Binop of expr * binop * expr
sast.ml:63:|   Unop of unop * expr
sast.ml:64:|   Id of string
sast.ml:65:|   Assign of string * expr
sast.ml:66:|   Noexpr
sast.ml:67:|   ListP of expr list
sast.ml:68:|   DictP of (expr * expr) list
sast.ml:69:|   Call of string * expr list    (* function call *)
sast.ml:70:|   CallDefault of expr * string * expr list
sast.ml:71:
```

```
sast.ml:72:
sast.ml:73:
sast.ml:74:type var_decl =
sast.ml:75:| Local of var_type * string * expr
sast.ml:76:
sast.ml:77:(* Statements *)
sast.ml:78:type stmt =
sast.ml:79:  Expr of expr
sast.ml:80:| Return of expr
sast.ml:81:| For of expr * expr * expr * stmt list
sast.ml:82:| If of expr * stmt list * stmt list
sast.ml:83:| While of expr * stmt list
sast.ml:84:
sast.ml:85:
sast.ml:86:(* Function Declaration *)
sast.ml:87:and func_decl = {
sast.ml:88:  returnType: var_type;
sast.ml:89:  name: string;
sast.ml:90:  args: formal list;
sast.ml:91:  body: stmt list;
sast.ml:92:  locals: formal list;
sast.ml:93:  pname: string; (* parent func name *)
sast.ml:94:}
sast.ml:95:
sast.ml:96:
sast.ml:97:(* Program entry point *)
sast.ml:98:type program = func_decl list
sast.ml:99:scanner.mll:1:{
scanner.mll:2:  open Parser
scanner.mll:3:  let unescape s =
scanner.mll:4:        Scanf.sscanf ("\"" ^ s ^ "\"") "%S%!" (fun x -> x)
scanner.mll:5:}
scanner.mll:6:let digit = ['0'-'9']
scanner.mll:7:let letter = ['a'-'z' 'A'-'Z']
scanner.mll:8:let variable = letter (letter | digit | '_') *
scanner.mll:9:let escape = '\\' ['\\' ''' '"' 'n' 'r' 't']
scanner.mll:10:let ascii = ([' '-'!' '#'-'[' ']'-'~'])
scanner.mll:11:rule token =
scanner.mll:12:parse [' ' '\t' '\r' '\n'] { token lexbuf }
scanner.mll:13:(* comment *)
scanner.mll:14:| "/*" { comment lexbuf }
scanner.mll:15:| "//" { sgcomment lexbuf }
scanner.mll:16:(* calculation *)
scanner.mll:17:| '+' { PLUS }
scanner.mll:18:| '-' { MINUS }
scanner.mll:19:| '*' { TIMES }
scanner.mll:20:| '/' { DIVIDE }
scanner.mll:21:| '%' { MOD }
scanner.mll:22:(* separator *)
scanner.mll:23:| ';' { SEMICOLON }
scanner.mll:24:| ',' { COMMA }
scanner.mll:25:| '=' { ASSIGN }
scanner.mll:26:| ':' { COLON }
scanner.mll:27:| '.' { DOT }
scanner.mll:28:(* logical operation *)
scanner.mll:29:| "&&" { AND }
scanner.mll:30:| "||" { OR }
scanner.mll:31:| "!" { NOT }
scanner.mll:32:| "if" { IF }
scanner.mll:33:| "else" { ELSE }
```

```
scanner.mll:34:| "for" { FOR }
scanner.mll:35:| "while" { WHILE}
scanner.mll:36:| "break" { BREAK }
scanner.mll:37:| "continue" { CONTINUE }
scanner.mll:38:| "return" {RETURN}
scanner.mll:39:(* comparator *)
scanner.mll:40:| '>' { GT }
scanner.mll:41:| ">=" { GEQ }
scanner.mll:42:| '<' { LT }
scanner.mll:43:| "<=" { LEQ }
scanner.mll:44:| "==" { EQ }
scanner.mll:45:| "!=" { NEQ }
scanner.mll:46:(* graph operator two *) (* *)
scanner.mll:47:| "->" { RIGHTLINK }
scanner.mll:48:| '$' { AMPERSAND }
scanner.mll:49:(* primary type *)
scanner.mll:50:| "void" { VOID }
scanner.mll:51:| "int" { INT }
scanner.mll:52:| "float" { FLOAT }
scanner.mll:53:| "string" { STRING }
scanner.mll:54:| "bool" { BOOL }
scanner.mll:55:| "node" { NODE }
scanner.mll:56:| "graph" { GRAPH }
scanner.mll:57:| "edge" { EDGE }
scanner.mll:58:| "list" { LIST }
scanner.mll:59:| "hashmap" { DICT }
scanner.mll:60:| "null" { NULL }
scanner.mll:61:(* integer and float *)
scanner.mll:62:| digit+ as lit { INT_LITERAL(int_of_string lit) }
scanner.mll:63:| digit+'.'digit* as lit { FLOAT_LITERAL(float_of_string
lit) }
scanner.mll:64:| '"' ((ascii | escape)* as lit) '"' {
STRING_LITERAL(unescape lit) }
scanner.mll:65:(* quote *)
scanner.mll:66:| '"'  { QUOTE }
scanner.mll:67:(* boolean operation *)
scanner.mll:68:| "true" | "false" as boollit {
BOOL_LITERAL(bool_of_string boollit)}
scanner.mll:69:(* bracket *)
scanner.mll:70:| '[' { LBRACKET }
scanner.mll:71:| ']' { RBRACKET }
scanner.mll:72:| '{' { LBRACE }
scanner.mll:73:| '}' { RBRACE }
scanner.mll:74:| '(' { LPAREN }
scanner.mll:75:| ')' { RPAREN }
scanner.mll:76:(* id *)
scanner.mll:77:| variable as id { ID(id) }
scanner.mll:78:| eof { EOF }
scanner.mll:79:
scanner.mll:80:and comment = parse
scanner.mll:81:|   "*/" {token lexbuf}
scanner.mll:82:| _ {comment lexbuf}
scanner.mll:83:
scanner.mll:84:and sgcomment = parse
scanner.mll:85:  | '\n'                                    { token lexbuf }
scanner.mll:86:  | eof                                     { EOF }
scanner.mll:87:  | _                                       { sgcomment
lexbuf }
scanner.mll:88:scannerprint.mll:1:{ open Printf }
scannerprint.mll:2:
```

```
scannerprint.mll:3:rule token = parse
scannerprint.mll:4:  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace
*)
scannerprint.mll:5:| "/*"     { comment lexbuf }           (* Comments *)
scannerprint.mll:6:| '('      { print_string "LPAREN " }
scannerprint.mll:7:| ')'      { print_string "RPAREN " }
scannerprint.mll:8:| '{'      { print_string "LBRACE " }
scannerprint.mll:9:| '}'      { print_string "RBRACE " }
scannerprint.mll:10:| ';'      { print_string "SEMI " }
scannerprint.mll:11:| "int"    { print_string "INT " }
scannerprint.mll:12:| ['0'-'9']+ { print_string "LITERAL " }
scannerprint.mll:13:| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* {
print_string "ID " }
scannerprint.mll:14:
scannerprint.mll:15:and comment = parse
scannerprint.mll:16:  "*/" { token lexbuf }
scannerprint.mll:17:| _    { comment lexbuf }
scannerprint.mll:18:
scannerprint.mll:19:{
scannerprint.mll:20:  let main () =
scannerprint.mll:21:    let lexbuf = Lexing.from_channel stdin in
scannerprint.mll:22:    try
scannerprint.mll:23:        while true do
scannerprint.mll:24:            ignore (token lexbuf)
scannerprint.mll:25:        done
scannerprint.mll:26:    with _ -> print_string "invalid_token\n"
scannerprint.mll:27:  let _ = Printexc.print main ()
scannerprint.mll:28:
scannerprint.mll:29:}
scannerprint.mll:30:semant.ml:1:open Sast
semant.ml:2:open Printf
semant.ml:3:
semant.ml:4:
semant.ml:5:module StringMap = Map.Make(String)
semant.ml:6:
semant.ml:7:(* Pretty-printing functions *)
semant.ml:8:let string_of_typ = function
semant.ml:9:    Int_t -> "int"
semant.ml:10:  | Float_t -> "float"
semant.ml:11:  | String_t -> "string"
semant.ml:12:  | Bool_t -> "bool"
semant.ml:13:  | Node_t -> "node"
semant.ml:14:  | Graph_t -> "graph"
semant.ml:15:  | List_Int_t -> "list<int>"
semant.ml:16:  | List_Float_t -> "list<float>"
semant.ml:17:  | List_String_t -> "list<string>"
semant.ml:18:  | List_Node_t -> "list<node>"
semant.ml:19:  | List_Graph_t -> "list<graph>"
semant.ml:20:  | List_Bool_t -> "list<bool>"
semant.ml:21:  | List_Null_t -> "list<null>"
semant.ml:22:  | Dict_Int_t -> "dict<int>"
semant.ml:23:  | Dict_Float_t -> "dict<float>"
semant.ml:24:  | Dict_String_t -> "dict<string>"
semant.ml:25:  | Dict_Node_t -> "dict<node>"
semant.ml:26:  | Dict_Graph_t -> "dict<graph>"
semant.ml:27:  | Void_t -> "void"
semant.ml:28:  | Null_t -> "null"
semant.ml:29:  | Edge_t -> "edge"
semant.ml:30:
semant.ml:31:let string_of_op = function
```

```
semant.ml:32:    Add -> "+"
semant.ml:33:  | Sub -> "-"
semant.ml:34:  | Mult -> "*"
semant.ml:35:  | Div -> "/"
semant.ml:36:  | Mod -> "%"
semant.ml:37:  | Equal -> "=="
semant.ml:38:  | Neq -> "!="
semant.ml:39:  | Less -> "<"
semant.ml:40:  | Leq -> "<="
semant.ml:41:  | Greater -> ">"
semant.ml:42:  | Geq -> ">="
semant.ml:43:  | And -> "and"
semant.ml:44:  | Or -> "or"
semant.ml:45:
semant.ml:46:
semant.ml:47:let string_of_uop = function
semant.ml:48:    Neg -> "-"
semant.ml:49:  | Not -> "not"
semant.ml:50:
semant.ml:51:let rec string_of_expr = function
semant.ml:52:    Num_Lit(Num_Int(l)) -> string_of_int l
semant.ml:53:  | Num_Lit(Num_Float(l)) -> string_of_float l
semant.ml:54:  | Null -> "null"
semant.ml:55:  | String_Lit(l) -> l
semant.ml:56:  | Bool_lit(true) -> "true"
semant.ml:57:  | Bool_lit(false) -> "false"
semant.ml:58:  | Node(_, e) -> "node(" ^ string_of_expr e ^ ")"
semant.ml:59:  | Graph_Link(e1, e2, e3) ->
semant.ml:60:    "graph_link(" ^ string_of_expr e1 ^ " " ^ "->" ^ " " ^
string_of_expr e2 ^ " " ^ string_of_expr e3 ^ ")"
semant.ml:61:  | Binop(e1, o, e2) ->
semant.ml:62:    string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^
string_of_expr e2
semant.ml:63:  | Unop(o, e) -> string_of_uop o ^ " " ^ string_of_expr e
semant.ml:64:  | Id(s) -> s
semant.ml:65:  | Assign(v, e) -> v ^ " = " ^ string_of_expr e
semant.ml:66:  | Noexpr -> ""
semant.ml:67:  | ListP(_) -> "list"
semant.ml:68:  | DictP(_) -> "dict"
semant.ml:69:  | Call(n, _) -> "function call " ^ n
semant.ml:70:  | CallDefault(e, n, _) -> "function call " ^
string_of_expr e ^ "." ^ n
semant.ml:71:
semant.ml:72:
semant.ml:73:exception SemanticError of string
semant.ml:74:
semant.ml:75:(* error message functions *)
semant.ml:76:let undeclared_function_error name =
semant.ml:77:  let msg = sprintf "Uh.. undeclared function %s :(" name in
semant.ml:78:  raise (SemanticError msg)
semant.ml:79:
semant.ml:80:let duplicate_formal_decl_error func name =
semant.ml:81:  let msg = sprintf "Uh.. duplicate formal %s in %s :(" name
func.name in
semant.ml:82:  raise (SemanticError msg)
semant.ml:83:
semant.ml:84:let duplicate_local_decl_error func name =
semant.ml:85:  let msg = sprintf "Uh.. duplicate local %s in %s :(" name
func.name in
semant.ml:86:  raise (SemanticError msg)
```

```
semant.ml:87:
semant.ml:88:let unidentify_error name =
semant.ml:89:  let msg = sprintf "Uh..undeclared identifier %s :(" name
in
semant.ml:90:  raise (SemanticError msg)
semant.ml:91:
semant.ml:92:let assign_error lvaluet rvaluet ex =
semant.ml:93:  let msg = sprintf "Uh..oh! Assigning %s = %s in %s is
illegal :(" lvaluet rvaluet ex in
semant.ml:94:  raise (SemanticError msg)
semant.ml:95:
semant.ml:96:let binary_error typ1 typ2 op ex =
semant.ml:97:  let msg = sprintf "Uh..oh! Binary Operator %s %s %s in %s
is illegal :(" typ1 op typ2 ex in
semant.ml:98:  raise (SemanticError msg)
semant.ml:99:
semant.ml:100:let unary_error typ op ex =
semant.ml:101:  let msg = sprintf "Uh..oh! Unary Operator %s %s in %s is
illegal :(" op typ ex in
semant.ml:102:  raise (SemanticError msg)
semant.ml:103:
semant.ml:104:let listType_error typ =
semant.ml:105:  let msg = sprintf "Uh.. This is a invalid list type: %s
:(" typ in
semant.ml:106:  raise (SemanticError msg)
semant.ml:107:
semant.ml:108:let invaid_dict_type_error typ =
semant.ml:109:  let msg = sprintf "Uh.. invalid dict type: %s :(" typ in
semant.ml:110:  raise (SemanticError msg)
semant.ml:111:
semant.ml:112:let list_element_type_inconsistent_error typ1 typ2 =
semant.ml:113:  let msg = sprintf "Uh.. List elements have different
types: %s and %s :(" typ1 typ2 in
semant.ml:114:  raise (SemanticError msg)
semant.ml:115:
semant.ml:116:let inconsistent_dict_element_type_error typ1 typ2 =
semant.ml:117:  let msg = sprintf "Uh.. dict can not contain objects of
different types: %s and %s :(" typ1 typ2 in
semant.ml:118:  raise (SemanticError msg)
semant.ml:119:
semant.ml:120:let unmatched_functionArg_error name =
semant.ml:121:  let msg = sprintf "Uh.. This args length does not match
in function call: %s :(" name in
semant.ml:122:  raise (SemanticError msg)
semant.ml:123:
semant.ml:124:let functionCompariable_error typ1 typ2 =
semant.ml:125:  let msg = sprintf "Uh..oh! incompatible argument type %s,
but %s is expected :(" typ1 typ2 in
semant.ml:126:  raise (SemanticError msg)
semant.ml:127:
semant.ml:128:let after_return_error _ =
semant.ml:129:  let msg = sprintf "Hmm? Something follow with a return
:(" in
semant.ml:130:  raise (SemanticError msg)
semant.ml:131:
semant.ml:132:let redefine_error _ =
semant.ml:133:  let msg = sprintf "Uh.. function may not be defined" in
semant.ml:134:  raise (SemanticError msg)
semant.ml:135:
semant.ml:136:let duplicate_func_error name =
```

```
semant.ml:137:  let msg = sprintf "Uh.. duplicate function declaration:
%s :(" name in
semant.ml:138:  raise (SemanticError msg)
semant.ml:139:
semant.ml:140:let unsupport_operation_error typ name =
semant.ml:141:  let msg = sprintf "Uh.. unsupport operation on type %s:
%s :(" typ name in
semant.ml:142:  raise (SemanticError msg)
semant.ml:143:
semant.ml:144:let listSize_error ex =
semant.ml:145:  let msg = sprintf "Uh.. list size method do not take
arguments: %s :(" ex in
semant.ml:146:  raise (SemanticError msg)
semant.ml:147:
semant.ml:148:let listPop_error ex =
semant.ml:149:  let msg = sprintf "Uh.. list pop method do not take
arguments: %s :(" ex in
semant.ml:150:  raise (SemanticError msg)
semant.ml:151:
semant.ml:152:let listGet_error ex =
semant.ml:153:  let msg = sprintf "Uh..list get method should only take
one argument of type int: %s :(" ex in
semant.ml:154:  raise (SemanticError msg)
semant.ml:155:
semant.ml:156:let listAdd_error typ ex =
semant.ml:157:  let msg = sprintf "Uh.. list add method should only take
one argument of type %s: %s :(" typ ex in
semant.ml:158:  raise (SemanticError msg)
semant.ml:159:
semant.ml:160:let listPush_error typ ex =
semant.ml:161:  let msg = sprintf "Uh.. list push method should only take
one argument of type %s: %s :(" typ ex in
semant.ml:162:  raise (SemanticError msg)
semant.ml:163:
semant.ml:164:let listRemove_error ex =
semant.ml:165:  let msg = sprintf "Uh.. list remove method should only
take one argument of type int: %s :(" ex in
semant.ml:166:  raise (SemanticError msg)
semant.ml:167:
semant.ml:168:let listSet_error typ ex =
semant.ml:169:  let msg = sprintf "Uh.. list set method should only take
two argument of type int and %s: %s :(" typ ex in
semant.ml:170:  raise (SemanticError msg)
semant.ml:171:
semant.ml:172:let empty_list_error ex =
semant.ml:173:  let msg = sprintf "Uh..oh! This empty list declaration:
%s is invalid :(" ex in
semant.ml:174:  raise (SemanticError msg)
semant.ml:175:
semant.ml:176:let invalid_dict_get_method_error ex =
semant.ml:177:  let msg = sprintf "Uh.. dict get method should only take
one argument of type int, string or node: %s :(" ex in
semant.ml:178:  raise (SemanticError msg)
semant.ml:179:
semant.ml:180:let invalid_dict_remove_method_error ex =
semant.ml:181:  let msg = sprintf "Uh.. dict remove method should only
take one argument of type int, string or node: %s :(" ex in
semant.ml:182:  raise (SemanticError msg)
semant.ml:183:
semant.ml:184:let invalid_dict_size_method_error ex =
```

```
semant.ml:185:  let msg = sprintf "Uh.. dict size method do not take
arguments: %s :(" ex in
semant.ml:186:  raise (SemanticError msg)
semant.ml:187:
semant.ml:188:let invalid_dict_keys_method_error ex =
semant.ml:189:  let msg = sprintf "Uh.. dict keys method do not take
arguments: %s :(" ex in
semant.ml:190:  raise (SemanticError msg)
semant.ml:191:
semant.ml:192:let invalid_dict_put_method_error typ ex =
semant.ml:193:  let msg = sprintf "Uh.. dict put method should only take
two argument of type (int, string or node) and %s: %s :(" typ ex in
semant.ml:194:  raise (SemanticError msg)
semant.ml:195:
semant.ml:196:let invalid_empty_dict_decl_error ex =
semant.ml:197:  let msg = sprintf "Uh.. This is invalid empty dict
declaration: %s :(" ex in
semant.ml:198:  raise (SemanticError msg)
semant.ml:199:
semant.ml:200:let graphRoot_error ex =
semant.ml:201:  let msg = sprintf "Uh.. root method do not take
arguments: %s" ex in
semant.ml:202:  raise (SemanticError msg)
semant.ml:203:
semant.ml:204:let graphSize_error ex =
semant.ml:205:  let msg = sprintf "Uh..size method do not take arguments:
%s" ex in
semant.ml:206:  raise (SemanticError msg)
semant.ml:207:
semant.ml:208:let graphNode_error ex =
semant.ml:209:  let msg = sprintf "Uh.. node method do not take
arguments: %s" ex in
semant.ml:210:  raise (SemanticError msg)
semant.ml:211:
semant.ml:212:let graphEdge_method_error ex =
semant.ml:213:  let msg = sprintf "Uh..edge method do not accept
arguments: %s" ex in
semant.ml:214:  raise (SemanticError msg)
semant.ml:215:
semant.ml:216:let graphLink_error ex =
semant.ml:217:  let msg = sprintf "Uh.. left side of graph should be node
type: %s" ex in
semant.ml:218:  raise (SemanticError msg)
semant.ml:219:
semant.ml:220:let graphEdge_error ex =
semant.ml:221:  let msg = sprintf "Uh.. There is a invalid graph edge at:
%s" ex in
semant.ml:222:  raise (SemanticError msg)
semant.ml:223:
semant.ml:224:let invalid_graph_list_node_at_error ex =
semant.ml:225:  let msg = sprintf "Uh.. invalid graph list node at: %s
:(" ex in
semant.ml:226:  raise (SemanticError msg)
semant.ml:227:
semant.ml:228:let unsupport_graph_list_edge_at_error ex =
semant.ml:229:  let msg = sprintf "Uh.. unsupport graph list edge at: %s
:(" ex in
semant.ml:230:  raise (SemanticError msg)
semant.ml:231:
semant.ml:232:let invalid_graph_root_as_error ex =
```

