GRACL (.grc)

Defne Sonmez	dys2109	System Architect
Eilam Lehrman	esl2160	Language Guru
Hadley Callaway	hcc2134	Manager
Maya Venkatraman	mv2731	System Architect
Pelin Cetin	pc2807	Tester

Table of Contents

I. INTRODUCTION

This manual describes GRACL (GRAph Concurrency Language), a language aiming to improve the efficiency of common graph algorithms by leveraging concurrency while allowing programmers to initialize and modify graphs easily with built-in data structures. GRACL also aims to make concurrency more user friendly and less error-prone, by taking care of lock and unlock operations under the hood for the user.

The syntax is inspired by some of the past projects focused on graphs (such as GRAIL from Spring 2017) with elements from Java, Python, and C. The following features are available to the programmer: Graphs, Nodes, Edges, hash tables mapping Nodes to Ints or Doubles, Edge lists, and Node lists. It is easier to construct directed graphs in GRACL, but the user is able to create undirected graphs as well by creating directed edges in both directions.

II. White Paper

1. Introduction & Motivation

GRACL (GRAph Concurrency Language) improves the efficiency of common graph algorithms such as Breadth-First-Search (BFS), Depth-First-Search (DFS), Dijkstra, and Traveling Salesman Problem. GRACL uses concurrency (with lightweight multithreading as in Go) and allows programmers to initialize and modify graphs easily with built-in data structures specific to our language. This combination enables the user to implement concurrent graph algorithms that converge more quickly than their traditional counterparts. Improved runtime efficiency has enormous potential to improve modern-day computing in areas such as transportation networks, metadata relation-building database systems, packet switching, neural networks, etc. We were inspired by some of the past projects focused on graphs (such as GRAIL from Spring 2017) to use syntax with elements from Java, Python, and C. We plan to make the following features available to the programmer: graphs, nodes, threads, and locks. We will discuss their implementation at greater length in our "Syntax" section.

2. Language Overview

GRACL is statically scoped, as well as strongly and statically typed to fully leverage the efficiency of the compiler. Like Java, GRACL creates copies of the references and passes them as values to methods.

GRACL supports mutable data types because immutability does not make sense for our application space. In typical graph searches, one must maintain a frontier or data structure of visited nodes. Immutability would force processes to create a new frontier or visited node structure every single time a new node is visited, making mutable objects far preferable.

We will ensure that threads are collected under the hood after they terminate and that orphan or zombie processes are properly reaped. There will be strict evaluation. While there will be no complete garbage collection, we may consider freeing graphs, nodes, lists, and strings for the user at the end of their program execution if time permits.

3. **Syntax**
    a. Primitives
        i. bool
        ii. char
        iii. double
        iv. int
        v. void
        vi. any - a type that allows nodes to store any type of data in their datafields
        vii.
    b. Objects

String	Character arrays with basic functionalities emulating Java Strings, primarily used for printing
List	A standard linked list [] with basic functionalities emulating Java LinkedLists
Node	An object that has a data field and a list of neighboring nodes
Graph	A list of nodes connected by directed or undirected edges representing a graph data type. Data types in the graph do not need to be homogeneous
Thread	A lightweight process sharing memory below the stack with other threads, much like a pthread in C. Initialized with a function that starts the routine the user wishes to have executed
RwLock	A reader/writer lock that allows multiple processes to read protected data simultaneously, but only allows one thread to write at once. No thread may read while another thread is writing
Tuple	A combination of two data types

    c. Integer and Double Operators
        i. +, -, /, *, %, +=, -=

    d. Logical Operators
        i. &&, ||, !, <, >, <=, >=, ==, !=

    e. Control Flow

/**/	Multiline comment
//	Single line comment

;	Signifies the end of a statement
if(...){} else{};	Conditional statements

for(...){};
for(... in ...){};
while(...){};
	Loops

int myFunc(int x) {
  return x;
};
	Example function

f. Node Properties and Built-In Functions

createNode(String name, any data, Node[] neighbors)	Creates a new node, setting its name, data, and list of neighbor tuples (weight, destinationNode). Returns an error if the passed-in name is not unique
.name	Provides a unique string label by which to access the node
.data	Returns data stored in the node
.neighbors	Returns a list of neighbor nodes
.updateName(String name)	Updates the name field on the node to be the new name passed in
.updateData(any data)	Updates the data field on the node to be the new data passed in
.rwLock() (Note: **possible)	Some concurrent graph algorithms require nodes to be marked as complete so multiple threads don't visit them. We propose creating locks for nodes so that no two threads try to mark a node as complete at the same time **We recognize that it may be difficult to expect the user to protect node accesses with locks. This is something we would appreciate feedback on in our proposal.

g. Graph Properties and Built-In Functions

createGraph(String formattedEdges)	Creates a new graph out of a user's string of edges. Each edge should be formatted start-weight-destination for an undirected edge or

	start-weight->destination for a directed edge. When creating each node, the start is taken as the node's name and the node's data field is left blank Example input: "B-4-C, C-7->D, A-3->C, D-3-B"
.nodes	Returns a list of nodes in the graph
.edges	Returns a string representation of all edges in the graph
.addDirectedEdge(Node A, Node B, int weight)	Adds to node A's neighbor list a directed edge to node B, representing it as a tuple (int weight, node destination)
.addUndirectedEdge(Node A, Node B, int weight)	Adds an undirected edge between node A and node B. Under the hood, this calls .addDirectedEdge() twice, once to add a directed edge from node A to node B and once to add a directed edge from node B to node A
.updateDirectedEdge(Node A, Node B, int weight)	Updates the directed edge from node A to node B to have a new weight
.updateUndirectedEdge(Node A, Node B, int weight)	Updates the undirected edge between node A and node B. Under the hood, this calls .updateDirectedEdge() twice, once to update the directed edge from node A to node B and once to update the directed edge from node B to node A
.removeDirectedEdge(Node A, Node B)	Removes the directed edge from node A to node B
.removeUndirectedEdge(Node A, Node B)	Removes the directed edge from node A to node B, and then removes the directed edge from node B to node A
.addNode(Node n)	Adds the passed-in node to the graph
.removeNode(Node n)	Removes the passed-in node from the graph and deletes corresponding edges

h. Thread Properties and Built-In Functions

createThread(any func, any param1, any param2, ...)	Creates a thread that begins by executing the function func applied to the given parameters. This returns a thread object which GRACL detaches upon completion behind the scenes

joinThreads(Thread[] threads)	Blocks until all threads specified in the list complete

i. RwLock Properties and Built-In Functions

createRwLock()	Creates an rwLock
.racquire()	Before entering a critical section of code, a thread has to wait to acquire the rwLock for reading. Once the lock is acquired it may execute the critical section
.wacquire()	Before entering a critical section of code, a thread has to wait to acquire the rwLock for writing. Once the lock is acquired it may execute the critical section
.rrelease()	After executing a critical section, a thread must release the rwLock for reading
.wrelease()	After executing a critical section, a thread must release the rwLock for writing

j. Other Functions

print(“Hello world”);	Print function that prints strings

4. **Basic Operations & Examples**
    a. Making and modifying a graph:

```
Graph g = createGraph("B-4-C, C-7->D, A-3->C, D-3-B");
A.updateData(True);
B.updateData(5);
C.updateData("Eilam");
D.updateData("GRACL");
g.removeData(D);
Node E = createNode("E", 7, [(6,A)]);
g.addNode(E);
```

    b. A simple concurrency example:

```
// Assuming this gets allocated on the heap, as in Java
String buf = "Hello";

RwLock lock = createRwLock();

void startRoutine() {
    lock.wacquire();
    print(buf);
    buf = "World";
    lock.wrelease();
}

for (int i = 0; i < 2; i++) {
    createThread(startRoutine);
}

/*
Sample output:

"Hello"
"World"

*/
```

c. Concurrent DFS:

```
Node[] path = [];
Node[] visited = [];
RwLock visitedLock = createRwLock();
RwLock pathLock = createRwLock();

bool goalTest(Node goal, Node current){
    return goal.name == current.name;
    // Searching for a specific node object using its unique name
}
```

```
void normalDFS(Graph graph, Node current, Node goal, Node[] visited, Node[] myPath){
    pathLock.racquire();
    // Another thread found the goal already and this thread should terminate
    if (path != []) {
        // Release the read lock on path
        pathLock.rrelease();
        return;
    } else {
        // Release the read lock on path
        pathLock.rrelease();
        myPath.add(current);
        // If goal found
        if (goalTest(goal, current)) {
            // Modify shared memory for path to goal
            pathLock.wacquire();
            path = myPath;
            // Release the write lock on path
            pathLock.wrelease();
            return;
        } else {
            // Add current to shared memory list of visited nodes
            visitedLock.wacquire();
            visited.add(current);
            // Release the write lock on visited
            visitedLock.wrelease();
            Node[] neighbors = current.neighbors;
            for ((weight, neighbor) in neighbors) {
                visitedLock.racquire();
                // If visited doesn't contain neighbor, call normal DFS on neighbor
                if (!visited.contains(neighbor)) {
                    // Release the read lock on visited
                    visitedLock.rrelease();
                    normalDFS(graph, neighbor, goal, visited, myPath);
                } else {
                    // Release the read lock on visited
                    visitedLock.rrelease();
                }
            }
            return;
        }
    }
}
```

```
Node[] multithreadDFS(Graph graph, Node start, Node goal) {
    if (goalTest(goal, start)) {
        return [];
    } else {
        Node[] neighbors = start.neighbors;
        visited.add(start);
        Thread[] threads = [];
        // Create a thread for each top-level child of start to perform search in parallel
        for ((weight, neighbor) in neighbors) {
            threads.add(createThread(normalDFS, graph, neighbor, goal, visited, [start]));
        }
        joinThreads(threads);
        return path;
    }
}
```

III. Language Tutorial

1. Downloading and Building GRACL

Download the tar.gz or clone from the main branch of https://github.com/pelincetin/GRACL and run make in the top-level directory to build the compiler:
$ make

To run all tests run:
$ **./testall.sh**

To compile and produce the LLVM of one testfile:
$ **./gracl.native -c ./tests/test-foo.grc**

To compile and run ./tests/test-foo.grc:
$ **./scripts/testone.sh foo**

2. Writing and Compiling Simple Programs

1. The following programs (or similar) can be written in GRACL and executed using our scripts in the ways described above.

Hello World

- All functions require the return type specified; all statements conclude with a semicolon

```
int main(){
    print("Hello world");
    return 0;
}
```

The next program, **Hatch & Sync**, illustrates the usage of our two concurrency-related keywords.

- Hatch allows the user to spawn threads using a nodelist, creating one thread per node in the nodelist. The user must pass hatch a void returning start routine that takes a node of the nodelist as its first parameter, but not include this parameter in the hatch call

```
int start_routine(Node n){
     synch n{
          print(n.data());
     }
     return 0;
}

int main(){
     Graph g;
     Node node1  = g.createNode("A");
     Node node2 = g.createNode("B");
     Node node3 = g.createNode("C");
     Node node4 = g.createNode("D");
     nl1 = g.nodes();
     hatch nl1 start_routine(){
          print("Code executed by the parent thread");
     }
     return 0;
}
```

The third program, **Create Graph,** illustrates Graph, Node and Edge creation

```
void foo(){
     Graph  g = createGraph(3);

     Node n1 = g.createNode(“A”);
     Node n2 = g.createNode(“B”);
     Node n3 = g.createNode(“C”);

     Edge e1 = g.createEdge(n1, n2, 1.2);

     return;
}
```

3. Compiler Flags

The compiler has four flags, given a file ./tests/test-foo.grc. To print the AST:
$ ./gracl.native -a ./tests/test-foo.grc

To print the SAST:
$ ./gracl.native -s ./tests/test-foo.grc

To print the LLVM output:
$ ./gracl.native -l ./tests/test-foo.grc

To check that the LLVM output is valid LLVM, and print it
$ ./gracl.native -c ./tests/test-foo.grc

IV. Language Reference Manual

1. **Introduction**
   This manual describes GRACL (GRAph Concurrency Language), a language aiming to improve the efficiency of common graph algorithms by leveraging concurrency while allowing programmers to initialize and modify graphs easily with built-in data structures specific to our language.

   The syntax is inspired by some of the past projects focused on graphs (such as GRAIL from Spring 2017) with elements from Java, Python, and C. We plan to make the following features available to the programmer: graphs, nodes, edges, integer and double hash tables, edge lists, and node lists. GRACL focuses on directed graphs but the user is able to create undirected graphs by calling directed edges twice.

2. Constants

Integer	Denoted int. Mathematical integers, eg 10 . Integer overflow is supported.
Doubles	Denoted double. Numbers that have a decimal point, e.g. 3.42 or 3.5E-10.
Boolean constants	Represented by the keywords True and False.
String literals	Series of ASCII characters delimited by double quotation marks.

3. Primitive Data Types

bool	Boolean value (True/False).
int	Integer.
double	A double-precision floating-point.

4. Derived Types

String	Array of ASCII characters.
NodeList	A collection of Node types. Under the hood it is implemented as a doubly linked list with O(n) lookup, they have an associated implicit mutex.
EdgeList	A collection of Edge types. Under the hood it is implemented as a doubly linked list with O(n) lookup, they have an associated implicit mutex.
IntTable	Collection of key value pairs with Node keys and int values. O(1) lookup time on keys, they have an associated implicit mutex. Has dynamic capacity; capacity doubles when the table is filled up. Uses the hash function h(item) = item % len and linear probing to resolve collisions.
DoubleTable	Collection of key value pairs with Node keys and double values. O(1) lookup time on keys, they have an associated implicit mutex. Has dynamic capacity; capacity doubles when the table is filled up. Uses the hash function h(item) = item % len and linear probing to resolve collisions.
Node	An object that has a String data field and an EdgeList representing connected edges, as well as an implicit mutex, and a visited boolean. Also has a Node

	precursor field, where the user can assign its preceding node; a cost field where the user can track the costs during graph algorithms; a bool deleted field marking whether the node has been deleted from the graph and a int parent_graph_id field denoting which graph the node belongs to.
Edge	An object that takes in two Node objects, represents a directed Edge, has a double representing the weight. A user may create two Edge objects to represent one undirected Edge. They have an associated implicit mutex and a bool deleted field representing whether it has been deleted from the graph.
Graph	A list of Node objects connected by directed or undirected Edges representing a Graph data type.
void	The return type of functions that do not return anything. A variable of type void cannot be declared.

5. Declarations
   Declarations in a program are of the following grammar:

   program: declarations eof
   declarations: declarations var_dec | declarations fun_dec | **ε**

   There is no specified order in which the arguments of a function call are evaluated.

   a. Object Declarators
      Each type has its own declarator, formatted in the following way:

      int i;
      bool b;
      double d;
      String s;
      NodeList nl;
      EdgeList el;
      IntTable it;
      DoubleTable dt;
      Node n;
      Edge e;
      Graph g;

      Local variables can also be declared and initialized at the same time.

      double e = 2.718;

e = foo();

Global variables can only be initialized to constant expressions outside the scope of any function. They cannot contain assignment, other references, or function calls. The value of the global variable can be changed later on in the flow of the program.

Every global value is assigned a default value that corresponds to its type unless specified by the user. The user should not rely on default values for derived types in their code.

double e = 2.718;

Object declarations are of the following grammar:

type: int | bool | double | void | string | graph | node | edge | inttable | doubletable | nodelist | edgelist
var_dec: **type id ;** | **type id** = expression **;**

b. Function Declarators
Every function declaration starts with first indicating the return type of the function, followed by its name and the arguments it will take in parentheses. Multiple arguments are separated by commas within the parentheses.

The scope of the function is indicated by curly brackets, starting with { and ending with }. Each function ends with a return statement right before the end of its scope. Any code in the scope of the function and *after* the return statement is not run.

GRACL will look for a function named main whose return type is an int to begin parsing the program.

If no return value is stated, GRACL automatically assigns a return value that corresponds to the return type of the function.

```
returnType functionName (type arg1, type arg2, ... ){
        //This is the scope
        …
        return statement;
}
```

Function declarations are of the following grammar:

type: int | bool | double | void | string | graph | node | edge | inttable | doubletable | nodelist | edgelist

formals_opt: formal_list | **ε**
formal_list: **type id** | formal_list , **type id**
func_body: func_body var_dec | func_body statement | **ε**
fun_dec: **type id (** formals_opt **) {** func_body **}**

6. Lexical Conventions

   a. Comments
   GRACL has two types of comments: multiline comment with /**/ and single-line comment with //.

   The /* characters introduce a comment; the */ characters terminate a comment. They do not indicate a comment when occurring within a string literal. Comments do not nest. Once the /* introducing a comment is seen, all other characters are ignored until the ending */ is encountered.

   The // characters don't need characters to terminate the comment. As soon as the user moves on to the next line, the comment will be terminated.

   GRACL does not support nested multiline comments. E.g.: /*... /*... */... */

   b. Identifiers
   An identifier, or name, is a sequence of letters, digits, and underscores (_). The first character cannot be a digit or underscore. Uppercase and lowercase letters are distinct. Name length is unlimited. The terms identifier and name are used interchangeably.

   c. Keywords
   All data types, *for*, *while*, *if*, *else*, *return, True*, *False*, *hatch,* and *synch* are keywords. Functions with the same headers as functions in the standard library cannot be declared. Keywords cannot be used elsewhere.

   d. Punctuators
   A punctuator is a symbol that has semantic significance but does not specify an operation to be performed. The punctuators [ ], ( ), and { } must occur in pairs, possibly separated by expressions, declarations, or statements.

7. Operator Conversions

String doubleToString(double a)	Takes in a double and returns a corresponding String. For example, doubleToString(14.7) would return "14.700000" on success. Program breaks on failure. Printing the resulting string will show six places after the decimal.
double intToDouble(int d)	Takes in an int and returns a corresponding double. For

	example, intToDouble(14) would return 14.0 on success. Program breaks on failure.

8. Expressions and Operators
    a. Assignment Operator

=	Assignment for all types.

    b. Integer and Double Operators

- +	Additive for types int and double.
* /	Multiplicative for types int and double.
%	Modulus operator; only works for integers.

    c. Logical Operators

== !=	Equality comparison for types int, double, and bool.
< <= >= >	Relational comparisons for types int and double.
!	Logical not for Boolean expressions.
&&	Logical and for Boolean expressions.
||	Logical or for Boolean expressions.

    d. IntTable Operators
    Note: The table and key should be variable names

table[Node key] = int value	Adds a new key-value pair to the IntTable.
table[Node key]	Returns the value corresponding to the key.

    e. DoubleTable Operators
    Note: The table and key should be variable names

table[Node key] = double value	Adds a new key-value pair to the DoubleTable.
table[Node key]	Returns the value corresponding to the key.

    f. Precedence
    The rows are ordered from highest to lowest precedence.

Operator	**Associativity**
( ) [ ] .	Left to Right.

! - (unary)	Right to Left.
* / %	Left to Right.
+ -	Left to Right.
hatch synch	Nonassociative
< <= >= >	Left to Right.
== !=	Left to Right.
&&	Left to Right.
||	Left to Right.
=	Right to Left.

9. Scope
    a. Block Scopes
    The scope of an identifier is limited to the block in which it is defined. Each block has its own scope. No conflict occurs if the same identifier is declared in two blocks. If one block encloses the other, the declaration in the enclosed block hides that in the enclosing block until the end of the enclosed block is reached. The hiding hierarchy is as follows: local variables hide the formal variables which hide the global variables. Blocks are defined by {}.

```
if(!gracl_is_great) {
        int die = 1;
} else {
        int live = 1;
}
die = 0;          // This will not compile.
```

    b. File Scopes
    Identifiers appearing outside of any block, function, or function prototype, have file scope. This scope continues to the end of the file.

```
int x;
if (plt_is_the_best_class) {
        x = x+ 1          // This is allowed.
}
```

10. Statements

Statements are of the following grammar:

statement_list: statement_list statement | **ε**

statement: expression **;** | **return** expression_opt **;** | **{** statement_list **}**
| **if (** expression **)** statement **else** statement | **for (** type **id in** expression **)** statement
| **while (** expression **)** statement
| **hatch** expression **id (** args_opt **)** statement | **synch id** statement

expression_opt: expression | **ε**

expression: **literal** | **func_literal** | **binary_literal** | **string_literal** | **id**
| expression + expression | expression **-** expression | expression ***** expression
| expression / expression | expression **%** expression | expression = expression
| expression **<** expression | expression ≤ expression | expression **&&** expression
| expression || expression | **-** expression | **!** expression | **id** = expression | **id (** args_opt **)**
| **id .** call_chain | **id [id]** | **id [id]** = expression | **(** expression **)**

call_chain: **id (** args_opt **)** | call_chain **. id (** args_opt **)**

args_opt: args_list | **ε**

args_list: expression | args_list **,** expression

a. Selection Statements

;	Signifies the end of a statement.
if(*condition*){ *statements* } | if(*condition*){ *statements* } else{ *statements* }	Conditional statement; curly braces only required for multiple statements.

b. Iteration Statements

for(*Node/Edge id* in *NodeList/EdgeList*) { *statements* }
 	
while(*condition*){ *statements* }
	Loops; curly braces only required for multiple statements. The for...in loop goes over all elements of a NodeList or an EdgeList. The user should not edit the list that the loop is iterating over.

c. Function-Call Statements

functionName**(** args_list **);**	The function-call statement is used when a defined function is called. Parameters in args_list can be any

	of the primitive types or objects or expressions that evaluate to those types or objects.

11. Concurrency

    There are two keywords that deal with concurrency, *hatch* and *synch. hatch* takes a NodeList, which contains all possible starting nodes for a graph algorithm, a user-defined function and its parameters. The first parameter is always the node taken from the NodeList passed in by the user; hence, it is omitted. It creates however many threads are specified by the length of the passed in NodeList and passes function(parameters) as the thread start routine. The curly braces following the *hatch* call denote code executed by the parent process before it begins to wait for the threads to terminate. The parent process may not execute code after the final curly brace until all child threads have been reaped.

    Here is an example:

    //look at type of nodelist , add arg to the end of func that is exactly the node being passed from the list

    hatch NodeList func(ARGS){

        // Code executed by the parent thread before threads joined.

    }

    // Code executed after threads reaped.

    hatch takes in a nodelist and generates one thread for each, with each node as the first argument in the function

    synch takes in a variable name of one of the following objects: Node, Edge, EdgeList, NodeList, IntTable, DoubleTable. Under the hood, synch takes implicit mutex of the object and uses that to acquire and release the lock within the curly braces. After the closing brace is parsed, the lock is released. This helps to prevent user deadlocking since every *synch* call includes release of the lock.

    Here is an example, the curly braces are required no matter how many statements:

```
synch visited {          // locks before access of shared data
        for ( Node n in visited ) {
                print(n.data());
        }
} // unlocks after shared data access is complete
```

12. Library Functions

Note: the .func(params) notation indicates that the given function is called like object.func(params) for the given referred to object

Note: If a function errors in a non-recoverable way, the program breaks

a. NodeList Properties and Built-in Functions

int .length_NL()	Returns the length of the list as an int on success. Returns -1 on failure.
bool .empty_NL()	Returns True if the list is empty and False if it is not.
int .removeNode(Node n)	Removes the passed-in item from the list. Returns 0 on success and -1 on failure.
Node .removeFirst_NL()	Removes the first item from the list. Returns the removed node. Program breaks if .removeFirst() is called on an empty NodeList.
Node .removeLast_NL()	Removes the last item from the list. Returns the removed node. Program breaks if .removeLast() is called on an empty NodeList.
void .appendNode(Node n)	Appends the passed-in Node to the end of the list.
void .prependNode(Node n)	Prepends the passed-in Node to the beginning of the list.
Node .head_NL()	Returns the head of the list
Node .tail_NL()	Returns the tail of the list
bool .includesNode(Node n)	Checks if Node n exists in the nodelist

b. EdgeList Properties and Built-in Functions

int .length_EL()	Returns the length of the list as an int on success. Returns -1 on failure.
bool .empty_EL()	Returns True if the list is empty and False if it is not.
int .removeEdge(Edge e)	Removes the passed-in item from the list. Returns 0 on success and -1 on failure.
Edge .removeFirst_EL()	Removes the first item from the list. Returns the removed edge. Program breaks if .removeFirst() is called on an empty EdgeList.
Edge .removeLast_EL()	Removes the last item from the list. Returns the removed edge. Program breaks if .removeLast() is called on an empty EdgeList.
int .appendEdge(Edge e)	Appends the passed-in Edge to the end of the list. Returns 0 on success and -1 on failure.
int .prependEdge(Edge e)	Prepends the passed-in Edge to the beginning of the list. Returns 0 on success and -1 on failure.
Edge .head_EL()	Returns the head of the list
Edge .tail_EL()	Returns the tail of the list

c. IntTable Properties and Built-in Functions

NodeList .intKeys()	Returns a list containing all Node keys in the hashtable.
int .deleteInt(Node n)	Deletes the key-value pair with the passed-in Node as the key from the IntTable. Returns 0 on success and -1 on failure.

bool .inInt(Node n)	Returns a bool as to whether or not a key exists in the IntTable.
int .hashCode_it(Node n)	Returns the result of the hash function.

d. DoubleTable Properties and Built-in Functions

NodeList .doubleKeys()	Returns a list containing all Node keys in the hashtable.
int .deleteDouble(Node n)	Deletes the key-value pair with the passed-in Node as the key from the DoubleTable. Returns 0 on success and -1 on failure.
bool .inDouble(Node n)	Returns a bool as to whether or not a key exists in the DoubleTable.
int .hashCode_dt(Node n)	Returns the result of the hash function.

e. Node Properties and Built-in Functions

String .data()	Returns data stored in the Node object.
NodeList .neighbors()	Returns a list of neighbor Nodes.
EdgeList .edges()	Returns a list of the node's edges.
bool .visited()	Returns a boolean representing if the Node has already been visited.
double .cost()	Returns the cost associated with the Node. Initialized to -1.0 .
Node .prec()	Returns the precursor assigned to the node by the user.
Node .setPrec(Node n)	Assigns the node n as the precursor of the node and returns its newly assigned precursor.
Node .updateData(String data)	Updates the data field on the Node to be the new data passed in and returns the updated node

Node .updateVisited(bool tf)	Updates the visited field on the Node to be the new bool passed in and returns the updated node
bool .nodeEquals(Node n)	Compares the two Node objects. Returns True if they are the same and returns False if they're not the same. Under the hood, this is implemented by comparing the implicit id associated with the Node object.
Edge .getEdge(Node n)	Returns the Edge object connecting the Node it's called on and Node n. If the two Node objects do not share an edge, then it returns NULL.
double .incrementCost()	Increments the node's cost by 1 and returns the node's new cost.
double .decrementCost()	Decrements the node's cost by 1 and returns the new cost.
double .updateCost(double new_cost)	Sets the cost of the node to the passed in double new_cost and returns it.

f. Graph Properties and Built-in Functions

NodeList .nodes()	Returns a list of Node objects in the graph.
Node .createNode(String data)	Creates a new Node, setting its data. It returns a pointer to a Node object that the user can save in a Node variable.
int .removeNodeGraph(Node n)	Removes the passed-in Node from the graph and deletes corresponding Edges. Returns 0 on success and -1 on failure.
Edge .addEdge(Node source, Node dest, double weight)	Adds the Edge e to a Node's EdgeList, and returns that Edge.
int .removeEdgeGraph(Edge e)	Removes the given Edge e and returns 0, returns -1 if the passed in Edge does not exist in the graph.

g. Edge Properties and Built-in Functions

void .updateEdge(double a)	Updates the directed Edge to have a new weight.
Node .start()	Returns the start Node of the Edge object.