```
semant.ml:233:  let msg = sprintf "Uh.. invalid graph root as: %s :(" ex
in
semant.ml:234:  raise (SemanticError msg)
semant.ml:235:
semant.ml:236:let wrongFuncReturn_error typ1 typ2 =
semant.ml:237:  let msg = sprintf "Ops, Wrong function return type: %s,
it expect %s" typ1 typ2 in
semant.ml:238:  raise (SemanticError msg)
semant.ml:239:
semant.ml:240:
semant.ml:241:let  match_list_type = function
semant.ml:242:    Int_t -> List_Int_t
semant.ml:243:  | Float_t -> List_Float_t
semant.ml:244:  | String_t -> List_String_t
semant.ml:245:  | Node_t -> List_Node_t
semant.ml:246:  | Graph_t -> List_Graph_t
semant.ml:247:  | Bool_t -> List_Bool_t
semant.ml:248:  | _ as t-> listType_error (string_of_typ t)
semant.ml:249:
semant.ml:250:let  reverse_match_list_type = function
semant.ml:251:    List_Int_t -> Int_t
semant.ml:252:  | List_Float_t -> Float_t
semant.ml:253:  | List_String_t -> String_t
semant.ml:254:  | List_Node_t -> Node_t
semant.ml:255:  | List_Graph_t -> Graph_t
semant.ml:256:  | List_Bool_t -> Bool_t
semant.ml:257:  | _ as t-> listType_error (string_of_typ t)
semant.ml:258:
semant.ml:259:let  match_dict_type = function
semant.ml:260:    Int_t -> Dict_Int_t
semant.ml:261:  | Float_t -> Dict_Float_t
semant.ml:262:  | String_t -> Dict_String_t
semant.ml:263:  | Node_t -> Dict_Node_t
semant.ml:264:  | Graph_t -> Dict_Graph_t
semant.ml:265:  | _ as t-> invaid_dict_type_error (string_of_typ t)
semant.ml:266:
semant.ml:267:let  reverse_match_dict_type = function
semant.ml:268:    Dict_Int_t -> Int_t
semant.ml:269:  | Dict_Float_t ->  Float_t
semant.ml:270:  | Dict_String_t -> String_t
semant.ml:271:  | Dict_Node_t -> Node_t
semant.ml:272:  | Dict_Graph_t -> Graph_t
semant.ml:273:  | _ as t-> invaid_dict_type_error (string_of_typ t)
semant.ml:274:
semant.ml:275:
semant.ml:276:(* list check helper function  *)
semant.ml:277:let check_valid_list_type typ =
semant.ml:278:  if typ = List_Int_t || typ = List_Float_t || typ =
List_String_t || typ = List_Node_t || typ = List_Graph_t || typ =
List_Bool_t then typ
semant.ml:279:  else listType_error (string_of_typ typ)
semant.ml:280:
semant.ml:281:let check_list_size_method ex es =
semant.ml:282:  match es with
semant.ml:283:    [] -> ()
semant.ml:284:  | _ -> listSize_error (string_of_expr ex)
semant.ml:285:
semant.ml:286:let check_list_pop_method ex es =
semant.ml:287:  match es with
semant.ml:288:    [] -> ()
```

```
semant.ml:289:  | _ -> listPop_error (string_of_expr ex)
semant.ml:290:
semant.ml:291:(* dict check helper function  *)
semant.ml:292:let check_valid_dict_type typ =
semant.ml:293:  if typ = Dict_Int_t || typ = Dict_Float_t || typ =
Dict_String_t || typ = Dict_Node_t || typ = Dict_Graph_t then typ
semant.ml:294:  else invaid_dict_type_error (string_of_typ typ)
semant.ml:295:
semant.ml:296:let check_dict_size_method ex es =
semant.ml:297:  match es with
semant.ml:298:    [] -> ()
semant.ml:299:  | _ -> invalid_dict_size_method_error (string_of_expr ex)
semant.ml:300:
semant.ml:301:let check_dict_keys_method ex es =
semant.ml:302:  match es with
semant.ml:303:    [] -> ()
semant.ml:304:  | _ -> invalid_dict_keys_method_error (string_of_expr ex)
semant.ml:305:
semant.ml:306:(* graph check helper function  *)
semant.ml:307:let check_graph_root_method ex es =
semant.ml:308:  match es with
semant.ml:309:    [] -> ()
semant.ml:310:  | _ -> graphRoot_error(string_of_expr ex)
semant.ml:311:
semant.ml:312:let check_graph_size_method ex es =
semant.ml:313:  match es with
semant.ml:314:    [] -> ()
semant.ml:315:  | _ -> graphSize_error(string_of_expr ex)
semant.ml:316:
semant.ml:317:let check_graph_nodes_method ex es =
semant.ml:318:  match es with
semant.ml:319:    [] -> ()
semant.ml:320:  | _ -> graphNode_error (string_of_expr ex)
semant.ml:321:
semant.ml:322:let check_graph_edges_method ex es =
semant.ml:323:  match es with
semant.ml:324:    [] -> ()
semant.ml:325:  | _ -> graphEdge_method_error (string_of_expr ex)
semant.ml:326:
semant.ml:327:let check_graph_list_node_at ex lt rt =
semant.ml:328:  if lt = Graph_t && rt = Node_t then () else
semant.ml:329:    invalid_graph_list_node_at_error (string_of_expr ex)
semant.ml:330:
semant.ml:331:let check_graph_root_as ex lt rt =
semant.ml:332:  if lt = Graph_t && rt = Node_t then () else
semant.ml:333:    invalid_graph_root_as_error (string_of_expr ex)
semant.ml:334:
semant.ml:335:let check_return_type func typ =
semant.ml:336:  let lvaluet = func.returnType and rvaluet = typ in
semant.ml:337:  match lvaluet with
semant.ml:338:    Float_t when rvaluet = Int_t -> ()
semant.ml:339:  | String_t when rvaluet = Null_t -> ()
semant.ml:340:  | Node_t when rvaluet = Null_t -> ()
semant.ml:341:  | Graph_t when rvaluet = Null_t -> ()
semant.ml:342:  | List_Int_t | List_String_t | List_Float_t | List_Node_t
| List_Graph_t | List_Bool_t when rvaluet = Null_t -> ()
semant.ml:343:  | Dict_Int_t | Dict_String_t | Dict_Float_t | Dict_Node_t
| Dict_Graph_t when rvaluet = Null_t -> ()
semant.ml:344:  (* for dict.keys() *)
```

```
semant.ml:345:  | List_Int_t | List_String_t | List_Node_t when rvaluet =
List_Null_t -> ()
semant.ml:346:  | _ -> if lvaluet == rvaluet then () else
semant.ml:347:      wrongFuncReturn_error (string_of_typ rvaluet)
(string_of_typ lvaluet)
semant.ml:348:
semant.ml:349:
semant.ml:350:(* get function obj from func_map, if not found, raise
error *)
semant.ml:351:let get_func_obj name func_map =
semant.ml:352:  try StringMap.find name func_map
semant.ml:353:  with Not_found -> undeclared_function_error name
semant.ml:354:
semant.ml:355:
semant.ml:356:(* Raise an exception if the given list has a duplicate *)
semant.ml:357:let report_duplicate exceptf list =
semant.ml:358:  let rec helper = function
semant.ml:359:      n1 :: n2 :: _ when n1 = n2 -> exceptf n1
semant.ml:360:    | _ :: t -> helper t
semant.ml:361:    | [] -> ()
semant.ml:362:  in helper (List.sort compare list)
semant.ml:363:
semant.ml:364:(* check function *)
semant.ml:365:let check_function func_map func =
semant.ml:366:  (* check duplicate formals *)
semant.ml:367:  let args = List.map (fun (Formal(_, n)) -> n) func.args
in
semant.ml:368:  report_duplicate (duplicate_formal_decl_error func) args;
semant.ml:369:
semant.ml:370:  (* check duplicate locals *)
semant.ml:371:  let locals = List.map (fun (Formal(_, n)) -> n)
func.locals in
semant.ml:372:  report_duplicate (duplicate_local_decl_error func)
locals;
semant.ml:373:
semant.ml:374:
semant.ml:375:  (* search locally, if not found, then recursively search
parent environment *)
semant.ml:376:  let rec type_of_identifier func s =
semant.ml:377:    let symbols = List.fold_left (fun m (Formal(t, n)) ->
StringMap.add n t m)
semant.ml:378:        StringMap.empty (func.args @ func.locals )
semant.ml:379:    in
semant.ml:380:    try StringMap.find s symbols
semant.ml:381:    with Not_found ->
semant.ml:382:      if func.name = "main" then unidentify_error s else
semant.ml:383:        (* recursively search parent environment *)
semant.ml:384:        type_of_identifier (StringMap.find func.pname
func_map) s
semant.ml:385:  in
semant.ml:386:  (* Raise an exception of the given rvalue type cannot be
assigned to
semant.ml:387:     the given lvalue type, noted that int could be
assinged to float type variable *)
semant.ml:388:  let check_assign lvaluet rvaluet ex = match lvaluet with
semant.ml:389:      Float_t when rvaluet = Int_t -> lvaluet
semant.ml:390:    | String_t when rvaluet = Null_t -> lvaluet
semant.ml:391:    | Node_t when rvaluet = Null_t -> lvaluet
semant.ml:392:    | Graph_t when rvaluet = Null_t -> lvaluet
```

```
semant.ml:393:    | List_Int_t | List_String_t | List_Float_t |
List_Node_t | List_Graph_t | List_Bool_t when rvaluet = Null_t -> lvaluet
semant.ml:394:    | Dict_Int_t | Dict_String_t | Dict_Float_t |
Dict_Node_t | Dict_Graph_t when rvaluet = Null_t -> lvaluet
semant.ml:395:    | List_Int_t | List_String_t | List_Node_t when rvaluet
= List_Null_t -> lvaluet
semant.ml:396:    | _ -> if lvaluet == rvaluet then lvaluet else
semant.ml:397:        assign_error (string_of_typ lvaluet) (string_of_typ
rvaluet) (string_of_expr ex)
semant.ml:398:  in
semant.ml:399:  (* Return the type of an expression or throw an exception
*)
semant.ml:400:  let rec expr = function
semant.ml:401:      Num_Lit(Num_Int _) -> Int_t
semant.ml:402:    | Num_Lit(Num_Float _) -> Float_t
semant.ml:403:    | Null -> Null_t
semant.ml:404:    | String_Lit _ -> String_t
semant.ml:405:    | Bool_lit _ -> Bool_t
semant.ml:406:    (* check node and graph *)
semant.ml:407:    | Node(_, _) -> Node_t
semant.ml:408:    | Graph_Link(e1, _, _) ->                          (*    *)
semant.ml:409:      let check_graph_link e1 =
semant.ml:410:        let typ = expr e1 in
semant.ml:411:        match typ with
semant.ml:412:          Node_t -> ()
semant.ml:413:        |_ -> graphLink_error (string_of_expr e1)
semant.ml:414:      in
semant.ml:415:      ignore(check_graph_link e1); Graph_t          (*    *)
semant.ml:416:    | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 =
expr e2 in
semant.ml:417:      (match op with
semant.ml:418:       (* +,-,*,/ *)
semant.ml:419:         Add | Sub | Mult | Div when t1 = Int_t && t2 =
Int_t -> Int_t
semant.ml:420:       |Add | Sub | Mult | Div when t1 = Float_t && t2 =
Float_t -> Float_t
semant.ml:421:       |Add | Sub | Mult | Div when t1 = Int_t && t2 =
Float_t -> Float_t
semant.ml:422:       |Add | Sub | Mult | Div when t1 = Float_t && t2 =
Int_t -> Float_t
semant.ml:423:       (* + - for graph *)
semant.ml:424:       | Add when t1 = Graph_t && t2 = Graph_t -> Graph_t
semant.ml:425:       | Sub when t1 = Graph_t && t2 = Graph_t ->
List_Graph_t
semant.ml:426:       | Sub when t1 = Graph_t && t2 = Node_t ->
List_Graph_t
semant.ml:427:       (* + for list *)
semant.ml:428:       | Add when t1 = List_Int_t && t2 = List_Int_t ->
List_Int_t
semant.ml:429:       | Add when t1 = List_Int_t && t2 = Int_t ->
List_Int_t
semant.ml:430:       | Add when t1 = List_Float_t && t2 = List_Float_t ->
List_Float_t
semant.ml:431:       | Add when t1 = List_Float_t && t2 = Float_t ->
List_Float_t
semant.ml:432:       | Add when t1 = List_Bool_t && t2 = List_Bool_t ->
List_Bool_t
semant.ml:433:       | Add when t1 = List_Bool_t && t2 = Bool_t ->
List_Bool_t
```

```
semant.ml:434:       | Add when t1 = List_String_t && t2 = List_String_t
-> List_String_t
semant.ml:435:       | Add when t1 = List_String_t && t2 = String_t ->
List_String_t
semant.ml:436:       | Add when t1 = List_Node_t && t2 = List_Node_t ->
List_Node_t
semant.ml:437:       | Add when t1 = List_Node_t && t2 = Node_t ->
List_Node_t
semant.ml:438:       | Add when t1 = List_Graph_t && t2 = List_Graph_t ->
List_Graph_t
semant.ml:439:       | Add when t1 = List_Graph_t && t2 = Graph_t ->
List_Graph_t
semant.ml:440:       (* - for list data *)
semant.ml:441:       | Sub when t1 = List_Int_t && t2 = Int_t ->
List_Int_t
semant.ml:442:       | Sub when t1 = List_Float_t && t2 = Float_t ->
List_Float_t
semant.ml:443:       | Sub when t1 = List_Bool_t && t2 = Bool_t ->
List_Bool_t
semant.ml:444:       | Sub when t1 = List_String_t && t2 = String_t ->
List_String_t
semant.ml:445:       | Sub when t1 = List_Node_t && t2 = Node_t ->
List_Node_t
semant.ml:446:       | Sub when t1 = List_Graph_t && t2 = Graph_t ->
List_Graph_t
semant.ml:447:       (* ==, != *)
semant.ml:448:       | Equal | Neq when t1 = t2 -> Bool_t
semant.ml:449:       (* <, <=, >, >= *)
semant.ml:450:       | Less | Leq | Greater | Geq when (t1 = Int_t || t1
= Float_t) && (t2 = Int_t || t2 = Float_t) -> Bool_t
semant.ml:451:       (* and, or *)
semant.ml:452:       | And | Or when t1 = Bool_t && t2 = Bool_t -> Bool_t
semant.ml:453:       (* mode *)
semant.ml:454:       | Mod when t1 = Int_t && t2 = Int_t -> Int_t
semant.ml:455:       | _ -> binary_error (string_of_typ t1)
(string_of_typ t2) (string_of_op op) (string_of_expr e)
semant.ml:456:       )
semant.ml:457:     | Unop(op, e) as ex -> let t = expr e in
semant.ml:458:       (match op with
semant.ml:459:          Neg when t = Int_t -> Int_t
semant.ml:460:       |Neg when t = Float_t -> Float_t
semant.ml:461:       | Not when t = Bool_t -> Bool_t
semant.ml:462:       | _ -> unary_error (string_of_typ t) (string_of_uop
op) (string_of_expr ex)
semant.ml:463:       )
semant.ml:464:     | Id s -> type_of_identifier func s
semant.ml:465:     | Assign(var, e) as ex -> let lt = type_of_identifier
func var and rt = expr e in
semant.ml:466:       check_assign lt rt ex
semant.ml:467:     | Noexpr -> Void_t
semant.ml:468:     | ListP([]) as ex -> empty_list_error (string_of_expr
ex)
semant.ml:469:     | ListP(es) ->
semant.ml:470:       let element_type =
semant.ml:471:         let determine_element_type ss = List.fold_left
semant.ml:472:             (fun l e -> (match l with
semant.ml:473:                    [] -> [expr e]
semant.ml:474:                  | t :: _ when t = (expr e) -> [t]
semant.ml:475:                  | t :: _ when (t = Graph_t && (expr e) =
Node_t) || (t = Node_t && (expr e) = Graph_t) -> [Graph_t]
```

```
semant.ml:476:                  | t :: _ when (t = Float_t && (expr e) =
Int_t) || (t = Int_t && (expr e) = Float_t) -> [Float_t]
semant.ml:477:                  | t :: _ ->
list_element_type_inconsistent_error (string_of_typ t) (string_of_typ
(expr e))
semant.ml:478:                ) [] ss
semant.ml:479:        in
semant.ml:480:        List.hd (determine_element_type es)
semant.ml:481:      in
semant.ml:482:      match_list_type element_type
semant.ml:483:    | DictP([]) as ex -> invalid_empty_dict_decl_error
(string_of_expr ex)
semant.ml:484:    | DictP(es) ->
semant.ml:485:      let element_type =
semant.ml:486:        let determine_element_type ss = List.fold_left
semant.ml:487:            (fun l (_, e) -> (match l with
semant.ml:488:                  [] -> [expr e]
semant.ml:489:                | t :: _  when t = (expr e) -> [t]
semant.ml:490:                | t :: _ ->
inconsistent_dict_element_type_error (string_of_typ t) (string_of_typ
(expr e))
semant.ml:491:                ) [] ss
semant.ml:492:        in
semant.ml:493:        List.hd (determine_element_type es)
semant.ml:494:      in
semant.ml:495:      match_dict_type element_type
semant.ml:496:    | Call(n, args) -> let func_obj = get_func_obj n
func_map in
semant.ml:497:      (* check function call such as the args length, args
type *)
semant.ml:498:      let check_funciton_call func args =
semant.ml:499:        let check_args_length l_arg r_arg = if (List.length
l_arg) = (List.length r_arg)
semant.ml:500:          then () else (unmatched_functionArg_error
func.name)
semant.ml:501:        in
semant.ml:502:        if List.mem func.name ["printb"; "print"; "printf";
"string"; "float"; "int"; "bool"] then ()
semant.ml:503:        else check_args_length func.args args;
semant.ml:504:        (* l_arg is a list of Formal(typ, name), r_arg is a
list of expr *)
semant.ml:505:        let check_args_type l_arg r_arg =
semant.ml:506:          List.iter2
semant.ml:507:            (fun (Formal(t, _)) r -> let r_typ = expr r in
if t = r_typ then () else
semant.ml:508:                functionCompariable_error (string_of_typ
r_typ) (string_of_typ t)
semant.ml:509:            )
semant.ml:510:            l_arg r_arg
semant.ml:511:        in
semant.ml:512:        if List.mem func.name ["printb"; "print"; "printf";
"string"; "float"; "int"; "bool"] then ()
semant.ml:513:        else check_args_type func.args args
semant.ml:514:      in
semant.ml:515:      ignore(check_funciton_call func_obj args);
func_obj.returnType
semant.ml:516:    | CallDefault(e, n, es) -> let typ = expr e in
semant.ml:517:      let check_list_get_method ex es =
semant.ml:518:        match es with
semant.ml:519:          [x] when (expr x) = Int_t -> ()
```

```
semant.ml:520:        | _ -> listGet_error (string_of_expr ex)
semant.ml:521:      in
semant.ml:522:      let check_list_add_method typ ex es =
semant.ml:523:        match es with
semant.ml:524:          [x] when (expr x) = (reverse_match_list_type typ)
-> ()
semant.ml:525:        | _ -> listAdd_error (string_of_typ
(reverse_match_list_type typ)) (string_of_expr ex)
semant.ml:526:      in
semant.ml:527:      let check_list_push_method typ ex es =
semant.ml:528:        match es with
semant.ml:529:          [x] when (expr x) = (reverse_match_list_type typ)
-> ()
semant.ml:530:        | _ -> listPush_error (string_of_typ
(reverse_match_list_type typ)) (string_of_expr ex)
semant.ml:531:      in
semant.ml:532:      let check_list_remove_method ex es =
semant.ml:533:        match es with
semant.ml:534:          [x] when (expr x) = Int_t -> ()
semant.ml:535:        | _ -> listRemove_error (string_of_expr ex)
semant.ml:536:      in
semant.ml:537:      let check_list_set_method typ ex es =
semant.ml:538:        match es with
semant.ml:539:          [index; value] when (expr index) = Int_t && (expr
value) = (reverse_match_list_type typ) -> ()
semant.ml:540:        | _ -> listSet_error (string_of_typ
(reverse_match_list_type typ)) (string_of_expr ex)
semant.ml:541:      in
semant.ml:542:      let check_dict_get_method ex es =
semant.ml:543:        match es with
semant.ml:544:          [x] when List.mem (expr x) [Int_t; String_t;
Node_t] -> ()
semant.ml:545:        | _ -> invalid_dict_get_method_error
(string_of_expr ex)
semant.ml:546:      in
semant.ml:547:      let check_dict_remove_method ex es =
semant.ml:548:        match es with
semant.ml:549:          [x] when List.mem (expr x) [Int_t; String_t;
Node_t] -> ()
semant.ml:550:        | _ -> invalid_dict_remove_method_error
(string_of_expr ex)
semant.ml:551:      in
semant.ml:552:      let check_dict_put_method typ ex es =
semant.ml:553:        match es with
semant.ml:554:          [key; value] when List.mem (expr key) [Int_t;
String_t; Node_t]
semant.ml:555:                            && ((expr value) =
(reverse_match_dict_type typ) || (expr value) = Null_t) -> ()
semant.ml:556:        | _ -> invalid_dict_put_method_error (string_of_typ
(reverse_match_dict_type typ)) (string_of_expr ex)
semant.ml:557:      in
semant.ml:558:      match typ with
semant.ml:559:        List_Int_t | List_Float_t | List_String_t |
List_Node_t | List_Graph_t | List_Bool_t ->
semant.ml:560:        (match n with
semant.ml:561:           "add" -> ignore(check_list_add_method typ e es);
typ
semant.ml:562:         | "push"  -> ignore(check_list_push_method typ e
es); typ
```

```
semant.ml:563:         | "remove" -> ignore(check_list_remove_method e
es); typ
semant.ml:564:         | "set" -> ignore(check_list_set_method typ e es);
typ
semant.ml:565:         (* | "concat" ->  *)
semant.ml:566:         | "pop" -> ignore(check_list_pop_method e es);
reverse_match_list_type typ
semant.ml:567:         | "get" -> ignore(check_list_get_method e es);
reverse_match_list_type typ
semant.ml:568:         | "size" -> ignore(check_list_size_method e es);
Int_t
semant.ml:569:         | _ -> unsupport_operation_error (string_of_typ
typ) n
semant.ml:570:        )
semant.ml:571:      | Dict_Int_t | Dict_Float_t | Dict_String_t |
Dict_Node_t | Dict_Graph_t ->
semant.ml:572:        (* key support type node, string, int *)
semant.ml:573:        (match n with
semant.ml:574:           "put" -> ignore(check_dict_put_method typ e es);
typ
semant.ml:575:         | "get" -> ignore(check_dict_get_method e es);
reverse_match_dict_type typ
semant.ml:576:         | "remove" -> ignore(check_dict_remove_method e
es); typ
semant.ml:577:         | "size" -> ignore(check_dict_size_method e es);
Int_t
semant.ml:578:         (* return List_Null_t here to bypass the semantic
check *)
semant.ml:579:         | "keys" -> ignore(check_dict_keys_method e es);
List_Null_t
semant.ml:580:         | _ -> unsupport_operation_error (string_of_typ
typ) n
semant.ml:581:        )
semant.ml:582:      | Node_t ->
semant.ml:583:        (match n with
semant.ml:584:         | "setVisited" -> Bool_t
semant.ml:585:         | "isVisted" -> Bool_t
semant.ml:586:         | _ -> unsupport_operation_error (string_of_typ
typ) n
semant.ml:587:        )
semant.ml:588:      | Graph_t ->
semant.ml:589:        (match n with
semant.ml:590:           "root" -> ignore(check_graph_root_method e es);
Node_t
semant.ml:591:         | "size" -> ignore(check_graph_size_method e es);
Int_t
semant.ml:592:         | "getAllNodes" -> ignore(check_graph_nodes_method
e es); List_Node_t
semant.ml:593:         | "edges" -> ignore(check_graph_edges_method e
es); List_Int_t
semant.ml:594:         | "setAllUnvisited" -> Int_t
semant.ml:595:         | "dfs" -> List_Node_t
semant.ml:596:         | "bfs" -> List_Node_t
semant.ml:597:         | "hasNode" -> Bool_t
semant.ml:598:         | "hasEdge" -> Bool_t
semant.ml:599:         | "addNode" -> Int_t
semant.ml:600:         | "addEdge" -> Int_t
semant.ml:601:         | "dijkstra" -> List_Node_t
semant.ml:602:         | _ -> unsupport_operation_error (string_of_typ
typ) n
```

```
semant.ml:603:        )
semant.ml:604:      | _ -> unsupport_operation_error (string_of_typ typ)
n
semant.ml:605:  in
semant.ml:606:  (* check statement *)
semant.ml:607:  let rec stmt = function
semant.ml:608:      Expr(e) -> ignore (expr e)
semant.ml:609:    | Return e -> ignore (check_return_type func (expr e))
semant.ml:610:    | For(e1, e2, e3, stls) ->
semant.ml:611:      ignore (expr e1); ignore (expr e2); ignore (expr e3);
ignore(stmt_list stls)
semant.ml:612:    | If(e, stls1, stls2) -> ignore(e); ignore(stmt_list
stls1); ignore(stmt_list stls2)
semant.ml:613:    | While(e, stls) -> ignore(e); ignore(stmt_list stls)
semant.ml:614:  and
semant.ml:615:    (* check statement list *)
semant.ml:616:    stmt_list = function
semant.ml:617:      Return _ :: ss when ss <> [] -> after_return_error ss
semant.ml:618:    | s::ss -> stmt s ; stmt_list ss
semant.ml:619:    | [] -> ()
semant.ml:620:
semant.ml:621:  in
semant.ml:622:  stmt_list func.body
semant.ml:623:
semant.ml:624:(* program here is a list of functions *)
semant.ml:625:let check program =
semant.ml:626:  let end_with s1 s2 =
semant.ml:627:    let len1 = String.length s1 and len2 = String.length s2
in
semant.ml:628:    if len1 < len2 then false
semant.ml:629:    else
semant.ml:630:      let last = String.sub s1 (len1-len2) len2 in
semant.ml:631:      if last = s2 then true else false
semant.ml:632:  in
semant.ml:633:  if List.mem true (List.map (fun f -> end_with f.name
"print") program)
semant.ml:634:  then redefine_error "_" else ();
semant.ml:635:  (* check duplicate function *)
semant.ml:636:  let m = StringMap.empty in
semant.ml:637:  ignore(List.map (fun f ->
semant.ml:638:      if StringMap.mem f.name m
semant.ml:639:      then (duplicate_func_error f.name)
semant.ml:640:      else StringMap.add f.name true m) program);
semant.ml:641:  (* Function declaration for a named function *)
semant.ml:642:  let built_in_funcs =
semant.ml:643:    let funcs = [
semant.ml:644:      (
semant.ml:645:        "print",
semant.ml:646:        { returnType = Void_t; name = "print"; args =
[Formal(String_t, "x")];
semant.ml:647:          locals = []; body = []; pname = "main"}
semant.ml:648:      );
semant.ml:649:      (
semant.ml:650:        "printb",
semant.ml:651:        { returnType = Void_t; name = "printb"; args =
[Formal(Bool_t, "x")];
semant.ml:652:          locals = []; body = []; pname = "main"}
semant.ml:653:      );
semant.ml:654:      (
semant.ml:655:        "printf",
```

```
semant.ml:656:        { returnType = Void_t; name = "printf"; args =
[Formal(String_t, "x")];
semant.ml:657:          locals = []; body = []; pname = "main"}
semant.ml:658:      );
semant.ml:659:      (
semant.ml:660:        "string",
semant.ml:661:        { returnType = String_t; name = "string"; args =
[Formal(String_t, "x")];
semant.ml:662:          locals = []; body = []; pname = "main"}
semant.ml:663:      );
semant.ml:664:      (
semant.ml:665:        "int",
semant.ml:666:        { returnType = Int_t; name = "int"; args =
[Formal(String_t, "x")];
semant.ml:667:          locals = []; body = []; pname = "main"}
semant.ml:668:      );
semant.ml:669:      (
semant.ml:670:        "float",
semant.ml:671:        { returnType = Float_t; name = "float"; args =
[Formal(String_t, "x")];
semant.ml:672:          locals = []; body = []; pname = "main"}
semant.ml:673:      );
semant.ml:674:      (
semant.ml:675:        "bool",
semant.ml:676:        { returnType = Bool_t; name = "bool"; args =
[Formal(String_t, "x")];
semant.ml:677:          locals = []; body = []; pname = "main"}
semant.ml:678:      )
semant.ml:679:    ]
semant.ml:680:    in
semant.ml:681:    let add_func funcs m =
semant.ml:682:      List.fold_left (fun m (n, func) -> StringMap.add n
func m) m funcs
semant.ml:683:    in
semant.ml:684:    add_func funcs StringMap.empty
semant.ml:685:  in
semant.ml:686:  (* collect all functions and store in map with key=name,
value=function *)
semant.ml:687:  let func_map = List.fold_left (fun m f -> StringMap.add
f.name f m) built_in_funcs program in
semant.ml:688:  let check_function_wrapper func m =
semant.ml:689:    func m
semant.ml:690:  in
semant.ml:691:  (**** Checking functions ****)
semant.ml:692:  List.iter (check_function_wrapper check_function
func_map) program
semant.ml:693:tests/code_gen/test-print.out:1:23
tests/code_gen/test-print.out:2:-1.200000
tests/code_gen/test-print.out:3:Hello, World
tests/code_gen/test-print.out:4:true
tests/code_gen/test-print.out:5:true
tests/code_gen/test-print.out:6:[1, 2, 3]
tests/code_gen/test-print.out:7:node 10
tests/code_gen/test-print.out:8:{a: 1, b: 2}
tests/code_gen/test-print.out:9:4115 ** 20.17 ** GIRAPHE
tests/code_gen/test-print.out:10:tests/code_gen/test-list.in:1:print("---
---------------general method for list------------------");
tests/code_gen/test-list.in:2:
tests/code_gen/test-list.in:3:list<int> a = [1, 2, 3];
tests/code_gen/test-list.in:4:list<float> b = [1.0, 2.0, 3.0];
```

```
tests/code_gen/test-list.in:5:list<string> c = ["a", "b", "c"];
tests/code_gen/test-list.in:6:list<bool> d = [true, false, true];
tests/code_gen/test-list.in:7:
tests/code_gen/test-list.in:8:print(a);
tests/code_gen/test-list.in:9:print(b);
tests/code_gen/test-list.in:10:print(c);
tests/code_gen/test-list.in:11:print(d);
tests/code_gen/test-list.in:12:
tests/code_gen/test-list.in:13:node n1 = node(7);
tests/code_gen/test-list.in:14:node n2 = node(8);
tests/code_gen/test-list.in:15:node n3 = node(9);
tests/code_gen/test-list.in:16:list<node> nd = [n1, n2, n3];
tests/code_gen/test-list.in:17:print(nd);
tests/code_gen/test-list.in:18:
tests/code_gen/test-list.in:19:list<graph> gp = [n1->n2, n2->n3, n3->n1];
tests/code_gen/test-list.in:20:print(gp);
tests/code_gen/test-list.in:21:
tests/code_gen/test-list.in:22:print("------------ test add, remove -----
----------");
tests/code_gen/test-list.in:23:
tests/code_gen/test-list.in:24:print("remove element not existing
a.remove(4), list doesn't change");
tests/code_gen/test-list.in:25:a.remove(4);
tests/code_gen/test-list.in:26:print(a);
tests/code_gen/test-list.in:27:print("adding element using list.add()");
tests/code_gen/test-list.in:28:a.add(5);
tests/code_gen/test-list.in:29:print(a);
tests/code_gen/test-list.in:30:print("remove element using -");
tests/code_gen/test-list.in:31:a=a-2;
tests/code_gen/test-list.in:32:print(a);
tests/code_gen/test-list.in:33:a=a-1;
tests/code_gen/test-list.in:34:a=a-5;
tests/code_gen/test-list.in:35:print("remove list to empty");
tests/code_gen/test-list.in:36:a=a-3;
tests/code_gen/test-list.in:37:print(a);
tests/code_gen/test-list.in:38:print("add element using +");
tests/code_gen/test-list.in:39:a=a+6;
tests/code_gen/test-list.in:40:print(a);
tests/code_gen/test-list.in:41:print("----------------------------------
--------");
tests/code_gen/test-list.in:42:
tests/code_gen/test-list.in:43:print("------------ test size, get, set --
-------------");
tests/code_gen/test-list.in:44:print(b);
tests/code_gen/test-list.in:45:print("Size:");
tests/code_gen/test-list.in:46:print(b.size());
tests/code_gen/test-list.in:47:print("Get first element:");
tests/code_gen/test-list.in:48:print(b.get(0));
tests/code_gen/test-list.in:49:print("Set function:");
tests/code_gen/test-list.in:50:b.set(1,5.0);
tests/code_gen/test-list.in:51:print(b);
tests/code_gen/test-list.in:52:print("----------------------------------
-------------");
tests/code_gen/test-dijkstra.out:1:From a to f
tests/code_gen/test-dijkstra.out:2:From a to g
tests/code_gen/test-dijkstra.out:3:tests/code_gen/test-while.out:1:0
tests/code_gen/test-while.out:2:1
tests/code_gen/test-while.out:3:true
tests/code_gen/test-while.out:4:tests/code_gen/test-list.out:1:----------
--------general method for list------------------
```

```
tests/code_gen/test-list.out:2:[1, 2, 3]
tests/code_gen/test-list.out:3:[1.000000, 2.000000, 3.000000]
tests/code_gen/test-list.out:4:[a, b, c]
tests/code_gen/test-list.out:5:[true, false, true]
tests/code_gen/test-list.out:6:[node 7
tests/code_gen/test-list.out:7:node 8
tests/code_gen/test-list.out:8:node 9
tests/code_gen/test-list.out:9:]
tests/code_gen/test-list.out:10:[--------------------------------------
tests/code_gen/test-list.out:11:node 7
tests/code_gen/test-list.out:12:node 8
tests/code_gen/test-list.out:13:edge 7 ->8: 0
tests/code_gen/test-list.out:14:--------------------------------------
tests/code_gen/test-list.out:15:--------------------------------------
tests/code_gen/test-list.out:16:node 8
tests/code_gen/test-list.out:17:node 9
tests/code_gen/test-list.out:18:edge 8 ->9: 0
tests/code_gen/test-list.out:19:--------------------------------------
tests/code_gen/test-list.out:20:--------------------------------------
tests/code_gen/test-list.out:21:node 9
tests/code_gen/test-list.out:22:node 7
tests/code_gen/test-list.out:23:edge 9 ->7: 0
tests/code_gen/test-list.out:24:--------------------------------------
tests/code_gen/test-list.out:25:]
tests/code_gen/test-list.out:26:------------ test add, remove -----------
----
tests/code_gen/test-list.out:27:remove element not existing a.remove(4),
list doesn't change
tests/code_gen/test-list.out:28:[1, 2, 3]
tests/code_gen/test-list.out:29:adding element using list.add()
tests/code_gen/test-list.out:30:[1, 2, 3, 5]
tests/code_gen/test-list.out:31:remove element using -
tests/code_gen/test-list.out:32:[1, 3, 5]
tests/code_gen/test-list.out:33:remove list to empty
tests/code_gen/test-list.out:34:[]
tests/code_gen/test-list.out:35:add element using +
tests/code_gen/test-list.out:36:[6]
tests/code_gen/test-list.out:37:----------------------------------------
--
tests/code_gen/test-list.out:38:------------ test size, get, set --------
-------
tests/code_gen/test-list.out:39:[1.000000, 2.000000, 3.000000]
tests/code_gen/test-list.out:40:Size:
tests/code_gen/test-list.out:41:3
tests/code_gen/test-list.out:42:Get first element:
tests/code_gen/test-list.out:43:1.000000
tests/code_gen/test-list.out:44:Set function:
tests/code_gen/test-list.out:45:[1.000000, 5.000000, 3.000000]
tests/code_gen/test-list.out:46:----------------------------------------
-------
tests/code_gen/test-list.out:47:tests/code_gen/test-if.out:1:small
tests/code_gen/test-if.out:2:12.000000
tests/code_gen/test-if.out:3:true
tests/code_gen/test-if.out:4:tests/code_gen/test-if.in:1:int a = 0;
tests/code_gen/test-if.in:2:if (a < 1) {
tests/code_gen/test-if.in:3:print("small");
tests/code_gen/test-if.in:4:}else {
tests/code_gen/test-if.in:5:print("large");
tests/code_gen/test-if.in:6:}
tests/code_gen/test-if.in:7:
```

```
tests/code_gen/test-if.in:8:float b = 12;
tests/code_gen/test-if.in:9:if (b > 10) {
tests/code_gen/test-if.in:10:    print(b);
tests/code_gen/test-if.in:11:}
tests/code_gen/test-if.in:12:
tests/code_gen/test-if.in:13:bool c = false;
tests/code_gen/test-if.in:14:if (!c) {
tests/code_gen/test-if.in:15:    print("true");
tests/code_gen/test-if.in:16:}
tests/code_gen/test-arith.in:1:print(1+2);
tests/code_gen/test-arith.in:2:print(2-3);
tests/code_gen/test-arith.in:3:print(5*15);
tests/code_gen/test-arith.in:4:print(9/3);
tests/code_gen/test-arith.in:5:print(8.2/4);
tests/code_gen/test-arith.in:6:print(5%2.3);
tests/code_gen/test-arith.in:7:print(1.2+1.3);
tests/code_gen/test-arith.in:8:print(8.2-1.5);
tests/code_gen/test-arith.in:9:
tests/code_gen/test-arith.in:10:int a = 0;
tests/code_gen/test-arith.in:11:int foo(int b) {
tests/code_gen/test-arith.in:12:  int a = 1;
tests/code_gen/test-arith.in:13:  int bar() {
tests/code_gen/test-arith.in:14:    return a+b;
tests/code_gen/test-arith.in:15:  }
tests/code_gen/test-arith.in:16:  return bar();
tests/code_gen/test-arith.in:17:}
tests/code_gen/test-arith.in:18:print(a);
tests/code_gen/test-arith.in:19:print(foo(3));
tests/code_gen/test-arith.in:20:tests/code_gen/test-bfs.out:1:a -> [1&b -
> 1&e -> [4&g -> 1&b, 2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]
tests/code_gen/test-bfs.out:2:Breath First Search from a
tests/code_gen/test-bfs.out:3:[node a
tests/code_gen/test-bfs.out:4:node b
tests/code_gen/test-bfs.out:5:node c
tests/code_gen/test-bfs.out:6:node d
tests/code_gen/test-bfs.out:7:node e
tests/code_gen/test-bfs.out:8:node g
tests/code_gen/test-bfs.out:9:node f
tests/code_gen/test-bfs.out:10:]
tests/code_gen/test-bfs.out:11:Set all Nodes to Unvisited
tests/code_gen/test-bfs.out:12:Breath First Search from c
tests/code_gen/test-bfs.out:13:[node c
tests/code_gen/test-bfs.out:14:node g
tests/code_gen/test-bfs.out:15:node f
tests/code_gen/test-bfs.out:16:node b
tests/code_gen/test-bfs.out:17:node e
tests/code_gen/test-bfs.out:18:]
tests/code_gen/test-bfs.out:19:Set all Nodes to Unvisited
tests/code_gen/test-bfs.out:20:tests/code_gen/test-print.in:1:print(23);
tests/code_gen/test-print.in:2:print(-1.2);
tests/code_gen/test-print.in:3:print("Hello, World");
tests/code_gen/test-print.in:4:print(true);
tests/code_gen/test-print.in:5:print(1>0.2);
tests/code_gen/test-print.in:6:print([1, 2, 3]);
tests/code_gen/test-print.in:7:print(node(10));
tests/code_gen/test-print.in:8:print({"a": 1, "b": 2});
tests/code_gen/test-print.in:9:
tests/code_gen/test-print.in:10:int a = 4115;
tests/code_gen/test-print.in:11:float b = 20.17;
tests/code_gen/test-print.in:12:string c = "GIRAPHE";
```

```
tests/code_gen/test-print.in:13:
tests/code_gen/test-print.in:14:printf("%d ** %.2f ** %s\n", a, b, c);
tests/code_gen/test-topo.out:1:node COMS 3101: Programming Language
Python
tests/code_gen/test-topo.out:2:node CSOR 4231: Analysis of Algorithms I
tests/code_gen/test-topo.out:3:node COMS 6232: Analysis of Algorithms II
tests/code_gen/test-topo.out:4:node COMS 4771: Machine Learning
tests/code_gen/test-topo.out:5:node COMS 4772: Advanced Machine Learning
tests/code_gen/test-topo.out:6:node COMS 4995: Deep Learning for Computer
Vision
tests/code_gen/test-topo.out:7:node COMS 1004: Introduction to Computer
Science and Programming in Java
tests/code_gen/test-topo.out:8:node CSEE 4119: Computer Networks
tests/code_gen/test-topo.out:9:node CSEE 4140: Networking Laboratory
tests/code_gen/test-topo.out:10:node COMS 4111: Introduction to Databases
tests/code_gen/test-topo.out:11:node COMS 6111: Advanced Database Systems
tests/code_gen/test-topo.out:12:node COMS 4118: Operating Systems
tests/code_gen/test-topo.out:13:tests/code_gen/test-arith.out:1:3
tests/code_gen/test-arith.out:2:-1
tests/code_gen/test-arith.out:3:75
tests/code_gen/test-arith.out:4:3
tests/code_gen/test-arith.out:5:2.050000
tests/code_gen/test-arith.out:6:0.400000
tests/code_gen/test-arith.out:7:2.500000
tests/code_gen/test-arith.out:8:6.700000
tests/code_gen/test-arith.out:9:0
tests/code_gen/test-arith.out:10:4
tests/code_gen/test-arith.out:11:tests/code_gen/test-
linkGraph.out:1:Using + to link graph
tests/code_gen/test-linkGraph.out:2:Link graphs: A -> 0$B ->3$E +  C ->
1$A + A -> 2$D
tests/code_gen/test-linkGraph.out:3:Shared nodes: A
tests/code_gen/test-linkGraph.out:4:-------------------------------------
-
tests/code_gen/test-linkGraph.out:5:node B
tests/code_gen/test-linkGraph.out:6:node E
tests/code_gen/test-linkGraph.out:7:node A
tests/code_gen/test-linkGraph.out:8:node C
tests/code_gen/test-linkGraph.out:9:node D
tests/code_gen/test-linkGraph.out:10:edge B ->E: 3
tests/code_gen/test-linkGraph.out:11:edge A ->B: 0
tests/code_gen/test-linkGraph.out:12:edge C ->A: 1
tests/code_gen/test-linkGraph.out:13:edge A ->D: 2
tests/code_gen/test-linkGraph.out:14:-------------------------------------
--
tests/code_gen/test-linkGraph.out:15:tests/code_gen/test-node-
function.in:1:node a = node(1);
tests/code_gen/test-node-function.in:2:node b = node("GIRAPHE");
tests/code_gen/test-node-function.in:3:node c = node(1>2);
tests/code_gen/test-node-function.in:4:print(a);
tests/code_gen/test-node-function.in:5:print(b);
tests/code_gen/test-node-function.in:6:print(c);
tests/code_gen/test-node-function.in:7:
tests/code_gen/test-node-function.in:8:node d = node("hi");
tests/code_gen/test-node-function.in:9:print("Is Node d Visited?");
tests/code_gen/test-node-function.in:10:print(d.isVisited());
tests/code_gen/test-node-function.in:11:print("Set Node d to Visited");
tests/code_gen/test-node-function.in:12:d.setVisited();
tests/code_gen/test-node-function.in:13:print(d.isVisited());
tests/code_gen/test-node-function.in:14:
```