Node .end()	Returns the end Node of the Edge object.
double .weight()	Returns the weight of the Edge object.
bool .edgeEquals(Edge e)	Compares the two Edge objects. Returns True if they are the same and returns False if they're not the same.

h. String Properties and Built-in Functions

int .stringLength()	Returns the length of the String.
bool .stringEquals(String s)	Returns True if both strings represent the same array of characters, False otherwise.

Note: String creation should be done by assigning a String literal to a String variable, similar to how primitive types are initialized. For example: String s = "GRACL";

i. General Library Functions and Constants

void print()	Prints String objects with a newline at the end.
void printi()	Prints int objects with a newline at the end.
infinity	Double type constant representing infinity. Useful for graph algorithms that have infinite edge weights. infinity - infinity and infinity/infinity are undefined
Graph createGraph()	Creates and returns an empty Graph.
NodeList createNodeList()	Creates and returns an empty NodeList.
EdgeList createEdgeList()	Creates and returns an empty EdgeList.
IntTable createIntTable()	Creates and returns an empty IntTable with Node keys and int values.
DoubleTable createDoubleTable()	Creates and returns an empty DoubleTable with Node keys and double values.

V. Project Plan

1. Planning Process

We created a timeline for ourselves that consisted of major milestones that we should achieve. We also made sure to communicate regularly in order to plan out at least one weekly meeting. Towards the end of the project, we strove to work in parallel to increase efficiency. We also met with Edwards frequently as he was our main source of direction on the project, and reassigned our priorities based on these meetings.

2. Specification Process

We started out with our initial proposal: Our first ideas were to have a graph that consisted of nodes and edges. As for concurrency, we were aiming to have thread and RwLock objects. Our feature specifications changed as we looked into different graph algorithms, such as Tarjan's, where we decided to add Int and Double tables. We also decided to add Edgelists andNnodelists after looking at such algorithms, attending office hours, and talking about various methods of implementation. Lastly, we vastly changed our features for concurrency by removing threads and Rwlock objects but instead, inspired by Java, adding a hatch and a synch function.

3. Development Process

We followed the compiler architecture's structure by starting off with the lexer, the parser, the semant file. Then we started to write the C libraries and modified the aforementioned files accordingly. Toward the end, we wrote graph algorithms, concurrent algorithms, and concurrent graph algorithms in C and in GRACL.

4. Testing Process

For every cfile written, we would test first in C, and then translate into GRACL. Additions to the compiler were tested directly in GRACL. Since we added a number of tests, we also wrote scripts in order to organize and streamline testing.We created two main scripts dedicated to testing purposes for GRACL tests: testall.sh, which runs all the tests and prints whether they passed, testone.sh, used to test one single program. Finally, we added a third script time-dfs.sh to test how concurrent DFS (depth-first-search) compares to normal DFS in terms of performance.

5. *Style Guide*

Variables were named like foo_bar_baz or in camelCase, types were labeled in upper case, single line definitions had the *let* and *in* on the same line, while larger functions had the *in* on a separate line after the function.

The C code followed standard C conventions.

6. ***Software Development Environment***
   The development environment was the Docker image provided, and most people used VS Code with its support for OCaml and Git.

7. ***Team Responsibilities***

Team Member	**Responsibilities**
Hadley Callaway, Manager	C Library and testing, GRACL Testing/graph algorithms
Pelin Cetin, Tester	Concurrency in C, Concurrent Graph Algorithms, Library and testing, Codegen, Data Structure and Algorithm Design, Dockerfile, GRACL Testing/Graph algorithms, Testing Scripts
Eilam Lehrman, Language Guru	Compiler Frontend, Semantics, Codegen, GRACL Testing, Concurrency
Defne Yagmur Sonmez, System Architect	C Library and testing, GRACL testing/graph algorithms
Maya Venkatraman, System Architect	C Library and testing, Concurrency, Collision Implementation, Data Structure and Algorithm Design, Concurrent Graph Algorithms

8. ***Project Timeline***

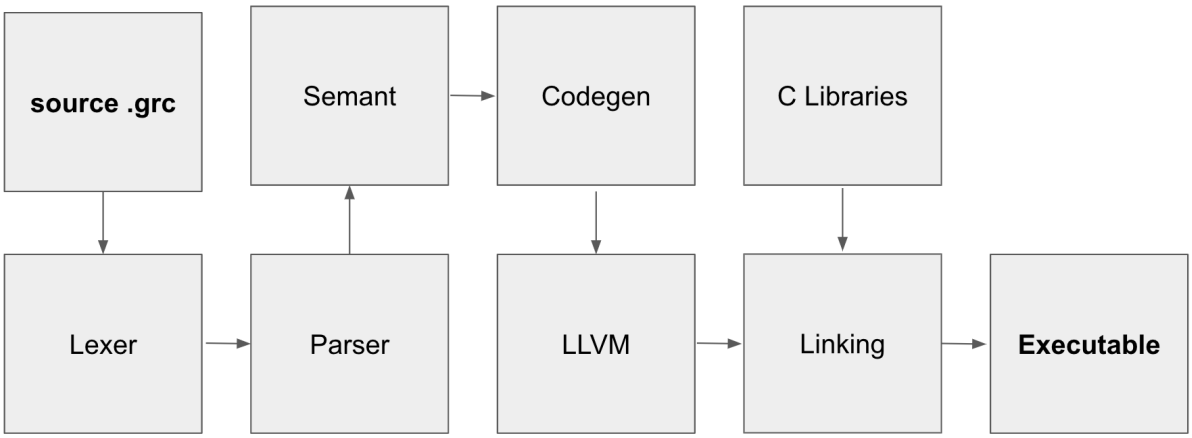
Date	Milestone
February 10, 2021	Language proposal and whitepaper complete
March 12, 2021	Language Reference Manual complete
March 12, 2021	Compiler frontend (Lexer, Parser) complete
March 24, 2021	Hello World runs
April 14, 2021	C Library done
April 20, 2021	Dijkstra works in GRACL
April 25, 2021	Concurrent Graph Algorithms work in GRACL

April 25, 2021	Concurrency complete
April 26, 2021	Presentation, Final Report complete

9. ***Project Log (see Appendix A)***

VI. Architectural Design

1. *The Compiler (Eilam, Pelin)*



The GRACL compiler, found in gracl.ml, begins by taking in a program, and then lexes it with scanner.mll to turn it into a stream of tokens and remove comments. These tokens are then passed to the parser (graclparser.mly), which ensures the program is syntactically correct, and then creates an AST with some added features (such as BlockEnd statements) that were not part of the original program, but help with semantic checking. The AST is then passed to semant.ml, which checks that the program is semantically correct, again with some added features (such as renaming every variable to ensure uniqueness within a function) to help in with the next step, code generation. Code generation is handled by codegen.ml, which traverses the SAST and builds up an LLVM module. The file functions.ml contains a list of every function in the standard library, and that list is then linked in where it is needed in semant (for checking those functions are called properly) and codegen (for generating the calls to those functions), this allows for the OCaml and C to interface purely through one file, so it is quite simple to add new built-in functions to the standard library. Once the LLVM is generated, it is passed to LLC, the LLVM compiler, to turn it into an assembly file. That assembly is then passed to the C compiler, and gets linked in with the pthread library and the standard library, and becomes an executable which can then be run. The files ast.ml and sast.ml define the types for the elements of the AST and the SAST, respectively.

An important note, the LLVM generated is platform dependent, and should only be run from the docker or some equivalent operating system. This is because the mutex and pthread related structs differ in LLVM on different operating systems.

2. *The C Libraries (Maya, Pelin, Defne, Hadley)*

The C libraries are linked to the compiler to provide the built-in graph functions to the user.

a. Lockedobject.h

The header file in which all object definitions reside. The objects defined in this file that are available to the user are Node, Edge, EdgeList, NodeList, IntTable, DoubleTable, Graph. Other objects defined in this file that are not exposed to the users include IntTableItem, IntTableLLItem, DoubleTableItem, DoubleTableLLItem.

b. node.h, node.c

The node header file and code for the node functions. Functions called on nodes are defined in these files.

c. edge.h , edge.c

The edge header file and code for the edge functions. Functions called on edges are defined in these files.

d. edgelist.h, edgelist.c

The EdgeList header file and code for the EdgeList functions. Functions called on EdgeLists (or involving EdgeLists) are defined in these files.

e. nodelist.h, nodelist.c

The NodeList header file and code for the NodeList functions. Functions called on NodeLists (or involving NodeLists) are defined in these files.

f. inttable.h, inttable.c

The IntTable header file and code for the IntTable functions. Functions called on IntTables are defined in these files. Collision hashing is implemented here.

g. doubletable.h, doubletable.

The DoubleTable header file and code for the DoubleTable functions. Functions called on DoubleTables are defined in these files. Collision hashing is implemented here.

h. graph.h, graph.c

The graph header file and code for the graph functions. Functions called on graphs are defined in these files. Collision hashing is implemented here.

i. **concurrency.c**

The functions associated with synch and hatch are defined in these files. Since these functions are not exposed to the user, the compiler side makes calls to the function in this file during synch and hatch blocks in the code.

j. **graclstdblib.c**
This files has **#include** statements for all the c files and the lockedobject.h file mentioned above. The Makefile for the compiler uses this file to generate the graclstdlib.o file and links it in.

VII. Testing Plan

Note: Concurrent tests are expected to fail due to non-deterministic output

1. ***Unit Testing (Lexer, Parser, Codegen)***

Unit testing was done manually with the different flags on the compiler, after each integration test was written and while its corresponding feature was developed, that test file would be checked first as an AST, then SAST, then the LLVM would be checked for validity, before finally checking the output.

2. ***Integration Testing***

Integration testing was done by testing every part of our language in GRACL. The integration tests were grouped by: identifiers, types, keywords, statements and blocks, control flow, built-in functions, comments, operators, variable and function declarations, graph types, graph functions, graph algorithms, concurrency and concurrent graph algorithms.

3. ***System Testing***

Hello world created and tested. More meaningful tests were run here. A few selected algorithms were fed to our compiler (such as Dijkstra) and the output verified. Concurrent algorithms (concurrent graph algorithms in particular) were fed to our compiler, but the output cannot typically be verified as concurrent tests have no regulated thread order. The test files are located in directories listed in the Test suite subsection below.

4. ***Automation***

The testall script was borrowed heavily from MicroC.
The test-script file under cfiles/tests loops over the C file executables coming in from running make and compares the output of the program with the filename_example.txt

files. Simply run bash **$ ./test-script.sh** under cfiles/tests and it will tell you if the C tests have passed or not.
The time-dfs script is a script that runs these normalDFS.grc and concdfs.grc 10 times each and compares the timing results and echo to the user. Simply run bash ./time-dfs.sh in the top directory.

5. ***Test suites***

The ./testall.sh runs every integration test that we have, and as noted above the concurrent ones are expected to fail.
./scripts/testone.sh tests the output of one test file, ./scripts/failone.sh tests the failing of one fail file, and ./scripts/breakdown.sh shows every part of how a test is run, from the source code, through the AST and SAST to the LLVM, finally ending with the output when the LLVM is linked into an executable that is run.

We tested the following features of our language:

- Arithmetic and logic operators
- Variable declaration and assignment
- Block Scoping
- Function Declaration
- Control flow. We checked if-else blocks whose bodies should or should not execute, and verified the number iterations of loops. We also tried stressing the language's ability to handle nested blocks.
- For loops
- Strings
- Built-in functions and objects
    - Node, Edge, EdgeList, NodeList, Graph, DoubleTable, IntTable
- Concurrency (hatch/synch keywords)
- Major algorithms
    - Dijkstra
    - DFS
    - Concurrent DFS
    - Concurrent Bidirectional Search
        - This has confusing output but illustrates that overall the two threads collide at node E in the middle of the graph as desired

6. ***Testing roles***

We borrowed the pretty-print behavior from MicroC for the AST/SAST. Pelin created scripts to automate our testing process both in GRACL and in C. Maya wrote a majority of C tests. Defne and Hadley wrote tests in C and GRACL. Maya and Pelin wrote concurrent graph integration tests. Hadley and Defne wrote Dijkstra both in C and GRACL.

VIII. Lessons Learned

1. Hadley Carolyn Callaway

   I went from never touching concurrency or functional programming languages to feeling confident working with both. Working on GRACL applied many of the concepts I studied over the past few years—language features, memory allocation, grammars, etc. Combining familiar concepts with new ones is a potent learning tool.

   Advice: Take advantage of each person's talents BUT do not silo. Empower your teammates to work on what they can know best and to which they can contribute the most. At the same time, take time to walk each other through the work you're doing, so that everyone gets the same learning benefit. For example, Eilam did the most work on the Codegen and spent time explaining what he was doing to me. That way, when it came time for me to add functions from the C types into the Codegen or to make sure the Codegen types matched the C types we had defined, I was able to do so confidently even though it wasn't my area of expertise.

2. Pelin Cetin

   I learned a new way of looking at coding languages. Before this class, I was taking coding languages for granted and now that I understand the behind the scenes, I really started appreciating them. I am now looking at languages I don't know with a new pair of eyes, wondering if it's normal or applicative order and being able to use what I've learned in the class.
   Coming into the class, I've had some functional language experience with Scala from Honors Data Structures but I didn't feel confident in it. Learning OCaml has opened new doors for me and now I am more than eager to take Professor Edwards' functional language class.
   Throughout the semester, I made an effort to work at every part of the project: compiler backend, the C library, the testing scripts, Dockerfile, and writing concurrent algorithms in our language. I feel so amazed that we were able to imagine a language and make it a reality and get to learn how everything works with each other.

   Advice: Clang and print statements are your friends. Get a good group and make sure that you guys communicate well and let each other know when you need help. If you don't communicate with your teammates, then your group will turn into a train wreck. Test everything, start early, go to office hours when you're confused. This class is hard but it's worth it for the happiness and the sweet sting of accomplishing something hard.

3. Eilam Lehrman

   I learned what a programming language actually is, and the process that turns a program into something the computer can actually execute. An important takeaway is that going from a programming language to assembly code is next to impossible to do in one step, but going step by step from a program to tokens to an AST to an SAST to code generation is a much more manageable and understandable process. This process also makes developing a compiler much easier, as when it came time to implement new features it was mostly a matter of

deciding which step could take care of it, and the earlier in pipeline the better. For example, the obj.method() notation was implemented by having the parser parse it as method(obj), so it could be handled by the existing func(var) structure for function calls in semant and codegen. However, something like block scoping could only be done in semantic checking as it requires using the symbol table that is built up from the AST, but it was done in a way to not impact Codegen. I found this to be useful considering how delicate the LLVM in codegen could be, I came to prefer the verbose OCaml errors over the segfaults LLVM offered. Implementing your own programming language seemed like an insurmountable task, but once I understood how to break it down into components with defined roles, it became much easier to approach.

Advice: Have a narrow idea of what you want your language to do, and do it right. It is easy to get lost in the details, but once you have the main bulk of a compiler running all the way through it is not that hard to add new features, compared to trying to get everything to work together all at once. Also, CLANG IS YOUR BEST FRIEND! If you know how to do something in C, you can do it in your language, but make sure you really do know how to do it in C. LLVM will seem daunting at first, but once you start working with it you will start to see the patterns behind how it works. Really make sure you understand every part of microC as best as you can, it is a great starting point for any compiler.

4. Defne Yagmur Sonmez

   I learned a lot about creating your own language and the steps that go into it. At first, I thought it would be very difficult and daunting but once broken down into its components and shared among group members it was a lot more approachable and feasible than I thought. I also naturally learned a lot about implementing graphs and concurrency. I had not done concurrency at all before this project and was hesitant on implementing it but was glad to be exposed to a completely new concept. Finally, I was once again reminded of the importance of planning and communication in group projects. I think we were able to work well together because we were organized about what should be done by the end of each week and had our own deadlines/expectations apart from the assignments of the class itself.

   Advice: Come up with a language idea that you think is feasible. It is likely that you may not be able to implement everything you want and you will have to change features so ready to be flexible about things. Lastly, maybe have a few milestones that you would be ok with if your language ended there. For example, if we weren't able to implement concurrency, we would have still been happy to have our graph features and have a graph algorithm.

5. Maya Venkatraman

I knew a lot about concurrency before I took the class, but had only ever used it in C/C++. Something really exciting about this process to me was learning about concurrency through the lens of other languages, such as Java, and learning how to implement that behavior in our language. I had no experience with functional programming before this class, so I also learned a large amount there. I also now feel comfortable understanding every step of compiler design, so I went from no idea how languages are implemented under the hood to a decent understanding of the steps that occur on behalf of the compiler when a program is run.

Advice: Make sure you pick a subject area you know well. Our project would have been extremely difficult if I hadn't been very familiar with both graph algorithms and concurrency. Leverage what you learned from your favorite classes and delve deeper in those areas. Go to the professor's office hours at least once every two weeks if you can. Test early and often.

Appendix A

commit f221bb61a90eb16f786c9dff107d3cbd307b416d
Author: E L <eilamemail@gmail.com>
Date: Mon Apr 26 21:08:38 2021 -0400

Small touch up

commit 9284a6bc7df5683987b9c6fa37ae703fa48cd90a
Merge: 7b753b7 f65cfd3
Author: defne sonmez <dys2109@columbia.edu>
Date: Mon Apr 26 15:04:46 2021 -0400

jgkldmkldslfkerge branch 'conc' of https://github.com/pelincetin/GRACL into conc

commit 7b753b7e2d650d6a18e3dc2303a6e3a790426b15
Author: defne sonmez <dys2109@columbia.edu>
Date: Mon Apr 26 15:04:25 2021 -0400

added more to dijkstra

commit f65cfd36c81ab6c9cc7b40cde5c707d913e2e90d
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 26 14:28:01 2021 -0400

progress on bidirection

commit be321127946d4a706a2becc6b4cf9d2008e2f7a8
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 26 13:43:54 2021 -0400

bidirection

commit 24920d1ae91e26c96888593032802b930d531376
Merge: 0f0bf0b bddd6a2
Author: pelincetin <pc2807@barnard.edu>

Date: Mon Apr 26 13:07:45 2021 -0400

    resolving merge conflicts

commit 0f0bf0baac649533ca3454074e24c05da21d6e3f
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 26 13:06:32 2021 -0400

    test-dfs update

commit bddd6a22be45b841f8db5766c48b0a8b28e1ad18
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 26 13:01:31 2021 -0400

    update the tests on main

commit 549c84fb613704a3e4d1741dc34830602787b753
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 26 02:44:42 2021 -0400

    updated concdfs

commit 77214877deb65c80370cc813ff02dea4072c4058
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 26 02:41:03 2021 -0400

    updated timer and concdfs

commit 7fca034370c0dd6a78c496b6a8329947e33a68bd
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 26 02:34:30 2021 -0400

    updated timer

commit 04e0933c012fc1e159fdab1e7aa7bc75f0b5fea5
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 26 00:51:56 2021 -0400

    edge test

commit d5a71e598adec0ce746b53fa5925087a12bffefa
Merge: fb19be6 072b256
Author: defne sonmez <dys2109@columbia.edu>
Date: Mon Apr 26 01:56:37 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc

commit fb19be6e96b491b1db32e53777c0e610e8b1199a
Author: defne sonmez <dys2109@columbia.edu>
Date: Mon Apr 26 01:56:26 2021 -0400

    tarjan attempt, compiles but doesnt produce anything

commit 072b2567b703fbc0a7262b23d169e4d3dd5f4173
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 26 00:47:56 2021 -0400

```
    conc: DFS optimzied

commit 2f3d587faec7b15f9ff3a26d56c63e29ba2a87e5
Author: Maya <mv2731@columbia.edu>
Date:   Mon Apr 26 00:36:26 2021 -0400

    conc: Giant graph normal dfs

commit adfa2e0118180cd2715f43b01c48f63e1e9dd985
Author: pelincetin <pc2807@barnard.edu>
Date:   Mon Apr 26 00:24:23 2021 -0400

    graph test in gracl

commit 868fba9bb0e5cf14dfefdc2a9eb613324ad5cb0b
Merge: fb3d647 7bdab4a
Author: Maya <mv2731@columbia.edu>
Date:   Mon Apr 26 00:18:09 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc

commit fb3d6479e3b5697d613a7f7553729da78f423a7a
Author: Maya <mv2731@columbia.edu>
Date:   Mon Apr 26 00:18:02 2021 -0400

    conc: Big graph conc dfs

commit 745b4ad77e43d3c841f99b99a061cd4f2597e805
Merge: 25760ca 7bdab4a
Author: pelincetin <pc2807@barnard.edu>
Date:   Sun Apr 25 23:42:14 2021 -0400

    Merge branch 'conc' into main

commit 25760ca73adb3ad7f99585048951521edf7d1ad5
Author: pelincetin <pc2807@barnard.edu>
Date:   Sun Apr 25 23:41:26 2021 -0400

    clean up conc

commit 31fc011deeac9635e900be556bfb4dc84940ce0c
Author: pelincetin <pc2807@barnard.edu>
Date:   Sun Apr 25 23:14:43 2021 -0400

    timer script

commit 7bdab4aec0cabcf83a62ddf6a2ef918c26b1d662
Author: pelincetin <pc2807@barnard.edu>
Date:   Sun Apr 25 23:41:26 2021 -0400

    clean up conc

commit 4233f2fa18adf18544783c7f3cc61b2a62ae98a6
Author: E L <eilamemail@gmail.com>
Date:   Sun Apr 25 23:30:53 2021 -0400
```

```
    Clean up codebase

commit 04c4aada7bff00569abbcd648dcf5e2032cf979f
Author: pelincetin <pc2807@barnard.edu>
Date:   Sun Apr 25 23:14:43 2021 -0400

    timer script

commit d3f89a453f1fd2199fb7053c04304fb627f6508c
Merge: 86eb797 dacc6af
Author: Maya <mv2731@columbia.edu>
Date:   Sun Apr 25 22:06:05 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc

commit 86eb7974cf9da8c4aa56f380ba89e310a8479c28
Author: Maya <mv2731@columbia.edu>
Date:   Sun Apr 25 22:05:57 2021 -0400

    conc: Conc DFS works

commit dacc6af789966c15b93b286df46bd2ef1b5586c2
Author: pelincetin <pc2807@barnard.edu>
Date:   Sun Apr 25 21:52:55 2021 -0400

    added the booleans to edge and node

commit de822fe961589e05c06c92f483d7e2d13a638952
Author: pelincetin <pc2807@barnard.edu>
Date:   Sun Apr 25 21:01:30 2021 -0400

    edge test

commit dbbd7073fbc85ce057bf4efc591dfcd554eb4764
Merge: ea3c2c3 059c9e3
Author: Maya <mv2731@columbia.edu>
Date:   Sun Apr 25 20:46:28 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc

commit ea3c2c31c2e6dacc937d1b152ecc289ba98ee979
Author: Maya <mv2731@columbia.edu>
Date:   Sun Apr 25 20:46:21 2021 -0400

    conc: Nodelisttest done

commit 059c9e39de2181143694a21d806498afaf214115
Author: Maya Venkatraman <mv2731@columbia.edu>
Date:   Sun Apr 25 20:46:07 2021 -0400

    Delete nodelist-test

commit 998e1cea4fbf3ee81491f8a61587bd7c358ef339
Author: Maya <mv2731@columbia.edu>
Date:   Sun Apr 25 20:44:47 2021 -0400
```

    conc: Nodelist test done

commit 42487239ff1fbf7f65856bbdbb7de0db61ca4f82
Merge: c888fca e4fb212
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 20:42:33 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc

commit c888fca058d8dbaafead0f97914536207b080c29
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 20:42:03 2021 -0400

    conc: Nodelist test done

commit e4fb2129c7ace0e960a5b4a78fb72d50f4c4bbf6
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 25 20:31:32 2021 -0400

    even more testing

commit 5e1657040991485eeef599859fba1d511924c71d
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 20:26:27 2021 -0400

    conc: Inttabletest done

commit 58e28066efb556f8c51ba332188b7ad40bdfeb8a
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 25 20:22:48 2021 -0400

    doubletable edgelist tests

commit 256ed60b0968a2b18e1daa348267685150a5fe26
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 25 19:56:51 2021 -0400

    edgelist test done

commit 14b65ae15009e56967c721c0266544aaf8199d70
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 25 19:45:40 2021 -0400