```
tests/code_gen/test-node-function.in:15:tests/code_gen/test-
Graph.in:1:node a = node("A");
tests/code_gen/test-Graph.in:2:node b = node("B");
tests/code_gen/test-Graph.in:3:node c = node("C");
tests/code_gen/test-Graph.in:4:node d = node("D");
tests/code_gen/test-Graph.in:5:node e = node("E");
tests/code_gen/test-Graph.in:6:node f = node("F");
tests/code_gen/test-Graph.in:7:
tests/code_gen/test-Graph.in:8:node m = node("M");
tests/code_gen/test-Graph.in:9:node h = node("H");
tests/code_gen/test-Graph.in:10:
tests/code_gen/test-Graph.in:11:node x = node("X");
tests/code_gen/test-Graph.in:12:node y = node("Y");
tests/code_gen/test-Graph.in:13:node z = node("Z");
tests/code_gen/test-Graph.in:14:
tests/code_gen/test-Graph.in:15:
tests/code_gen/test-Graph.in:16:print(" A -> E + A -> B -> D + A -> C ->F
");
tests/code_gen/test-Graph.in:17:
tests/code_gen/test-Graph.in:18:graph g = a -> 1$e + a -> 2$b -> 3$d + a
-> 4$c -> 5$f;
tests/code_gen/test-Graph.in:19:print(g);
tests/code_gen/test-Graph.in:20:
tests/code_gen/test-Graph.in:21:print("Get Graph Size:");
tests/code_gen/test-Graph.in:22:print(g.size());
tests/code_gen/test-Graph.in:23:
tests/code_gen/test-Graph.in:24:print("Add Nodes to Graph");
tests/code_gen/test-Graph.in:25:g.addNode(m);
tests/code_gen/test-Graph.in:26:print(g.size());
tests/code_gen/test-Graph.in:27:print(g);
tests/code_gen/test-Graph.in:28:
tests/code_gen/test-Graph.in:29:print("Judge whether Nodes exist");
tests/code_gen/test-Graph.in:30:print(g.hasNode(x));
tests/code_gen/test-Graph.in:31:print(g.hasNode(a));
tests/code_gen/test-Graph.in:32:
tests/code_gen/test-Graph.in:33:print("Add Edge to Graph, Support new
Nodes");
tests/code_gen/test-Graph.in:34:g.addEdge(e, f, 5);
tests/code_gen/test-Graph.in:35:g.addEdge(a, z, 10);
tests/code_gen/test-Graph.in:36:print(g.size());
tests/code_gen/test-Graph.in:37:list<node> nodes = g.getAllNodes();
tests/code_gen/test-Graph.in:38:print(nodes);
tests/code_gen/test-Graph.in:39:print(g);
tests/code_gen/test-Graph.in:40:
tests/code_gen/test-Graph.in:41:print("Judge whether Edges exist between
two Nodes");
tests/code_gen/test-Graph.in:42:print(g.hasEdge(c,d));
tests/code_gen/test-Graph.in:43:print(g.hasEdge(a,b));
tests/code_gen/test-Graph.in:44:
tests/code_gen/test-Graph.in:45:
tests/code_gen/test-Graph.in:46:
tests/code_gen/test-Graph.in:47:tests/code_gen/test-node-
function.out:1:node 1
tests/code_gen/test-node-function.out:2:node GIRAPHE
tests/code_gen/test-node-function.out:3:node false
tests/code_gen/test-node-function.out:4:Is Node d Visited?
tests/code_gen/test-node-function.out:5:false
tests/code_gen/test-node-function.out:6:Set Node d to Visited
tests/code_gen/test-node-function.out:7:true
```

```
tests/code_gen/test-node-function.out:8:tests/code_gen/test-Graph.out:1:
A -> E + A -> B -> D + A -> C ->F
tests/code_gen/test-Graph.out:2:-------------------------------------
tests/code_gen/test-Graph.out:3:node A
tests/code_gen/test-Graph.out:4:node E
tests/code_gen/test-Graph.out:5:node B
tests/code_gen/test-Graph.out:6:node D
tests/code_gen/test-Graph.out:7:node C
tests/code_gen/test-Graph.out:8:node F
tests/code_gen/test-Graph.out:9:edge A ->E: 1
tests/code_gen/test-Graph.out:10:edge B ->D: 3
tests/code_gen/test-Graph.out:11:edge A ->B: 2
tests/code_gen/test-Graph.out:12:edge C ->F: 5
tests/code_gen/test-Graph.out:13:edge A ->C: 4
tests/code_gen/test-Graph.out:14:-------------------------------------
tests/code_gen/test-Graph.out:15:Get Graph Size:
tests/code_gen/test-Graph.out:16:6
tests/code_gen/test-Graph.out:17:Add Nodes to Graph
tests/code_gen/test-Graph.out:18:7
tests/code_gen/test-Graph.out:19:-------------------------------------
tests/code_gen/test-Graph.out:20:node A
tests/code_gen/test-Graph.out:21:node E
tests/code_gen/test-Graph.out:22:node B
tests/code_gen/test-Graph.out:23:node D
tests/code_gen/test-Graph.out:24:node C
tests/code_gen/test-Graph.out:25:node F
tests/code_gen/test-Graph.out:26:node M
tests/code_gen/test-Graph.out:27:edge A ->E: 1
tests/code_gen/test-Graph.out:28:edge B ->D: 3
tests/code_gen/test-Graph.out:29:edge A ->B: 2
tests/code_gen/test-Graph.out:30:edge C ->F: 5
tests/code_gen/test-Graph.out:31:edge A ->C: 4
tests/code_gen/test-Graph.out:32:-------------------------------------
tests/code_gen/test-Graph.out:33:Judge whether Nodes exist
tests/code_gen/test-Graph.out:34:false
tests/code_gen/test-Graph.out:35:true
tests/code_gen/test-Graph.out:36:Add Edge to Graph, Support new Nodes
tests/code_gen/test-Graph.out:37:8
tests/code_gen/test-Graph.out:38:[node A
tests/code_gen/test-Graph.out:39:node E
tests/code_gen/test-Graph.out:40:node B
tests/code_gen/test-Graph.out:41:node D
tests/code_gen/test-Graph.out:42:node C
tests/code_gen/test-Graph.out:43:node F
tests/code_gen/test-Graph.out:44:node M
tests/code_gen/test-Graph.out:45:node Z
tests/code_gen/test-Graph.out:46:]
tests/code_gen/test-Graph.out:47:-------------------------------------
tests/code_gen/test-Graph.out:48:node A
tests/code_gen/test-Graph.out:49:node E
tests/code_gen/test-Graph.out:50:node B
tests/code_gen/test-Graph.out:51:node D
tests/code_gen/test-Graph.out:52:node C
tests/code_gen/test-Graph.out:53:node F
tests/code_gen/test-Graph.out:54:node M
tests/code_gen/test-Graph.out:55:node Z
tests/code_gen/test-Graph.out:56:edge A ->E: 1
tests/code_gen/test-Graph.out:57:edge B ->D: 3
tests/code_gen/test-Graph.out:58:edge A ->B: 2
tests/code_gen/test-Graph.out:59:edge C ->F: 5
```

```
tests/code_gen/test-Graph.out:60:edge A ->C: 4
tests/code_gen/test-Graph.out:61:edge E ->F: 5
tests/code_gen/test-Graph.out:62:edge A ->Z: 10
tests/code_gen/test-Graph.out:63:--------------------------------------
tests/code_gen/test-Graph.out:64:Judge whether Edges exist between two
Nodes
tests/code_gen/test-Graph.out:65:false
tests/code_gen/test-Graph.out:66:true
tests/code_gen/test-Graph.out:67:tests/code_gen/test-dfs.out:1:a -> [1&b
-> 1&e -> [4&g -> 1&b, 2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]
tests/code_gen/test-dfs.out:2:Depth First Search from c
tests/code_gen/test-dfs.out:3:[node c
tests/code_gen/test-dfs.out:4:node f
tests/code_gen/test-dfs.out:5:node g
tests/code_gen/test-dfs.out:6:node b
tests/code_gen/test-dfs.out:7:node e
tests/code_gen/test-dfs.out:8:]
tests/code_gen/test-dfs.out:9:Set all Nodes to Unvisited
tests/code_gen/test-dfs.out:10:Depth First Search from b
tests/code_gen/test-dfs.out:11:[node b
tests/code_gen/test-dfs.out:12:node e
tests/code_gen/test-dfs.out:13:node c
tests/code_gen/test-dfs.out:14:node f
tests/code_gen/test-dfs.out:15:node g
tests/code_gen/test-dfs.out:16:]
tests/code_gen/test-dfs.out:17:tests/code_gen/test-list2.in:1:print("----
--------------test contains, concatenate-----------------");
tests/code_gen/test-list2.in:2:
tests/code_gen/test-list2.in:3:list<int> a = [1, 2, 3];
tests/code_gen/test-list2.in:4:
tests/code_gen/test-list2.in:5:print("test whether list contains 2, 4");
tests/code_gen/test-list2.in:6:print(a.contains(2));
tests/code_gen/test-list2.in:7:print(a.contains(4));
tests/code_gen/test-list2.in:8:
tests/code_gen/test-list2.in:9:list<int> b = [4, 5, 6];
tests/code_gen/test-list2.in:10:
tests/code_gen/test-list2.in:11:list<int> c = a + b;
tests/code_gen/test-list2.in:12:print("print concatenate [1,2,3] and
[4,5,6]");
tests/code_gen/test-list2.in:13:print(c);
tests/code_gen/test-list2.in:14:
tests/code_gen/test-list2.in:15:
tests/code_gen/test-list2.in:16:print("---------------------------------
--------------------------");
tests/code_gen/test-linkGraph.in:1:node a = node("A");
tests/code_gen/test-linkGraph.in:2:node b = node("B");
tests/code_gen/test-linkGraph.in:3:node c = node("C");
tests/code_gen/test-linkGraph.in:4:node d = node("D");
tests/code_gen/test-linkGraph.in:5:node e = node("E");
tests/code_gen/test-linkGraph.in:6:
tests/code_gen/test-linkGraph.in:7:print("Using + to link graph");
tests/code_gen/test-linkGraph.in:8:
tests/code_gen/test-linkGraph.in:9:print("Link graphs: A -> 0$B ->3$E +
C -> 1$A + A -> 2$D");
tests/code_gen/test-linkGraph.in:10:print("Shared nodes: A");
tests/code_gen/test-linkGraph.in:11:
tests/code_gen/test-linkGraph.in:12:graph g = a -> 0$b ->3$e +  c -> 1$a
+ a -> 2$d;
tests/code_gen/test-linkGraph.in:13:
tests/code_gen/test-linkGraph.in:14:print(g);
```

```
tests/code_gen/test-linkGraph.in:15:tests/code_gen/test-filter.out:1:1 :
Restaurant
tests/code_gen/test-filter.out:2:2 : Unversity Building
tests/code_gen/test-filter.out:3:3 : Library
tests/code_gen/test-filter.out:4:4 : SuperMarket
tests/code_gen/test-filter.out:5:5 : Pharmacy
tests/code_gen/test-filter.out:6:
tests/code_gen/test-filter.out:7:-- try to find places of target type --
tests/code_gen/test-filter.out:8:1
tests/code_gen/test-filter.out:9:
tests/code_gen/test-filter.out:10:--------- have found target -----------
-
tests/code_gen/test-filter.out:11:---------     target      -------------
-
tests/code_gen/test-filter.out:12:node Xian Famous Food
tests/code_gen/test-filter.out:13:--------- shortest path to target  ----
-
tests/code_gen/test-filter.out:14:[node Mudd
tests/code_gen/test-filter.out:15:node Butler Library
tests/code_gen/test-filter.out:16:node Vine
tests/code_gen/test-filter.out:17:node WestSide Market
tests/code_gen/test-filter.out:18:node Duane Reade
tests/code_gen/test-filter.out:19:node Xian Famous Food
tests/code_gen/test-filter.out:20:]
tests/code_gen/test-filter.out:21:
tests/code_gen/test-filter.out:22:
tests/code_gen/test-filter.out:23:
tests/code_gen/test-filter.out:24:--------- have found target -----------
-
tests/code_gen/test-filter.out:25:---------     target      -------------
-
tests/code_gen/test-filter.out:26:node Vine
tests/code_gen/test-filter.out:27:--------- shortest path to target  ----
-
tests/code_gen/test-filter.out:28:[node Mudd
tests/code_gen/test-filter.out:29:node Butler Library
tests/code_gen/test-filter.out:30:node Vine
tests/code_gen/test-filter.out:31:]
tests/code_gen/test-filter.out:32:
tests/code_gen/test-filter.out:33:
tests/code_gen/test-filter.out:34:tests/code_gen/test-while.in:1:int a =
0;
tests/code_gen/test-while.in:2:while (a < 2) {
tests/code_gen/test-while.in:3:   print(a);
tests/code_gen/test-while.in:4:   a = a + 1;
tests/code_gen/test-while.in:5:}
tests/code_gen/test-while.in:6:
tests/code_gen/test-while.in:7:
tests/code_gen/test-while.in:8:bool c = true;
tests/code_gen/test-while.in:9:while (c) {
tests/code_gen/test-while.in:10:  print(c);
tests/code_gen/test-while.in:11:  c = ! c;
tests/code_gen/test-while.in:12:}
tests/code_gen/test-topo.in:1:node a = node("COMS 1004: Introduction to
Computer Science and Programming in Java");
tests/code_gen/test-topo.in:2:node b = node("COMS 4111: Introduction to
Databases");
tests/code_gen/test-topo.in:3:node c = node("COMS 6111: Advanced Database
Systems");
```

```
tests/code_gen/test-topo.in:4:node d = node("CSEE 4119: Computer
Networks");
tests/code_gen/test-topo.in:5:node e = node("CSEE 4140: Networking
Laboratory");
tests/code_gen/test-topo.in:6:node f = node("COMS 4118: Operating
Systems");
tests/code_gen/test-topo.in:7:node g = node("COMS 3101: Programming
Language Python");
tests/code_gen/test-topo.in:8:node h = node("CSOR 4231: Analysis of
Algorithms I");
tests/code_gen/test-topo.in:9:node i1 = node("COMS 6232: Analysis of
Algorithms II");
tests/code_gen/test-topo.in:10:node j = node("COMS 4771: Machine
Learning");
tests/code_gen/test-topo.in:11:node k = node("COMS 4772: Advanced Machine
Learning");
tests/code_gen/test-topo.in:12:node l = node("COMS 4995: Deep Learning
for Computer Vision");
tests/code_gen/test-topo.in:13:
tests/code_gen/test-topo.in:14:graph ga = a -> b -> f + b -> c + a -> d -
> f + d -> e + h -> f + g -> h -> j -> l + h -> i1 + j -> k;
tests/code_gen/test-topo.in:15:
tests/code_gen/test-topo.in:16:int i=0;
tests/code_gen/test-topo.in:17:list<node> n = ga.getAllNodes();
tests/code_gen/test-topo.in:18:list<node> tmp = n;
tests/code_gen/test-topo.in:19:list<node> res = n;
tests/code_gen/test-topo.in:20:int stmp = 0;
tests/code_gen/test-topo.in:21:ga.setAllUnvisited();
tests/code_gen/test-topo.in:22:
tests/code_gen/test-topo.in:23:while(i<ga.size()){
tests/code_gen/test-topo.in:24:   if(n.get(i).isVisited()){
tests/code_gen/test-topo.in:25:         i = i + 1;
tests/code_gen/test-topo.in:26:   }else{
tests/code_gen/test-topo.in:27:         tmp = ga.dfs(n.get(i));
tests/code_gen/test-topo.in:28:         stmp = tmp.size();
tests/code_gen/test-topo.in:29:         while(stmp > 0) {
tests/code_gen/test-topo.in:30:               res.add(tmp.get(stmp-1));
tests/code_gen/test-topo.in:31:               stmp = stmp - 1;
tests/code_gen/test-topo.in:32:         }
tests/code_gen/test-topo.in:33:         i = i + 1;
tests/code_gen/test-topo.in:34:   }
tests/code_gen/test-topo.in:35:}
tests/code_gen/test-topo.in:36:
tests/code_gen/test-topo.in:37:while(i > 0) {
tests/code_gen/test-topo.in:38:   print(res.get(i-1+ga.size()));
tests/code_gen/test-topo.in:39:   i=i-1;
tests/code_gen/test-topo.in:40:}
tests/code_gen/test-filter.in:1:node a = node("Mudd");
tests/code_gen/test-filter.in:2:node b = node("Vine");
tests/code_gen/test-filter.in:3:node c = node("Xian Famous Food");
tests/code_gen/test-filter.in:4:node d = node("Butler Library");
tests/code_gen/test-filter.in:5:node e = node("WestSide Market");
tests/code_gen/test-filter.in:6:node f = node("Duane Reade");
tests/code_gen/test-filter.in:7:
tests/code_gen/test-filter.in:8:
tests/code_gen/test-filter.in:9:graph gh = a -> 1$d ->8$c + a -> 1$d ->
2$b -> 1$e -> 1$f -> 1$c + a ->5$b;
tests/code_gen/test-filter.in:10:list<node> path = [a];
tests/code_gen/test-filter.in:11:node cur = a;
tests/code_gen/test-filter.in:12:int curVal = 0;
```

```
tests/code_gen/test-filter.in:13:print("1 : Restaurant");
tests/code_gen/test-filter.in:14:print("2 : Unversity Building");
tests/code_gen/test-filter.in:15:print("3 : Library");
tests/code_gen/test-filter.in:16:print("4 : SuperMarket");
tests/code_gen/test-filter.in:17:print("5 : Pharmacy");
tests/code_gen/test-filter.in:18:print("");
tests/code_gen/test-filter.in:19:hashmap<int> hmap = { a : 2, b : 1, c :
1, d : 3, e : 4, f : 5};
tests/code_gen/test-filter.in:20:
tests/code_gen/test-filter.in:21:void filter(node sour, int target) {
tests/code_gen/test-filter.in:22:    print("-- try to find places of
target type --");
tests/code_gen/test-filter.in:23:    print(target);
tests/code_gen/test-filter.in:24:    list<node> nodes = gh.getAllNodes();
tests/code_gen/test-filter.in:25:    int size = nodes.size();
tests/code_gen/test-filter.in:26:    int i = 0;
tests/code_gen/test-filter.in:27:    while (i < size) {
tests/code_gen/test-filter.in:28:        cur = nodes.get(i);
tests/code_gen/test-filter.in:29:        curVal = hmap.get(cur);
tests/code_gen/test-filter.in:30:        if (curVal == target) {
tests/code_gen/test-filter.in:31:            print("");
tests/code_gen/test-filter.in:32:            print("--------- have found
target ------------");
tests/code_gen/test-filter.in:33:            print("---------     target
--------------");
tests/code_gen/test-filter.in:34:            print(cur);
tests/code_gen/test-filter.in:35:            print("--------- shortest
path to target  -----");
tests/code_gen/test-filter.in:36:            path = gh.dijkstra(sour,
cur);
tests/code_gen/test-filter.in:37:            print(path);
tests/code_gen/test-filter.in:38:            print("");
tests/code_gen/test-filter.in:39:            print("");
tests/code_gen/test-filter.in:40:        }
tests/code_gen/test-filter.in:41:        i = i + 1;
tests/code_gen/test-filter.in:42:    }
tests/code_gen/test-filter.in:43:}
tests/code_gen/test-filter.in:44:filter(a, 1);
tests/code_gen/test-filter.in:45:
tests/code_gen/test-filter.in:46:tests/code_gen/test-dijkstra.in:1:node a
= node("a");
tests/code_gen/test-dijkstra.in:2:node b = node("b");
tests/code_gen/test-dijkstra.in:3:node c = node("c");
tests/code_gen/test-dijkstra.in:4:node d = node("d");
tests/code_gen/test-dijkstra.in:5:node e = node("e");
tests/code_gen/test-dijkstra.in:6:node f = node("f");
tests/code_gen/test-dijkstra.in:7:node g = node("g");
tests/code_gen/test-dijkstra.in:8:
tests/code_gen/test-dijkstra.in:9:
tests/code_gen/test-dijkstra.in:10:graph gh = a -> 1$b ->1$e -> 4$g ->1$b
+ a -> 1$b ->1$e ->2$c + a -> 5$c -> 1$g +a -> 5$c -> 1$f->1$c + a ->
3$d->2$c +a -> 3$d -> 3$f;
tests/code_gen/test-dijkstra.in:11:
tests/code_gen/test-dijkstra.in:12:print("From a to f");
tests/code_gen/test-dijkstra.in:13:gh.dijkstra(a,f);
tests/code_gen/test-dijkstra.in:14:print("From a to g");
tests/code_gen/test-dijkstra.in:15:gh.dijkstra(a, g);
tests/code_gen/test-dijkstra.in:16:tests/code_gen/test-dfs.in:1:node a =
node("a");
tests/code_gen/test-dfs.in:2:node b = node("b");
```

```
tests/code_gen/test-dfs.in:3:node c = node("c");
tests/code_gen/test-dfs.in:4:node d = node("d");
tests/code_gen/test-dfs.in:5:node e = node("e");
tests/code_gen/test-dfs.in:6:node f = node("f");
tests/code_gen/test-dfs.in:7:node g = node("g");
tests/code_gen/test-dfs.in:8:
tests/code_gen/test-dfs.in:9:print("a -> [1&b -> 1&e -> [4&g -> 1&b,
2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]");
tests/code_gen/test-dfs.in:10:
tests/code_gen/test-dfs.in:11:graph gh = a -> 1$b ->1$e -> 4$g ->1$b + a
-> 1$b ->1$e ->2$c + a -> 5$c -> 1$g +a -> 5$c -> 1$f->1$c + a -> 3$d-
>2$c +a -> 3$d -> 3$f;
tests/code_gen/test-dfs.in:12:
tests/code_gen/test-dfs.in:13:print("Depth First Search from c");
tests/code_gen/test-dfs.in:14:list<node> res = gh.dfs(c);
tests/code_gen/test-dfs.in:15:print(res);
tests/code_gen/test-dfs.in:16:print("Set all Nodes to Unvisited");
tests/code_gen/test-dfs.in:17:gh.setAllUnvisited();
tests/code_gen/test-dfs.in:18:print("Depth First Search from b");
tests/code_gen/test-dfs.in:19:list<node> ret = gh.dfs(b);
tests/code_gen/test-dfs.in:20:print(ret);
tests/code_gen/test-bfs.in:1:node a = node("a");
tests/code_gen/test-bfs.in:2:node b = node("b");
tests/code_gen/test-bfs.in:3:node c = node("c");
tests/code_gen/test-bfs.in:4:node d = node("d");
tests/code_gen/test-bfs.in:5:node e = node("e");
tests/code_gen/test-bfs.in:6:node f = node("f");
tests/code_gen/test-bfs.in:7:node g = node("g");
tests/code_gen/test-bfs.in:8:
tests/code_gen/test-bfs.in:9:print("a -> [1&b -> 1&e -> [4&g -> 1&b,
2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]");
tests/code_gen/test-bfs.in:10:
tests/code_gen/test-bfs.in:11:graph gh = a -> 1$b ->1$e -> 4$g ->1$b + a
-> 1$b ->1$e ->2$c + a -> 5$c -> 1$g +a -> 5$c -> 1$f->1$c + a -> 3$d-
>2$c +a -> 3$d -> 3$f;
tests/code_gen/test-bfs.in:12:
tests/code_gen/test-bfs.in:13:print("Breath First Search from a");
tests/code_gen/test-bfs.in:14:list<node> res = gh.bfs(a);
tests/code_gen/test-bfs.in:15:print(res);
tests/code_gen/test-bfs.in:16:print("Set all Nodes to Unvisited");
tests/code_gen/test-bfs.in:17:gh.setAllUnvisited();
tests/code_gen/test-bfs.in:18:
tests/code_gen/test-bfs.in:19:print("Breath First Search from c");
tests/code_gen/test-bfs.in:20:list<node> ret = gh.bfs(c);
tests/code_gen/test-bfs.in:21:print(ret);
tests/code_gen/test-bfs.in:22:print("Set all Nodes to Unvisited");
tests/code_gen/test-bfs.in:23:gh.setAllUnvisited();
tests/code_gen/test-list2.out:1:------------------test contains,
concatenate------------------
tests/code_gen/test-list2.out:2:test whether list contains 2, 4
tests/code_gen/test-list2.out:3:true
tests/code_gen/test-list2.out:4:false
tests/code_gen/test-list2.out:5:print concatenate [1,2,3] and [4,5,6]
tests/code_gen/test-list2.out:6:[1, 2, 3, 4, 5, 6]
tests/code_gen/test-list2.out:7:----------------------------------------
--------------------
tests/code_gen/test-list2.out:8:tests/test2/test-linkGraph.in:1:node a =
node("A");
tests/test2/test-linkGraph.in:2:node b = node("B");
tests/test2/test-linkGraph.in:3:node c = node("C");
```

```
tests/test2/test-linkGraph.in:4:node d = node("D");
tests/test2/test-linkGraph.in:5:node e = node("E");
tests/test2/test-linkGraph.in:6:
tests/test2/test-linkGraph.in:7:print("Using + to link graph");
tests/test2/test-linkGraph.in:8:
tests/test2/test-linkGraph.in:9:print("Link graphs: A -> 0$B ->3$E +  C -
> 1$A + A -> 2$D");
tests/test2/test-linkGraph.in:10:print("Shared nodes: A");
tests/test2/test-linkGraph.in:11:
tests/test2/test-linkGraph.in:12:graph g = a -> 0$b ->3$e +  c -> 1$a + a
-> 2$d;
tests/test2/test-linkGraph.in:13:
tests/test2/test-linkGraph.in:14:print(g);
tests/test2/test-linkGraph.in:15:tests/output/test-print.out:1:23
tests/output/test-print.out:2:-1.200000
tests/output/test-print.out:3:Hello, World
tests/output/test-print.out:4:true
tests/output/test-print.out:5:true
tests/output/test-print.out:6:[1, 2, 3]
tests/output/test-print.out:7:node  0: 10
tests/output/test-print.out:8:{a: 1, b: 2}
tests/output/test-print.out:9:4115 ** 20.17 ** GIRAPHE
tests/output/test-print.out:10:tests/output/test-while.out:1:0
tests/output/test-while.out:2:1
tests/output/test-while.out:3:true
tests/output/test-while.out:4:tests/output/test-list.out:1:--------------
----general method for list------------------
tests/output/test-list.out:2:[1, 2, 3]
tests/output/test-list.out:3:[1.000000, 2.000000, 3.000000]
tests/output/test-list.out:4:[a, b, c]
tests/output/test-list.out:5:[true, false, true]
tests/output/test-list.out:6:[node  2: 7
tests/output/test-list.out:7:node  1: 8
tests/output/test-list.out:8:node  0: 9
tests/output/test-list.out:9:]
tests/output/test-list.out:10:[--------------------------------------
tests/output/test-list.out:11:Nodes:
tests/output/test-list.out:12:node  2: 7
tests/output/test-list.out:13:node  1: 8
tests/output/test-list.out:14:
tests/output/test-list.out:15:Edges:
tests/output/test-list.out:16:edge  2 ->  1: 0
tests/output/test-list.out:17:--------------------------------------
tests/output/test-list.out:18:--------------------------------------
tests/output/test-list.out:19:Nodes:
tests/output/test-list.out:20:node  1: 8
tests/output/test-list.out:21:node  0: 9
tests/output/test-list.out:22:
tests/output/test-list.out:23:Edges:
tests/output/test-list.out:24:edge  1 ->  0: 0
tests/output/test-list.out:25:--------------------------------------
tests/output/test-list.out:26:--------------------------------------
tests/output/test-list.out:27:Nodes:
tests/output/test-list.out:28:node  0: 9
tests/output/test-list.out:29:node  2: 7
tests/output/test-list.out:30:
tests/output/test-list.out:31:Edges:
tests/output/test-list.out:32:edge  0 ->  2: 0
tests/output/test-list.out:33:--------------------------------------
tests/output/test-list.out:34:]
```

```
tests/output/test-list.out:35:------------ test add, remove -------------
--
tests/output/test-list.out:36:remove element not existing a.remove(4),
list doesn't change
tests/output/test-list.out:37:[1, 2, 3]
tests/output/test-list.out:38:adding element using list.add()
tests/output/test-list.out:39:[1, 2, 3, 5]
tests/output/test-list.out:40:remove element using -
tests/output/test-list.out:41:[1, 3, 5]
tests/output/test-list.out:42:remove list to empty
tests/output/test-list.out:43:[]
tests/output/test-list.out:44:add element using +
tests/output/test-list.out:45:[6]
tests/output/test-list.out:46:-----------------------------------------
tests/output/test-list.out:47:------------ test size, get, set ----------
-----
tests/output/test-list.out:48:[1.000000, 2.000000, 3.000000]
tests/output/test-list.out:49:Size:
tests/output/test-list.out:50:3
tests/output/test-list.out:51:Get first element:
tests/output/test-list.out:52:1.000000
tests/output/test-list.out:53:Set function:
tests/output/test-list.out:54:[1.000000, 5.000000, 3.000000]
tests/output/test-list.out:55:-----------------------------------------
-----
tests/output/test-list.out:56:tests/output/test-if.out:1:small
tests/output/test-if.out:2:12.000000
tests/output/test-if.out:3:true
tests/output/test-if.out:4:tests/output/test-arith.out:1:3
tests/output/test-arith.out:2:-1
tests/output/test-arith.out:3:75
tests/output/test-arith.out:4:3
tests/output/test-arith.out:5:2.050000
tests/output/test-arith.out:6:0.400000
tests/output/test-arith.out:7:2.500000
tests/output/test-arith.out:8:6.700000
tests/output/test-arith.out:9:0
tests/output/test-arith.out:10:4
tests/output/test-arith.out:11:tests/output/test-node-function.out:1:node
3: 1
tests/output/test-node-function.out:2:node  2: GIRAPHE
tests/output/test-node-function.out:3:node  1: false
tests/output/test-node-function.out:4:Is Node d Visited?
tests/output/test-node-function.out:5:false
tests/output/test-node-function.out:6:Set Node d to Visited
tests/output/test-node-function.out:7:true
tests/output/test-node-
function.out:8:tests/test_semantic.sh:1:#!/bin/bash
tests/test_semantic.sh:2:
tests/test_semantic.sh:3:NC='\033[0m'
tests/test_semantic.sh:4:CYAN='\033[0;36m'
tests/test_semantic.sh:5:GREEN='\033[0;32m'
tests/test_semantic.sh:6:RED='\033[0;31m'
tests/test_semantic.sh:7:
tests/test_semantic.sh:8:result=true
tests/test_semantic.sh:9:
tests/test_semantic.sh:10:INPUT_FILES="semantic_check/*.in"
tests/test_semantic.sh:11:printf "${CYAN}Running Semantic Check
tests...\n${NC}"
tests/test_semantic.sh:12:
```

```
tests/test_semantic.sh:13:for input_file in $INPUT_FILES; do
tests/test_semantic.sh:14:    output_file=${input_file/.in/.out}
tests/test_semantic.sh:15:    semantic_check/semantic_check < $input_file
| cmp -s $output_file -
tests/test_semantic.sh:16:    if [ "$?" -eq 0 ]; then
tests/test_semantic.sh:17:       printf "%-65s ${GREEN}SUCCESS\n${NC}" "
- checking $input_file..."
tests/test_semantic.sh:18:    else
tests/test_semantic.sh:19:       printf "%-65s ${RED}ERROR\n${NC}" "  -
checking $input_file..." 1>&2
tests/test_semantic.sh:20:        result=false
tests/test_semantic.sh:21:    fi
tests/test_semantic.sh:22:done
tests/test_semantic.sh:23:
tests/test_semantic.sh:24:exit 0
tests/test_semantic.sh:25:
tests/test_semantic.sh:26:# if $result; then
tests/test_semantic.sh:27:#     exit 0
tests/test_semantic.sh:28:# else
tests/test_semantic.sh:29:#     exit 1
tests/test_semantic.sh:30:# fi
tests/test_semantic.sh:31:tests/bfs.sh:1:#!/bin/bash
tests/bfs.sh:2:
tests/bfs.sh:3:NC='\033[0m'
tests/bfs.sh:4:CYAN='\033[0;36m'
tests/bfs.sh:5:GREEN='\033[0;32m'
tests/bfs.sh:6:RED='\033[0;31m'
tests/bfs.sh:7:
tests/bfs.sh:8:result=true
tests/bfs.sh:9:
tests/bfs.sh:10:INPUT_FILES="bfs/*.in"
tests/bfs.sh:11:printf "${CYAN}Running code_gen tests...\n${NC}"
tests/bfs.sh:12:
tests/bfs.sh:13:
tests/bfs.sh:14:
tests/bfs.sh:15:for input_file in $INPUT_FILES; do
tests/bfs.sh:16:    output_file=${input_file/.in/.out}
tests/bfs.sh:17:    sh ./giraphe.sh $input_file | cat > bfs.out
tests/bfs.sh:18:
tests/bfs.sh:19:done
tests/bfs.sh:20:
tests/bfs.sh:21:exit 0
tests/bfs.sh:22:
tests/bfs.sh:23:# if $result; then
tests/bfs.sh:24:#     exit 0
tests/bfs.sh:25:# else
tests/bfs.sh:26:#     exit 1
tests/bfs.sh:27:# fi
tests/bfs.sh:28:tests/bfs/test-bfs.in:1:node a = node("a");
tests/bfs/test-bfs.in:2:node b = node("b");
tests/bfs/test-bfs.in:3:node c = node("c");
tests/bfs/test-bfs.in:4:node d = node("d");
tests/bfs/test-bfs.in:5:node e = node("e");
tests/bfs/test-bfs.in:6:node f = node("f");
tests/bfs/test-bfs.in:7:node g = node("g");
tests/bfs/test-bfs.in:8:
tests/bfs/test-bfs.in:9:print("a -> [1&b -> 1&e -> [4&g -> 1&b, 2&c], 5&c
-> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]");
tests/bfs/test-bfs.in:10:
```

```
tests/bfs/test-bfs.in:11:graph gh = a -> 1$b ->1$e -> 4$g ->1$b + a ->
1$b ->1$e ->2$c + a -> 5$c -> 1$g +a -> 5$c -> 1$f->1$c + a -> 3$d->2$c
+a -> 3$d -> 3$f;
tests/bfs/test-bfs.in:12:
tests/bfs/test-bfs.in:13:print("Breath First Search from a");
tests/bfs/test-bfs.in:14:list<node> res = gh.bfs(a);
tests/bfs/test-bfs.in:15:print(res);
tests/bfs/test-bfs.in:16:print("Set all Nodes to Unvisited");
tests/bfs/test-bfs.in:17:gh.setAllUnvisited();
tests/bfs/test-bfs.in:18:
tests/bfs/test-bfs.in:19:print("Breath First Search from c");
tests/bfs/test-bfs.in:20:list<node> ret = gh.bfs(c);
tests/bfs/test-bfs.in:21:print(ret);
tests/bfs/test-bfs.in:22:print("Set all Nodes to Unvisited");
tests/bfs/test-bfs.in:23:gh.setAllUnvisited();
tests/test2.out:1:Using + to link graph
tests/test2.out:2:Link graphs: A -> 0$B ->3$E +  C -> 1$A + A -> 2$D
tests/test2.out:3:Shared nodes: A
tests/test2.out:4:-------------------------------------
tests/test2.out:5:node B
tests/test2.out:6:node E
tests/test2.out:7:node A
tests/test2.out:8:node C
tests/test2.out:9:node D
tests/test2.out:10:edge B ->E: 3
tests/test2.out:11:edge A ->B: 0
tests/test2.out:12:edge C ->A: 1
tests/test2.out:13:edge A ->D: 2
tests/test2.out:14:-------------------------------------
tests/test2.out:15:tests/new.sh:1:#!/bin/bash
tests/new.sh:2:
tests/new.sh:3:NC='\033[0m'
tests/new.sh:4:CYAN='\033[0;36m'
tests/new.sh:5:GREEN='\033[0;32m'
tests/new.sh:6:RED='\033[0;31m'
tests/new.sh:7:
tests/new.sh:8:result=true
tests/new.sh:9:
tests/new.sh:10:INPUT_FILES="test2/*.in"
tests/new.sh:11:printf "${CYAN}Running code_gen tests...\n${NC}"
tests/new.sh:12:
tests/new.sh:13:
tests/new.sh:14:
tests/new.sh:15:for input_file in $INPUT_FILES; do
tests/new.sh:16:    output_file=${input_file/.in/.out}
tests/new.sh:17:    sh ./giraphe.sh $input_file | cat > test2.out
tests/new.sh:18:
tests/new.sh:19:done
tests/new.sh:20:
tests/new.sh:21:exit 0
tests/new.sh:22:
tests/new.sh:23:# if $result; then
tests/new.sh:24:#     exit 0
tests/new.sh:25:# else
tests/new.sh:26:#     exit 1
tests/new.sh:27:# fi
tests/new.sh:28:tests/tests/test-list.in:1:print("------------------
general method for list------------------");
tests/tests/test-list.in:2:
tests/tests/test-list.in:3:list<int> a = [1, 2, 3];
```

```
tests/tests/test-list.in:4:list<float> b = [1.0, 2.0, 3.0];
tests/tests/test-list.in:5:list<string> c = ["a", "b", "c"];
tests/tests/test-list.in:6:list<bool> d = [true, false, true];
tests/tests/test-list.in:7:
tests/tests/test-list.in:8:print(a);
tests/tests/test-list.in:9:print(b);
tests/tests/test-list.in:10:print(c);
tests/tests/test-list.in:11:print(d);
tests/tests/test-list.in:12:
tests/tests/test-list.in:13:node n1 = node(7);
tests/tests/test-list.in:14:node n2 = node(8);
tests/tests/test-list.in:15:node n3 = node(9);
tests/tests/test-list.in:16:list<node> nd = [n1, n2, n3];
tests/tests/test-list.in:17:print(nd);
tests/tests/test-list.in:18:
tests/tests/test-list.in:19:list<graph> gp = [n1->n2, n2->n3, n3->n1];
tests/tests/test-list.in:20:print(gp);
tests/tests/test-list.in:21:
tests/tests/test-list.in:22:print("------------ test add, remove --------
-------");
tests/tests/test-list.in:23:
tests/tests/test-list.in:24:print("remove element not existing
a.remove(4), list doesn't change");
tests/tests/test-list.in:25:a.remove(4);
tests/tests/test-list.in:26:print(a);
tests/tests/test-list.in:27:print("adding element using list.add()");
tests/tests/test-list.in:28:a.add(5);
tests/tests/test-list.in:29:print(a);
tests/tests/test-list.in:30:print("remove element using -");
tests/tests/test-list.in:31:a=a-2;
tests/tests/test-list.in:32:print(a);
tests/tests/test-list.in:33:a=a-1;
tests/tests/test-list.in:34:a=a-5;
tests/tests/test-list.in:35:print("remove list to empty");
tests/tests/test-list.in:36:a=a-3;
tests/tests/test-list.in:37:print(a);
tests/tests/test-list.in:38:print("add element using +");
tests/tests/test-list.in:39:a=a+6;
tests/tests/test-list.in:40:print(a);
tests/tests/test-list.in:41:print("--------------------------------------
-----");
tests/tests/test-list.in:42:
tests/tests/test-list.in:43:print("------------ test size, get, set -----
----------");
tests/tests/test-list.in:44:print(b);
tests/tests/test-list.in:45:print("Size:");
tests/tests/test-list.in:46:print(b.size());
tests/tests/test-list.in:47:print("Get first element:");
tests/tests/test-list.in:48:print(b.get(0));
tests/tests/test-list.in:49:print("Set function:");
tests/tests/test-list.in:50:b.set(1,5.0);
tests/tests/test-list.in:51:print(b);
tests/tests/test-list.in:52:print("--------------------------------------
----------");
tests/tests/test-if.in:1:int a = 0;
tests/tests/test-if.in:2:if (a < 1) {
tests/tests/test-if.in:3:   print("small");
tests/tests/test-if.in:4:}else {
tests/tests/test-if.in:5:   print("large");
tests/tests/test-if.in:6:}
```

```
tests/tests/test-if.in:7:
tests/tests/test-if.in:8:float b = 12;
tests/tests/test-if.in:9:if (b > 10) {
tests/tests/test-if.in:10:  print(b);
tests/tests/test-if.in:11:}
tests/tests/test-if.in:12:
tests/tests/test-if.in:13:bool c = false;
tests/tests/test-if.in:14:if (!c) {
tests/tests/test-if.in:15:  print("true");
tests/tests/test-if.in:16:}
tests/tests/test-arith.in:1:print(1+2);
tests/tests/test-arith.in:2:print(2-3);
tests/tests/test-arith.in:3:print(5*15);
tests/tests/test-arith.in:4:print(9/3);
tests/tests/test-arith.in:5:print(8.2/4);
tests/tests/test-arith.in:6:print(5%2.3);
tests/tests/test-arith.in:7:print(1.2+1.3);
tests/tests/test-arith.in:8:print(8.2-1.5);
tests/tests/test-arith.in:9:
tests/tests/test-arith.in:10:int a = 0;
tests/tests/test-arith.in:11:int foo(int b) {
tests/tests/test-arith.in:12:  int a = 1;
tests/tests/test-arith.in:13:  int bar() {
tests/tests/test-arith.in:14:    return a+b;
tests/tests/test-arith.in:15:  }
tests/tests/test-arith.in:16:  return bar();
tests/tests/test-arith.in:17:}
tests/tests/test-arith.in:18:print(a);
tests/tests/test-arith.in:19:print(foo(3));
tests/tests/test-arith.in:20:tests/tests/test-print.in:1:print(23);
tests/tests/test-print.in:2:print(-1.2);
tests/tests/test-print.in:3:print("Hello, World");
tests/tests/test-print.in:4:print(true);
tests/tests/test-print.in:5:print(1>0.2);
tests/tests/test-print.in:6:print([1, 2, 3]);
tests/tests/test-print.in:7:print(node(10));
tests/tests/test-print.in:8:print({"a": 1, "b": 2});
tests/tests/test-print.in:9:
tests/tests/test-print.in:10:int a = 4115;
tests/tests/test-print.in:11:float b = 20.17;
tests/tests/test-print.in:12:string c = "GIRAPHE";
tests/tests/test-print.in:13:
tests/tests/test-print.in:14:printf("%d ** %.2f ** %s\n", a, b, c);
tests/tests/test-node-function.in:1:node a = node(1);
tests/tests/test-node-function.in:2:node b = node("GIRAPHE");
tests/tests/test-node-function.in:3:node c = node(1>2);
tests/tests/test-node-function.in:4:print(a);
tests/tests/test-node-function.in:5:print(b);
tests/tests/test-node-function.in:6:print(c);
tests/tests/test-node-function.in:7:
tests/tests/test-node-function.in:8:node d = node("hi");
tests/tests/test-node-function.in:9:print("Is Node d Visited?");
tests/tests/test-node-function.in:10:print(d.isVisited());
tests/tests/test-node-function.in:11:print("Set Node d to Visited");
tests/tests/test-node-function.in:12:d.setVisited();
tests/tests/test-node-function.in:13:print(d.isVisited());
tests/tests/test-node-function.in:14:
tests/tests/test-node-function.in:15:tests/tests/test-while.in:1:int a =
0;
tests/tests/test-while.in:2:while (a < 2) {
```