    Added to hatch test

commit c3602a83537496beddb4689db46369dce68416f9
Merge: 64a654f 6370918
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 25 19:38:58 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc
    Get most recent

commit 64a654fa55e0caccce43717eb608847c1094056a
Author: E L <eilamemail@gmail.com>

Date: Sun Apr 25 19:38:19 2021 -0400

    Hatch working

commit 63709188709293e1b7f32de928d474a8b44e630c
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 25 19:20:23 2021 -0400

    beefy tests

commit b80a654d093834dedf79a9b48ca899db26929614
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 25 19:07:08 2021 -0400

    graph test done

commit 0f516e65d5a07bd2224979f806c3bb1874281238
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 25 18:47:12 2021 -0400

    progress on graph test

commit 42de9426be0a3d6b0d98d5307dfc0b7daca7437d
Merge: 7c6c0cf ecfc053
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 25 19:13:19 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc
    Getting most recent

commit 7c6c0cf58dfee2733283893c1b6bec2c3e5c69b9
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 25 19:12:49 2021 -0400

    Added hatch to codegen

commit ecfc053903bd079822e0de02a0ac0c92a7514ad3
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 19:11:07 2021 -0400

    conc: Finish edge-test error behavior

commit 81fd7d9f5cabaea27cb73e2336edae502fd445e4
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 18:59:34 2021 -0400

    conc: Make removeNodeGraph more robust

commit f7a71536135a8052a6515b1157d6a12ad1b1667e
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 18:46:23 2021 -0400

    conc: Add edge delete true in nodeRemove

commit 95bda6d19fdd7fab584e0f6a903d183801929620
Author: Maya <mv2731@columbia.edu>

Date: Sun Apr 25 18:27:59 2021 -0400

    conc: Fix node-test to have all error handling

commit 757dcdb824d4ac5bd6bee43ccaa882f54a126266
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 17:36:52 2021 -0400

    conc: Add error handle

commit d460b5a4893d7bb18a59b93c0bb04e74ce6ac50c
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 16:57:51 2021 -0400

    introduce the deleted bool to nodes and edges

commit 2a9a54caba9e26b295c9520a5d1de6f997a3d39f
Merge: 3222e24 1ca5ee3
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 15:19:04 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc

commit 3222e24b84ac7d81c697cae3a46e592193c3ac59
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 15:18:58 2021 -0400

    conc: Test graph collision

commit 1ca5ee3177c1b940ce61423ac107b82a83de8fdc
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 25 15:17:59 2021 -0400

    hatch-test makefile

commit bd73a056327e94f30a23e12cecce27665907026c
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 25 15:15:13 2021 -0400

    hatch tests

commit 1de2db09ac71893d5e6402ed37c36740a757a81f
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 25 15:04:30 2021 -0400

    merge

commit 612fcbf58a693e88773d04163d30e1cc99d40293
Author: defne sonmez <dys2109@columbia.edu>
Date: Sun Apr 25 15:11:20 2021 -0400

    commented out print statements

commit f2742ec28795d8998f693957f629fcc15b074f4a
Merge: 836832e 197677a
Author: defne sonmez <dys2109@columbia.edu>

Date: Sun Apr 25 15:08:57 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc
    5 millionth

commit 197677a5a9f23863ac2ed8ac7ee2f16c38cae29d
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 15:07:53 2021 -0400

    conc: Readd important file :D

commit 836832e6a9435154f2697665664d9889f009e351
Merge: 4c35100 058639a
Author: defne sonmez <dys2109@columbia.edu>
Date: Sun Apr 25 15:07:34 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc
    to push

commit 4c351005f2bc4c2e33e9437ef8d7acf2bc79065e
Author: defne sonmez <dys2109@columbia.edu>
Date: Sun Apr 25 15:06:49 2021 -0400

    added more testing for neighbors in c

commit 058639ae5afdfa250c37f3f4bd5ada56e0e4a35f
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 15:03:07 2021 -0400

    conc: Moved normalDFS

commit 1fd5ed1ff7ad1b910cf5fd96c121b86e43d535bf
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 14:49:24 2021 -0400

    conc: Fix errors in collision graph

commit e3bfd7b8d95418a71dd588fa4bb5956a0435698f
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 14:47:58 2021 -0400

    conc: Graph collision implemented

commit 5929896e6b2f1231e02a31e3c1725caaff4e9eaa
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 14:28:08 2021 -0400

    conc: Clean up algos

commit 148fc1a61efdcf2aac6c4871a2766d009dcb97ab
Merge: e598d05 dba659b
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 14:12:35 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc

commit e598d05244e37df6b7eda66519ecedcc31f49fda
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 25 14:12:28 2021 -0400

    removed frees

commit 170291de096c6b22ff18759b0b66b7a57b7e31be
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 25 04:12:24 2021 -0400

    Hatch unwrapper done

commit dba659b9894284f1a0b85dd8f0d0da1b4fe12b05
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 25 00:29:59 2021 -0400

    Concdfs parses

commit de1fdc8cfe1fb933a9d4885185d0008b7190fe12
Author: E L <eilamemail@gmail.com>
Date: Sat Apr 24 23:48:46 2021 -0400

    Ready for hatch

commit 15435650d640e1ee0a0803e78331de2199767de5
Merge: ad30301 ebbde55
Author: E L <eilamemail@gmail.com>
Date: Sat Apr 24 22:49:12 2021 -0400

    Merge branch 'conc' of https://github.com/pelincetin/GRACL into conc
    int isprime(int n){
        for (int i = 2; i <= floor(sqrt(n)); i++){
            if (n % i == 0) return 0;
        }
        return 1;
    }

commit ad303012be3f66001450b35f6a23cf47509418b3
Author: E L <eilamemail@gmail.com>
Date: Sat Apr 24 22:48:59 2021 -0400

    Fix mutex type, thus fixing for loops

commit ebbde5513c6075f55ea1a96f6beb1edc8b3f38fb
Author: Maya <mv2731@columbia.edu>
Date: Sat Apr 24 19:22:16 2021 -0400

    conc: Comments for Eilam

commit 24b6f8fc8c1bcf0ec8435b120758c644da3396e8
Author: Maya <mv2731@columbia.edu>
Date: Sat Apr 24 19:04:00 2021 -0400

    conc: Wrote hatch_end

commit d22c7b1d9d1fd4d3f0d89a804c7133655af3e7ea

Author: Maya <mv2731@columbia.edu>
Date: Sat Apr 24 18:19:38 2021 -0400

    conc: Added neighbors func, wrote concdfs in grc

commit 233ccd8038db6101c9e20471e149e8d9619b263f
Author: E L <eilamemail@gmail.com>
Date: Sat Apr 24 18:11:27 2021 -0400

    Fixing tests

commit 13956fdc09e532e88bb603e1b2c8c17f74d09d1b
Author: Maya <mv2731@columbia.edu>
Date: Sat Apr 24 17:35:28 2021 -0400

    conc: Fix exit behavior, conc DFS working

commit a2a5ad70887a92149b8a686a6fe9ceaa12970247
Author: pelincetin <pc2807@barnard.edu>
Date: Sat Apr 24 14:57:27 2021 -0400

    nodelist edgelist mutex

commit 2f55b870b25e4c4073de6e6c29c4e69b7dda625b
Author: pelincetin <pc2807@barnard.edu>
Date: Sat Apr 24 14:53:56 2021 -0400

    progress on normaldfs

commit 05b5b06b5fd3797b9a3c210b230687b0f0a24e6e
Author: pelincetin <pc2807@barnard.edu>
Date: Fri Apr 23 18:06:51 2021 -0400

    i will never stop printing yo

commit f940e47d8394bea8784b222d0f8d1c3eb9f1fd69
Merge: aac71e7 de7336c
Author: pelincetin <pc2807@barnard.edu>
Date: Fri Apr 23 17:16:35 2021 -0400

    merge

commit aac71e751ed38e11a776e2cb5cf9eaa567fa7e91
Merge: 85992a0 efd6a08
Author: pelincetin <pc2807@barnard.edu>
Date: Fri Apr 23 17:12:26 2021 -0400

    Merge branch 'main' into conc

commit 85992a075b1b7c50bdfcef0077953d012e0fae0a
Author: pelincetin <pc2807@barnard.edu>
Date: Fri Apr 23 17:12:11 2021 -0400

    dfs progress

commit de7336c6ff72f257240ac6e5ba6ef7d8d4893c72

Author: defne sonmez <dys2109@columbia.edu>
Date: Fri Apr 23 16:09:40 2021 -0400

    dijkstra and nodefull work, dijkstra.out file has problems

commit 0889ca684a2369dc48a9cd5d36de527546abf406
Merge: 331b25d bd14f36
Author: defne sonmez <dys2109@columbia.edu>
Date: Fri Apr 23 12:09:43 2021 -0400

    Merge branch 'dikstra' of https://github.com/pelincetin/GRACL into dikstra
    sure bro

commit bd14f3603c12289d1ffce1511a4448f3e8d797cb
Author: E L <eilamemail@gmail.com>
Date: Fri Apr 23 12:08:36 2021 -0400

    add print debug

commit 79251a3870debc9928fc6ec58b98864dd40cbb7e
Author: E L <eilamemail@gmail.com>
Date: Fri Apr 23 12:00:54 2021 -0400

    Fix branching error in for loops

commit c9070297503e529059ed2d18b68d36bdbcb3ce8e
Merge: f1db1ff b3d0dc4
Author: E L <eilamemail@gmail.com>
Date: Fri Apr 23 11:33:04 2021 -0400

    Merge branch 'main' into dikstra
    yo

commit 331b25d123febf65f255b648f27462044547c65e
Merge: 4a5d64f b3d0dc4
Author: defne sonmez <dys2109@columbia.edu>
Date: Fri Apr 23 00:22:47 2021 -0400

    Merge branch 'main' into dikstra

    merging for for loops

commit 4a5d64f891307c1f6f838c0fb5b34b44e0ac2eb5
Merge: f1db1ff efd6a08
Author: defne sonmez <dys2109@columbia.edu>
Date: Fri Apr 23 00:12:48 2021 -0400

    Merge branch 'main' into dikstra

    for testing purposes

commit b3d0dc47d723f2ffc295e27d3c92c16b0c75c858
Merge: efd6a08 091f88b
Author: E L <eilamemail@gmail.com>
Date: Fri Apr 23 00:01:22 2021 -0400

```
    Merge branch 'ctypes' into main
    BEEG MERGE, full(?) graph functionality

commit 091f88b177788b6b668e8f121b613fb4aa9eecdd
Author: E L <eilamemail@gmail.com>
Date:   Thu Apr 22 23:57:05 2021 -0400

    Fixed bug with repeated for loops

commit 6b8482fce288be075a30bd88cb5f9eb97705c1fe
Author: E L <eilamemail@gmail.com>
Date:   Thu Apr 22 23:36:25 2021 -0400

    Loops tested and working

commit f6a1ef8e1b748cc0b8473749c64edb075de92ebe
Author: E L <eilamemail@gmail.com>
Date:   Thu Apr 22 23:18:03 2021 -0400

    Node for loop seemingly working

commit f1db1ff29ed47000a7e1b23eb6d7687cc76cd0f2
Author: defne sonmez <dys2109@columbia.edu>
Date:   Thu Apr 22 22:54:00 2021 -0400

    compiles but segfaults when running

commit 342d8975864d3115ea9ac37060591243def91073
Author: defne sonmez <dys2109@columbia.edu>
Date:   Thu Apr 22 20:07:02 2021 -0400

    modified dijkstra but still getting error

commit b1a6ccbe5871e6a0fde6a08b86f836585c5efa90
Author: E L <eilamemail@gmail.com>
Date:   Thu Apr 22 03:11:38 2021 -0400

    Semant check for loops

commit f12e8950676634691df0092616d1f55da6f2b39e
Author: E L <eilamemail@gmail.com>
Date:   Thu Apr 22 01:57:45 2021 -0400

    Fixed error handling on table functions

commit e02ac5872ea3198ec7c9b82e42ee46d05b9557b5
Author: defne sonmez <dys2109@columbia.edu>
Date:   Wed Apr 21 23:18:05 2021 -0400

    discovered and solved new bug, didn't have current=source checking

commit 45476170ee61a5960996c3eaedf9ad479a474a2e
Author: defne sonmez <dys2109@columbia.edu>
Date:   Wed Apr 21 22:49:33 2021 -0400

    edge case errors fixed
```

commit f906aabbe23a6c3c6f2794e4104eb16efbf83184
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 21 20:25:45 2021 -0400

    c dijkstra is working except for edge cases

commit b1b442cf727f4916d8386dbb5822e78bd12b2fe6
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 21 19:58:19 2021 -0400

    working on it

commit 28daec2161a94242de83c5c0f22870ce5155ab3b
Author: pelincetin <pc2807@barnard.edu>
Date: Wed Apr 21 18:20:01 2021 -0400

    Dockerfile

commit efd6a08d91bc1d1098fbfa0c01b9bcfc8b0fd709
Author: pelincetin <pc2807@barnard.edu>
Date: Wed Apr 21 18:19:21 2021 -0400

    Dockefile

commit dc0f5f45b4129d0d0b2df882978ed2d4ac36cfd9
Author: pelincetin <pc2807@barnard.edu>
Date: Wed Apr 21 18:18:07 2021 -0400

    Dockerfile

commit 3c4fa4f6d62020a197b20ae1fe2d0eb63135c324
Merge: b8fbb53 22a4069
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 21 14:43:32 2021 -0400

    Merge branch 'ctypes' of https://github.com/pelincetin/GRACL into dikstra

commit b8fbb53431b4cd13869d46ff4c8c87c45d40b128
Merge: fca0f8d 8cf472e
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 21 14:42:47 2021 -0400

    Merge branch 'main' of https://github.com/pelincetin/GRACL into dikstra

commit 22a4069e672cbacfc65208110988380f165d7729
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 21 14:40:29 2021 -0400

    updated codegen types to be accurate

commit f861c940d6bb123a32ecf67d950b409d519c67e8
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 21 13:57:42 2021 -0400

    node and edge tests pass with stringEquals update

commit 6ba5e05b8d89e2259685b6c1accd49e01f4f74f6
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 21 13:44:12 2021 -0400

    graph/it/dt grc tests pass with size 0 & -1 checks

commit 340d8b64ebfc8910424a5295e981cf4af49d939d
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 21 13:37:58 2021 -0400

    graph/dt/it handle errors for sizes less than 1

commit ba275a346961cb316ac5a7dd8d1f99863db84373
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 20 10:44:32 2021 -0400

    Fixed semant check on ==/!=

commit 45859321ff21a0a2bda8313a069abde8a504c05d
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Mon Apr 19 13:55:26 2021 -0400

    nodefull passes GRC test suite

commit 3800bcfbe13c14b5c8dd46219e9b3310f13a7696
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Mon Apr 19 13:55:13 2021 -0400

    IT and DT tests pass c test suite

commit 86d6030cf5e5fc11807ffbe51b6dea8785a45fd6
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Mon Apr 19 01:27:20 2021 -0400

    all grc type tests working except for node

commit 63b6dd5998bfecb884683d1e2e7352cbe88001ee
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Mon Apr 19 00:22:35 2021 -0400

    edge passing grc test

commit c39e6e253530abed6474a68de728e6cdff73c4f5
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Mon Apr 19 00:11:51 2021 -0400

    IT and DT tests working in grc

commit 5308218eaee61cfba4ee625846b64e92b24ff5b5
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 18 23:58:13 2021 -0400

    Access/Insert working, needs edge case testing

commit b3ba466e3fd45a230f00d7357f22c3ce566957e4

Author: E L <eilamemail@gmail.com>
Date: Sun Apr 18 23:14:57 2021 -0400

    yo

commit cece9afe465e0b8ef142874ab95f92117305ddad
Merge: 79a724b 8cf472e
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 18 23:14:17 2021 -0400

    Merge main in

commit 79a724b45da7eaf1a453aa9a1ea4f8407d3499e1
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 18 23:12:25 2021 -0400

    Add test for table syntax

commit c05217ddb28e414217211f43adb9b05f939de97c
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Apr 18 23:04:08 2021 -0400

    testing graph, dt, it in grc

commit 809af6a19dfff6d64e45b55273b202d325a8689a
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Apr 18 22:04:22 2021 -0400

    added IT and DT functions to functions.ml

commit f8a43e3f14ba0a399395c8a8d50a49239c496d2d
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Apr 18 21:58:02 2021 -0400

    changed node name to n for consistency

commit 83e4da8b16eb29515a3cb4c1511126177eeddb23
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Apr 18 21:50:48 2021 -0400

    IT fully tested in C

commit e19db7efde55e008c4fe2e3003d77f2f4ed2c920
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Apr 18 21:38:50 2021 -0400

    DT fully tested in C

commit f352f76582d3d1b3eb3decf4f2116130fcbcba16
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 18 18:57:35 2021 -0400

    fix testing for concurrency

commit 380ccc5d77a6d6aa095cc3cdf7037f8052baf7d9
Author: E L <eilamemail@gmail.com>

Date: Sun Apr 18 18:52:07 2021 -0400

    Synch working in GRACL

commit 23a1704d215cbac5ff0ae1bfa8777b558070307b
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 18 17:17:59 2021 -0400

    start to get synch working on the compiler

commit 219ecf726bd146aead382da6553e5089911910a0
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 18 17:08:05 2021 -0400

    new conc test

commit a66f20a1387226b38ce1bf61ff896c22e29a5810
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 18 16:59:39 2021 -0400

    ctypes: Implement graph id logic

commit 360d0032a0ad810169203bc6c10d077d96b48bfa
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 18 16:00:14 2021 -0400

    ctypes: Fix linking

commit c9edcd47306fc785290533d7434e88a27ea492e9
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 18 15:45:40 2021 -0400

    testing for sync

commit 85cf0da355b6fa1671398d77205f640cd50cc8ac
Merge: 0c31f2c 2faac17
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 18 15:40:49 2021 -0400

    Merge branch 'ctypes' of https://github.com/pelincetin/GRACL into ctypes

commit 0c31f2cdabca4d3a8f42c9ec98d58a3bf9b0dfd3
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 18 15:40:35 2021 -0400

    ctypes: Add .h's to avoid code bloat

commit e6aae06723b042d434f83df000b2e2487aa697e0
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 18 15:08:09 2021 -0400

    move the concurrency file into cfiles, add the dfs code written w edwards

commit 5709ff5f7969f09288b671fd221adb2d7431f3ce
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 18 15:03:20 2021 -0400

    concurrency is BACK

commit e13a2c61bc3b8d0cadd254fad0298ba161987860
Merge: 22cd3ff 8cf472e
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 18 14:50:12 2021 -0400

    Merge branch 'main' into conc

commit 2faac17e5deffe33fd4b53d31369077586743889
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Apr 18 14:18:36 2021 -0400

    comments

commit 8cf472eac98924976d55532cc5d52f026618de03
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 18 13:16:38 2021 -0400

    Test heap functionality

commit 6b892a591cb20f96c4ce1b4dc7dec3a74e664747
Author: E L <eilamemail@gmail.com>
Date: Sat Apr 17 02:49:19 2021 -0400

    Finished scoping, added more tests

commit 9c82954552d410c7c91376b09c8211c4820889a6
Author: Maya <mv2731@columbia.edu>
Date: Wed Apr 14 21:00:41 2021 -0400

    ctypes: Fixed infinite loop in delete

commit 4404ca5f4df609229b5dfcae33c038f72a8ef8f7
Author: Maya <mv2731@columbia.edu>
Date: Wed Apr 14 20:33:22 2021 -0400

    ctypes: Fix naming standards

commit df0a34cd7b71e13ccce4be3233ec120a5da1dbbd
Author: E L <eilamemail@gmail.com>
Date: Wed Apr 14 14:42:43 2021 -0400

    Fix blocks not generating new semant block

commit 73ddf306469581126209b4a0b8cb057990f2d4e9
Author: E L <eilamemail@gmail.com>
Date: Wed Apr 14 01:22:27 2021 -0400

    yo

commit d966c7ac5a9ce34021a628c823a781c035ca0314
Author: E L <eilamemail@gmail.com>
Date: Wed Apr 14 00:56:37 2021 -0400

Block scope seems to work

commit 90875884cbe7f7af117b503a69265da7efdc875d
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 14 00:49:17 2021 -0400

added inttable test (close to doubletable)

commit 13a7291f826baa3165fac220b1f5f5e2c2b6184c
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 14 00:45:54 2021 -0400

add more deletes to dt test

commit 0706425ffd62ed13165f90348415fabd007b1a9b
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 14 00:44:54 2021 -0400

cheged dt variable names to it for inttable

commit f42bdb8d6272e739c58814ce23655c7db550d465
Author: Maya <mv2731@columbia.edu>
Date: Wed Apr 14 00:36:06 2021 -0400

ctypes: a bit to fix on double/int table

commit cafd6f57440b2a00b47edd60f19064ef455090be
Merge: 2414853 5796327
Author: Maya <mv2731@columbia.edu>
Date: Wed Apr 14 00:31:53 2021 -0400

Merge branch 'ctypes' of https://github.com/pelincetin/GRACL into ctypes

commit 24148539a4f92bc025b3984539f970f932808974
Author: Maya <mv2731@columbia.edu>
Date: Wed Apr 14 00:31:47 2021 -0400

ctypes: Fix inttable

commit 5796327f2ef657d1837a0bc516eb0944089386e0
Merge: d171ba4 a6834ae
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 14 00:26:57 2021 -0400

merge remote ctypes into local ctypes

commit d171ba4349a666df31424017ee075d33faa97ad6
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 14 00:25:02 2021 -0400

doubletable test written

commit a6834ae65a2ddcc8082164ff7014515a0f02319d
Author: Maya <mv2731@columbia.edu>
Date: Wed Apr 14 00:23:55 2021 -0400

    ctypes: Fix doubletable?

commit 3d5a1cefced0eadd4b23130d4888abfd54fa6ad7
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 23:54:42 2021 -0400

    basic double table test

commit f397339aab15863b33e7335a9be63ba78d468a3d
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 13 23:43:24 2021 -0400

    Keeping tests working

commit 1c401fe86ee47a0cf0486db78db51242fbbbe6e7
Merge: ff67907 e107b98
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 23:28:27 2021 -0400

    Merge branch 'main' into ctypes

commit ff679072069cf81c1fda716abee76e7fd8d23dca
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 23:17:05 2021 -0400

    fix c type tests

commit 35337b9f502b6b3ad849ce39c31dcc03bd62451c
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 13 23:08:20 2021 -0400

    Got semantic checking for block scoping

commit 129a9ea6cf558360ddf40707c5e7ba18e89cfebe
Merge: 1440bff 97cf636
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 22:55:43 2021 -0400

    ctypes merge with doubletable stuff

commit 1440bff73eb7e9ae93242fa31459af33b5c16b73
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 22:52:43 2021 -0400

    change set_prec to setPrec

commit 97cf6365bcc87f03cae22b361cdf8e083d48dd14
Author: Maya <mv2731@columbia.edu>
Date: Tue Apr 13 22:48:22 2021 -0400

    fix var name

commit 2c61bcf3568f8ee83430b09f65cad64f3ecc2348
Author: Maya <mv2731@columbia.edu>
Date: Tue Apr 13 22:41:30 2021 -0400

    ctypes: Add new inttable data structures to locked object

commit 53babce8dc4c74b7d800f517d59021bb4690a8c8
Author: Maya <mv2731@columbia.edu>
Date: Tue Apr 13 22:05:33 2021 -0400

    ctypes: working on implementing collision doubletable

commit 4343618d88b62f63c974a1b3f1ee5d989d9d68bd
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 14:46:23 2021 -0400

    add node functions

commit a8aa957b6a01679a784f8cd026c032e45a6a6bb5
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 14:36:01 2021 -0400

    getEdge added and tested in c

commit 661d325d64496e9c82dae975114a4a2978eacecb
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 13:01:56 2021 -0400

    all tests of edge in c are passing

commit 70ca2f7de94b0b9d9c16a6473c22a3226e229409
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 13:01:46 2021 -0400

    remove unneccessary line from edge

commit f3adb16acfb3aa5d4e0ca36553571fab0240290a
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Apr 13 13:01:34 2021 -0400

    remove comment from node-test

commit e107b98148bc4b0c188d0c0fd48bebfc3fba4472
Merge: 4ad6a18 05fc964
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 13 01:36:09 2021 -0400