```
tests/tests/test-while.in:3:print(a);
tests/tests/test-while.in:4:a = a + 1;
tests/tests/test-while.in:5:}
tests/tests/test-while.in:6:
tests/tests/test-while.in:7:
tests/tests/test-while.in:8:bool c = true;
tests/tests/test-while.in:9:while (c) {
tests/tests/test-while.in:10:    print(c);
tests/tests/test-while.in:11:    c = ! c;
tests/tests/test-while.in:12:}
tests/giraphe.sh:1:# Check whether the file "utils.bc" exist
tests/giraphe.sh:2:file="lib.bc"
tests/giraphe.sh:3:if [ ! -e "$file" ]
tests/giraphe.sh:4:then
tests/giraphe.sh:5:    clang -emit-llvm -o lib.bc -c ../lib.c -Wno-
varargs
tests/giraphe.sh:6:fi
tests/giraphe.sh:7:
tests/giraphe.sh:8:if [ $# -eq 1 ]
tests/giraphe.sh:9:then
tests/giraphe.sh:10:    ../giraphe.native <$1 >a.ll
tests/giraphe.sh:11:else
tests/giraphe.sh:12:    ../giraphe.native $1 <$2 >a.ll
tests/giraphe.sh:13:fi
tests/giraphe.sh:14:clang -Wno-override-module lib.bc a.ll -o $1.exe
tests/giraphe.sh:15:./$1.exe
tests/giraphe.sh:16:rm a.ll
tests/giraphe.sh:17:rm ./$1.exe
tests/giraphe.sh:18:
tests/giraphe.sh:19:# /usr/local/opt/llvm38/bin/clang-3.8
tests/giraphe.sh:20:tests/parser/_graph.in:1:a->b;
tests/parser/_graph.in:2:tests/parser/_function.in:1:int func(int a, int
b) {
tests/parser/_function.in:2:    int c = 0;
tests/parser/_function.in:3:    return a + b + c;
tests/parser/_function.in:4:}
tests/parser/_function.in:5:func(1, 2);
tests/parser/_function.in:6:func(a, b);
tests/parser/_node.in:1:node(1);
tests/parser/_node.in:2:node("a");
tests/parser/_node.in:3:node(false);
tests/parser/_node.in:4:node([1,"2"]);
tests/parser/_node.in:5:tests/parser/_node.out:1:Expr(Node(Num_Lit(1)));
tests/parser/_node.out:2:Expr(Node(String_Lit(a)));
tests/parser/_node.out:3:Expr(Node(Bool_lit(false)));
tests/parser/_node.out:4:Expr(Node(List(Num_Lit(1), String_Lit(2))));
tests/parser/_node.out:5:tests/parser/parserize.ml:1:open Ast
tests/parser/parserize.ml:2:open Printf
tests/parser/parserize.ml:3:
tests/parser/parserize.ml:4:(* Unary operators *)
tests/parser/parserize.ml:5:let txt_of_unop = function
tests/parser/parserize.ml:6:  | Not -> "Not"
tests/parser/parserize.ml:7:  | Neg -> "Sub"
tests/parser/parserize.ml:8:
tests/parser/parserize.ml:9:(* Binary operators *)
tests/parser/parserize.ml:10:let txt_of_binop = function
tests/parser/parserize.ml:11:  (* Arithmetic *)
tests/parser/parserize.ml:12:  | Add -> "Add"
tests/parser/parserize.ml:13:  | Sub -> "Sub"
tests/parser/parserize.ml:14:  | Mult -> "Mult"
```

```
tests/parser/parserize.ml:15:  | Div -> "Div"
tests/parser/parserize.ml:16:  | Mod -> "Mod"
tests/parser/parserize.ml:17:  (* Boolean *)
tests/parser/parserize.ml:18:  | Or -> "Or"
tests/parser/parserize.ml:19:  | And -> "And"
tests/parser/parserize.ml:20:  | Equal -> "Equal"
tests/parser/parserize.ml:21:  | Neq -> "Neq"
tests/parser/parserize.ml:22:  | Less -> "Less"
tests/parser/parserize.ml:23:  | Leq -> "Leq"
tests/parser/parserize.ml:24:  | Greater -> "Greater"
tests/parser/parserize.ml:25:  | Geq -> "Geq"
tests/parser/parserize.ml:26:
tests/parser/parserize.ml:27:let txt_of_var_type = function
tests/parser/parserize.ml:28:  | Void_t -> "void"
tests/parser/parserize.ml:29:  | Null_t -> "null"
tests/parser/parserize.ml:30:  | Int_t -> "int"
tests/parser/parserize.ml:31:  | Float_t -> "float"
tests/parser/parserize.ml:32:  | String_t -> "string"
tests/parser/parserize.ml:33:  | Bool_t -> "bool"
tests/parser/parserize.ml:34:  | Node_t -> "node"
tests/parser/parserize.ml:35:  | Graph_t -> "graph"
tests/parser/parserize.ml:36:  | Dict_Int_t -> "dict<int>"
tests/parser/parserize.ml:37:  | Dict_Float_t -> "dict<float>"
tests/parser/parserize.ml:38:  | Dict_String_t -> "dict<string>"
tests/parser/parserize.ml:39:  | Dict_Node_t -> "dict<node>"
tests/parser/parserize.ml:40:  | Dict_Graph_t -> "dict<graph>"
tests/parser/parserize.ml:41:  | List_Int_t -> "list<int>"
tests/parser/parserize.ml:42:  | List_Float_t -> "list<float>"
tests/parser/parserize.ml:43:  | List_Bool_t -> "list<bool>"
tests/parser/parserize.ml:44:  | List_String_t -> "list<string>"
tests/parser/parserize.ml:45:  | List_Node_t -> "list<node>"
tests/parser/parserize.ml:46:  | List_Graph_t -> "list<graph>"
tests/parser/parserize.ml:47:
tests/parser/parserize.ml:48:let txt_of_formal = function
tests/parser/parserize.ml:49:| Formal(vtype, name) -> sprintf "Formal(%s,
%s)" (txt_of_var_type vtype) name
tests/parser/parserize.ml:50:
tests/parser/parserize.ml:51:
tests/parser/parserize.ml:52:let txt_of_formal_list formals =
tests/parser/parserize.ml:53:  let rec aux acc = function
tests/parser/parserize.ml:54:    | [] -> sprintf "%s" (String.concat ", "
(List.rev acc))
tests/parser/parserize.ml:55:    | fml :: tl -> aux (txt_of_formal fml ::
acc) tl
tests/parser/parserize.ml:56:  in aux [] formals
tests/parser/parserize.ml:57:
tests/parser/parserize.ml:58:let txt_of_num = function
tests/parser/parserize.ml:59:  | Num_Int(x) -> string_of_int x
tests/parser/parserize.ml:60:  | Num_Float(x) -> string_of_float x
tests/parser/parserize.ml:61:
tests/parser/parserize.ml:62:(* Expressions *)
tests/parser/parserize.ml:63:let rec txt_of_expr = function
tests/parser/parserize.ml:64:  | Num_Lit(x) -> sprintf "Num_Lit(%s)"
(txt_of_num x)
tests/parser/parserize.ml:65:  | Bool_lit(x) -> sprintf "Bool_lit(%s)"
(string_of_bool x)
tests/parser/parserize.ml:66:  | String_Lit(x) -> sprintf
"String_Lit(%s)" x
tests/parser/parserize.ml:67:  | Null -> sprintf "Null"
```

```
tests/parser/parserize.ml:68:  | Node(x) -> sprintf "Node(%s)"
(txt_of_expr x)
tests/parser/parserize.ml:69:  | Unop(op, e) -> sprintf "Unop(%s, %s)"
(txt_of_unop op) (txt_of_expr e)
tests/parser/parserize.ml:70:  | Binop(e1, op, e2) -> sprintf "Binop(%s,
%s, %s)"
tests/parser/parserize.ml:71:      (txt_of_expr e1) (txt_of_binop op)
(txt_of_expr e2)
tests/parser/parserize.ml:72:  | Graph_Link(e1, e2, e3) -> sprintf
"Graph_Link(%s, %s, WithEdge, %s)"
tests/parser/parserize.ml:73:      (txt_of_expr e1) (txt_of_expr e2)
(txt_of_expr e3)
tests/parser/parserize.ml:74:  | Id(x) -> sprintf "Id(%s)" x
tests/parser/parserize.ml:75:  | Assign(e1, e2) -> sprintf "Assign(%s,
%s)" e1 (txt_of_expr e2)
tests/parser/parserize.ml:76:  | Noexpr -> sprintf "Noexpression"
tests/parser/parserize.ml:77:  | ListP(l) -> sprintf "List(%s)"
(txt_of_list l)
tests/parser/parserize.ml:78:  | DictP(d) -> sprintf "Dict(%s)"
(txt_of_dict d)
tests/parser/parserize.ml:79:  | Call(f, args) -> sprintf "Call(%s,
[%s])" (f) (txt_of_list args)
tests/parser/parserize.ml:80:  | CallDefault(e, f, args) -> sprintf
"CallDefault(%s, %s, [%s])" (txt_of_expr e) f (txt_of_list args)
tests/parser/parserize.ml:81:
tests/parser/parserize.ml:82:(*Variable Declaration*)
tests/parser/parserize.ml:83:and txt_of_var_decl = function
tests/parser/parserize.ml:84:  | Local(var, name, e1) -> sprintf
"Local(%s, %s, %s)"
tests/parser/parserize.ml:85:    (txt_of_var_type var) name (txt_of_expr
e1)
tests/parser/parserize.ml:86:
tests/parser/parserize.ml:87:(* Lists *)
tests/parser/parserize.ml:88:and txt_of_list = function
tests/parser/parserize.ml:89:  | [] -> ""
tests/parser/parserize.ml:90:  | [x] -> txt_of_expr x
tests/parser/parserize.ml:91:  | _ as l -> String.concat ", " (List.map
txt_of_expr l)
tests/parser/parserize.ml:92:
tests/parser/parserize.ml:93:(* Dict *)
tests/parser/parserize.ml:94:and txt_of_dict_key_value = function
tests/parser/parserize.ml:95:  | (key, value) -> sprintf
"key:%s,value:%s" (txt_of_expr key) (txt_of_expr value)
tests/parser/parserize.ml:96:
tests/parser/parserize.ml:97:and txt_of_dict = function
tests/parser/parserize.ml:98:  | [] -> ""
tests/parser/parserize.ml:99:  | [x] -> txt_of_dict_key_value x
tests/parser/parserize.ml:100:  | _ as d -> String.concat ", " (List.map
txt_of_dict_key_value d)
tests/parser/parserize.ml:101:
tests/parser/parserize.ml:102:(* Functions Declaration *)
tests/parser/parserize.ml:103:and txt_of_func_decl f =
tests/parser/parserize.ml:104:  sprintf "%s %s (%s) {%s}"
tests/parser/parserize.ml:105:    (txt_of_var_type f.returnType) f.name
(txt_of_formal_list f.args) (txt_of_stmts f.body)
tests/parser/parserize.ml:106:
tests/parser/parserize.ml:107:(* Statements *)
tests/parser/parserize.ml:108:and txt_of_stmt = function
tests/parser/parserize.ml:109:  | Expr(expr) -> sprintf "Expr(%s);"
(txt_of_expr expr)
```

```
tests/parser/parserize.ml:110:  | Func(f) -> sprintf "Func(%s)"
(txt_of_func_decl f)
tests/parser/parserize.ml:111:  | Return(expr) -> sprintf "Return(%s);"
(txt_of_expr expr)
tests/parser/parserize.ml:112:  | For(e1,e2,e3,s) -> sprintf
"For(%s;%s;%s){%s}"
tests/parser/parserize.ml:113:    (txt_of_expr e1) (txt_of_expr e2)
(txt_of_expr e3) (txt_of_stmts s)
tests/parser/parserize.ml:114:  | If(e1,s1,s2) -> sprintf "If(%s){%s}
Else{%s}"
tests/parser/parserize.ml:115:    (txt_of_expr e1) (txt_of_stmts s1)
(txt_of_stmts s2)
tests/parser/parserize.ml:116:  | While(e1, s) -> sprintf "While(%s){%s}"
tests/parser/parserize.ml:117:    (txt_of_expr e1) (txt_of_stmts s)
tests/parser/parserize.ml:118:  | Var_dec(var) -> sprintf "Var_dec(%s);"
(txt_of_var_decl var)
tests/parser/parserize.ml:119:and txt_of_stmts stmts =
tests/parser/parserize.ml:120:  let rec aux acc = function
tests/parser/parserize.ml:121:      | [] -> sprintf "%s" (String.concat
"\n" (List.rev acc))
tests/parser/parserize.ml:122:      | stmt :: tl -> aux (txt_of_stmt stmt
:: acc) tl
tests/parser/parserize.ml:123:  in aux [] stmts
tests/parser/parserize.ml:124:
tests/parser/parserize.ml:125:(* Program entry point *)
tests/parser/parserize.ml:126:let _ =
tests/parser/parserize.ml:127:  let lexbuf = Lexing.from_channel stdin in
tests/parser/parserize.ml:128:  let program = Parser.program
Scanner.token lexbuf in
tests/parser/parserize.ml:129:  let result = txt_of_stmts program in
tests/parser/parserize.ml:130:  print_endline result
tests/parser/parserize.ml:131:tests/parser/_dict.out:1:Expr(Dict());
tests/parser/_dict.out:2:Expr(Dict(key:String_Lit(a),value:String_Lit(b))
);
tests/parser/_dict.out:3:Expr(Dict(key:String_Lit(a),value:String_Lit(b),
key:String_Lit(c),value:String_Lit(d)));
tests/parser/_dict.out:4:Expr(Dict(key:String_Lit(a),value:Num_Lit(1),
key:String_Lit(c),value:Bool_lit(true)));
tests/parser/_dict.out:5:tests/parser/_list.in:1:[];
tests/parser/_list.in:2:[1,2,3];
tests/parser/_list.in:3:[1,"2",2.0,true,false,1+1];
tests/parser/_list.in:4:tests/parser/_arithmetic.out:1:Expr(Binop(Num_Lit
(1), Add, Num_Lit(5.4)));
tests/parser/_arithmetic.out:2:Expr(Binop(Num_Lit(1), Sub,
Num_Lit(5.4)));
tests/parser/_arithmetic.out:3:Expr(Binop(Num_Lit(1), Mult,
Num_Lit(5.4)));
tests/parser/_arithmetic.out:4:Expr(Binop(Num_Lit(1), Div,
Num_Lit(5.4)));
tests/parser/_arithmetic.out:5:Expr(Binop(Num_Lit(1), Mod,
Num_Lit(5.4)));
tests/parser/_arithmetic.out:6:Expr(Unop(Sub, Num_Lit(42)));
tests/parser/_arithmetic.out:7:Expr(Binop(Num_Lit(1), Add, Unop(Sub,
Num_Lit(43))));
tests/parser/_arithmetic.out:8:Expr(Binop(Binop(Num_Lit(1), Mult,
Num_Lit(2)), Add, Binop(Num_Lit(3), Mult, Num_Lit(4))));
tests/parser/_arithmetic.out:9:Expr(Binop(Binop(Binop(Num_Lit(1), Div,
Num_Lit(2)), Mod, Num_Lit(3)), Mod, Num_Lit(4)));
tests/parser/_arithmetic.out:10:Expr(Binop(Binop(Num_Lit(1), Add,
Num_Lit(2)), Sub, Binop(Num_Lit(3), Div, Num_Lit(4))));
```

```
tests/parser/_arithmetic.out:11:Expr(Binop(Binop(Num_Lit(1), Mult,
Num_Lit(2)), Add, Num_Lit(3)));
tests/parser/_arithmetic.out:12:tests/parser/_conditional.in:1:aList =
["str1","str2","str3"];
tests/parser/_conditional.in:2:int i;
tests/parser/_conditional.in:3:for(i = 0; i<= 5; i=i+1){
tests/parser/_conditional.in:4:    if (str == "str2"){
tests/parser/_conditional.in:5:        3+2;
tests/parser/_conditional.in:6:    }
tests/parser/_conditional.in:7:    else{
tests/parser/_conditional.in:8:        /* do something */
tests/parser/_conditional.in:9:    }
tests/parser/_conditional.in:10:}
tests/parser/_conditional.in:11:tests/parser/_function.out:1:Func(int
func (Formal(int, a), Formal(int, b)) {Var_dec(Local(int, c,
Num_Lit(0)));
tests/parser/_function.out:2:Return(Binop(Binop(Id(a), Add, Id(b)), Add,
Id(c)));})
tests/parser/_function.out:3:Expr(Call(func, [Num_Lit(1), Num_Lit(2)]));
tests/parser/_function.out:4:Expr(Call(func, [Id(a), Id(b)]));
tests/parser/_function.out:5:tests/parser/_literals.out:1:Expr(Num_Lit(20
));
tests/parser/_literals.out:2:Expr(Num_Lit(20.));
tests/parser/_literals.out:3:Expr(String_Lit(str));
tests/parser/_literals.out:4:Expr(Bool_lit(true));
tests/parser/_literals.out:5:Expr(Bool_lit(false));
tests/parser/_literals.out:6:tests/parser/_conditional.out:1:Expr(Assign(
aList, List(String_Lit(str1), String_Lit(str2), String_Lit(str3))));
tests/parser/_conditional.out:2:Var_dec(Local(int, i, Noexpression));
tests/parser/_conditional.out:3:For(Assign(i, Num_Lit(0));Binop(Id(i),
Leq, Num_Lit(5));Assign(i, Binop(Id(i), Add,
Num_Lit(1))))){If(Binop(Id(str), Equal,
String_Lit(str2))){Expr(Binop(Num_Lit(3), Add, Num_Lit(2)));} Else{}}
tests/parser/_conditional.out:4:tests/parser/_dict.in:1:{};
tests/parser/_dict.in:2:{"a":"b"};
tests/parser/_dict.in:3:{"a":"b","c":"d"};
tests/parser/_dict.in:4:{"a":1,"c":true};
tests/parser/_dict.in:5:tests/parser/_arithmetic.in:1:1 + 5.4;
tests/parser/_arithmetic.in:2:1 - 5.4;
tests/parser/_arithmetic.in:3:1 * 5.4;
tests/parser/_arithmetic.in:4:1 / 5.4;
tests/parser/_arithmetic.in:5:1 % 5.4;
tests/parser/_arithmetic.in:6:-42;
tests/parser/_arithmetic.in:7:1 + -43;
tests/parser/_arithmetic.in:8:1 * 2 + 3 * 4;
tests/parser/_arithmetic.in:9:1 / 2 % 3 % 4;
tests/parser/_arithmetic.in:10:1 + 2 - 3 / 4;
tests/parser/_arithmetic.in:11:1 * 2 + 3;
tests/parser/_graph.out:1:Expr(Graph_Link(Id(a), RLink, Id(b), WithEdge,
Null));
tests/parser/_graph.out:2:tests/parser/Makefile:1:OCAMLC = ocamlc
tests/parser/Makefile:2:OBJS = ../../_build/parser.cmo
../../_build/scanner.cmo parserize.cmo
tests/parser/Makefile:3:INCLUDES = -I ../../_build
tests/parser/Makefile:4:
tests/parser/Makefile:5:default: parserize
tests/parser/Makefile:6:
tests/parser/Makefile:7:all:
tests/parser/Makefile:8:    cd ..; make all
tests/parser/Makefile:9:
```

```
tests/parser/Makefile:10:parserize: $(OBJS)
tests/parser/Makefile:11:	$(OCAMLC) $(INCLUDES) -o parserize $(OBJS)
tests/parser/Makefile:12:
tests/parser/Makefile:13:%.cmo: %.ml
tests/parser/Makefile:14:	$(OCAMLC) $(INCLUDES) -c $<
tests/parser/Makefile:15:
tests/parser/Makefile:16:%.cmi: %.mli
tests/parser/Makefile:17:	$(OCAMLC) $(INCLUDES) -c $<
tests/parser/Makefile:18:
tests/parser/Makefile:19:.PHONY: clean
tests/parser/Makefile:20:clean:
tests/parser/Makefile:21:	rm -f parserize *.cmo *.cmi
tests/parser/Makefile:22:
tests/parser/Makefile:23:# parserize.cmo: ../../_build/parser.cmi
../../_build/ast.cmi
tests/parser/Makefile:24:# parserize.cmx: ../../_build/parser.cmi
../../_build/ast.cmi
tests/parser/Makefile:25:tests/parser/_literals.in:1:20;
tests/parser/_literals.in:2:20.0;
tests/parser/_literals.in:3:"str";
tests/parser/_literals.in:4:true;
tests/parser/_literals.in:5:false;
tests/parser/_literals.in:6:tests/parser/_list.out:1:Expr(List());
tests/parser/_list.out:2:Expr(List(Num_Lit(1), Num_Lit(2), Num_Lit(3)));
tests/parser/_list.out:3:Expr(List(Num_Lit(1), String_Lit(2),
Num_Lit(2.), Bool_lit(true), Bool_lit(false), Binop(Num_Lit(1), Add,
Num_Lit(1))));
tests/parser/_list.out:4:tests/semantic_check/_illegal_assignment.out:1:U
h..oh! Assigning int = float in a = 3.1 is illegal :(
tests/semantic_check/_illegal_assignment.out:2:tests/semantic_check/_inva
lid_graph_size.in:1:node a = node("1");
tests/semantic_check/_invalid_graph_size.in:2:node b = node("1");
tests/semantic_check/_invalid_graph_size.in:3:node c = node("1");
tests/semantic_check/_invalid_graph_size.in:4:graph g = a -> b -> c;
tests/semantic_check/_invalid_graph_size.in:5:g.size(1);
tests/semantic_check/_invalid_graph_size.in:6:tests/semantic_check/_undec
lared_variable.in:1:int a = 1;
tests/semantic_check/_undeclared_variable.in:2:a + b;
tests/semantic_check/_illegal_binary_operation5.in:1:1.1%1;
tests/semantic_check/_invalid_graph_link.in:1:int a = 1;
tests/semantic_check/_invalid_graph_link.in:2:node b = node("1");
tests/semantic_check/_invalid_graph_link.in:3:a -> b;
tests/semantic_check/_invalid_graph_link.in:4:tests/semantic_check/_illeg
al_binary_operation4.out:1:Uh..oh! Binary Operator bool and int in true
and 1 is illegal :(
tests/semantic_check/_illegal_binary_operation4.out:2:tests/semantic_chec
k/_unmatched_func_arg_len.out:1:Uh.. This args length does not match in
function call: main.foo :(
tests/semantic_check/_unmatched_func_arg_len.out:2:tests/semantic_check/_
invalid_graph_nodes.out:1:Uh.. node method do not take arguments: g
tests/semantic_check/_invalid_graph_nodes.out:2:tests/semantic_check/_inv
alid_graph_size.out:1:Uh..size method do not take arguments: g
tests/semantic_check/_invalid_graph_size.out:2:tests/semantic_check/_inco
mpatible_func_arg_type.in:1:int foo(int a, int b) {
tests/semantic_check/_incompatible_func_arg_type.in:2:    return a + b;
tests/semantic_check/_incompatible_func_arg_type.in:3:}
tests/semantic_check/_incompatible_func_arg_type.in:4:
tests/semantic_check/_incompatible_func_arg_type.in:5:foo("1",3);
tests/semantic_check/_illegal_unary_operation2.out:1:Uh..oh! Unary
Operator not int in not 1 is illegal :(
```