    Merge branch 'main' of https://github.com/pelincetin/GRACL into main

commit 05fc964ba13589dc9396c0fe5314d1c878e02699
Author: pelincetin <pc2807@barnard.edu>
Date: Tue Apr 13 01:35:43 2021 -0400

    update the makefile

commit 4ad6a18f985a4e65b549f9c82aa75b41234935d4
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 13 01:34:19 2021 -0400

    List tests working

commit 8545df78461473ae2e9fb67d0e7491a932a237e3
Merge: db42069 879200c
Author: pelincetin <pc2807@barnard.edu>
Date: Tue Apr 13 01:32:28 2021 -0400

    merge

commit 62b531bd83da9a38b72badcfa0a262fcf4fb3706
Author: pelincetin <pc2807@barnard.edu>
Date: Tue Apr 13 01:22:17 2021 -0400

    fix script

commit 95094bf1aed24d68853dbc1ab3fe3c09a5960303
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 13 01:18:22 2021 -0400

    deleting

commit 5d7e637aad0b36ff2c13028dd45fb5dc81745c72
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 13 01:16:48 2021 -0400

    yo

commit d996e60b691695f5920eb75b8525e83a25978df6
Merge: db42069 1ca9896
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 13 01:15:04 2021 -0400

    Updated semant, removed concurrency

commit 1ca9896ae0f5a97720fc06b93846d1b781f0eeda
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 13 00:32:28 2021 -0400

    Removed concurrency

commit db420698821c50dae1d3084327b0f3f11fcdafe3
Merge: 4690271 3e392dd
Author: Maya Venkatraman <mv2731@columbia.edu>
Date: Tue Apr 13 00:31:16 2021 -0400

    Merge pull request #8 from pelincetin/collision-handle

    Collision handle

commit 3e392dde7971257cb1a1ca891291616e2dab7efa
Author: Maya <mv2731@columbia.edu>
Date: Tue Apr 13 00:25:00 2021 -0400

    collision-handle: add graph id

commit 1e6b58b7a4e064d68d52b768e4870154074aea39
Author: Maya <mv2731@columbia.edu>

Date: Tue Apr 13 00:15:08 2021 -0400

    collision-handle: Fix arg order

commit 4f229ff3b58e6b8fcaf250eac714d7b8b8b6abe1
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 12 23:23:28 2021 -0400

    collision-handle: graph tested up

commit f86b4847b1a0bc8609c7ba615036e8c541c6eb7c
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 12 23:14:13 2021 -0400

    collision-handle: working on graph-test

commit bf15a466a98d799e78a8f03305d952f2491c7935
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 12 22:14:52 2021 -0400

    collision-handle: Add id to node type

commit 0d85efc8c46c9aee90a48188a531ee4508b4d34a
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 12 21:58:34 2021 -0400

    collision-handle: edgelist fully tested

commit 31faa5c4cd3efc8b96fc518b224c696305828055
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 12 21:36:45 2021 -0400

    collision-handle: Nodelist all tested

commit 8a46ad2949c2758d397b6c71b486f94ad0be09d2
Merge: 7ae9bcc 3ff593d
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 12 20:03:07 2021 -0400

    Merge branch 'collision-handle' of https://github.com/pelincetin/GRACL into collision-handle

commit 7ae9bcc76fff1effa1315d024949e47c4df884a0
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 12 20:02:59 2021 -0400

    collision-handle: nodetest segfault fixed ty pelin lol

commit 53d0b34fff73aaf9c0e059350909609d1fa7d091
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 12 17:21:32 2021 -0400

    segfaulting node

commit 3ff593dd439e187463859dc211c2b77a2f2c5717
Author: defne sonmez <dys2109@columbia.edu>
Date: Mon Apr 12 17:15:30 2021 -0400

changed same segfault in here

commit c788507c35c9cbb08f9f7e62f9a56cc221f87ce1
Author: defne sonmez <dys2109@columbia.edu>
Date: Mon Apr 12 17:12:59 2021 -0400

fixed segfault and tested for remove first in edgelist

commit 59c5a652f8acb5e9930264e4a01bce7467fc6ff1
Author: defne sonmez <dys2109@columbia.edu>
Date: Mon Apr 12 14:55:33 2021 -0400

tested removal

commit 81d7f26d44c4efd52b7758018b3d527f9004f4dd
Merge: fb225f8 906b4e7
Author: E L <eilamemail@gmail.com>
Date: Mon Apr 12 00:46:50 2021 -0400

merge

commit 906b4e7a30569376f0a1e95f93c193881594fcbc
Author: E L <eilamemail@gmail.com>
Date: Mon Apr 12 00:42:49 2021 -0400

It ends...?

commit 9f518319cc31c4c9476a6ccb28d0886e6adaedcf
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 11 23:29:24 2021 -0400

It continues...passes all non-graph related tests

commit 1396dbdcf4385f069c06e9598e8f221c90168cfa
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 11 22:54:53 2021 -0400

It continues...most tests pass

commit fb225f8e693b4ce123b65bc36143e095c389680e
Author: E L <eilamemail@gmail.com>
Date: Sun Apr 11 00:44:51 2021 -0400

Revert "Parser change, should fix initialization order"
Rolling back to allow for proper fix
This reverts commit dc3b24626b69e77a3d2d557fd525c8aa7d02c50e.

commit dc3b24626b69e77a3d2d557fd525c8aa7d02c50e
Author: E L <eilamemail@gmail.com>
Date: Thu Apr 8 23:10:21 2021 -0400

Parser change, should fix initialization order

commit 1cc1eaa4210e11937fc6e69e27d912245572dc4f
Author: E L <eilamemail@gmail.com>

Date: Thu Apr 8 22:51:18 2021 -0400

    It begins

commit 879200cce01446af920b6e483dccd4c5bc00e323
Author: pelincetin <pc2807@barnard.edu>
Date: Thu Apr 8 19:58:55 2021 -0400

    codegen changed

commit 4a27b0f98c69bf44b1f6d0d5745986ade52d8f31
Author: pelincetin <pc2807@barnard.edu>
Date: Thu Apr 8 19:28:43 2021 -0400

    initialize tests

commit 80b8511d17eaef65f28c62511175cec630b7e367
Author: pelincetin <pc2807@barnard.edu>
Date: Thu Apr 8 19:22:16 2021 -0400

    all functions compile

commit fca0f8d0ed94c1268e98cbc2a685ec50c0e9d3bc
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Thu Apr 8 19:21:59 2021 -0400

    dijkstra written but untested

commit c7be40ee23f4e64754dd3e2c8c6b7dbf52ae1791
Author: Maya <mv2731@columbia.edu>
Date: Thu Apr 8 19:15:13 2021 -0400

    collision-handle: Remove common and organize into type-specific files, fix linking

commit e923348a40d70984fe75dfcff1b04cfca7c12772
Author: pelincetin <pc2807@barnard.edu>
Date: Thu Apr 8 18:41:42 2021 -0400

    more progress

commit 60674c275a27f4169c48d65ec4aecebe572e1b6c
Author: pelincetin <pc2807@barnard.edu>
Date: Thu Apr 8 18:27:39 2021 -0400

    progress

commit 03d0d90fcbb543183cce15767278b1830674b798
Author: pelincetin <pc2807@barnard.edu>
Date: Thu Apr 8 18:16:22 2021 -0400

    search is done

commit 164787f9f158a4f1a67539e34c644fba79e1b19c
Author: pelincetin <pc2807@barnard.edu>
Date: Thu Apr 8 18:02:31 2021 -0400

initialize

commit 04625148819572c04eb079b17e697127cf48c44e
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Thu Apr 8 17:20:39 2021 -0400

revised appendEdge; fixed test outputs

commit 587256cb5fd9fa906aeac57d8397833b6c844c5f
Author: Maya <mv2731@columbia.edu>
Date: Thu Apr 8 17:14:17 2021 -0400

collision-handle: Spell precursor correctly

commit 76b054db811e3e3ce84fa91fe947a3bbf641eda0
Author: Maya <mv2731@columbia.edu>
Date: Thu Apr 8 17:13:35 2021 -0400

collision-handle: Add prec and cost to node

commit a27ecd665af55ffc331b1397889be01dd3d1f334
Author: Maya <mv2731@columbia.edu>
Date: Thu Apr 8 16:59:47 2021 -0400

collision-handle: Clean up common functions

commit 7383269f9a59759daf3d5a27efd85750ca926448
Author: Maya <mv2731@columbia.edu>
Date: Thu Apr 8 16:30:10 2021 -0400

collision-handle: Redesigned the graph underlying hash to have Edgelist value rather than NL, wrote removeNodeGraph

commit 019aab0d7751d577d0a34522ffb3f4bce2da4de7
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Thu Apr 8 15:09:03 2021 -0400

EL and NL grc test files (not yet working)

commit 270707fee670ff8c6e39abb86dc934ba0a99b92f
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Thu Apr 8 15:08:03 2021 -0400

shortened head and tail functions

commit a14262c3af2d0470baf0ebec2ad504580a6ffcf2
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Thu Apr 8 14:10:09 2021 -0400

Deleted duplicate definitions createNL/createNL

commit 5873b5e112a51944c23c76f8a3df3aeda62f387e
Merge: d591395 4690271
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Thu Apr 8 14:00:02 2021 -0400

Merge branch 'main' of https://github.com/pelincetin/GRACL into graph-algos

commit d59139582e623da55a89b5d103752fb3904dbde9
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Thu Apr 8 13:56:46 2021 -0400

Added C tests for EL and NL tail

commit 4690271d6bc3dbdd51f0bb98f21ce606464f06a6
Author: pelincetin <pc2807@barnard.edu>
Date: Thu Apr 8 13:23:51 2021 -0400

merge

commit 65cac6c065e6e97fca8aeb83ea41837ea1a8555c
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Thu Apr 8 13:02:32 2021 -0400

Changed EL remove functions to edge

commit 03325482af195f0a42bf3d0643247dd862c0508c
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Thu Apr 8 12:58:12 2021 -0400

Changed functions to have _EL and _NL at the end

commit 7ccf94ccde5905353de9f9f3d7632f93053ecc11
Author: pelincetin <pc2807@barnard.edu>
Date: Wed Apr 7 21:37:27 2021 -0400

more merge conflicts

commit 45cacccb3acd13f41aae7100c7626db2bc45807
Merge: c5f8de7 daf1637
Author: pelincetin <pc2807@barnard.edu>
Date: Wed Apr 7 21:31:33 2021 -0400

merge conflict

commit c5f8de7fa2f7275996ee0c15aef0e45313521722
Author: pelincetin <pc2807@barnard.edu>
Date: Wed Apr 7 21:24:53 2021 -0400

remove node implemented

commit f43cd557a1cbff0862612d7d3686c18b25500b2f
Merge: dd3b441 468b861
Author: pelincetin <pc2807@barnard.edu>
Date: Wed Apr 7 20:31:08 2021 -0400

merge

commit dd3b441c9d118800fe2e19528e69b45f435ab555
Author: pelincetin <pc2807@barnard.edu>
Date: Wed Apr 7 20:27:57 2021 -0400

initialize removeNode

commit d753db7779d339311341c69bd8e6dbdf843dd199
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 7 13:57:33 2021 -0400

delete accidental test script output

commit a053640e884024e10ef0d6b106db71c4b551634a
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 7 13:54:04 2021 -0400

Fixed nl el in functions

commit fd547a7f849620913cd5e8ee7e41079d83d5e4c1
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 7 13:43:02 2021 -0400

added head/tail to functions.ml

commit c24a943145fd2ea4145250b77f39d68ada4fdb27
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Apr 7 13:31:46 2021 -0400

added head+tail funcs to lists in c, not codegen

commit daf1637fd09749feb3eb63b77e7c8e2e3107fda5
Author: E L <eilamemail@gmail.com>
Date: Wed Apr 7 00:20:41 2021 -0400

Fixed node neighbors

commit fe9e9cc50649aacb1c0c21d717f974bb94338819
Author: defne sonmez <dys2109@columbia.edu>
Date: Tue Apr 6 23:23:49 2021 -0400

fixed createnode and addedge, also created print-functions.c and replaced print statements in test files

commit f71330d993613d2c9d6f1e3946fc9118b1989b07
Author: defne sonmez <dys2109@columbia.edu>
Date: Tue Apr 6 23:11:07 2021 -0400

moved createnodelist, neighbors to common functions

commit 468b86150ae647e9d4b959046db52eedd82feed4
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 6 18:38:54 2021 -0400

Added note for testing another node function

commit fdf9bbfade35309c96147528300bf1b4dd0d37eb
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 6 18:14:47 2021 -0400

Cleaned up some warnings

commit 0ad0b23e5783a3a48789d3118489052270596bcf
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 6 17:32:32 2021 -0400

    Finished casting/string functions

commit 6d6d847e81af970f8fe73e63a624e17fcd51903e
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 6 16:37:12 2021 -0400

    Add c functions to functions.ml

commit 6ebb774cb309909667207a53f98d30643461223e
Merge: 6984e15 fc003f6
Author: LetsGitStartedAlready <47231353+LetsGitStartedAlready@users.noreply.github.com>
Date: Tue Apr 6 16:07:25 2021 -0400

    Merge pull request #6 from pelincetin/stdlib

    Add standard library file for linking in C code efficiently

commit fc003f6a7d9fdf0123363715840cb300b635e962
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 6 01:37:19 2021 -0400

    Linking solved

commit 40ef5d1bf4bcbb21fd52904fd7e781706d6d79e1
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 6 01:27:14 2021 -0400

    Testing new stdlib configuration

commit dd172700aae827fa37f46655276e56c8a65c4da6
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 6 00:52:26 2021 -0400

    Add rest of graph functions to functions.ml

commit 26a5a1e29994f6f301d50d40c3cf5dd3bac9f7dc
Author: E L <eilamemail@gmail.com>
Date: Tue Apr 6 00:36:16 2021 -0400

    change to graclstdlib

commit 22cd3ff553f89149c2c3084e6a4878ef6ec9e7c7
Merge: d81f466 6984e15
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 5 18:54:56 2021 -0400

    merge conflict

commit 6984e153355c438c6fda9a17b2a6059a916c2e11
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 5 16:57:25 2021 -0400

    get all the types in ocaml

commit b796e11b515f0bb9275dd637e63c4fa12dd723e7
Merge: 169595b 32db54d
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 5 16:31:23 2021 -0400

    yo

commit 32db54d3a2a90dd06235082f21c2c6bbf8cf0c28
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 5 14:32:37 2021 -0400

    don't want these files

commit f28483d4b27c89c593e0e0095a9f6db733c8f33d
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 5 14:25:05 2021 -0400

    test files compile and run

commit e4b42f831eda072212b9ef227785b587f02f9b76
Author: pelincetin <pc2807@barnard.edu>
Date: Mon Apr 5 14:08:52 2021 -0400

    compilation errors fixed, one header file

commit c4924991409fa0fd564d988c6e98408abe057c04
Author: Maya <mv2731@columbia.edu>
Date: Mon Apr 5 02:10:08 2021 -0400

    ctypes: 2 more funcitons to impelment in graph.c

commit 555db0bd72da286d775566e21602309bf1363d4c
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 4 21:50:24 2021 -0400

    ctypes: Make common functions.h

commit 2ce1f8cff9674c8eb94d2456d38081d8adf178cc
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 4 21:46:19 2021 -0400

    ctypes: Remove malloc from commonFunctions.h

commit 4bdf156465f2aae799a47c47bf1eafa509e5ab91
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 4 21:42:02 2021 -0400

    ctypes: Adjusted spacing in nodelist and modified edgelist removeEdge

commit 685ad52bef8dec0e4055b44aea0fbf0dbf3bc5cf
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 4 21:38:02 2021 -0400

    ctypes: Adjust appendEdge

commit f718bd131191764b50acc49c0df8f6d96e068348
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 4 21:32:44 2021 -0400

    ctypes: Fix removeNode nodelist.c

commit 32e96375e788abaf285a64465c84a3f60bcecec d
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 4 21:19:04 2021 -0400

    ctypes: Modifying edgelist function for error in remove

commit 0b383998e2c0ddb1847e4397c1db659b380765e7
Author: Maya <mv2731@columbia.edu>
Date: Sun Apr 4 21:04:32 2021 -0400

    ctypes: Organized into .c and .h

commit faeca50657a0c031f39be63acd3ee5b98ca2006a
Author: pelincetin <pc2807@barnard.edu>
Date: Sat Apr 3 19:12:57 2021 -0400

    comments

commit c6c882eba5f45fbb5ea671473e14978be292c01f
Author: pelincetin <pc2807@barnard.edu>
Date: Sat Apr 3 18:42:56 2021 -0400

    initial graph

commit 169595bf5fdd1fca34eeb432e45f2c8149ad1ffd
Merge: 1fdb3ac 0525d35
Author: LetsGitStartedAlready <47231353+LetsGitStartedAlready@users.noreply.github.com>
Date: Sat Apr 3 15:37:36 2021 -0400

    Merge pull request #5 from pelincetin/node-test

    Add node into GRACL + ease of adding future functions

commit 31b6074efd12f48dae130bbd90d80a791d9f95d4
Author: E L <eilamemail@gmail.com>
Date: Sat Apr 3 15:23:04 2021 -0400

    Malloc understood

commit 0525d35d3c3848a5dcf263d2d9c957f0afb37876
Author: E L <eilamemail@gmail.com>
Date: Sat Apr 3 15:02:55 2021 -0400

    QOL change to functions file

commit 22f44824286354c78ab7d9193a95eefbed5c711f
Author: pelincetin <pc2807@barnard.edu>
Date: Sat Apr 3 15:02:27 2021 -0400

lol2

commit 3515b4a9a4d8d9f31322a87d0075bc05317e5fce
Author: pelincetin <pc2807@barnard.edu>
Date: Sat Apr 3 15:00:34 2021 -0400

lol

commit 0793b1ca909db5806beaff35f4d497f2e0aa7c89
Author: E L <eilamemail@gmail.com>
Date: Sat Apr 3 01:09:20 2021 -0400

Streamlined builtin adding process

commit 25f6c1e4ef3094b3dc3351ff839d110cc99059a9
Author: E L <eilamemail@gmail.com>
Date: Fri Apr 2 23:54:07 2021 -0400

Node implemented in GRACL

commit 3071b47b2e52710688518e5b5e69a147b4ce7a3b
Author: E L <eilamemail@gmail.com>
Date: Wed Mar 31 02:20:10 2021 -0400

Got node basics working

commit 2dbccfc81dffed8059ce363b56432021d892a847
Author: pelincetin <pc2807@barnard.edu>
Date: Wed Mar 31 01:20:33 2021 -0400

work on node, almost got it working. for now pretend like createnode is not called on graphs

commit a497c647c0d2de41bd49442780b80b73aff70cf6
Merge: 1fdb3ac 693bb8f
Author: pelincetin <pc2807@barnard.edu>
Date: Tue Mar 30 23:28:24 2021 -0400

Merge branch 'ctypes' into node-test

commit 1fdb3ac520666e7331dda78d9c45ec21adc96ba9
Merge: 3ab0b3c 556a0c8
Author: LetsGitStartedAlready <47231353+LetsGitStartedAlready@users.noreply.github.com>
Date: Tue Mar 30 23:16:56 2021 -0400

Merge pull request #4 from pelincetin/methods

Parse method calls as function calls

commit 556a0c8fd5b163873246ed5ad203112a6787c67d
Author: E L <eilamemail@gmail.com>
Date: Tue Mar 30 23:16:10 2021 -0400

Add methods as syntactic sugar

commit 3ab0b3c7431b8fc11c78a8496954f13d4259138b
Merge: 0ee3579 35adf2b

Author: LetsGitStartedAlready <47231353+LetsGitStartedAlready@users.noreply.github.com>
Date: Tue Mar 30 23:08:53 2021 -0400

    Merge pull request #3 from pelincetin/infinity

    Adding Infinity

commit 693bb8f4bb541832bc1b1c85755cf15094fa8e63
Author: pelincetin <pc2807@barnard.edu>
Date: Tue Mar 30 14:51:07 2021 -0400

    test infrastructure initialized

commit 219155cc71e097b7b84095f9f8a82bab6b65f225
Author: pelincetin <pc2807@barnard.edu>
Date: Tue Mar 30 11:57:58 2021 -0400

    test directory, change makefiles, remove locks from edgelist and nodelist

commit 0ee35793caf3996d27d5a6d28b9721c09ca9f743
Author: Pelin Cetin <35610616+pelincetin@users.noreply.github.com>
Date: Mon Mar 29 21:47:39 2021 -0400

    Update README.md

commit 25576e74bbac88e2c41ac91bfaf50d2365561a25
Author: defne sonmez <dys2109@columbia.edu>
Date: Sun Mar 28 20:27:18 2021 -0400

    added and tested all functionalities for nodelist, should continue testing more edge cases

commit bcf1aa2b4750eb3d6945919fb41ec631c53cc117
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Mar 28 18:33:23 2021 -0400

    Edgelist working and tested; updated makefile

commit 1e5c133df1bda368931ea44542f94d0f5f17c7c1
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Mar 28 17:54:54 2021 -0400

    Working on testing edgelist

commit 984e2743b57b3f685f70062c33c3e839a4273620
Author: Maya <mv2731@columbia.edu>
Date: Sun Mar 28 15:22:06 2021 -0400

    ctypes: Delete comment

commit 5c95f92958586100fc87da159b9e3829b50cdc23
Merge: e134f79 de62057
Author: Maya <mv2731@columbia.edu>
Date: Sun Mar 28 15:21:21 2021 -0400

    Merge branch 'ctypes' of https://github.com/pelincetin/GRACL into ctypes

commit e134f7964cdce182815109620824f26b09a4d5ef
Author: Maya <mv2731@columbia.edu>
Date: Sun Mar 28 15:21:12 2021 -0400

    ctypes: remove fully stubbed out

commit de620571f2ec9308b25d295ee3e35250c14967a6
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Mar 28 15:02:56 2021 -0400

    Added comment

commit be0436aa20830cdb03eb4fad1d75cd733d699c74
Merge: 2b8ccab ca1a8b9
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Mar 28 14:35:40 2021 -0400

    Merge branch 'ctypes' of https://github.com/pelincetin/GRACL into ctypes

commit 2b8ccab7e0b552b2221e9a2680131b9ef6f70b5b
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sun Mar 28 14:34:52 2021 -0400

    Edge written and tested

commit ca1a8b92cb155b600949f00d3acff67345b5b223
Author: Maya <mv2731@columbia.edu>
Date: Sun Mar 28 13:58:52 2021 -0400

    ctypes: Need to talk about edge equality test

commit 32cc5080cb5a9653f17d34a4fcca94d7f0f52e4f
Author: Maya <mv2731@columbia.edu>
Date: Sun Mar 28 13:54:05 2021 -0400

    ctypes: Add prepend function, only remove e remains

commit f82a1b095d7e1edcf4d5ce3aa52c626b4c46c231
Author: Maya <mv2731@columbia.edu>
Date: Sun Mar 28 13:42:10 2021 -0400

    ctypes: Finish append

commit 1079e31eceec823aeccc1269d2c5f1b79578263e
Author: E L <eilamemail@gmail.com>
Date: Sun Mar 28 03:49:33 2021 -0400

    removed method from SAST

commit ae4b710bd337ac1aff82c5465ce2299f0b961daf
Author: Maya <mv2731@columbia.edu>
Date: Sun Mar 28 02:26:16 2021 -0400

    ctypes: Only 3 methods left to implement in edgelist

commit eb05ccdcda7a369f339fece1c22f1da16b3657c9

Author: Maya <mv2731@columbia.edu>
Date: Sun Mar 28 01:59:43 2021 -0400

    ctypes: Progress on edgelist functions

commit a28962c1be7bc3a6ba6e0929d947af5d5a084eac
Author: Maya <mv2731@columbia.edu>
Date: Sun Mar 28 01:56:38 2021 -0400

    ctypes: Work on edgelist

commit f8c47ee955a22af9dccc47a5d2747bf8f2d165f8
Author: E L <eilamemail@gmail.com>
Date: Sat Mar 27 19:56:19 2021 -0400

    Removed method from AST, all calls treated the same

commit a655276431519f66a3e8009ffd51701ffd201ec7
Author: pelincetin <pc2807@barnard.edu>
Date: Fri Mar 26 14:48:58 2021 -0400

    testing for node works, todo: separate the tests, also fixed the types in edge

commit e7cbf895ef6c75d44fda1628ba50f0be56487dad
Author: pelincetin <pc2807@barnard.edu>
Date: Fri Mar 26 14:19:10 2021 -0400

    fixed compilation errors, created a Makefile

commit c5886693adfafc336b8219280e23ef2011d4b9ff
Author: Maya <mv2731@columbia.edu>
Date: Fri Mar 26 14:04:03 2021 -0400

    ctypes: Rethink edges in node

commit ae0afa92c2ec7825779095833898e4a634321d02
Author: Maya <mv2731@columbia.edu>
Date: Fri Mar 26 13:38:11 2021 -0400

    ctypes: Add edge.c functions

commit ea51deda128f4d22a51ec155ae884d5d02955436
Author: Maya <mv2731@columbia.edu>
Date: Fri Mar 26 13:30:10 2021 -0400

    ctypes: Stub out functions for nodes

commit 9d6ea1a259533d665d11df46cb0bcdf2487702d1
Author: pelincetin <pc2807@barnard.edu>
Date: Fri Mar 26 12:40:36 2021 -0400

    node intermediary step

commit ef7ed6b7e46f36e516c54adc10dad62663457395
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Fri Mar 26 12:27:18 2021 -0400

changes to lockedobject

commit 485bc444a0e7c76de91a56f3bc8a54e99739f1cf
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Fri Mar 26 11:52:38 2021 -0400

added node functions

commit 35adf2b34af44ed487cf8a4a10fb27890d819eed
Author: E L <eilamemail@gmail.com>
Date: Fri Mar 26 00:47:53 2021 -0400

Infinity tentatively working

commit 01ae4c09bc3645f4ac20e07f96bf2d0212040f15
Merge: e7a2d65 f98d84b
Author: LetsGitStartedAlready <47231353+LetsGitStartedAlready@users.noreply.github.com>
Date: Thu Mar 25 20:33:56 2021 -0400

Merge pull request #2 from pelincetin/neq

Restore full suite of microC operations

commit e7a2d65b69d8162001dcd93fcf63d691212ac9f1
Author: E L <eilamemail@gmail.com>
Date: Thu Mar 25 20:32:57 2021 -0400

Added double testing to decinit

commit f98d84bfd7407bb178ff8a4705be533f9932d3b6
Author: Maya <mv2731@columbia.edu>
Date: Wed Mar 24 21:24:43 2021 -0400

neq: Add test of global decs with comparators

commit ce666e88f751a8ad5e4b85a23bbeceaa07cfbeea
Author: Maya <mv2731@columbia.edu>
Date: Wed Mar 24 21:00:39 2021 -0400

neq: Add and run successful geq test

commit 4610edfec903cacaae8809db9451a284fd32a852
Merge: 0b975a3 a236b38
Author: Maya <mv2731@columbia.edu>
Date: Wed Mar 24 20:35:22 2021 -0400

Merge branch 'neq' of https://github.com/pelincetin/GRACL into neq

commit 0b975a394df7e8c4544e033bf27c7a51ca3a2374
Author: Maya <mv2731@columbia.edu>
Date: Wed Mar 24 20:35:00 2021 -0400

neq: Add test for geq

commit a236b3823714d5193f0d727fa9c4be96e01208d4

Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Mar 24 20:27:47 2021 -0400

    add geq

commit 05b054995c76984e68e349aa98608ff8eac638d3
Author: E L <eilamemail@gmail.com>
Date: Wed Mar 24 00:50:07 2021 -0400

    Fixed op3 test

commit a117216fa2deb30ac5489792a74a9ef5011779d1
Author: Maya <mv2731@columbia.edu>
Date: Wed Mar 24 00:43:49 2021 -0400

    neq: Enable not equals and greater than

commit 0502dcaaa32ebfd99f7697047ff29b4a009c28b4
Author: E L <eilamemail@gmail.com>
Date: Tue Mar 23 23:51:30 2021 -0400

    Testing suite fully operational

commit e448876114d45d3b55d9821474d4d31b22544ae0
Merge: 11e3257 9eca7d4
Author: E L <eilamemail@gmail.com>
Date: Mon Mar 22 23:10:43 2021 -0400

    Merge branch 'decinit' into main

commit 11e3257ee17f7052173513cfc369adb0dc557f80
Merge: 3bca4fe f3ead5f
Author: LetsGitStartedAlready <47231353+LetsGitStartedAlready@users.noreply.github.com>
Date: Mon Mar 22 23:00:45 2021 -0400

    Merge pull request #1 from pelincetin/decinit

    Add Declare/Initialization functionality

commit 9eca7d45c9207763a31ab8d53269c60baa08278e
Author: E L <eilamemail@gmail.com>
Date: Mon Mar 22 22:59:19 2021 -0400

    Finished decinit

commit d81f4662fca6f001d57e43e84d3568dc837da972
Author: Maya <mv2731@columbia.edu>
Date: Mon Mar 22 21:42:09 2021 -0400

    conc: Discover nodelist hatch issue; implement basic begin and end functions for hatch

commit 90bd2cb7b0e569e23036e4990f2223ea1988faeb
Author: Maya <mv2731@columbia.edu>
Date: Mon Mar 22 20:43:46 2021 -0400

    conc: Finish basic synch functions in graccllib

commit 3bca4fe2ab7e196caba85e818e64595f4c9ad23c
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Mon Mar 22 17:06:23 2021 -0400

    Edited README

commit 50e17d5b077a56cc20d9d330ba59957df0d99598
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Mon Mar 22 14:39:56 2021 -0400

    Updated README

commit f3ead5f07ef312d831b5eb527276d208c8bb6333
Author: E L <eilamemail@gmail.com>
Date: Mon Mar 22 01:56:38 2021 -0400

    Got local decinit, and global for basic types

commit d4e39f5747e057a6293f43dd290620ace2f3644a
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Mar 21 20:09:35 2021 +0000

    fixed a logic error, more error checking

commit d09fd6770a5223d7ce719dba8662740e8b8a2c2f
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Mar 21 19:48:17 2021 +0000

    clean target mistake

commit 98cdbab731c8aec795f0bc40ea09f74b2ba2066a
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Mar 21 19:39:56 2021 +0000

    basic implementation of sync and test

commit 970d78b4c8a468d0da9c03aaad604a70c34a87ee
Author: pelincetin <pc2807@barnard.edu>
Date: Sun Mar 21 19:14:28 2021 +0000

    initialize

commit 5b6936652a9dca73f5daf391a9b6d3a10cd008f3
Author: E L <eilamemail@gmail.com>
Date: Sun Mar 21 03:01:19 2021 -0400

    Decinit SAST working

commit 9a4fd0d39d34db132cb380ae86cdaaa1c9066e10
Author: E L <eilamemail@gmail.com>
Date: Sat Mar 20 19:32:49 2021 -0400

    Fixed additional tests

commit 99da4101133a67aa4315d9a59bd194c56ade6c73

Author: Hadley Callaway <hccallaway@gmail.com>
Date: Sat Mar 20 17:15:24 2021 -0400

    started implementing types -- all are untested

commit 9974bb6834be592c518a9c5094e946f56a29d624
Author: E L <eilamemail@gmail.com>
Date: Sat Mar 20 00:44:10 2021 -0400

    Added useful testing scripts

commit 9cbb7fd821aca4dd611addacec4e4ffd96527393
Author: E L <eilamemail@gmail.com>
Date: Fri Mar 19 18:00:33 2021 -0400

    Added testing

commit 4c946907072de541884232e204ca992b1a7aec25
Author: E L <eilamemail@gmail.com>
Date: Thu Mar 18 18:55:16 2021 -0400

    Changed to grc extension

commit bd1884a0d38de59083cb84d27c800948f45b648d
Author: E L <eilamemail@gmail.com>
Date: Thu Mar 18 18:54:12 2021 -0400

    Hello, World

commit 88960c39b0769a067ddd51a78b645f4d881f6614
Author: E L <eilamemail@gmail.com>
Date: Tue Mar 16 22:00:44 2021 -0400

    Got semant to compile, still requires further checking

commit 925f5e2b40776ba334aaf08491ee2327f464398c
Author: E L <eilamemail@gmail.com>
Date: Tue Mar 16 20:49:08 2021 -0400

    AST works, SAST in progress, bind is main issue

commit f84e81e2f5b0330eaf9cfa2f2ac68c62c714d2d3
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Wed Feb 24 16:21:18 2021 -0500

    Added if with no else statement

commit 65b585dbd57005575bd9bf881a10a30a91d00e94
Author: Hadley Callaway <hccallaway@gmail.com>
Date: Tue Feb 23 20:54:07 2021 -0500

    Changed synch syntax to get rid of lock

commit 77149016ae0c6c3c95818963d6bab6b0eb240cb5
Author: E L <eilamemail@gmail.com>
Date: Tue Feb 23 17:18:14 2021 -0500

Added our more java-like features

commit 0981ed94bee8e4dc4a354f4b74a29ef5ac993166
Author: E L <eilamemail@gmail.com>
Date: Tue Feb 23 02:13:18 2021 -0500

Added string literals to parser

commit 6ac1bf63ecccc1b112dd268b2972c9d90e779186
Author: E L <eilamemail@gmail.com>
Date: Tue Feb 23 02:07:57 2021 -0500

Added hatch/synch

commit c273b84c1bc83c16339520b7be6fcb89cfb0f043
Author: E L <eilamemail@gmail.com>
Date: Tue Feb 23 00:27:12 2021 -0500

Removed microcparse, replaced with graclparser

commit 772d578d0a146b9caa28c0fb2ec7f1cf2a3d7366
Author: E L <eilamemail@gmail.com>
Date: Tue Feb 23 00:24:22 2021 -0500

Changed name to graclparser

commit 7b9cba752cd586c053c1ddb27144f2318fc19973
Author: E L <eilamemail@gmail.com>
Date: Mon Feb 22 23:49:30 2021 -0500

Scanner complete, adding syntax to parser

commit 8a807f63c234f91e0d737f31ba40f6646012088f
Author: E L <eilamemail@gmail.com>
Date: Sun Feb 21 19:58:58 2021 -0500

First go, lexer almost complete

commit 3f5362eb69efc3a4679697dd020710666c188864
Author: pelincetin <pc2807@barnard.edu>
Date: Thu Feb 18 17:01:19 2021 -0500

microc

commit c7cbc9840be0a298450c57b2922f66e230175c2d
Author: E L <eilamemail@gmail.com>
Date: Tue Feb 16 20:07:21 2021 -0500

OK now I got it right, I gotta eat...

commit d5cb0ead7b8d5d5e890f186670fdf806e92ddd5d
Author: E L <eilamemail@gmail.com>
Date: Tue Feb 16 20:06:04 2021 -0500

Whoops, missed something in the parser

commit 959619ce7c8e5ccf9e98d4a1c9e18ccf54a0ec22
Author: E L <eilamemail@gmail.com>
Date: Tue Feb 16 20:02:21 2021 -0500

    First draft of parser

commit 309c105ab93d812e3f0904767e5ac2c8f9410e1f
Author: pelincetin <pc2807@barnard.edu>
Date: Tue Feb 16 18:53:05 2021 -0500

    init commit

commit 8e19f5fc24aa2e4edea5828209bbbb3f7be062ed
Author: Pelin Cetin <35610616+pelincetin@users.noreply.github.com>
Date: Sun Feb 14 16:38:59 2021 -0500