```
tests/semantic_check/_illegal_unary_operation2.out:2:tests/semantic_check
/_illegal_binary_operation1.out:1:Uh..oh! Binary Operator int + string in
1 + hh is illegal :(
tests/semantic_check/_illegal_binary_operation1.out:2:tests/semantic_chec
k/_invalid_list_set1.out:1:Uh.. list set method should only take two
argument of type int and int: a :(
tests/semantic_check/_invalid_list_set1.out:2:tests/semantic_check/_inval
id_list_size.out:1:Uh.. list size method do not take arguments: a :(
tests/semantic_check/_invalid_list_size.out:2:tests/semantic_check/_illeg
al_binary_operation4.in:1:true && 1;
tests/semantic_check/_illegal_binary_operation4.in:2:tests/semantic_check
/_invalid_list_get2.out:1:Uh..list get method should only take one
argument of type int: a :(
tests/semantic_check/_invalid_list_get2.out:2:tests/semantic_check/_inval
id_graph_link.out:1:Uh.. left side of graph should be node type: a
tests/semantic_check/_invalid_graph_link.out:2:tests/semantic_check/_ille
gal_binary_operation1.in:1:1+"hh";
tests/semantic_check/_invalid_list_type1.in:1:list<int> a = [1,2];
tests/semantic_check/_invalid_list_type1.in:2:[a,a];
tests/semantic_check/_illegal_unary_operation2.in:1:! 1;
tests/semantic_check/_illegal_unary_operation2.in:2:tests/semantic_check/
semantic_check.ml:1:(* Program entry point *)
tests/semantic_check/semantic_check.ml:2:open Printf
tests/semantic_check/semantic_check.ml:3:
tests/semantic_check/semantic_check.ml:4:let _ =
tests/semantic_check/semantic_check.ml:5:    let lexbuf =
Lexing.from_channel stdin in
tests/semantic_check/semantic_check.ml:6:    let ast = Parser.program
Scanner.token lexbuf in
tests/semantic_check/semantic_check.ml:7:    let sast = Checker.convert
ast in
tests/semantic_check/semantic_check.ml:8:    try
tests/semantic_check/semantic_check.ml:9:        Semant.check sast
tests/semantic_check/semantic_check.ml:10:    with
tests/semantic_check/semantic_check.ml:11:        Semant.SemanticError(m)
-> print_endline m
tests/semantic_check/semantic_check.ml:12:        (*
Semant.SemanticError(m) -> raise (Failure(m)) *)
tests/semantic_check/semantic_check.ml:13:tests/semantic_check/_invalid_l
ist_size.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_size.in:2:a.size(1);
tests/semantic_check/_illegal_unary_operation1.out:1:Uh..oh! Unary
Operator - string in - hh is illegal :(
tests/semantic_check/_illegal_unary_operation1.out:2:tests/semantic_check
/_invalid_list_add1.out:1:Uh.. list add method should only take one
argument of type int: a :(
tests/semantic_check/_invalid_list_add1.out:2:tests/semantic_check/_inval
id_return_type.in:1:int foo() {
tests/semantic_check/_invalid_return_type.in:2:    return "1";
tests/semantic_check/_invalid_return_type.in:3:}
tests/semantic_check/_illegal_assignment.in:1:int a = 3.1;
tests/semantic_check/_invalid_graph_edges.out:1:Uh..edge method do not
accept arguments: g
tests/semantic_check/_invalid_graph_edges.out:2:tests/semantic_check/_inv
alid_list_set2.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_set2.in:2:a.set("1", 2);
tests/semantic_check/_invalid_list_get1.out:1:Uh..list get method should
only take one argument of type int: a :(
tests/semantic_check/_invalid_list_get1.out:2:tests/semantic_check/_undec
lared_variable.out:1:Uh..undeclared identifier b :(
```

```
tests/semantic_check/_undeclared_variable.out:2:tests/semantic_check/_ill
egal_binary_operation3.in:1:1 < "hh";
tests/semantic_check/_illegal_binary_operation2.in:1:1 == 1.1;
tests/semantic_check/_invalid_list_set2.out:1:Uh.. list set method should
only take two argument of type int and int: a :(
tests/semantic_check/_invalid_list_set2.out:2:tests/semantic_check/_inval
id_list_push2.out:1:Uh.. list push method should only take one argument
of type int: a :(
tests/semantic_check/_invalid_list_push2.out:2:tests/semantic_check/_inva
lid_expr_after_return.out:1:Hmm? Something follow with a return :(
tests/semantic_check/_invalid_expr_after_return.out:2:tests/semantic_chec
k/_invalid_list_add2.out:1:Uh.. list add method should only take one
argument of type int: a :(
tests/semantic_check/_invalid_list_add2.out:2:tests/semantic_check/_inval
id_list_type1.out:1:Uh.. This is a invalid list type: list<int> :(
tests/semantic_check/_invalid_list_type1.out:2:tests/semantic_check/_inva
lid_list_pop.out:1:Uh.. list pop method do not take arguments: a :(
tests/semantic_check/_invalid_list_pop.out:2:tests/semantic_check/_invali
d_return_type.out:1:Ops, Wrong function return type: string, it expect
int
tests/semantic_check/_invalid_return_type.out:2:tests/semantic_check/_inv
alid_empty_list.out:1:Uh..oh! This empty list declaration: list is
invalid :(
tests/semantic_check/_invalid_empty_list.out:2:tests/semantic_check/_inva
lid_graph_edges.in:1:node a = node("1");
tests/semantic_check/_invalid_graph_edges.in:2:node b = node("1");
tests/semantic_check/_invalid_graph_edges.in:3:node c = node("1");
tests/semantic_check/_invalid_graph_edges.in:4:graph g = a->b->c;
tests/semantic_check/_invalid_graph_edges.in:5:g.edges(1);
tests/semantic_check/_invalid_graph_edges.in:6:tests/semantic_check/_inva
lid_list_push2.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_push2.in:2:a.push("1");
tests/semantic_check/_invalid_list_set3.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_set3.in:2:a.set(1, "2");
tests/semantic_check/_redefine_print_func.in:1:int print() {
tests/semantic_check/_redefine_print_func.in:2:    return 1;
tests/semantic_check/_redefine_print_func.in:3:}
tests/semantic_check/_redefine_print_func.in:4:
tests/semantic_check/_redefine_print_func.in:5:print();
tests/semantic_check/_invalid_graph_edge_at.in:1:node a = node("1");
tests/semantic_check/_invalid_graph_edge_at.in:2:node b = node("1");
tests/semantic_check/_invalid_graph_edge_at.in:3:graph g = a -> b;
tests/semantic_check/_invalid_graph_edge_at.in:4:g@(a,1);
tests/semantic_check/_invalid_graph_edge_at.in:5:tests/semantic_check/_in
valid_list_add1.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_add1.in:2:a.add(1, 2);
tests/semantic_check/_invalid_graph_edge_at.out:1:Uh.. There is a invalid
graph edge at: g
tests/semantic_check/_invalid_graph_edge_at.out:2:tests/semantic_check/_i
llegal_binary_operation5.out:1:Uh..oh! Binary Operator float % int in 1.1
% 1 is illegal :(
tests/semantic_check/_illegal_binary_operation5.out:2:tests/semantic_chec
k/_invalid_list_get2.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_get2.in:2:a.get("1");
tests/semantic_check/_inconsistent_list_element_type.out:1:Uh.. List
elements have different types: int and string :(
tests/semantic_check/_inconsistent_list_element_type.out:2:tests/semantic
_check/_invalid_graph_root.in:1:node a = node("1");
tests/semantic_check/_invalid_graph_root.in:2:node b = node("1");
tests/semantic_check/_invalid_graph_root.in:3:node c = node("1");
```

```
tests/semantic_check/_invalid_graph_root.in:4:graph g = a->b->c;
tests/semantic_check/_invalid_graph_root.in:5:g.root(1);
tests/semantic_check/_invalid_graph_root.in:6:tests/semantic_check/_inval
id_graph_root.out:1:Uh.. root method do not take arguments: g
tests/semantic_check/_invalid_graph_root.out:2:tests/semantic_check/_inco
nsistent_list_element_type.in:1:list<int> a = [1, "a"];
tests/semantic_check/_invalid_list_remove2.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_remove2.in:2:a.remove("1");
tests/semantic_check/_invalid_list_pop.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_pop.in:2:a.pop(1);
tests/semantic_check/_invalid_empty_list.in:1:list<int> a = [];
tests/semantic_check/_illegal_unary_operation1.in:1:-"hh";
tests/semantic_check/_invalid_list_get1.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_get1.in:2:a.get(1, 2);
tests/semantic_check/Makefile:1:OCAMLC = ocamlc
tests/semantic_check/Makefile:2:OBJS = ../../_build/parser.cmo
../../_build/scanner.cmo ../../_build/ast.cmo ../../_build/sast.cmo
../../_build/checker.cmo ../../_build/semant.cmo semantic_check.cmo
tests/semantic_check/Makefile:3:INCLUDES = -I ../../_build
tests/semantic_check/Makefile:4:
tests/semantic_check/Makefile:5:default: semantic_check
tests/semantic_check/Makefile:6:
tests/semantic_check/Makefile:7:all:
tests/semantic_check/Makefile:8:  cd ..; make all
tests/semantic_check/Makefile:9:
tests/semantic_check/Makefile:10:semantic_check: $(OBJS)
tests/semantic_check/Makefile:11: $(OCAMLC) $(INCLUDES) -o semantic_check
$(OBJS)
tests/semantic_check/Makefile:12:
tests/semantic_check/Makefile:13:%.cmo: %.ml
tests/semantic_check/Makefile:14: $(OCAMLC) $(INCLUDES) -c $<
tests/semantic_check/Makefile:15:
tests/semantic_check/Makefile:16:%.cmi: %.mli
tests/semantic_check/Makefile:17: $(OCAMLC) $(INCLUDES) -c $<
tests/semantic_check/Makefile:18:
tests/semantic_check/Makefile:19:.PHONY: clean
tests/semantic_check/Makefile:20:clean:
tests/semantic_check/Makefile:21: rm -f semantic_check *.cmo *.cmi
tests/semantic_check/Makefile:22:tests/semantic_check/_invalid_expr_after
_return.in:1:int foo() {
tests/semantic_check/_invalid_expr_after_return.in:2:    return 1;
tests/semantic_check/_invalid_expr_after_return.in:3:    int a = 1;
tests/semantic_check/_invalid_expr_after_return.in:4:}
tests/semantic_check/_invalid_list_add2.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_add2.in:2:a.add("1");
tests/semantic_check/_illegal_binary_operation2.out:1:Uh..oh! Binary
Operator int == float in 1 == 1.1 is illegal :(
tests/semantic_check/_illegal_binary_operation2.out:2:tests/semantic_chec
k/_invalid_list_set3.out:1:Uh.. list set method should only take two
argument of type int and int: a :(
tests/semantic_check/_invalid_list_set3.out:2:tests/semantic_check/_incom
patible_func_arg_type.out:1:Uh..oh! incompatible argument type string,
but int is expected :(
tests/semantic_check/_incompatible_func_arg_type.out:2:tests/semantic_che
ck/_invalid_list_remove1.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_remove1.in:2:a.remove(1, 2);
tests/semantic_check/_invalid_list_push1.out:1:Uh.. list push method
should only take one argument of type int: a :(
```

```
tests/semantic_check/_invalid_list_push1.out:2:tests/semantic_check/_ille
gal_binary_operation3.out:1:Uh..oh! Binary Operator int < string in 1 <
hh is illegal :(
tests/semantic_check/_illegal_binary_operation3.out:2:tests/semantic_chec
k/_invalid_graph_nodes.in:1:node a = node("1");
tests/semantic_check/_invalid_graph_nodes.in:2:node b = node("1");
tests/semantic_check/_invalid_graph_nodes.in:3:node c = node("1");
tests/semantic_check/_invalid_graph_nodes.in:4:graph g = a->b->c;
tests/semantic_check/_invalid_graph_nodes.in:5:g.getAllNodes(1);
tests/semantic_check/_invalid_graph_nodes.in:6:tests/semantic_check/_inva
lid_list_set1.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_set1.in:2:a.set(1);
tests/semantic_check/_invalid_list_remove2.out:1:Uh.. list remove method
should only take one argument of type int: a :(
tests/semantic_check/_invalid_list_remove2.out:2:tests/semantic_check/_in
valid_list_remove1.out:1:Uh.. list remove method should only take one
argument of type int: a :(
tests/semantic_check/_invalid_list_remove1.out:2:tests/semantic_check/_in
valid_list_push1.in:1:list<int> a = [1,2,3];
tests/semantic_check/_invalid_list_push1.in:2:a.push(1, 2);
tests/semantic_check/_redefine_print_func.out:1:Uh.. function may not be
defined
tests/semantic_check/_redefine_print_func.out:2:tests/semantic_check/_unm
atched_func_arg_len.in:1:int foo(int a, int b) {
tests/semantic_check/_unmatched_func_arg_len.in:2:    return a + b;
tests/semantic_check/_unmatched_func_arg_len.in:3:}
tests/semantic_check/_unmatched_func_arg_len.in:4:
tests/semantic_check/_unmatched_func_arg_len.in:5:foo(1,2,3);
tests/topo.out:1:node COMS 3101: Programming Language Python
tests/topo.out:2:node CSOR 4231: Analysis of Algorithms I
tests/topo.out:3:node COMS 6232: Analysis of Algorithms II
tests/topo.out:4:node COMS 4771: Machine Learning
tests/topo.out:5:node COMS 4772: Advanced Machine Learning
tests/topo.out:6:node COMS 4995: Deep Learning for Computer Vision
tests/topo.out:7:node COMS 1004: Introduction to Computer Science and
Programming in Java
tests/topo.out:8:node CSEE 4119: Computer Networks
tests/topo.out:9:node CSEE 4140: Networking Laboratory
tests/topo.out:10:node COMS 4111: Introduction to Databases
tests/topo.out:11:node COMS 6111: Advanced Database Systems
tests/topo.out:12:node COMS 4118: Operating Systems
tests/topo.out:13:tests/test_scanner.sh:1:#!/bin/bash
tests/test_scanner.sh:2:
tests/test_scanner.sh:3:NC='\033[0m'
tests/test_scanner.sh:4:CYAN='\033[0;36m'
tests/test_scanner.sh:5:GREEN='\033[0;32m'
tests/test_scanner.sh:6:RED='\033[0;31m'
tests/test_scanner.sh:7:
tests/test_scanner.sh:8:result=true
tests/test_scanner.sh:9:
tests/test_scanner.sh:10:INPUT_FILES="scanner/*.in"
tests/test_scanner.sh:11:printf "${CYAN}Running scanner tests...\n${NC}"
tests/test_scanner.sh:12:
tests/test_scanner.sh:13:for input_file in $INPUT_FILES; do
tests/test_scanner.sh:14:    output_file=${input_file/.in/.out}
tests/test_scanner.sh:15:    scanner/tokenize < $input_file | cmp -s
$output_file -
tests/test_scanner.sh:16:    if [ "$?" -eq 0 ]; then
tests/test_scanner.sh:17:       printf "%-65s ${GREEN}SUCCESS\n${NC}" "
- checking $input_file..."
```

```
tests/test_scanner.sh:18:    else
tests/test_scanner.sh:19:       printf "%-65s ${RED}ERROR\n${NC}" "  -
checking $input_file..." 1>&2
tests/test_scanner.sh:20:         result=false
tests/test_scanner.sh:21:    fi
tests/test_scanner.sh:22:done
tests/test_scanner.sh:23:
tests/test_scanner.sh:24:exit 0
tests/test_scanner.sh:25:
tests/test_scanner.sh:26:# if $result; then
tests/test_scanner.sh:27:#  exit 0
tests/test_scanner.sh:28:# else
tests/test_scanner.sh:29:#  exit 1
tests/test_scanner.sh:30:# fi
tests/test_scanner.sh:31:tests/topo/test-topo.in:1:node a = node("COMS
1004: Introduction to Computer Science and Programming in Java");
tests/topo/test-topo.in:2:node b = node("COMS 4111: Introduction to
Databases");
tests/topo/test-topo.in:3:node c = node("COMS 6111: Advanced Database
Systems");
tests/topo/test-topo.in:4:node d = node("CSEE 4119: Computer Networks");
tests/topo/test-topo.in:5:node e = node("CSEE 4140: Networking
Laboratory");
tests/topo/test-topo.in:6:node f = node("COMS 4118: Operating Systems");
tests/topo/test-topo.in:7:node g = node("COMS 3101: Programming Language
Python");
tests/topo/test-topo.in:8:node h = node("CSOR 4231: Analysis of
Algorithms I");
tests/topo/test-topo.in:9:node i1 = node("COMS 6232: Analysis of
Algorithms II");
tests/topo/test-topo.in:10:node j = node("COMS 4771: Machine Learning");
tests/topo/test-topo.in:11:node k = node("COMS 4772: Advanced Machine
Learning");
tests/topo/test-topo.in:12:node l = node("COMS 4995: Deep Learning for
Computer Vision");
tests/topo/test-topo.in:13:
tests/topo/test-topo.in:14:graph ga = a -> b -> f + b -> c + a -> d -> f
+ d -> e + h -> f + g -> h -> j -> l + h -> i1 + j -> k;
tests/topo/test-topo.in:15:
tests/topo/test-topo.in:16:int i=0;
tests/topo/test-topo.in:17:list<node> n = ga.getAllNodes();
tests/topo/test-topo.in:18:list<node> tmp = n;
tests/topo/test-topo.in:19:list<node> res = n;
tests/topo/test-topo.in:20:int stmp = 0;
tests/topo/test-topo.in:21:ga.setAllUnvisited();
tests/topo/test-topo.in:22:
tests/topo/test-topo.in:23:while(i<ga.size()){
tests/topo/test-topo.in:24: if(n.get(i).isVisited()){
tests/topo/test-topo.in:25:       i = i + 1;
tests/topo/test-topo.in:26: }else{
tests/topo/test-topo.in:27:       tmp = ga.dfs(n.get(i));
tests/topo/test-topo.in:28:       stmp = tmp.size();
tests/topo/test-topo.in:29:       while(stmp > 0) {
tests/topo/test-topo.in:30:            res.add(tmp.get(stmp-1));
tests/topo/test-topo.in:31:            stmp = stmp - 1;
tests/topo/test-topo.in:32:       }
tests/topo/test-topo.in:33:       i = i + 1;
tests/topo/test-topo.in:34: }
tests/topo/test-topo.in:35:}
tests/topo/test-topo.in:36:
```

```
tests/topo/test-topo.in:37:while(i > 0) {
tests/topo/test-topo.in:38: print(res.get(i-1+ga.size()));
tests/topo/test-topo.in:39: i=i-1;
tests/topo/test-topo.in:40:}
tests/topo.sh:1:#!/bin/bash
tests/topo.sh:2:
tests/topo.sh:3:NC='\033[0m'
tests/topo.sh:4:CYAN='\033[0;36m'
tests/topo.sh:5:GREEN='\033[0;32m'
tests/topo.sh:6:RED='\033[0;31m'
tests/topo.sh:7:
tests/topo.sh:8:result=true
tests/topo.sh:9:
tests/topo.sh:10:INPUT_FILES="topo/*.in"
tests/topo.sh:11:printf "${CYAN}Running code_gen tests...\n${NC}"
tests/topo.sh:12:
tests/topo.sh:13:
tests/topo.sh:14:
tests/topo.sh:15:for input_file in $INPUT_FILES; do
tests/topo.sh:16:    output_file=${input_file/.in/.out}
tests/topo.sh:17:    sh ./giraphe.sh $input_file | cat > topo.out
tests/topo.sh:18:
tests/topo.sh:19:done
tests/topo.sh:20:
tests/topo.sh:21:exit 0
tests/topo.sh:22:
tests/topo.sh:23:# if $result; then
tests/topo.sh:24:#     exit 0
tests/topo.sh:25:# else
tests/topo.sh:26:#     exit 1
tests/topo.sh:27:# fi
tests/topo.sh:28:tests/test_parser.sh:1:#!/bin/bash
tests/test_parser.sh:2:
tests/test_parser.sh:3:NC='\033[0m'
tests/test_parser.sh:4:CYAN='\033[0;36m'
tests/test_parser.sh:5:GREEN='\033[0;32m'
tests/test_parser.sh:6:RED='\033[0;31m'
tests/test_parser.sh:7:
tests/test_parser.sh:8:result=true
tests/test_parser.sh:9:
tests/test_parser.sh:10:INPUT_FILES="parser/*.in"
tests/test_parser.sh:11:printf "${CYAN}Running Parser tests...\n${NC}"
tests/test_parser.sh:12:
tests/test_parser.sh:13:for input_file in $INPUT_FILES; do
tests/test_parser.sh:14:    output_file=${input_file/.in/.out}
tests/test_parser.sh:15:    parser/parserize < $input_file | cmp -s
$output_file -
tests/test_parser.sh:16:    if [ "$?" -eq 0 ]; then
tests/test_parser.sh:17:       printf "%-65s ${GREEN}SUCCESS\n${NC}" "  -
checking $input_file..."
tests/test_parser.sh:18:    else
tests/test_parser.sh:19:       printf "%-65s ${RED}ERROR\n${NC}" "  -
checking $input_file..." 1>&2
tests/test_parser.sh:20:    result=false
tests/test_parser.sh:21:    fi
tests/test_parser.sh:22:done
tests/test_parser.sh:23:
tests/test_parser.sh:24:cd ../compiler;
tests/test_parser.sh:25:ocamlyacc -v parser.mly;
tests/test_parser.sh:26:
```

```
tests/test_parser.sh:27:exit 0
tests/test_parser.sh:28:
tests/test_parser.sh:29:# if $result; then
tests/test_parser.sh:30:#   exit 0
tests/test_parser.sh:31:# else
tests/test_parser.sh:32:#   exit 1
tests/test_parser.sh:33:# fi
tests/test_parser.sh:34:tests/filter.out:1:1 : Restaurant
tests/filter.out:2:2 : Unversity Building
tests/filter.out:3:3 : Library
tests/filter.out:4:4 : SuperMarket
tests/filter.out:5:5 : Pharmacy
tests/filter.out:6:
tests/filter.out:7:-- try to find places of target type --
tests/filter.out:8:1
tests/filter.out:9:
tests/filter.out:10:--------- have found target ------------
tests/filter.out:11:---------     target      --------------
tests/filter.out:12:node Xian Famous Food
tests/filter.out:13:--------- shortest path to target  -----
tests/filter.out:14:[node Mudd
tests/filter.out:15:node Butler Library
tests/filter.out:16:node Vine
tests/filter.out:17:node WestSide Market
tests/filter.out:18:node Duane Reade
tests/filter.out:19:node Xian Famous Food
tests/filter.out:20:]
tests/filter.out:21:
tests/filter.out:22:
tests/filter.out:23:
tests/filter.out:24:--------- have found target ------------
tests/filter.out:25:---------     target      --------------
tests/filter.out:26:node Vine
tests/filter.out:27:--------- shortest path to target  -----
tests/filter.out:28:[node Mudd
tests/filter.out:29:node Butler Library
tests/filter.out:30:node Vine
tests/filter.out:31:]
tests/filter.out:32:
tests/filter.out:33:
tests/filter.out:34:tests/scanner/_boolean_operation.out:1:BOOL_LITERAL
tests/scanner/_boolean_operation.out:2:BOOL_LITERAL
tests/scanner/_boolean_operation.out:3:tests/scanner/_quote.out:1:QUOTE
tests/scanner/_quote.out:2:tests/scanner/_primary_type.in:1:void int
float string bool node graph edge list hashmap null
tests/scanner/_primary_type.in:2:tests/scanner/_graph_operator.out:1:RIGH
TLINK
tests/scanner/_graph_operator.out:2:tests/scanner/_graph_operator.in:1:->
tests/scanner/_graph_operator.in:2:tests/scanner/_bracket.out:1:LEFTBRACK
ET
tests/scanner/_bracket.out:2:RIGHTBRACKET
tests/scanner/_bracket.out:3:LEFTCURLYBRACKET
tests/scanner/_bracket.out:4:RIGHTCURLYBRACKET
tests/scanner/_bracket.out:5:LEFTROUNDBRACKET
tests/scanner/_bracket.out:6:RIGHTROUNDBRACKET
tests/scanner/_bracket.out:7:tests/scanner/_arithmetic.out:1:PLUS
tests/scanner/_arithmetic.out:2:MINUS
tests/scanner/_arithmetic.out:3:TIMES
tests/scanner/_arithmetic.out:4:DIVIDE
tests/scanner/_arithmetic.out:5:MOD
```

```
tests/scanner/_arithmetic.out:6:tests/scanner/_integer_float.in:1:10 10.0
11.1
tests/scanner/_boolean_operation.in:1:true false
tests/scanner/_primary_type.out:1:VOID
tests/scanner/_primary_type.out:2:INT
tests/scanner/_primary_type.out:3:FLOAT
tests/scanner/_primary_type.out:4:STRING
tests/scanner/_primary_type.out:5:BOOL
tests/scanner/_primary_type.out:6:NODE
tests/scanner/_primary_type.out:7:GRAPH
tests/scanner/_primary_type.out:8:EDGE
tests/scanner/_primary_type.out:9:LIST
tests/scanner/_primary_type.out:10:DICT
tests/scanner/_primary_type.out:11:NULL
tests/scanner/_primary_type.out:12:tests/scanner/_bracket.in:1:[ ] { } (
)
tests/scanner/_comment.out:1:
tests/scanner/_comment.out:2:tests/scanner/_quote.in:1:"
tests/scanner/_quote.in:2:tests/scanner/_comparator.out:1:GREATER
tests/scanner/_comparator.out:2:GREATEREQUAL
tests/scanner/_comparator.out:3:SMALLER
tests/scanner/_comparator.out:4:SMALLEREQUAL
tests/scanner/_comparator.out:5:tests/scanner/_arithmetic.in:1:+ - * / %
tests/scanner/_integer_float.out:1:INT_LITERAL
tests/scanner/_integer_float.out:2:FLOAT_LITERAL
tests/scanner/_integer_float.out:3:FLOAT_LITERAL
tests/scanner/_integer_float.out:4:tests/scanner/Makefile:1:OCAMLC =
ocamlc
tests/scanner/Makefile:2:OBJS = ../../_build/scanner.cmo tokenize.cmo
tests/scanner/Makefile:3:INCLUDES = -I ../../_build
tests/scanner/Makefile:4:
tests/scanner/Makefile:5:default: tokenize
tests/scanner/Makefile:6:
tests/scanner/Makefile:7:all:
tests/scanner/Makefile:8:   cd ..; make all
tests/scanner/Makefile:9:
tests/scanner/Makefile:10:tokenize: $(OBJS)
tests/scanner/Makefile:11:  $(OCAMLC) $(INCLUDES) -o tokenize $(OBJS)
tests/scanner/Makefile:12:
tests/scanner/Makefile:13:%.cmo: %.ml
tests/scanner/Makefile:14:  $(OCAMLC) $(INCLUDES) -c $<
tests/scanner/Makefile:15:
tests/scanner/Makefile:16:%.cmi: %.mli
tests/scanner/Makefile:17:  $(OCAMLC) $(INCLUDES) -c $<
tests/scanner/Makefile:18:
tests/scanner/Makefile:19:.PHONY: clean
tests/scanner/Makefile:20:clean:
tests/scanner/Makefile:21:  rm -f tokenize *.cmo *.cmi
tests/scanner/Makefile:22:
tests/scanner/Makefile:23:tokenize.cmo:
tests/scanner/Makefile:24:tokenize.cmx:
tests/scanner/Makefile:25:tests/scanner/_comparator.in:1:> >= < <=
tests/scanner/tokenize.ml:1:open Parser
tests/scanner/tokenize.ml:2:
tests/scanner/tokenize.ml:3:let stringify = function
tests/scanner/tokenize.ml:4:  (* calculation *)
tests/scanner/tokenize.ml:5:  | PLUS -> "PLUS"   | MINUS -> "MINUS"
tests/scanner/tokenize.ml:6:  | TIMES -> "TIMES" | DIVIDE -> "DIVIDE"
tests/scanner/tokenize.ml:7:  | MOD -> "MOD"
tests/scanner/tokenize.ml:8:  (* separator *)
```

```
tests/scanner/tokenize.ml:9:  | SEMICOLON -> "SEMICOLON"   | COMMA ->
"COMMA"
tests/scanner/tokenize.ml:10:  | ASSIGN -> "ASSIGN"         | COLON ->
"COLON"
tests/scanner/tokenize.ml:11:  | DOT -> "DOT"
tests/scanner/tokenize.ml:12:  (* logical operation *)
tests/scanner/tokenize.ml:13:  | AND -> "AND"      | OR -> "OR"
tests/scanner/tokenize.ml:14:  | NOT -> "NOT"      | IF -> "IF"
tests/scanner/tokenize.ml:15:  | ELSE -> "ELSE"    | FOR -> "FOR"
tests/scanner/tokenize.ml:16:  | WHILE -> "WHILE"  | BREAK -> "BREAK"
tests/scanner/tokenize.ml:17:  | CONTINUE -> "CONTINUE"
tests/scanner/tokenize.ml:18:  (* comparator *)
tests/scanner/tokenize.ml:19:  | EQ -> "EQUAL"          | NEQ ->
"NOTEQUAL"
tests/scanner/tokenize.ml:20:  | GT -> "GREATER"      | GEQ ->
"GREATEREQUAL"
tests/scanner/tokenize.ml:21:  | LT -> "SMALLER"      | LEQ ->
"SMALLEREQUAL"
tests/scanner/tokenize.ml:22:  (* graph operator *)
tests/scanner/tokenize.ml:23:  | RIGHTLINK -> "RIGHTLINK"
tests/scanner/tokenize.ml:24:  | AMPERSAND -> "AMPERSAND"
tests/scanner/tokenize.ml:25:  (* identifier *)
tests/scanner/tokenize.ml:26:  | ID(string) -> "ID"
tests/scanner/tokenize.ml:27:  (* primary type *)
tests/scanner/tokenize.ml:28:  | INT -> "INT"          | FLOAT -> "FLOAT"
tests/scanner/tokenize.ml:29:  | STRING -> "STRING"    | BOOL -> "BOOL"
tests/scanner/tokenize.ml:30:  | NODE -> "NODE"        | GRAPH -> "GRAPH"
tests/scanner/tokenize.ml:31:  | LIST -> "LIST"        | DICT -> "DICT"
tests/scanner/tokenize.ml:32:  | NULL -> "NULL"        | VOID -> "VOID"
tests/scanner/tokenize.ml:33:  | EDGE -> "EDGE"
tests/scanner/tokenize.ml:34:  (* quote *)
tests/scanner/tokenize.ml:35:  | QUOTE -> "QUOTE"
tests/scanner/tokenize.ml:36:  (* boolean operation *)
tests/scanner/tokenize.ml:37:  (* bracket *)
tests/scanner/tokenize.ml:38:  | LBRACKET -> "LEFTBRACKET"          |
RBRACKET -> "RIGHTBRACKET"
tests/scanner/tokenize.ml:39:  | LBRACE -> "LEFTCURLYBRACKET" | RBRACE ->
"RIGHTCURLYBRACKET"
tests/scanner/tokenize.ml:40:  | LPAREN -> "LEFTROUNDBRACKET" | RPAREN ->
"RIGHTROUNDBRACKET"
tests/scanner/tokenize.ml:41:  (* End-of-File *)
tests/scanner/tokenize.ml:42:  | EOF -> "EOF"
tests/scanner/tokenize.ml:43:  (* Literals *)
tests/scanner/tokenize.ml:44:  | INT_LITERAL(int) -> "INT_LITERAL"
tests/scanner/tokenize.ml:45:  | FLOAT_LITERAL(float) -> "FLOAT_LITERAL"
tests/scanner/tokenize.ml:46:  | STRING_LITERAL(string) ->
"STRING_LITERAL"
tests/scanner/tokenize.ml:47:  | BOOL_LITERAL(bool) -> "BOOL_LITERAL"
tests/scanner/tokenize.ml:48:  | RETURN -> "RETURN"
tests/scanner/tokenize.ml:49:
tests/scanner/tokenize.ml:50:let _ =
tests/scanner/tokenize.ml:51:  let lexbuf = Lexing.from_channel stdin in
tests/scanner/tokenize.ml:52:  let rec print_tokens = function
tests/scanner/tokenize.ml:53:    | EOF -> " "
tests/scanner/tokenize.ml:54:    | token ->
tests/scanner/tokenize.ml:55:      print_endline (stringify token);
tests/scanner/tokenize.ml:56:      print_tokens (Scanner.token lexbuf) in
tests/scanner/tokenize.ml:57:  print_tokens (Scanner.token lexbuf)
tests/scanner/tokenize.ml:58:tests/filter/test-filter.in:1:node a =
node("Mudd");
```

```
tests/filter/test-filter.in:2:node b = node("Vine");
tests/filter/test-filter.in:3:node c = node("Xian Famous Food");
tests/filter/test-filter.in:4:node d = node("Butler Library");
tests/filter/test-filter.in:5:node e = node("WestSide Market");
tests/filter/test-filter.in:6:node f = node("Duane Reade");
tests/filter/test-filter.in:7:
tests/filter/test-filter.in:8:
tests/filter/test-filter.in:9:graph gh = a -> 1$d ->8$c + a -> 1$d -> 2$b
-> 1$e -> 1$f -> 1$c + a ->5$b;
tests/filter/test-filter.in:10:list<node> path = [a];
tests/filter/test-filter.in:11:node cur = a;
tests/filter/test-filter.in:12:int curVal = 0;
tests/filter/test-filter.in:13:print("1 : Restaurant");
tests/filter/test-filter.in:14:print("2 : Unversity Building");
tests/filter/test-filter.in:15:print("3 : Library");
tests/filter/test-filter.in:16:print("4 : SuperMarket");
tests/filter/test-filter.in:17:print("5 : Pharmacy");
tests/filter/test-filter.in:18:print("");
tests/filter/test-filter.in:19:hashmap<int> hmap = { a : 2, b : 1, c : 1,
d : 3, e : 4, f : 5};
tests/filter/test-filter.in:20:
tests/filter/test-filter.in:21:void filter(node sour, int target) {
tests/filter/test-filter.in:22:    print("-- try to find places of target
type --");
tests/filter/test-filter.in:23:    print(target);
tests/filter/test-filter.in:24:    list<node> nodes = gh.getAllNodes();
tests/filter/test-filter.in:25:    int size = nodes.size();
tests/filter/test-filter.in:26:    int i = 0;
tests/filter/test-filter.in:27:    while (i < size) {
tests/filter/test-filter.in:28:        cur = nodes.get(i);
tests/filter/test-filter.in:29:        curVal = hmap.get(cur);
tests/filter/test-filter.in:30:        if (curVal == target) {
tests/filter/test-filter.in:31:            print("");
tests/filter/test-filter.in:32:            print("--------- have found
target ------------");
tests/filter/test-filter.in:33:            print("---------     target
--------------");
tests/filter/test-filter.in:34:            print(cur);
tests/filter/test-filter.in:35:            print("--------- shortest path
to target  -----");
tests/filter/test-filter.in:36:            path = gh.dijkstra(sour, cur);
tests/filter/test-filter.in:37:            print(path);
tests/filter/test-filter.in:38:            print("");
tests/filter/test-filter.in:39:            print("");
tests/filter/test-filter.in:40:        }
tests/filter/test-filter.in:41:        i = i + 1;
tests/filter/test-filter.in:42:    }
tests/filter/test-filter.in:43:}
tests/filter/test-filter.in:44:filter(a, 1);
tests/filter/test-filter.in:45:
tests/filter/test-filter.in:46:tests/bfs.out:1:a -> [1&b -> 1&e -> [4&g -
> 1&b, 2&c], 5&c -> [1&g, 1&f -> 1&c], 3&d -> [2&c, 3&f]]
tests/bfs.out:2:Breath First Search from a
tests/bfs.out:3:[node a
tests/bfs.out:4:node b
tests/bfs.out:5:node c
tests/bfs.out:6:node d
tests/bfs.out:7:node e
tests/bfs.out:8:node g
tests/bfs.out:9:node f
```

```
tests/bfs.out:10:]
tests/bfs.out:11:Set all Nodes to Unvisited
tests/bfs.out:12:Breath First Search from c
tests/bfs.out:13:[node c
tests/bfs.out:14:node g
tests/bfs.out:15:node f
tests/bfs.out:16:node b
tests/bfs.out:17:node e
tests/bfs.out:18:]
tests/bfs.out:19:Set all Nodes to Unvisited
tests/bfs.out:20:tests/filter.sh:1:#!/bin/bash
tests/filter.sh:2:
tests/filter.sh:3:NC='\033[0m'
tests/filter.sh:4:CYAN='\033[0;36m'
tests/filter.sh:5:GREEN='\033[0;32m'
tests/filter.sh:6:RED='\033[0;31m'
tests/filter.sh:7:
tests/filter.sh:8:result=true
tests/filter.sh:9:
tests/filter.sh:10:INPUT_FILES="filter/*.in"
tests/filter.sh:11:printf "${CYAN}Running code_gen tests...\n${NC}"
tests/filter.sh:12:
tests/filter.sh:13:
tests/filter.sh:14:
tests/filter.sh:15:for input_file in $INPUT_FILES; do
tests/filter.sh:16:    output_file=${input_file/.in/.out}
tests/filter.sh:17:    sh ./giraphe.sh $input_file | cat > filter.out
tests/filter.sh:18:
tests/filter.sh:19:done
tests/filter.sh:20:
tests/filter.sh:21:exit 0
tests/filter.sh:22:
tests/filter.sh:23:# if $result; then
tests/filter.sh:24:#   exit 0
tests/filter.sh:25:# else
tests/filter.sh:26:#   exit 1
tests/filter.sh:27:# fi
tests/filter.sh:28:tests/Makefile:1:default: test
tests/Makefile:2:
tests/Makefile:3:all: clean
tests/Makefile:4:     cd ..; make clean
tests/Makefile:5:
tests/Makefile:6:test: clean build
tests/Makefile:7:     clang -emit-llvm -o lib.bc -c ../lib.c -Wno-
varargs
tests/Makefile:8:     bash ./test_scanner.sh
tests/Makefile:9:     bash ./test_parser.sh
tests/Makefile:10:    bash ./test_semantic.sh
tests/Makefile:11:    bash ./test_code_gen.sh
tests/Makefile:12:
tests/Makefile:13:build:
tests/Makefile:14:    cd scanner; make
tests/Makefile:15:    cd parser; make
tests/Makefile:16:    cd semantic_check; make
tests/Makefile:17:
tests/Makefile:18:.PHONY: clean
tests/Makefile:19:clean:
tests/Makefile:20:    rm -f lib.bc
tests/Makefile:21:    cd scanner; make clean
tests/Makefile:22:    cd parser; make clean
```

```
tests/Makefile:23:    cd semantic_check; make clean
tests/Makefile:24:tests/test_code_gen.sh:1:#!/bin/bash
tests/test_code_gen.sh:2:
tests/test_code_gen.sh:3:NC='\033[0m'
tests/test_code_gen.sh:4:CYAN='\033[0;36m'
tests/test_code_gen.sh:5:GREEN='\033[0;32m'
tests/test_code_gen.sh:6:RED='\033[0;31m'
tests/test_code_gen.sh:7:
tests/test_code_gen.sh:8:result=true
tests/test_code_gen.sh:9:
tests/test_code_gen.sh:10:INPUT_FILES="code_gen/*.in"
tests/test_code_gen.sh:11:printf "${CYAN}Running code_gen
tests...\n${NC}"
tests/test_code_gen.sh:12:
tests/test_code_gen.sh:13:
tests/test_code_gen.sh:14:
tests/test_code_gen.sh:15:for input_file in $INPUT_FILES; do
tests/test_code_gen.sh:16:    output_file=${input_file/.in/.out}
tests/test_code_gen.sh:17:    sh ./giraphe.sh $input_file | cmp -s
$output_file -
tests/test_code_gen.sh:18:    if [ "$?" -eq 0 ]; then
tests/test_code_gen.sh:19:       printf "%-65s ${GREEN}SUCCESS\n${NC}" "
- checking $input_file..."
tests/test_code_gen.sh:20:    else
tests/test_code_gen.sh:21:       printf "%-65s ${RED}ERROR\n${NC}" "  -
checking $input_file..." 1>&2
tests/test_code_gen.sh:22:        result=false
tests/test_code_gen.sh:23:    fi
tests/test_code_gen.sh:24:done
tests/test_code_gen.sh:25:
tests/test_code_gen.sh:26:exit 0
tests/test_code_gen.sh:27:
tests/test_code_gen.sh:28:# if $result; then
tests/test_code_gen.sh:29:# exit 0
tests/test_code_gen.sh:30:# else
tests/test_code_gen.sh:31:# exit 1
tests/test_code_gen.sh:32:# fi
tests/test_code_gen.sh:33:tests/topo.in:1:node a = node("COMS 1004:
Introduction to Computer Science and Programming in Java");
tests/topo.in:2:node b = node("COMS 4111: Introduction to Databases");
tests/topo.in:3:node c = node("COMS 6111: Advanced Database Systems");
tests/topo.in:4:node d = node("CSEE 4119: Computer Networks");
tests/topo.in:5:node e = node("CSEE 4140: Networking Laboratory");
tests/topo.in:6:node f = node("COMS 4118: Operating Systems");
tests/topo.in:7:node g = node("COMS 3101: Programming Language Python");
tests/topo.in:8:node h = node("CSOR 4231: Analysis of Algorithms I");
tests/topo.in:9:node i1 = node("COMS 6232: Analysis of Algorithms II");
tests/topo.in:10:node j = node("COMS 4771: Machine Learning");
tests/topo.in:11:node k = node("COMS 4772: Advanced Machine Learning");
tests/topo.in:12:node l = node("COMS 4995: Deep Learning for Computer
Vision");
tests/topo.in:13:
tests/topo.in:14:graph ga = a -> b -> f + b -> c + a -> d -> f + d -> e +
h -> f + g -> h -> j -> l + h -> i1 + j -> k;
tests/topo.in:15:
tests/topo.in:16:int i=0;
tests/topo.in:17:list<node> n = ga.getAllNodes();
tests/topo.in:18:list<node> tmp = n;
tests/topo.in:19:list<node> res = n;
tests/topo.in:20:int stmp = 0;
```

```
tests/topo.in:21:ga.setAllUnvisited();
tests/topo.in:22:
tests/topo.in:23:while(i<ga.size()){
tests/topo.in:24:    if(n.get(i).isVisited()){
tests/topo.in:25:        i = i + 1;
tests/topo.in:26:    }else{
tests/topo.in:27:        tmp = ga.dfs(n.get(i));
tests/topo.in:28:        stmp = tmp.size();
tests/topo.in:29:        while(stmp > 0) {
tests/topo.in:30:            res.add(tmp.get(stmp-1));
tests/topo.in:31:            stmp = stmp - 1;
tests/topo.in:32:        }
tests/topo.in:33:        i = i + 1;
tests/topo.in:34:    }
tests/topo.in:35:}
tests/topo.in:36:
tests/topo.in:37:while(i > 0) {
tests/topo.in:38:    print(res.get(i-1+ga.size()));
tests/topo.in:39:    i=i-1;
tests/topo.in:40:}
tests/topo.in:41:
```