    Initial commit

Appendix B

1. LLVM generated for test-dijkstra.grc

```
; ModuleID = 'GRACL'
source_filename = "GRACL"

%Node = type { %Mutex, i32, i8*, i8, { %Mutex, %EdgeListItem*, %EdgeListItem* }*, %Node*, i32, i32, i8 }
%Mutex = type { { i32, i32, i32, i32, i32, i16, i16, %pthread_list } }
%pthread_list = type { %pthread_list*, %pthread_list* }
%EdgeListItem = type { { %Mutex, double, %Node*, %Node*, i8 }*, %EdgeListItem*, %EdgeListItem* }
%NodeListItem = type { %Node*, %NodeListItem*, %NodeListItem* }

@fmt = global [4 x i8] c"%d\0A\00"
@cast = global [3 x i8] c"%f\00"
@fmt.1 = global [4 x i8] c"%s\0A\00"
@str = private unnamed_addr constant [2 x i8] c"A\00"
@str.2 = private unnamed_addr constant [2 x i8] c"B\00"
@str.3 = private unnamed_addr constant [2 x i8] c"C\00"
@str.4 = private unnamed_addr constant [2 x i8] c"D\00"
@str.5 = private unnamed_addr constant [2 x i8] c"E\00"
@str.6 = private unnamed_addr constant [2 x i8] c"F\00"
@str.7 = private unnamed_addr constant [2 x i8] c"G\00"
@str.8 = private unnamed_addr constant [2 x i8] c"H\00"
@str.9 = private unnamed_addr constant [2 x i8] c"I\00"
@str.10 = private unnamed_addr constant [2 x i8] c"J\00"
@str.11 = private unnamed_addr constant [2 x i8] c"K\00"
@str.12 = private unnamed_addr constant [2 x i8] c"L\00"
@str.13 = private unnamed_addr constant [2 x i8] c"M\00"
@str.14 = private unnamed_addr constant [2 x i8] c"N\00"
@str.15 = private unnamed_addr constant [2 x i8] c"O\00"
@str.16 = private unnamed_addr constant [23 x i8] c"Path should be A-B-D-F\00"
@str.17 = private unnamed_addr constant [23 x i8] c"Path should be A-B-D-E\00"
@str.18 = private unnamed_addr constant [17 x i8] c"Path should be E\00"
@str.19 = private unnamed_addr constant [19 x i8] c"Path should be A-B\00"
@str.20 = private unnamed_addr constant [19 x i8] c"Path should be D-E\00"
```

```
@str.21 = private unnamed_addr constant [21 x i8] c"Path should be B-D-F\00"
@str.22 = private unnamed_addr constant [21 x i8] c"Path should be B-D-E\00"
@str.23 = private unnamed_addr constant [17 x i8] c"Path should be D\00"
@str.24 = private unnamed_addr constant [17 x i8] c"Path should be C\00"
@str.25 = private unnamed_addr constant [22 x i8] c"Path should not exist\00"
@str.26 = private unnamed_addr constant [22 x i8] c"Path should not exist\00"
@str.27 = private unnamed_addr constant [33 x i8] c"Path should be A-B-D-F-N-H-I-K-G\00"
@str.28 = private unnamed_addr constant [25 x i8] c"Path should be M-O-J-I-K\00"
@str.29 = private unnamed_addr constant [22 x i8] c"Path should not exist\00"
@str.30 = private unnamed_addr constant [21 x i8] c"Path should be I-K-G\00"
@str.31 = private unnamed_addr constant [21 x i8] c"Path should be C-M-O\00"

declare i32 @printf(i8*, ...)

declare i32 @__sprintf_chk(i8*, i32, i64, i8*, ...)

declare i64 @strlen(i8*)

declare i32 @strcmp(i8*, i8*)

declare i32 @_synch_start(i8*)

declare i32 @_synch_end(i8*)

define %Node* @getCheapestNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %unsettledNodes) {
entry:
  %unsettledNodes1 = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
    store { %Mutex, %NodeListItem*, %NodeListItem* }* %unsettledNodes, { %Mutex, %NodeListItem*, %NodeListItem* }**
%unsettledNodes1
  %var1_cheapestNode = alloca %Node*
  %var3_n = alloca %Node*
  %var2_cheapestCost = alloca double
  %var1_cheapestNode2 = load %Node*, %Node** %var1_cheapestNode
  store double 0x7FF0000000000000, double* %var2_cheapestCost
  %list = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %item = alloca %NodeListItem*
    %unsettledNodes3 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%unsettledNodes1
    store { %Mutex, %NodeListItem*, %NodeListItem* }* %unsettledNodes3, { %Mutex, %NodeListItem*, %NodeListItem* }**
%list
  %list4 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %list
    %list_gep = getelementptr inbounds { %Mutex, %NodeListItem*, %NodeListItem* }, { %Mutex, %NodeListItem*,
%NodeListItem* }* %list4, i32 0, i32 1
  %item_ptr = load %NodeListItem*, %NodeListItem** %list_gep
  store %NodeListItem* %item_ptr, %NodeListItem** %item
  br label %for

for:                                              ; preds = %end_for, %entry
  %item14 = load %NodeListItem*, %NodeListItem** %item
  %bool = icmp ne %NodeListItem* %item14, null
  br i1 %bool, label %for_body, label %formerge

for_body:                                         ; preds = %for
  %item5 = load %NodeListItem*, %NodeListItem** %item
  %item_gep = getelementptr inbounds %NodeListItem, %NodeListItem* %item5, i32 0, i32 0
```

```
  %element = load %Node*, %Node** %item_gep
  store %Node* %element, %Node** %var3_n
  %var3_n6 = load %Node*, %Node** %var3_n
  %var3_n7 = load %Node*, %Node** %var3_n
  %cost_result = call double @cost(%Node* %var3_n7)
  %var2_cheapestCost8 = load double, double* %var2_cheapestCost
  %tmp = fcmp olt double %cost_result, %var2_cheapestCost8
  br i1 %tmp, label %then, label %else

end_for:                                          ; preds = %ifmerge
  %item12 = load %NodeListItem*, %NodeListItem** %item
  %item_gep13 = getelementptr inbounds %NodeListItem, %NodeListItem* %item12, i32 0, i32 1
  %next = load %NodeListItem*, %NodeListItem** %item_gep13
  store %NodeListItem* %next, %NodeListItem** %item
  br label %for

ifmerge:                                          ; preds = %else, %then
  br label %end_for

then:                                             ; preds = %for_body
  %var3_n9 = load %Node*, %Node** %var3_n
  %cost_result10 = call double @cost(%Node* %var3_n9)
  store double %cost_result10, double* %var2_cheapestCost
  %var3_n11 = load %Node*, %Node** %var3_n
  store %Node* %var3_n11, %Node** %var1_cheapestNode
  br label %ifmerge

else:                                             ; preds = %for_body
  br label %ifmerge

formerge:                                         ; preds = %for
  %var1_cheapestNode15 = load %Node*, %Node** %var1_cheapestNode
  ret %Node* %var1_cheapestNode15
}

define { %Mutex, %NodeListItem*, %NodeListItem* }* @getShortestPath(%Node* %source, %Node* %goal, { %Mutex,
%NodeListItem*, %NodeListItem* }* %nl) {
entry:
  %source1 = alloca %Node*
  store %Node* %source, %Node** %source1
  %goal2 = alloca %Node*
  store %Node* %goal, %Node** %goal2
  %nl3 = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %nl, { %Mutex, %NodeListItem*, %NodeListItem* }** %nl3
  %var2_current = alloca %Node*
  %var1_path = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %var3_notfound = alloca i1
  %createNodeList_result = call { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()
   store { %Mutex, %NodeListItem*, %NodeListItem* }* %createNodeList_result, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var1_path
  %goal4 = load %Node*, %Node** %goal2
  store %Node* %goal4, %Node** %var2_current
  %goal5 = load %Node*, %Node** %goal2
  %source6 = load %Node*, %Node** %source1
  %nodeEquals_result = call i1 @nodeEquals(%Node* %source6, %Node* %goal5)
```

```
  br i1 %nodeEquals_result, label %then, label %else

ifmerge:                                          ; preds = %ifmerge11, %then
    %var1_path31 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var1_path
  ret { %Mutex, %NodeListItem*, %NodeListItem* }* %var1_path31

then:                                             ; preds = %entry
  %var2_current7 = load %Node*, %Node** %var2_current
    %var1_path8 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var1_path
   %prependNode_result = call i32 @prependNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %var1_path8, %Node*
%var2_current7)
  br label %ifmerge

else:                                             ; preds = %entry
  %goal9 = load %Node*, %Node** %goal2
  %nl10 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %nl3
  %includesNode_result = call i1 @includesNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %nl10, %Node* %goal9)
  %tmp = xor i1 %includesNode_result, true
  br i1 %tmp, label %then12, label %else14

ifmerge11:                                        ; preds = %merge, %then12
  br label %ifmerge

then12:                                           ; preds = %else
  %createNodeList_result13 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()
     store { %Mutex, %NodeListItem*, %NodeListItem* }* %createNodeList_result13, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var1_path
  br label %ifmerge11

else14:                                           ; preds = %else
  store i1 true, i1* %var3_notfound
  br label %while

while:                                            ; preds = %ifmerge26, %else14
  %var3_notfound30 = load i1, i1* %var3_notfound
  br i1 %var3_notfound30, label %while_body, label %merge

while_body:                                       ; preds = %while
  %var2_current15 = load %Node*, %Node** %var2_current
    %var1_path16 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var1_path
   %prependNode_result17 = call i32 @prependNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %var1_path16, %Node*
%var2_current15)
  %source18 = load %Node*, %Node** %source1
  %var2_current19 = load %Node*, %Node** %var2_current
  %nodeEquals_result20 = call i1 @nodeEquals(%Node* %var2_current19, %Node* %source18)
  br i1 %nodeEquals_result20, label %then22, label %else23

ifmerge21:                                        ; preds = %else23, %then22
  %var3_notfound24 = load i1, i1* %var3_notfound
  %tmp25 = icmp eq i1 %var3_notfound24, true
  br i1 %tmp25, label %then27, label %else29
```

```
then22:                                           ; preds = %while_body
  store i1 false, i1* %var3_notfound
  br label %ifmerge21

else23:                                           ; preds = %while_body
  br label %ifmerge21

ifmerge26:                                        ; preds = %else29, %then27
  br label %while

then27:                                           ; preds = %ifmerge21
  %var2_current28 = load %Node*, %Node** %var2_current
  %prec_result = call %Node* @prec(%Node* %var2_current28)
  store %Node* %prec_result, %Node** %var2_current
  br label %ifmerge26

else29:                                           ; preds = %ifmerge21
  br label %ifmerge26

merge:                                            ; preds = %while
  br label %ifmerge11
}

define { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }*
}*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %g, %Node* %source, %Node* %goal) {
entry:
  %g1 = alloca { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*,
i32, i32, i32 }*
  store { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32,
i32 }* %g, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32,
i32, i32 }** %g1
  %source2 = alloca %Node*
  store %Node* %source, %Node** %source2
  %goal3 = alloca %Node*
  store %Node* %goal, %Node** %goal3
  %var3_currentNode = alloca %Node*
  %var1_settledNodes = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %var5_edge = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var6_adjacentNode = alloca %Node*
  %var4_edges = alloca { %Mutex, %EdgeListItem*, %EdgeListItem* }*
  %var2_unsettledNodes = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %var7_newCost = alloca double
  %source4 = load %Node*, %Node** %source2
  %updateCost_result = call double @updateCost(%Node* %source4, double 0.000000e+00)
  %createNodeList_result = call { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %createNodeList_result, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var1_settledNodes
  %createNodeList_result5 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()
     store { %Mutex, %NodeListItem*, %NodeListItem* }* %createNodeList_result5, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var2_unsettledNodes
  %source6 = load %Node*, %Node** %source2
     %var2_unsettledNodes7 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var2_unsettledNodes
   %appendNode_result = call i32 @appendNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %var2_unsettledNodes7,
%Node* %source6)
```

```
 br label %while

while:                                            ; preds = %formerge, %entry
    %var2_unsettledNodes60 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var2_unsettledNodes
 %length_NL_result = call i32 @length_NL({ %Mutex, %NodeListItem*, %NodeListItem* }* %var2_unsettledNodes60)
 %tmp61 = icmp sgt i32 %length_NL_result, 0
 br i1 %tmp61, label %while_body, label %merge

while_body:                                       ; preds = %while
    %var2_unsettledNodes8 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var2_unsettledNodes
    %getCheapestNode_result = call %Node* @getCheapestNode({ %Mutex, %NodeListItem*, %NodeListItem* }*
%var2_unsettledNodes8)
 store %Node* %getCheapestNode_result, %Node** %var3_currentNode
 %var3_currentNode9 = load %Node*, %Node** %var3_currentNode
    %var2_unsettledNodes10 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var2_unsettledNodes
 %includesNode_result = call i1 @includesNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %var2_unsettledNodes10,
%Node* %var3_currentNode9)
 br i1 %includesNode_result, label %then, label %else

ifmerge:                                          ; preds = %else, %then
 %var3_currentNode13 = load %Node*, %Node** %var3_currentNode
 %edges_result = call { %Mutex, %EdgeListItem*, %EdgeListItem* }* @edges(%Node* %var3_currentNode13)
  store { %Mutex, %EdgeListItem*, %EdgeListItem* }* %edges_result, { %Mutex, %EdgeListItem*, %EdgeListItem* }**
%var4_edges
 %list = alloca { %Mutex, %EdgeListItem*, %EdgeListItem* }*
 %item = alloca %EdgeListItem*
  %var4_edges14 = load { %Mutex, %EdgeListItem*, %EdgeListItem* }*, { %Mutex, %EdgeListItem*, %EdgeListItem* }**
%var4_edges
 store { %Mutex, %EdgeListItem*, %EdgeListItem* }* %var4_edges14, { %Mutex, %EdgeListItem*, %EdgeListItem* }** %list
 %list15 = load { %Mutex, %EdgeListItem*, %EdgeListItem* }*, { %Mutex, %EdgeListItem*, %EdgeListItem* }** %list
   %list_gep = getelementptr inbounds { %Mutex, %EdgeListItem*, %EdgeListItem* }, { %Mutex, %EdgeListItem*,
%EdgeListItem* }* %list15, i32 0, i32 1
 %item_ptr = load %EdgeListItem*, %EdgeListItem** %list_gep
 store %EdgeListItem* %item_ptr, %EdgeListItem** %item
 br label %for

then:                                             ; preds = %while_body
 %var3_currentNode11 = load %Node*, %Node** %var3_currentNode
    %var2_unsettledNodes12 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var2_unsettledNodes
 %removeNode_result = call i32 @removeNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %var2_unsettledNodes12,
%Node* %var3_currentNode11)
 br label %ifmerge

else:                                             ; preds = %while_body
 br label %ifmerge

for:                                              ; preds = %end_for, %ifmerge
 %item56 = load %EdgeListItem*, %EdgeListItem** %item
 %bool = icmp ne %EdgeListItem* %item56, null
 br i1 %bool, label %for_body, label %formerge
```

```
for_body:                                         ; preds = %for
  %item16 = load %EdgeListItem*, %EdgeListItem** %item
  %item_gep = getelementptr inbounds %EdgeListItem, %EdgeListItem* %item16, i32 0, i32 0
  %element = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %item_gep
  store { %Mutex, double, %Node*, %Node*, i8 }* %element, { %Mutex, double, %Node*, %Node*, i8 }** %var5_edge
  %var5_edge17 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var5_edge
  %var5_edge18 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var5_edge
  %end_result = call %Node* @end({ %Mutex, double, %Node*, %Node*, i8 }* %var5_edge18)
  store %Node* %end_result, %Node** %var6_adjacentNode
  %var6_adjacentNode19 = load %Node*, %Node** %var6_adjacentNode
   %var1_settledNodes20 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var1_settledNodes
   %includesNode_result21 = call i1 @includesNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %var1_settledNodes20,
%Node* %var6_adjacentNode19)
  %tmp = xor i1 %includesNode_result21, true
  br i1 %tmp, label %then23, label %else53

end_for:                                          ; preds = %ifmerge22
  %item54 = load %EdgeListItem*, %EdgeListItem** %item
  %item_gep55 = getelementptr inbounds %EdgeListItem, %EdgeListItem* %item54, i32 0, i32 1
  %next = load %EdgeListItem*, %EdgeListItem** %item_gep55
  store %EdgeListItem* %next, %EdgeListItem** %item
  br label %for

ifmerge22:                                        ; preds = %else53, %ifmerge47
  br label %end_for

then23:                                           ; preds = %for_body
  %var3_currentNode24 = load %Node*, %Node** %var3_currentNode
  %cost_result = call double @cost(%Node* %var3_currentNode24)
  %var5_edge25 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var5_edge
  %weight_result = call double @weight({ %Mutex, double, %Node*, %Node*, i8 }* %var5_edge25)
  %tmp26 = fadd double %cost_result, %weight_result
  store double %tmp26, double* %var7_newCost
  %var6_adjacentNode27 = load %Node*, %Node** %var6_adjacentNode
  %cost_result28 = call double @cost(%Node* %var6_adjacentNode27)
  %tmp29 = fcmp oeq double %cost_result28, -1.000000e+00
  %var7_newCost30 = load double, double* %var7_newCost
  %var6_adjacentNode31 = load %Node*, %Node** %var6_adjacentNode
  %cost_result32 = call double @cost(%Node* %var6_adjacentNode31)
  %tmp33 = fcmp olt double %var7_newCost30, %cost_result32
  %tmp34 = or i1 %tmp29, %tmp33
  br i1 %tmp34, label %then36, label %else42

ifmerge35:                                        ; preds = %else42, %then36
  %var6_adjacentNode43 = load %Node*, %Node** %var6_adjacentNode
     %var2_unsettledNodes44 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var2_unsettledNodes
       %includesNode_result45 = call i1 @includesNode({ %Mutex, %NodeListItem*, %NodeListItem* }*
%var2_unsettledNodes44, %Node* %var6_adjacentNode43)
  %tmp46 = xor i1 %includesNode_result45, true
  br i1 %tmp46, label %then48, label %else52

then36:                                           ; preds = %then23
  %var7_newCost37 = load double, double* %var7_newCost
```

```
  %var6_adjacentNode38 = load %Node*, %Node** %var6_adjacentNode
  %updateCost_result39 = call double @updateCost(%Node* %var6_adjacentNode38, double %var7_newCost37)
  %var3_currentNode40 = load %Node*, %Node** %var3_currentNode
  %var6_adjacentNode41 = load %Node*, %Node** %var6_adjacentNode
  %setPrec_result = call %Node* @setPrec(%Node* %var6_adjacentNode41, %Node* %var3_currentNode40)
  br label %ifmerge35

else42:                                           ; preds = %then23
  br label %ifmerge35

ifmerge47:                                        ; preds = %else52, %then48
  br label %ifmerge22

then48:                                           ; preds = %ifmerge35
  %var6_adjacentNode49 = load %Node*, %Node** %var6_adjacentNode
    %var2_unsettledNodes50 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var2_unsettledNodes
      %appendNode_result51 = call i32 @appendNode({ %Mutex, %NodeListItem*, %NodeListItem* }*
%var2_unsettledNodes50, %Node* %var6_adjacentNode49)
  br label %ifmerge47

else52:                                           ; preds = %ifmerge35
  br label %ifmerge47

else53:                                           ; preds = %for_body
  br label %ifmerge22

formerge:                                         ; preds = %for
  %var3_currentNode57 = load %Node*, %Node** %var3_currentNode
   %var1_settledNodes58 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var1_settledNodes
   %appendNode_result59 = call i32 @appendNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %var1_settledNodes58,
%Node* %var3_currentNode57)
  br label %while

merge:                                            ; preds = %while
   %var1_settledNodes62 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var1_settledNodes
  %goal63 = load %Node*, %Node** %goal3
  %source64 = load %Node*, %Node** %source2
   %getShortestPath_result = call { %Mutex, %NodeListItem*, %NodeListItem* }* @getShortestPath(%Node* %source64,
%Node* %goal63, { %Mutex, %NodeListItem*, %NodeListItem* }* %var1_settledNodes62)
  ret { %Mutex, %NodeListItem*, %NodeListItem* }* %getShortestPath_result
}

define i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %nl) {
entry:
  %nl1 = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %nl, { %Mutex, %NodeListItem*, %NodeListItem* }** %nl1
  %var1_n = alloca %Node*
  %nl2 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %nl1
  %empty_NL_result = call i1 @empty_NL({ %Mutex, %NodeListItem*, %NodeListItem* }* %nl2)
  %tmp = xor i1 %empty_NL_result, true
  br i1 %tmp, label %then, label %else
```

```
ifmerge:                                          ; preds = %else, %formerge
  ret i32 0

then:                                             ; preds = %entry
  %list = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %item = alloca %NodeListItem*
  %nl3 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %nl1
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %nl3, { %Mutex, %NodeListItem*, %NodeListItem* }** %list
  %list4 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %list
    %list_gep = getelementptr inbounds { %Mutex, %NodeListItem*, %NodeListItem* }, { %Mutex, %NodeListItem*,
%NodeListItem* }* %list4, i32 0, i32 1
  %item_ptr = load %NodeListItem*, %NodeListItem** %list_gep
  store %NodeListItem* %item_ptr, %NodeListItem** %item
  br label %for

for:                                              ; preds = %end_for, %then
  %item10 = load %NodeListItem*, %NodeListItem** %item
  %bool = icmp ne %NodeListItem* %item10, null
  br i1 %bool, label %for_body, label %formerge

for_body:                                         ; preds = %for
  %item5 = load %NodeListItem*, %NodeListItem** %item
  %item_gep = getelementptr inbounds %NodeListItem, %NodeListItem* %item5, i32 0, i32 0
  %element = load %Node*, %Node** %item_gep
  store %Node* %element, %Node** %var1_n
  %var1_n6 = load %Node*, %Node** %var1_n
  %var1_n7 = load %Node*, %Node** %var1_n
  %data_result = call i8* @data(%Node* %var1_n7)
  %print = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* %data_result)
  br label %end_for

end_for:                                          ; preds = %for_body
  %item8 = load %NodeListItem*, %NodeListItem** %item
  %item_gep9 = getelementptr inbounds %NodeListItem, %NodeListItem* %item8, i32 0, i32 1
  %next = load %NodeListItem*, %NodeListItem** %item_gep9
  store %NodeListItem* %next, %NodeListItem** %item
  br label %for

formerge:                                         ; preds = %for
  br label %ifmerge

else:                                             ; preds = %entry
  br label %ifmerge
}

define i32 @main() {
entry:
  %var18_e2 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var22_e6 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var35_e20 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var11_nodeJ = alloca %Node*
  %var14_nodeM = alloca %Node*
  %var21_e5 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var38_e23 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var33_e18 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
```

```
  %var12_nodeK = alloca %Node*
  %var19_e3 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var1_g = alloca { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }*
  %var3_nodeB = alloca %Node*
  %var25_e9 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var4_nodeC = alloca %Node*
  %var8_nodeG = alloca %Node*
  %var9_nodeH = alloca %Node*
  %var24_e8 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var13_nodeL = alloca %Node*
  %var15_nodeN = alloca %Node*
  %var29_e14 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var30_e15 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var2_nodeA = alloca %Node*
  %var26_e10 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var10_nodeI = alloca %Node*
  %var20_e4 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var36_e21 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var16_nodeO = alloca %Node*
  %var39_e24 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var34_e19 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var5_nodeD = alloca %Node*
  %var23_e7 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var28_e13 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var37_e22 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var7_nodeF = alloca %Node*
  %var6_nodeE = alloca %Node*
  %var31_e16 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var27_e12 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var32_e17 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var41_path = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %var17_e1 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var40_e25 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
    %createGraph_result = call { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }* @createGraph(i32 15)
  store { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32,
i32 }* %createGraph_result, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
  %var1_g1 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }** %var1_g
  %createNode_result = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex,
%NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g1, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str, i32 0, i32
0))
  store %Node* %createNode_result, %Node** %var2_nodeA
  %var1_g2 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }** %var1_g
  %createNode_result3 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex,
%NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g2, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.2, i32 0,
i32 0))
  store %Node* %createNode_result3, %Node** %var3_nodeB
```

```
  %var1_g4 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }** %var1_g
  %createNode_result5 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex,
%NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g4, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.3, i32 0,
i32 0))
  store %Node* %createNode_result5, %Node** %var4_nodeC
  %var1_g6 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }** %var1_g
  %createNode_result7 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex,
%NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g6, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.4, i32 0,
i32 0))
  store %Node* %createNode_result7, %Node** %var5_nodeD
  %var1_g8 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }** %var1_g
  %createNode_result9 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex,
%NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g8, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.5, i32 0,
i32 0))
  store %Node* %createNode_result9, %Node** %var6_nodeE
    %var1_g10 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
   %createNode_result11 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, {
%Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g10, i8* getelementptr inbounds ([2 x i8], [2 x i8]*
@str.6, i32 0, i32 0))
  store %Node* %createNode_result11, %Node** %var7_nodeF
    %var1_g12 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
   %createNode_result13 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, {
%Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g12, i8* getelementptr inbounds ([2 x i8], [2 x i8]*
@str.7, i32 0, i32 0))
  store %Node* %createNode_result13, %Node** %var8_nodeG
    %var1_g14 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
   %createNode_result15 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, {
%Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g14, i8* getelementptr inbounds ([2 x i8], [2 x i8]*
@str.8, i32 0, i32 0))
  store %Node* %createNode_result15, %Node** %var9_nodeH
    %var1_g16 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
   %createNode_result17 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, {
%Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g16, i8* getelementptr inbounds ([2 x i8], [2 x i8]*
@str.9, i32 0, i32 0))
  store %Node* %createNode_result17, %Node** %var10_nodeI
    %var1_g18 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
   %createNode_result19 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, {
%Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g18, i8* getelementptr inbounds ([2 x i8], [2 x i8]*
@str.10, i32 0, i32 0))
```

```
 store %Node* %createNode_result19, %Node** %var11_nodeJ
   %var1_g20 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result21 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g20, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.11, i32 0, i32 0))
 store %Node* %createNode_result21, %Node** %var12_nodeK
   %var1_g22 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result23 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g22, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.12, i32 0, i32 0))
 store %Node* %createNode_result23, %Node** %var13_nodeL
   %var1_g24 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result25 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g24, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.13, i32 0, i32 0))
 store %Node* %createNode_result25, %Node** %var14_nodeM
   %var1_g26 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result27 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g26, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.14, i32 0, i32 0))
 store %Node* %createNode_result27, %Node** %var15_nodeN
   %var1_g28 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result29 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g28, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.15, i32 0, i32 0))
 store %Node* %createNode_result29, %Node** %var16_nodeO
 %var3_nodeB30 = load %Node*, %Node** %var3_nodeB
 %var2_nodeA31 = load %Node*, %Node** %var2_nodeA
   %var1_g32 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g32, %Node* %var2_nodeA31, %Node* %var3_nodeB30, double 1.000000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result, { %Mutex, double, %Node*, %Node*, i8 }** %var17_e1
 %var4_nodeC33 = load %Node*, %Node** %var4_nodeC
 %var2_nodeA34 = load %Node*, %Node** %var2_nodeA
   %var1_g35 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result36 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g35, %Node* %var2_nodeA34, %Node* %var4_nodeC33, double 1.500000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result36, { %Mutex, double, %Node*, %Node*, i8 }** %var18_e2
 %var5_nodeD37 = load %Node*, %Node** %var5_nodeD
```

```
 %var3_nodeB38 = load %Node*, %Node** %var3_nodeB
   %var1_g39 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result40 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g39, %Node* %var3_nodeB38, %Node* %var5_nodeD37, double 1.200000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result40, { %Mutex, double, %Node*, %Node*, i8 }** %var19_e3
 %var7_nodeF41 = load %Node*, %Node** %var7_nodeF
 %var3_nodeB42 = load %Node*, %Node** %var3_nodeB
   %var1_g43 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result44 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g43, %Node* %var3_nodeB42, %Node* %var7_nodeF41, double 1.500000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result44, { %Mutex, double, %Node*, %Node*, i8 }** %var20_e4
 %var6_nodeE45 = load %Node*, %Node** %var6_nodeE
 %var4_nodeC46 = load %Node*, %Node** %var4_nodeC
   %var1_g47 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result48 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g47, %Node* %var4_nodeC46, %Node* %var6_nodeE45, double 1.000000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result48, { %Mutex, double, %Node*, %Node*, i8 }** %var21_e5
 %var6_nodeE49 = load %Node*, %Node** %var6_nodeE
 %var5_nodeD50 = load %Node*, %Node** %var5_nodeD
   %var1_g51 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result52 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g51, %Node* %var5_nodeD50, %Node* %var6_nodeE49, double 2.000000e+00)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result52, { %Mutex, double, %Node*, %Node*, i8 }** %var22_e6
 %var7_nodeF53 = load %Node*, %Node** %var7_nodeF
 %var5_nodeD54 = load %Node*, %Node** %var5_nodeD
   %var1_g55 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result56 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g55, %Node* %var5_nodeD54, %Node* %var7_nodeF53, double 1.000000e+00)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result56, { %Mutex, double, %Node*, %Node*, i8 }** %var23_e7
 %var6_nodeE57 = load %Node*, %Node** %var6_nodeE
 %var7_nodeF58 = load %Node*, %Node** %var7_nodeF
   %var1_g59 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result60 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g59, %Node* %var7_nodeF58, %Node* %var6_nodeE57, double 5.000000e+00)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result60, { %Mutex, double, %Node*, %Node*, i8 }** %var24_e8
 %var15_nodeN61 = load %Node*, %Node** %var15_nodeN
 %var7_nodeF62 = load %Node*, %Node** %var7_nodeF
```

```
  %var1_g63 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result64 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g63, %Node* %var7_nodeF62, %Node* %var15_nodeN61, double 1.000000e+00)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result64, { %Mutex, double, %Node*, %Node*, i8 }** %var25_e9
 %var13_nodeL65 = load %Node*, %Node** %var13_nodeL
 %var15_nodeN66 = load %Node*, %Node** %var15_nodeN
  %var1_g67 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result68 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g67, %Node* %var15_nodeN66, %Node* %var13_nodeL65, double 2.000000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result68, { %Mutex, double, %Node*, %Node*, i8 }** %var26_e10
 %var9_nodeH69 = load %Node*, %Node** %var9_nodeH
 %var15_nodeN70 = load %Node*, %Node** %var15_nodeN
  %var1_g71 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result72 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g71, %Node* %var15_nodeN70, %Node* %var9_nodeH69, double 2.000000e+00)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result72, { %Mutex, double, %Node*, %Node*, i8 }** %var27_e12
 %var11_nodeJ73 = load %Node*, %Node** %var11_nodeJ
 %var9_nodeH74 = load %Node*, %Node** %var9_nodeH
  %var1_g75 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result76 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g75, %Node* %var9_nodeH74, %Node* %var11_nodeJ73, double 5.000000e+00)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result76, { %Mutex, double, %Node*, %Node*, i8 }** %var28_e13
 %var10_nodeI77 = load %Node*, %Node** %var10_nodeI
 %var9_nodeH78 = load %Node*, %Node** %var9_nodeH
  %var1_g79 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result80 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g79, %Node* %var9_nodeH78, %Node* %var10_nodeI77, double 1.000000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result80, { %Mutex, double, %Node*, %Node*, i8 }** %var29_e14
 %var10_nodeI81 = load %Node*, %Node** %var10_nodeI
 %var11_nodeJ82 = load %Node*, %Node** %var11_nodeJ
  %var1_g83 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result84 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g83, %Node* %var11_nodeJ82, %Node* %var10_nodeI81, double 3.000000e+00)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result84, { %Mutex, double, %Node*, %Node*, i8 }** %var30_e15
 %var8_nodeG85 = load %Node*, %Node** %var8_nodeG
 %var10_nodeI86 = load %Node*, %Node** %var10_nodeI
```

```
  %var1_g87 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result88 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g87, %Node* %var10_nodeI86, %Node* %var8_nodeG85, double 4.000000e+00)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result88, { %Mutex, double, %Node*, %Node*, i8 }** %var31_e16
 %var8_nodeG89 = load %Node*, %Node** %var8_nodeG
 %var2_nodeA90 = load %Node*, %Node** %var2_nodeA
  %var1_g91 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result92 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g91, %Node* %var2_nodeA90, %Node* %var8_nodeG89, double 4.500000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result92, { %Mutex, double, %Node*, %Node*, i8 }** %var32_e17
 %var12_nodeK93 = load %Node*, %Node** %var12_nodeK
 %var10_nodeI94 = load %Node*, %Node** %var10_nodeI
  %var1_g95 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result96 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g95, %Node* %var10_nodeI94, %Node* %var12_nodeK93, double 2.000000e+00)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result96, { %Mutex, double, %Node*, %Node*, i8 }** %var33_e18
 %var8_nodeG97 = load %Node*, %Node** %var8_nodeG
 %var12_nodeK98 = load %Node*, %Node** %var12_nodeK
  %var1_g99 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result100 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g99, %Node* %var12_nodeK98, %Node* %var8_nodeG97, double 1.000000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result100, { %Mutex, double, %Node*, %Node*, i8 }** %var34_e19
 %var14_nodeM101 = load %Node*, %Node** %var14_nodeM
 %var6_nodeE102 = load %Node*, %Node** %var6_nodeE
  %var1_g103 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result104 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g103, %Node* %var6_nodeE102, %Node* %var14_nodeM101, double 1.000000e+01)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result104, { %Mutex, double, %Node*, %Node*, i8 }** %var35_e20
 %var14_nodeM105 = load %Node*, %Node** %var14_nodeM
 %var4_nodeC106 = load %Node*, %Node** %var4_nodeC
  %var1_g107 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result108 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g107, %Node* %var4_nodeC106, %Node* %var14_nodeM105, double 8.000000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result108, { %Mutex, double, %Node*, %Node*, i8 }** %var36_e21
```

```
 %var16_nodeO109 = load %Node*, %Node** %var16_nodeO
 %var14_nodeM110 = load %Node*, %Node** %var14_nodeM
   %var1_g111 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result112 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g111, %Node* %var14_nodeM110, %Node* %var16_nodeO109, double 5.000000e+00)
   store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result112, { %Mutex, double, %Node*, %Node*, i8 }** %var37_e22
 %var11_nodeJ113 = load %Node*, %Node** %var11_nodeJ
 %var16_nodeO114 = load %Node*, %Node** %var16_nodeO
   %var1_g115 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result116 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g115, %Node* %var16_nodeO114, %Node* %var11_nodeJ113, double 2.000000e+00)
   store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result116, { %Mutex, double, %Node*, %Node*, i8 }** %var38_e23
 %var14_nodeM117 = load %Node*, %Node** %var14_nodeM
 %var11_nodeJ118 = load %Node*, %Node** %var11_nodeJ
   %var1_g119 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result120 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g119, %Node* %var11_nodeJ118, %Node* %var14_nodeM117, double 3.000000e+00)
   store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result120, { %Mutex, double, %Node*, %Node*, i8 }** %var39_e24
 %var11_nodeJ121 = load %Node*, %Node** %var11_nodeJ
 %var14_nodeM122 = load %Node*, %Node** %var14_nodeM
   %var1_g123 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result124 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g123, %Node* %var14_nodeM122, %Node* %var11_nodeJ121, double 1.000000e+01)
   store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result124, { %Mutex, double, %Node*, %Node*, i8 }** %var40_e25
  %print = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([23 x i8], [23 x i8]* @str.16, i32 0, i32 0))
 %var7_nodeF125 = load %Node*, %Node** %var7_nodeF
 %var2_nodeA126 = load %Node*, %Node** %var2_nodeA
   %var1_g127 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g127, %Node* %var2_nodeA126, %Node* %var7_nodeF125)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %var41_path128 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
 %printNl_result = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path128)
```

```
  %print129 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([23 x i8], [23 x i8]* @str.17, i32 0, i32 0))
  %var6_nodeE130 = load %Node*, %Node** %var6_nodeE
  %var2_nodeA131 = load %Node*, %Node** %var2_nodeA
    %var1_g132 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result133 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g132, %Node* %var2_nodeA131, %Node* %var6_nodeE130)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result133, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
    %var41_path134 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result135 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path134)
  %print136 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([17 x i8], [17 x i8]* @str.18, i32 0, i32 0))
  %var6_nodeE137 = load %Node*, %Node** %var6_nodeE
  %var6_nodeE138 = load %Node*, %Node** %var6_nodeE
    %var1_g139 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result140 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g139, %Node* %var6_nodeE138, %Node* %var6_nodeE137)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result140, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
    %var41_path141 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result142 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path141)
  %print143 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([19 x i8], [19 x i8]* @str.19, i32 0, i32 0))
  %var3_nodeB144 = load %Node*, %Node** %var3_nodeB
  %var2_nodeA145 = load %Node*, %Node** %var2_nodeA
    %var1_g146 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result147 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g146, %Node* %var2_nodeA145, %Node* %var3_nodeB144)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result147, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
    %var41_path148 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result149 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path148)
  %print150 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([19 x i8], [19 x i8]* @str.20, i32 0, i32 0))
  %var6_nodeE151 = load %Node*, %Node** %var6_nodeE
  %var5_nodeD152 = load %Node*, %Node** %var5_nodeD
    %var1_g153 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result154 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g153, %Node* %var5_nodeD152, %Node* %var6_nodeE151)
```

```
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result154, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var41_path
   %var41_path155 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var41_path
  %printNl_result156 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path155)
  %print157 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr
inbounds ([21 x i8], [21 x i8]* @str.21, i32 0, i32 0))
  %var7_nodeF158 = load %Node*, %Node** %var7_nodeF
  %var3_nodeB159 = load %Node*, %Node** %var3_nodeB
    %var1_g160 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result161 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*,
%EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g160, %Node*
%var3_nodeB159, %Node* %var7_nodeF158)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result161, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var41_path
   %var41_path162 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var41_path
  %printNl_result163 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path162)
  %print164 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr
inbounds ([21 x i8], [21 x i8]* @str.22, i32 0, i32 0))
  %var6_nodeE165 = load %Node*, %Node** %var6_nodeE
  %var3_nodeB166 = load %Node*, %Node** %var3_nodeB
    %var1_g167 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result168 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*,
%EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g167, %Node*
%var3_nodeB166, %Node* %var6_nodeE165)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result168, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var41_path
   %var41_path169 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var41_path
  %printNl_result170 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path169)
  %print171 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr
inbounds ([17 x i8], [17 x i8]* @str.23, i32 0, i32 0))
  %var5_nodeD172 = load %Node*, %Node** %var5_nodeD
  %var5_nodeD173 = load %Node*, %Node** %var5_nodeD
    %var1_g174 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result175 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*,
%EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g174, %Node*
%var5_nodeD173, %Node* %var5_nodeD172)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result175, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var41_path
   %var41_path176 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var41_path
  %printNl_result177 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path176)
  %print178 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr
inbounds ([17 x i8], [17 x i8]* @str.24, i32 0, i32 0))
  %var4_nodeC179 = load %Node*, %Node** %var4_nodeC
  %var4_nodeC180 = load %Node*, %Node** %var4_nodeC
```

```
  %var1_g181 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result182 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g181, %Node* %var4_nodeC180, %Node* %var4_nodeC179)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result182, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %var41_path183 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result184 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path183)
  %print185 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([22 x i8], [22 x i8]* @str.25, i32 0, i32 0))
  %var4_nodeC186 = load %Node*, %Node** %var4_nodeC
  %var3_nodeB187 = load %Node*, %Node** %var3_nodeB
  %var1_g188 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result189 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g188, %Node* %var3_nodeB187, %Node* %var4_nodeC186)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result189, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %var41_path190 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result191 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path190)
  %print192 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([22 x i8], [22 x i8]* @str.26, i32 0, i32 0))
  %var2_nodeA193 = load %Node*, %Node** %var2_nodeA
  %var7_nodeF194 = load %Node*, %Node** %var7_nodeF
  %var1_g195 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result196 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g195, %Node* %var7_nodeF194, %Node* %var2_nodeA193)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result196, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %var41_path197 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result198 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path197)
  %print199 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([33 x i8], [33 x i8]* @str.27, i32 0, i32 0))
  %var8_nodeG200 = load %Node*, %Node** %var8_nodeG
  %var2_nodeA201 = load %Node*, %Node** %var2_nodeA
  %var1_g202 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result203 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g202, %Node* %var2_nodeA201, %Node* %var8_nodeG200)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result203, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %var41_path204 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
```

```
  %printNl_result205 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path204)
  %print206 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([25 x i8], [25 x i8]* @str.28, i32 0, i32 0))
  %var12_nodeK207 = load %Node*, %Node** %var12_nodeK
  %var14_nodeM208 = load %Node*, %Node** %var14_nodeM
    %var1_g209 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result210 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g209, %Node* %var14_nodeM208, %Node* %var12_nodeK207)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result210, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
    %var41_path211 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result212 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path211)
  %print213 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([22 x i8], [22 x i8]* @str.29, i32 0, i32 0))
  %var14_nodeM214 = load %Node*, %Node** %var14_nodeM
  %var12_nodeK215 = load %Node*, %Node** %var12_nodeK
    %var1_g216 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result217 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g216, %Node* %var12_nodeK215, %Node* %var14_nodeM214)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result217, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
    %var41_path218 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result219 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path218)
  %print220 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([21 x i8], [21 x i8]* @str.30, i32 0, i32 0))
  %var8_nodeG221 = load %Node*, %Node** %var8_nodeG
  %var10_nodeI222 = load %Node*, %Node** %var10_nodeI
    %var1_g223 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %dijkstra_result224 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g223, %Node* %var10_nodeI222, %Node* %var8_nodeG221)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result224, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
    %var41_path225 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result226 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path225)
  %print227 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([21 x i8], [21 x i8]* @str.31, i32 0, i32 0))
  %var16_nodeO228 = load %Node*, %Node** %var16_nodeO
  %var4_nodeC229 = load %Node*, %Node** %var4_nodeC
    %var1_g230 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
```

```
  %dijkstra_result231 = call { %Mutex, %NodeListItem* , %NodeListItem* }* @dijkstra({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem*  }*  }*,  {  %Mutex,  %NodeListItem*,  %NodeListItem*  }*,  i32,  i32,  i32  }*  %var1_g230,  %Node* %var4_nodeC229, %Node* %var16_nodeO228)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %dijkstra_result231, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
   %var41_path232 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var41_path
  %printNl_result233 = call i32 @printNl({ %Mutex, %NodeListItem*, %NodeListItem* }* %var41_path232)
  ret i32 0
}

declare double @cost(%Node*)

declare { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()

declare i1 @nodeEquals(%Node*, %Node*)

declare i32 @prependNode({ %Mutex, %NodeListItem*, %NodeListItem* }*, %Node*)

declare i1 @includesNode({ %Mutex, %NodeListItem*, %NodeListItem* }*, %Node*)

declare %Node* @prec(%Node*)

declare double @updateCost(%Node*, double)

declare i32 @appendNode({ %Mutex, %NodeListItem*, %NodeListItem* }*, %Node*)

declare i32 @removeNode({ %Mutex, %NodeListItem*, %NodeListItem* }*, %Node*)

declare { %Mutex, %EdgeListItem*, %EdgeListItem* }* @edges(%Node*)

declare %Node* @end({ %Mutex, double, %Node*, %Node*, i8 }*)

declare double @weight({ %Mutex, double, %Node*, %Node*, i8 }*)

declare %Node* @setPrec(%Node*, %Node*)

declare i32 @length_NL({ %Mutex, %NodeListItem*, %NodeListItem* }*)

declare i1 @empty_NL({ %Mutex, %NodeListItem*, %NodeListItem* }*)

declare i8* @data(%Node*)

declare { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* @createGraph(i32)

declare %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, i8*)

declare { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, %Node*, %Node*, double)
```

2. LLVM generated for test-concdfs.grc

```
; ModuleID = 'GRACL'
source_filename = "GRACL"
```

```
%Node = type { %Mutex, i32, i8*, i8, { %Mutex, %EdgeListItem*, %EdgeListItem* }*, %Node*, i32, i32, i8 }
%Mutex = type { { i32, i32, i32, i32, i32, i16, i16, %pthread_list } }
%pthread_list = type { %pthread_list*, %pthread_list* }
%EdgeListItem = type { { %Mutex, double, %Node*, %Node*, i8 }*, %EdgeListItem*, %EdgeListItem* }
%NodeListItem = type { %Node*, %NodeListItem*, %NodeListItem* }
%hatch_args = type { %Node*, %Node*, { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }* }

@fmt = global [4 x i8] c"%d\0A\00"
@cast = global [3 x i8] c"%f\00"
@fmt.1 = global [4 x i8] c"%s\0A\00"
@goal_found = global i1 false
@str = private unnamed_addr constant [2 x i8] c"A\00"
@str.2 = private unnamed_addr constant [2 x i8] c"B\00"
@str.3 = private unnamed_addr constant [2 x i8] c"C\00"
@str.4 = private unnamed_addr constant [2 x i8] c"D\00"
@str.5 = private unnamed_addr constant [2 x i8] c"E\00"
@str.6 = private unnamed_addr constant [2 x i8] c"F\00"
@str.7 = private unnamed_addr constant [2 x i8] c"G\00"
@str.8 = private unnamed_addr constant [2 x i8] c"H\00"
@str.9 = private unnamed_addr constant [2 x i8] c"I\00"
@str.10 = private unnamed_addr constant [2 x i8] c"J\00"
@str.11 = private unnamed_addr constant [2 x i8] c"K\00"
@str.12 = private unnamed_addr constant [2 x i8] c"L\00"
@str.13 = private unnamed_addr constant [2 x i8] c"M\00"
@str.14 = private unnamed_addr constant [2 x i8] c"N\00"
@str.15 = private unnamed_addr constant [2 x i8] c"O\00"
@str.16 = private unnamed_addr constant [2 x i8] c"P\00"
@str.17 = private unnamed_addr constant [2 x i8] c"Q\00"
@str.18 = private unnamed_addr constant [2 x i8] c"R\00"
@str.19 = private unnamed_addr constant [2 x i8] c"S\00"
@str.20 = private unnamed_addr constant [2 x i8] c"T\00"
@str.21 = private unnamed_addr constant [2 x i8] c"U\00"
@str.22 = private unnamed_addr constant [2 x i8] c"V\00"
@str.23 = private unnamed_addr constant [2 x i8] c"W\00"
@str.24 = private unnamed_addr constant [2 x i8] c"X\00"
@str.25 = private unnamed_addr constant [2 x i8] c"Y\00"

declare i32 @printf(i8*, ...)

declare i32 @__sprintf_chk(i8*, i32, i64, i8*, ...)

declare i64 @strlen(i8*)

declare i32 @strcmp(i8*, i8*)

declare i32 @_synch_start(i8*)

declare i32 @_synch_end(i8*)

define i1 @goalTest(%Node* %goal, %Node* %current) {
entry:
  %goal1 = alloca %Node*
  store %Node* %goal, %Node** %goal1
```

```
  %current2 = alloca %Node*
  store %Node* %current, %Node** %current2
  %goal3 = load %Node*, %Node** %goal1
  %current4 = load %Node*, %Node** %current2
  %nodeEquals_result = call i1 @nodeEquals(%Node* %current4, %Node* %goal3)
  ret i1 %nodeEquals_result
}

define i32 @normalDFS(%Node* %current, %Node* %goal, { %Mutex, %NodeListItem*, %NodeListItem* }* %myPath, { %Mutex, %NodeListItem*, %NodeListItem* }* %path) {
entry:
  %current1 = alloca %Node*
  store %Node* %current, %Node** %current1
  %goal2 = alloca %Node*
  store %Node* %goal, %Node** %goal2
  %myPath3 = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %myPath, { %Mutex, %NodeListItem*, %NodeListItem* }** %myPath3
  %path4 = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %path, { %Mutex, %NodeListItem*, %NodeListItem* }** %path4
  %var2_neighbors = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %var3_n = alloca %Node*
  %var1_done = alloca i1
  %var4_recurse = alloca i1
  %var5_n = alloca %Node*
  store i1 false, i1* %var1_done
  %goal_found = load i1, i1* @goal_found
  br i1 %goal_found, label %then, label %else

ifmerge:                                          ; preds = %else, %then
  %var1_done5 = load i1, i1* %var1_done
  %tmp = xor i1 %var1_done5, true
  %goal_found6 = load i1, i1* @goal_found
  %tmp7 = xor i1 %goal_found6, true
  %tmp8 = and i1 %tmp, %tmp7
  br i1 %tmp8, label %then10, label %else77

then:                                             ; preds = %entry
  store i1 true, i1* %var1_done
  br label %ifmerge

else:                                             ; preds = %entry
  br label %ifmerge

ifmerge9:                                         ; preds = %else77, %ifmerge15
  ret i32 0

then10:                                           ; preds = %ifmerge
  %current11 = load %Node*, %Node** %current1
    %myPath12 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %myPath3
    %appendNode_result = call i32 @appendNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %myPath12, %Node* %current11)
  %current13 = load %Node*, %Node** %current1
  %goal14 = load %Node*, %Node** %goal2
  %goalTest_result = call i1 @goalTest(%Node* %goal14, %Node* %current13)
```

```
  br i1 %goalTest_result, label %then16, label %else25

ifmerge15:                                        ; preds = %ifmerge31, %formerge
  br label %ifmerge9

then16:                                           ; preds = %then10
  %list = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %item = alloca %NodeListItem*
    %myPath17 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%myPath3
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %myPath17, { %Mutex, %NodeListItem*, %NodeListItem* }** %list
  %list18 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %list
    %list_gep = getelementptr inbounds { %Mutex, %NodeListItem*, %NodeListItem* }, { %Mutex, %NodeListItem*,
%NodeListItem* }* %list18, i32 0, i32 1
  %item_ptr = load %NodeListItem*, %NodeListItem** %list_gep
  store %NodeListItem* %item_ptr, %NodeListItem** %item
  br label %for

for:                                              ; preds = %end_for, %then16
  %item24 = load %NodeListItem*, %NodeListItem** %item
  %bool = icmp ne %NodeListItem* %item24, null
  br i1 %bool, label %for_body, label %formerge

for_body:                                         ; preds = %for
  %item19 = load %NodeListItem*, %NodeListItem** %item
  %item_gep = getelementptr inbounds %NodeListItem, %NodeListItem* %item19, i32 0, i32 0
  %element = load %Node*, %Node** %item_gep
  store %Node* %element, %Node** %var5_n
  %var5_n20 = load %Node*, %Node** %var5_n
  %var5_n21 = load %Node*, %Node** %var5_n
  %data_result = call i8* @data(%Node* %var5_n21)
  %print = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* %data_result)
  br label %end_for

end_for:                                          ; preds = %for_body
  %item22 = load %NodeListItem*, %NodeListItem** %item
  %item_gep23 = getelementptr inbounds %NodeListItem, %NodeListItem* %item22, i32 0, i32 1
  %next = load %NodeListItem*, %NodeListItem** %item_gep23
  store %NodeListItem* %next, %NodeListItem** %item
  br label %for

formerge:                                         ; preds = %for
  store i1 true, i1* @goal_found
  store i1 true, i1* %var1_done
  br label %ifmerge15

else25:                                           ; preds = %then10
  %var1_done26 = load i1, i1* %var1_done
  %tmp27 = xor i1 %var1_done26, true
  %goal_found28 = load i1, i1* @goal_found
  %tmp29 = xor i1 %goal_found28, true
  %tmp30 = and i1 %tmp27, %tmp29
  br i1 %tmp30, label %then32, label %else76

ifmerge31:                                        ; preds = %else76, %formerge75
```

```
  br label %ifmerge15

then32:                                           ; preds = %else25
  %current33 = load %Node*, %Node** %current1
  %neighbors_result = call { %Mutex, %NodeListItem*, %NodeListItem* }* @neighbors(%Node* %current33)
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %neighbors_result, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var2_neighbors
  %list34 = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %item35 = alloca %NodeListItem*
  %var2_neighbors36 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var2_neighbors
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %var2_neighbors36, { %Mutex, %NodeListItem*, %NodeListItem* }**
%list34
  %list37 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %list34
    %list_gep38 = getelementptr inbounds { %Mutex, %NodeListItem*, %NodeListItem* }, { %Mutex, %NodeListItem*,
%NodeListItem* }* %list37, i32 0, i32 1
  %item_ptr39 = load %NodeListItem*, %NodeListItem** %list_gep38
  store %NodeListItem* %item_ptr39, %NodeListItem** %item35
  br label %for40

for40:                                            ; preds = %end_for42, %then32
  %item73 = load %NodeListItem*, %NodeListItem** %item35
  %bool74 = icmp ne %NodeListItem* %item73, null
  br i1 %bool74, label %for_body41, label %formerge75

for_body41:                                       ; preds = %for40
  %item43 = load %NodeListItem*, %NodeListItem** %item35
  %item_gep44 = getelementptr inbounds %NodeListItem, %NodeListItem* %item43, i32 0, i32 0
  %element45 = load %Node*, %Node** %item_gep44
  store %Node* %element45, %Node** %var3_n
  %var3_n46 = load %Node*, %Node** %var3_n
  store i1 false, i1* %var4_recurse
  %var3_n47 = load %Node*, %Node** %var3_n
  %void_ptr = bitcast %Node* %var3_n47 to i8*
  %synch_start = call i32 @_synch_start(i8* %void_ptr)
  %var3_n48 = load %Node*, %Node** %var3_n
  %visited_result = call i1 @visited(%Node* %var3_n48)
  %tmp49 = xor i1 %visited_result, true
  %goal_found50 = load i1, i1* @goal_found
  %tmp51 = xor i1 %goal_found50, true
  %tmp52 = and i1 %tmp49, %tmp51
  br i1 %tmp52, label %then54, label %else56

end_for42:                                        ; preds = %ifmerge63
  %item70 = load %NodeListItem*, %NodeListItem** %item35
  %item_gep71 = getelementptr inbounds %NodeListItem, %NodeListItem* %item70, i32 0, i32 1
  %next72 = load %NodeListItem*, %NodeListItem** %item_gep71
  store %NodeListItem* %next72, %NodeListItem** %item35
  br label %for40

ifmerge53:                                        ; preds = %else56, %then54
  %var3_n57 = load %Node*, %Node** %var3_n
  %void_ptr58 = bitcast %Node* %var3_n57 to i8*
  %synch_end = call i32 @_synch_end(i8* %void_ptr58)
  %var4_recurse59 = load i1, i1* %var4_recurse
```

```
  %goal_found60 = load i1, i1* @goal_found
  %tmp61 = xor i1 %goal_found60, true
  %tmp62 = and i1 %var4_recurse59, %tmp61
  br i1 %tmp62, label %then64, label %else69

then54:                                           ; preds = %for_body41
  store i1 true, i1* %var4_recurse
  %var3_n55 = load %Node*, %Node** %var3_n
  %updateVisited_result = call %Node* @updateVisited(%Node* %var3_n55, i1 true)
  br label %ifmerge53

else56:                                           ; preds = %for_body41
  br label %ifmerge53

ifmerge63:                                        ; preds = %else69, %then64
  br label %end_for42

then64:                                           ; preds = %ifmerge53
  %path65 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %path4
    %myPath66 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%myPath3
  %goal67 = load %Node*, %Node** %goal2
  %var3_n68 = load %Node*, %Node** %var3_n
    %normalDFS_result = call i32 @normalDFS(%Node* %var3_n68, %Node* %goal67, { %Mutex, %NodeListItem*,
%NodeListItem* }* %myPath66, { %Mutex, %NodeListItem*, %NodeListItem* }* %path65)
  br label %ifmerge63

else69:                                           ; preds = %ifmerge53
  br label %ifmerge63

formerge75:                                       ; preds = %for40
  br label %ifmerge31

else76:                                           ; preds = %else25
  br label %ifmerge31

else77:                                           ; preds = %ifmerge
  br label %ifmerge9
}

define void @normalDFS_start(%Node* %current, %Node* %goal, { %Mutex, %NodeListItem*, %NodeListItem* }* %myPath,
{ %Mutex, %NodeListItem*, %NodeListItem* }* %path) {
entry:
  %current1 = alloca %Node*
  store %Node* %current, %Node** %current1
  %goal2 = alloca %Node*
  store %Node* %goal, %Node** %goal2
  %myPath3 = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %myPath, { %Mutex, %NodeListItem*, %NodeListItem* }** %myPath3
  %path4 = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %path, { %Mutex, %NodeListItem*, %NodeListItem* }** %path4
  %var1_individual_nl = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %var2_n = alloca %Node*
  %createNodeList_result = call { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()
```

```
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %createNodeList_result, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var1_individual_nl
  %list = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %item = alloca %NodeListItem*
    %myPath5 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%myPath3
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %myPath5, { %Mutex, %NodeListItem*, %NodeListItem* }** %list
  %list6 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %list
    %list_gep = getelementptr inbounds { %Mutex, %NodeListItem*, %NodeListItem* }, { %Mutex, %NodeListItem*,
%NodeListItem* }* %list6, i32 0, i32 1
  %item_ptr = load %NodeListItem*, %NodeListItem** %list_gep
  store %NodeListItem* %item_ptr, %NodeListItem** %item
  br label %for

for:                                              ; preds = %end_for, %entry
  %item13 = load %NodeListItem*, %NodeListItem** %item
  %bool = icmp ne %NodeListItem* %item13, null
  br i1 %bool, label %for_body, label %formerge

for_body:                                         ; preds = %for
  %item7 = load %NodeListItem*, %NodeListItem** %item
  %item_gep = getelementptr inbounds %NodeListItem, %NodeListItem* %item7, i32 0, i32 0
  %element = load %Node*, %Node** %item_gep
  store %Node* %element, %Node** %var2_n
  %var2_n8 = load %Node*, %Node** %var2_n
  %var2_n9 = load %Node*, %Node** %var2_n
   %var1_individual_nl10 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var1_individual_nl
    %appendNode_result = call i32 @appendNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %var1_individual_nl10,
%Node* %var2_n9)
  br label %end_for

end_for:                                          ; preds = %for_body
  %item11 = load %NodeListItem*, %NodeListItem** %item
  %item_gep12 = getelementptr inbounds %NodeListItem, %NodeListItem* %item11, i32 0, i32 1
  %next = load %NodeListItem*, %NodeListItem** %item_gep12
  store %NodeListItem* %next, %NodeListItem** %item
  br label %for

formerge:                                         ; preds = %for
  %current14 = load %Node*, %Node** %current1
  %updateVisited_result = call %Node* @updateVisited(%Node* %current14, i1 true)
  %path15 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %path4
   %var1_individual_nl16 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var1_individual_nl
  %goal17 = load %Node*, %Node** %goal2
  %current18 = load %Node*, %Node** %current1
    %normalDFS_result = call i32 @normalDFS(%Node* %current18, %Node* %goal17, { %Mutex, %NodeListItem*,
%NodeListItem* }* %var1_individual_nl16, { %Mutex, %NodeListItem*, %NodeListItem* }* %path15)
  ret void
}

define i32 @main() {
entry:
  %var22_node11 = alloca %Node*
```

```
  %var30_node19 = alloca %Node*
  %var12_node10 = alloca %Node*
  %var25_node14 = alloca %Node*
  %var47_e20 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var15_e3 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var20_e8 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var26_node15 = alloca %Node*
  %var57_myPath = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %var1_g = alloca { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }*
  %var9_node7 = alloca %Node*
  %var52_e25 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var39_e12 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var56_children = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %var53_e26 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var3_node1 = alloca %Node*
  %var8_node6 = alloca %Node*
  %var24_node13 = alloca %Node*
  %var58_path = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %var14_e2 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var38_e11 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var41_e14 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var33_node22 = alloca %Node*
  %var34_node23 = alloca %Node*
  %var21_e9 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var19_e7 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var11_node9 = alloca %Node*
  %var2_done = alloca i1
  %var10_node8 = alloca %Node*
  %var27_node16 = alloca %Node*
  %var37_e10 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var42_e15 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var48_e21 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var49_e22 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var45_e18 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var36_node25 = alloca %Node*
  %var6_node4 = alloca %Node*
  %var59_n = alloca %Node*
  %var13_e1 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var23_node12 = alloca %Node*
  %var55_goal = alloca %Node*
  %var32_node21 = alloca %Node*
  %var44_e17 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var54_parent = alloca %Node*
  %var16_e4 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var17_e5 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var18_e6 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var31_node20 = alloca %Node*
  %var5_node3 = alloca %Node*
  %var46_e19 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var4_node2 = alloca %Node*
  %var7_node5 = alloca %Node*
  %var40_e13 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var51_e24 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var35_node24 = alloca %Node*
```

```
  %var29_node18 = alloca %Node*
  %var43_e16 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var50_e23 = alloca { %Mutex, double, %Node*, %Node*, i8 }*
  %var28_node17 = alloca %Node*
   %var1_g1 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}*, i32, i32, i32 }** %var1_g
  store i1 false, i1* %var2_done
  %var3_node12 = load %Node*, %Node** %var3_node1
  %var4_node23 = load %Node*, %Node** %var4_node2
  %var5_node34 = load %Node*, %Node** %var5_node3
  %var6_node45 = load %Node*, %Node** %var6_node4
  %var7_node56 = load %Node*, %Node** %var7_node5
  %var8_node67 = load %Node*, %Node** %var8_node6
  %var9_node78 = load %Node*, %Node** %var9_node7
  %var10_node89 = load %Node*, %Node** %var10_node8
  %var11_node910 = load %Node*, %Node** %var11_node9
  %var12_node1011 = load %Node*, %Node** %var12_node10
  %var13_e112 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var13_e1
  %var14_e213 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var14_e2
  %var15_e314 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var15_e3
  %var16_e415 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var16_e4
  %var17_e516 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var17_e5
  %var18_e617 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var18_e6
  %var19_e718 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var19_e7
  %var20_e819 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var20_e8
  %var21_e920 = load { %Mutex, double, %Node*, %Node*, i8 }*, { %Mutex, double, %Node*, %Node*, i8 }** %var21_e9
   %createGraph_result = call { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }* @createGraph(i32 12)
  store { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32,
i32 }* %createGraph_result, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
    %var1_g21 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex,
%NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g21, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str, i32 0,
i32 0))
  store %Node* %createNode_result, %Node** %var3_node1
    %var1_g22 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
   %createNode_result23 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, {
%Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g22, i8* getelementptr inbounds ([2 x i8], [2 x i8]*
@str.2, i32 0, i32 0))
  store %Node* %createNode_result23, %Node** %var4_node2
    %var1_g24 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*,
%NodeListItem* }*, i32, i32, i32 }** %var1_g
   %createNode_result25 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, {
%Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g24, i8* getelementptr inbounds ([2 x i8], [2 x i8]*
@str.3, i32 0, i32 0))
  store %Node* %createNode_result25, %Node** %var5_node3
```

```
  %var1_g26 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result27 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g26, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.4, i32 0, i32 0))
 store %Node* %createNode_result27, %Node** %var6_node4
  %var1_g28 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result29 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g28, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.5, i32 0, i32 0))
 store %Node* %createNode_result29, %Node** %var7_node5
  %var1_g30 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result31 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g30, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.6, i32 0, i32 0))
 store %Node* %createNode_result31, %Node** %var8_node6
  %var1_g32 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result33 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g32, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.7, i32 0, i32 0))
 store %Node* %createNode_result33, %Node** %var9_node7
  %var1_g34 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result35 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g34, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.8, i32 0, i32 0))
 store %Node* %createNode_result35, %Node** %var10_node8
  %var1_g36 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result37 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g36, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.9, i32 0, i32 0))
 store %Node* %createNode_result37, %Node** %var11_node9
  %var1_g38 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result39 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g38, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.10, i32 0, i32 0))
 store %Node* %createNode_result39, %Node** %var12_node10
  %var1_g40 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result41 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g40, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.11, i32 0, i32 0))
```

```
 store %Node* %createNode_result41, %Node** %var22_node11
   %var1_g42 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result43 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g42, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.12, i32 0, i32 0))
 store %Node* %createNode_result43, %Node** %var23_node12
   %var1_g44 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result45 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g44, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.13, i32 0, i32 0))
 store %Node* %createNode_result45, %Node** %var24_node13
   %var1_g46 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result47 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g46, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.14, i32 0, i32 0))
 store %Node* %createNode_result47, %Node** %var25_node14
   %var1_g48 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result49 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g48, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.15, i32 0, i32 0))
 store %Node* %createNode_result49, %Node** %var26_node15
   %var1_g50 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result51 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g50, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.16, i32 0, i32 0))
 store %Node* %createNode_result51, %Node** %var27_node16
   %var1_g52 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result53 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g52, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.17, i32 0, i32 0))
 store %Node* %createNode_result53, %Node** %var28_node17
   %var1_g54 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result55 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g54, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.18, i32 0, i32 0))
 store %Node* %createNode_result55, %Node** %var29_node18
   %var1_g56 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
```

```
  %createNode_result57 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g56, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.19, i32 0, i32 0))
 store %Node* %createNode_result57, %Node** %var30_node19
  %var1_g58 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result59 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g58, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.20, i32 0, i32 0))
 store %Node* %createNode_result59, %Node** %var31_node20
  %var1_g60 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result61 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g60, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.21, i32 0, i32 0))
 store %Node* %createNode_result61, %Node** %var32_node21
  %var1_g62 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result63 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g62, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.22, i32 0, i32 0))
 store %Node* %createNode_result63, %Node** %var33_node22
  %var1_g64 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result65 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g64, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.23, i32 0, i32 0))
 store %Node* %createNode_result65, %Node** %var34_node23
  %var1_g66 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result67 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g66, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.24, i32 0, i32 0))
 store %Node* %createNode_result67, %Node** %var35_node24
  %var1_g68 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %createNode_result69 = call %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g68, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @str.25, i32 0, i32 0))
 store %Node* %createNode_result69, %Node** %var36_node25
 %var4_node270 = load %Node*, %Node** %var4_node2
 %var3_node171 = load %Node*, %Node** %var3_node1
  %var1_g72 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g72, %Node* %var3_node171, %Node* %var4_node270, double 1.400000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result, { %Mutex, double, %Node*, %Node*, i8 }** %var13_e1
```

```
  %var7_node573 = load %Node*, %Node** %var7_node5
  %var4_node274 = load %Node*, %Node** %var4_node2
    %var1_g75 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result76 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g75, %Node* %var4_node274, %Node* %var7_node573, double 1.300000e+01)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result76, { %Mutex, double, %Node*, %Node*, i8 }** %var14_e2
  %var11_node977 = load %Node*, %Node** %var11_node9
  %var7_node578 = load %Node*, %Node** %var7_node5
    %var1_g79 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result80 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g79, %Node* %var7_node578, %Node* %var11_node977, double 4.300000e+01)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result80, { %Mutex, double, %Node*, %Node*, i8 }** %var15_e3
  %var22_node1181 = load %Node*, %Node** %var22_node11
  %var11_node982 = load %Node*, %Node** %var11_node9
    %var1_g83 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result84 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g83, %Node* %var11_node982, %Node* %var22_node1181, double 5.000000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result84, { %Mutex, double, %Node*, %Node*, i8 }** %var37_e10
  %var23_node1285 = load %Node*, %Node** %var23_node12
  %var22_node1186 = load %Node*, %Node** %var22_node11
    %var1_g87 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result88 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g87, %Node* %var22_node1186, %Node* %var23_node1285, double 2.000000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result88, { %Mutex, double, %Node*, %Node*, i8 }** %var38_e11
  %var24_node1389 = load %Node*, %Node** %var24_node13
  %var23_node1290 = load %Node*, %Node** %var23_node12
    %var1_g91 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result92 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g91, %Node* %var23_node1290, %Node* %var24_node1389, double 3.333000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result92, { %Mutex, double, %Node*, %Node*, i8 }** %var39_e12
  %var25_node1493 = load %Node*, %Node** %var25_node14
  %var24_node1394 = load %Node*, %Node** %var24_node13
    %var1_g95 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result96 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g95, %Node* %var24_node1394, %Node* %var25_node1493, double 3.333000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result96, { %Mutex, double, %Node*, %Node*, i8 }** %var40_e13
  %var26_node1597 = load %Node*, %Node** %var26_node15
```

```
 %var24_node1398 = load %Node*, %Node** %var24_node13
   %var1_g99 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result100 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g99, %Node* %var24_node1398, %Node* %var26_node1597, double 3.333000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result100, { %Mutex, double, %Node*, %Node*, i8 }** %var41_e14
 %var26_node15101 = load %Node*, %Node** %var26_node15
 %var25_node14102 = load %Node*, %Node** %var25_node14
   %var1_g103 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result104 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g103, %Node* %var25_node14102, %Node* %var26_node15101, double 4.500000e-01)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result104, { %Mutex, double, %Node*, %Node*, i8 }** %var42_e15
 %var5_node3105 = load %Node*, %Node** %var5_node3
 %var3_node1106 = load %Node*, %Node** %var3_node1
   %var1_g107 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result108 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g107, %Node* %var3_node1106, %Node* %var5_node3105, double 1.300000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result108, { %Mutex, double, %Node*, %Node*, i8 }** %var16_e4
 %var8_node6109 = load %Node*, %Node** %var8_node6
 %var5_node3110 = load %Node*, %Node** %var5_node3
   %var1_g111 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result112 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g111, %Node* %var5_node3110, %Node* %var8_node6109, double 4.300000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result112, { %Mutex, double, %Node*, %Node*, i8 }** %var17_e5
 %var9_node7113 = load %Node*, %Node** %var9_node7
 %var5_node3114 = load %Node*, %Node** %var5_node3
   %var1_g115 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result116 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g115, %Node* %var5_node3114, %Node* %var9_node7113, double 1.300000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result116, { %Mutex, double, %Node*, %Node*, i8 }** %var18_e6
 %var27_node16117 = load %Node*, %Node** %var27_node16
 %var9_node7118 = load %Node*, %Node** %var9_node7
   %var1_g119 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result120 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g119, %Node* %var9_node7118, %Node* %var27_node16117, double 8.000000e+00)
```

```
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result120, { %Mutex, double, %Node*, %Node*, i8 }** %var43_e16
 %var28_node17121 = load %Node*, %Node** %var28_node17
 %var9_node7122 = load %Node*, %Node** %var9_node7
  %var1_g123 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result124 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g123, %Node* %var9_node7122, %Node* %var28_node17121, double 1.350000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result124, { %Mutex, double, %Node*, %Node*, i8 }** %var44_e17
 %var29_node18125 = load %Node*, %Node** %var29_node18
 %var28_node17126 = load %Node*, %Node** %var28_node17
  %var1_g127 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result128 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g127, %Node* %var28_node17126, %Node* %var29_node18125, double 5.777770e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result128, { %Mutex, double, %Node*, %Node*, i8 }** %var45_e18
 %var30_node19129 = load %Node*, %Node** %var30_node19
 %var29_node18130 = load %Node*, %Node** %var29_node18
  %var1_g131 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result132 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g131, %Node* %var29_node18130, %Node* %var30_node19129, double 5.500000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result132, { %Mutex, double, %Node*, %Node*, i8 }** %var46_e19
 %var28_node17133 = load %Node*, %Node** %var28_node17
 %var30_node19134 = load %Node*, %Node** %var30_node19
  %var1_g135 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result136 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g135, %Node* %var30_node19134, %Node* %var28_node17133, double -4.500000e+01)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result136, { %Mutex, double, %Node*, %Node*, i8 }** %var47_e20
 %var12_node10137 = load %Node*, %Node** %var12_node10
 %var8_node6138 = load %Node*, %Node** %var8_node6
  %var1_g139 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
 %addEdge_result140 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g139, %Node* %var8_node6138, %Node* %var12_node10137, double 4.300000e+01)
 store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result140, { %Mutex, double, %Node*, %Node*, i8 }** %var19_e7
 %var31_node20141 = load %Node*, %Node** %var31_node20
 %var12_node10142 = load %Node*, %Node** %var12_node10
```

```
  %var1_g143 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result144 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g143, %Node* %var12_node10142, %Node* %var31_node20141, double -5.000000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result144, { %Mutex, double, %Node*, %Node*, i8 }** %var48_e21
  %var32_node21145 = load %Node*, %Node** %var32_node21
  %var12_node10146 = load %Node*, %Node** %var12_node10
  %var1_g147 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result148 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g147, %Node* %var12_node10146, %Node* %var32_node21145, double -3.000000e+00)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result148, { %Mutex, double, %Node*, %Node*, i8 }** %var49_e22
  %var33_node22149 = load %Node*, %Node** %var33_node22
  %var32_node21150 = load %Node*, %Node** %var32_node21
  %var1_g151 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result152 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g151, %Node* %var32_node21150, %Node* %var33_node22149, double 1.000000e+02)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result152, { %Mutex, double, %Node*, %Node*, i8 }** %var50_e23
  %var34_node23153 = load %Node*, %Node** %var34_node23
  %var31_node20154 = load %Node*, %Node** %var31_node20
  %var1_g155 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result156 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g155, %Node* %var31_node20154, %Node* %var34_node23153, double 6.600000e+01)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result156, { %Mutex, double, %Node*, %Node*, i8 }** %var51_e24
  %var6_node4157 = load %Node*, %Node** %var6_node4
  %var3_node1158 = load %Node*, %Node** %var3_node1
  %var1_g159 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result160 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g159, %Node* %var3_node1158, %Node* %var6_node4157, double 1.300000e+01)
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result160, { %Mutex, double, %Node*, %Node*, i8 }** %var20_e8
  %var10_node8161 = load %Node*, %Node** %var10_node8
  %var6_node4162 = load %Node*, %Node** %var6_node4
  %var1_g163 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result164 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g163, %Node* %var6_node4162, %Node* %var10_node8161, double 4.300000e+01)
```

```
  store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result164, { %Mutex, double, %Node*, %Node*, i8 }** %var21_e9
  %var35_node24165 = load %Node*, %Node** %var35_node24
  %var10_node8166 = load %Node*, %Node** %var10_node8
    %var1_g167 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result168 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g167, %Node* %var10_node8166, %Node* %var35_node24165, double 1.000000e+03)
    store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result168, { %Mutex, double, %Node*, %Node*, i8 }** %var52_e25
  %var36_node25169 = load %Node*, %Node** %var36_node25
  %var35_node24170 = load %Node*, %Node** %var35_node24
    %var1_g171 = load { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }** %var1_g
  %addEdge_result172 = call { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }* %var1_g171, %Node* %var35_node24170, %Node* %var36_node25169, double 6.600000e+01)
    store { %Mutex, double, %Node*, %Node*, i8 }* %addEdge_result172, { %Mutex, double, %Node*, %Node*, i8 }** %var53_e26
  %var3_node1173 = load %Node*, %Node** %var3_node1
  store %Node* %var3_node1173, %Node** %var54_parent
  %var34_node23174 = load %Node*, %Node** %var34_node23
  store %Node* %var34_node23174, %Node** %var55_goal
  %createNodeList_result = call { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %createNodeList_result, { %Mutex, %NodeListItem*, %NodeListItem* }** %var56_children
  %createNodeList_result175 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()
     store { %Mutex, %NodeListItem*, %NodeListItem* }* %createNodeList_result175, { %Mutex, %NodeListItem*, %NodeListItem* }** %var57_myPath
  %var54_parent176 = load %Node*, %Node** %var54_parent
  %var57_myPath177 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var57_myPath
    %appendNode_result = call i32 @appendNode({ %Mutex, %NodeListItem*, %NodeListItem* }* %var57_myPath177, %Node* %var54_parent176)
  %var54_parent178 = load %Node*, %Node** %var54_parent
  %updateVisited_result = call %Node* @updateVisited(%Node* %var54_parent178, i1 true)
  %var55_goal179 = load %Node*, %Node** %var55_goal
  %var54_parent180 = load %Node*, %Node** %var54_parent
  %nodeEquals_result = call i1 @nodeEquals(%Node* %var54_parent180, %Node* %var55_goal179)
  br i1 %nodeEquals_result, label %then, label %else

ifmerge:                                          ; preds = %hatch_formerge, %formerge
  ret i32 0

then:                                             ; preds = %entry
  %list = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %item = alloca %NodeListItem*
  %var57_myPath181 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %var57_myPath
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %var57_myPath181, { %Mutex, %NodeListItem*, %NodeListItem* }** %list
  %list182 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %list
```

```
    %list_gep = getelementptr inbounds { %Mutex, %NodeListItem*, %NodeListItem* }, { %Mutex, %NodeListItem*,
%NodeListItem* }* %list182, i32 0, i32 1
  %item_ptr = load %NodeListItem*, %NodeListItem** %list_gep
  store %NodeListItem* %item_ptr, %NodeListItem** %item
  br label %for

for:                                              ; preds = %end_for, %then
  %item188 = load %NodeListItem*, %NodeListItem** %item
  %bool = icmp ne %NodeListItem* %item188, null
  br i1 %bool, label %for_body, label %formerge

for_body:                                         ; preds = %for
  %item183 = load %NodeListItem*, %NodeListItem** %item
  %item_gep = getelementptr inbounds %NodeListItem, %NodeListItem* %item183, i32 0, i32 0
  %element = load %Node*, %Node** %item_gep
  store %Node* %element, %Node** %var59_n
  %var59_n184 = load %Node*, %Node** %var59_n
  %var59_n185 = load %Node*, %Node** %var59_n
  %data_result = call i8* @data(%Node* %var59_n185)
  %print = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt.1, i32 0, i32 0), i8* %data_result)
  br label %end_for

end_for:                                          ; preds = %for_body
  %item186 = load %NodeListItem*, %NodeListItem** %item
  %item_gep187 = getelementptr inbounds %NodeListItem, %NodeListItem* %item186, i32 0, i32 1
  %next = load %NodeListItem*, %NodeListItem** %item_gep187
  store %NodeListItem* %next, %NodeListItem** %item
  br label %for

formerge:                                         ; preds = %for
  br label %ifmerge

else:                                             ; preds = %entry
  %var54_parent189 = load %Node*, %Node** %var54_parent
  %neighbors_result = call { %Mutex, %NodeListItem*, %NodeListItem* }* @neighbors(%Node* %var54_parent189)
   store { %Mutex, %NodeListItem*, %NodeListItem* }* %neighbors_result, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var56_children
  %createNodeList_result190 = call { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()
     store { %Mutex, %NodeListItem*, %NodeListItem* }* %createNodeList_result190, { %Mutex, %NodeListItem*,
%NodeListItem* }** %var58_path
  %nl_length = alloca i32
   %var56_children191 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var56_children
  %length_NL_result = call i32 @length_NL({ %Mutex, %NodeListItem*, %NodeListItem* }* %var56_children191)
  store i32 %length_NL_result, i32* %nl_length
  %pthread_array = alloca i64*
  %args_array = alloca %hatch_args*
  %length = load i32, i32* %nl_length
  %sext_length = sext i32 %length to i64
  %bytes = mul i64 8, %sext_length
  %malloc = call i8* @malloc(i64 %bytes)
  %cast_mem = bitcast i8* %malloc to i64*
  store i64* %cast_mem, i64** %pthread_array
  %length192 = load i32, i32* %nl_length
  %sext_length193 = sext i32 %length192 to i64
```

```
  %bytes194 = mul i64 32, %sext_length193
  %malloc195 = call i8* @malloc(i64 %bytes194)
  %cast_mem196 = bitcast i8* %malloc195 to %hatch_args*
  store %hatch_args* %cast_mem196, %hatch_args** %args_array
  %i = alloca i32
  store i32 0, i32* %i
  %list197 = alloca { %Mutex, %NodeListItem*, %NodeListItem* }*
  %item198 = alloca %NodeListItem*
   %var56_children199 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem*
}** %var56_children
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %var56_children199, { %Mutex, %NodeListItem*, %NodeListItem* }**
%list197
  %list200 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %list197
    %list_gep201 = getelementptr inbounds { %Mutex, %NodeListItem*, %NodeListItem* }, { %Mutex, %NodeListItem*,
%NodeListItem* }* %list200, i32 0, i32 1
  %item_ptr202 = load %NodeListItem*, %NodeListItem** %list_gep201
  store %NodeListItem* %item_ptr202, %NodeListItem** %item198
  br label %hatch_for

hatch_for:                                        ; preds = %hatch_end_for, %else
  %item233 = load %NodeListItem*, %NodeListItem** %item198
  %bool234 = icmp ne %NodeListItem* %item233, null
  br i1 %bool234, label %hatch_for_body, label %parent_actions

hatch_for_body:                                   ; preds = %hatch_for
  %item203 = load %NodeListItem*, %NodeListItem** %item198
  %item_gep204 = getelementptr inbounds %NodeListItem, %NodeListItem* %item203, i32 0, i32 0
  %val = load %Node*, %Node** %item_gep204
  %arg = load %hatch_args*, %hatch_args** %args_array
  %i205 = load i32, i32* %i
  %i206 = sext i32 %i205 to i64
  %gep1 = getelementptr inbounds %hatch_args, %hatch_args* %arg, i64 %i206
  %field_gep = getelementptr inbounds %hatch_args, %hatch_args* %gep1, i32 0, i32 0
  store %Node* %val, %Node** %field_gep
  %var55_goal207 = load %Node*, %Node** %var55_goal
  %arg208 = load %hatch_args*, %hatch_args** %args_array
  %i209 = load i32, i32* %i
  %i210 = sext i32 %i209 to i64
  %gep1211 = getelementptr inbounds %hatch_args, %hatch_args* %arg208, i64 %i210
  %field_gep212 = getelementptr inbounds %hatch_args, %hatch_args* %gep1211, i32 0, i32 1
  store %Node* %var55_goal207, %Node** %field_gep212
  %var57_myPath213 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var57_myPath
  %arg214 = load %hatch_args*, %hatch_args** %args_array
  %i215 = load i32, i32* %i
  %i216 = sext i32 %i215 to i64
  %gep1217 = getelementptr inbounds %hatch_args, %hatch_args* %arg214, i64 %i216
  %field_gep218 = getelementptr inbounds %hatch_args, %hatch_args* %gep1217, i32 0, i32 2
  store { %Mutex, %NodeListItem*, %NodeListItem* }* %var57_myPath213, { %Mutex, %NodeListItem*, %NodeListItem* }**
%field_gep218
   %var58_path219 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }**
%var58_path
  %arg220 = load %hatch_args*, %hatch_args** %args_array
  %i221 = load i32, i32* %i
  %i222 = sext i32 %i221 to i64
```

```
  %gep1223 = getelementptr inbounds %hatch_args, %hatch_args* %arg220, i64 %i222
  %field_gep224 = getelementptr inbounds %hatch_args, %hatch_args* %gep1223, i32 0, i32 3
   store { %Mutex, %NodeListItem*, %NodeListItem* }* %var58_path219, { %Mutex, %NodeListItem*, %NodeListItem* }**
%field_gep224
  %pthreads = load i64*, i64** %pthread_array
  %i225 = load i32, i32* %i
  %i226 = sext i32 %i225 to i64
  %pthread_gep = getelementptr inbounds i64, i64* %pthreads, i64 %i226
  %args = load %hatch_args*, %hatch_args** %args_array
  %i227 = load i32, i32* %i
  %i228 = sext i32 %i227 to i64
  %args_gep = getelementptr inbounds %hatch_args, %hatch_args* %args, i64 %i228
  %cast_ptr = bitcast %hatch_args* %args_gep to i8*
      %pthread_create = call i32 @pthread_create(i64* %pthread_gep, { i64, [48 x i8] }* null, i8* (i8*)*
@hatch_unwrapper_normalDFS_start, i8* %cast_ptr)
  br label %hatch_end_for

hatch_end_for:                                    ; preds = %hatch_for_body
  %item229 = load %NodeListItem*, %NodeListItem** %item198
  %item_gep230 = getelementptr inbounds %NodeListItem, %NodeListItem* %item229, i32 0, i32 1
  %next231 = load %NodeListItem*, %NodeListItem** %item_gep230
  store %NodeListItem* %next231, %NodeListItem** %item198
  %i232 = load i32, i32* %i
  %add_one = add nsw i32 %i232, 1
  store i32 %add_one, i32* %i
  br label %hatch_for

parent_actions:                                   ; preds = %hatch_for
  br label %hatch_formerge

hatch_formerge:                                   ; preds = %parent_actions
  %threads = load i64*, i64** %pthread_array
  %length235 = load i32, i32* %nl_length
  %hatch_end = call i32 @hatch_end(i64* %threads, i32 %length235)
  br label %ifmerge
}

declare i1 @nodeEquals(%Node*, %Node*)

declare i32 @appendNode({ %Mutex, %NodeListItem*, %NodeListItem* }*, %Node*)

declare i8* @data(%Node*)

declare { %Mutex, %NodeListItem*, %NodeListItem* }* @neighbors(%Node*)

declare i1 @visited(%Node*)

declare %Node* @updateVisited(%Node*, i1)

declare { %Mutex, %NodeListItem*, %NodeListItem* }* @createNodeList()

declare { { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32,
i32, i32 }* @createGraph(i32)
```

```
declare %Node* @createNode({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, i8*)

declare { %Mutex, double, %Node*, %Node*, i8 }* @addEdge({ { %Node*, { %Mutex, %EdgeListItem*, %EdgeListItem* }* }*, { %Mutex, %NodeListItem*, %NodeListItem* }*, i32, i32, i32 }*, %Node*, %Node*, double)

define i8* @hatch_unwrapper_normalDFS_start(i8*) {
entry:
  %void_ptr = alloca i8*
  %wrapper = alloca %hatch_args*
  store i8* %0, i8** %void_ptr
  %void_ptr1 = load i8*, i8** %void_ptr
  %cast_ptr = bitcast i8* %void_ptr1 to %hatch_args*
  store %hatch_args* %cast_ptr, %hatch_args** %wrapper
  %struct_ptr = load %hatch_args*, %hatch_args** %wrapper
  %arg_gep = getelementptr inbounds %hatch_args, %hatch_args* %struct_ptr, i32 0, i32 0
  %arg = load %Node*, %Node** %arg_gep
  %struct_ptr2 = load %hatch_args*, %hatch_args** %wrapper
  %arg_gep3 = getelementptr inbounds %hatch_args, %hatch_args* %struct_ptr2, i32 0, i32 1
  %arg4 = load %Node*, %Node** %arg_gep3
  %struct_ptr5 = load %hatch_args*, %hatch_args** %wrapper
  %arg_gep6 = getelementptr inbounds %hatch_args, %hatch_args* %struct_ptr5, i32 0, i32 2
  %arg7 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %arg_gep6
  %struct_ptr8 = load %hatch_args*, %hatch_args** %wrapper
  %arg_gep9 = getelementptr inbounds %hatch_args, %hatch_args* %struct_ptr8, i32 0, i32 3
  %arg10 = load { %Mutex, %NodeListItem*, %NodeListItem* }*, { %Mutex, %NodeListItem*, %NodeListItem* }** %arg_gep9
    call void @normalDFS_start(%Node* %arg, %Node* %arg4, { %Mutex, %NodeListItem*, %NodeListItem* }* %arg7, { %Mutex, %NodeListItem*, %NodeListItem* }* %arg10)
  ret i8* null
}

declare i32 @length_NL({ %Mutex, %NodeListItem*, %NodeListItem* }*)

declare i8* @malloc(i64)

declare i32 @pthread_create(i64*, { i64, [48 x i8] }*, i8* (i8*)*, i8*)

declare i32 @hatch_end(i64*, i32)
```

Appendix C

```
#!/bin/sh

#Taken from microC

LLI="lli"
LLC="llc"
CC="cc"
GRACL="./gracl.native"

# Set time limit for all operations
```

```
ulimit -t 30

globallog=testall.log
rm -f $globallog
error=0
globalerror=0

keep=0

Usage() {
   echo "Usage: testall.sh [options] [.grc files]"
   echo "-k    Keep intermediate files"
   echo "-h    Print this help"
   exit 1
}

SignalError() {
   if [ $error -eq 0 ] ; then
   echo "FAILED"
   error=1
   fi
   echo "  $1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile.  Differences, if any, written to difffile
Compare() {
   generatedfiles="$generatedfiles $3"
   echo diff -b $1 $2 ">" $3 1>&2
   diff -b "$1" "$2" > "$3" 2>&1 || {
   SignalError "$1 differs"
   echo "FAILED $1 differs from $2" 1>&2
   }
}

# Run <args>
# Report the command, run it, and report any errors
Run() {
   echo $* 1>&2
   eval $* || {
   SignalError "$1 failed on $*"
   return 1
```

```
    }
}

# RunFail <args>
# Report the command, run it, and expect an error
RunFail() {
    echo $* 1>&2
    eval $* && {
    SignalError "failed: $* did not report an error"
    return 1
    }
    return 0
}

Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                              s/.grc//'`
    reffile=`echo $1 | sed 's/.grc$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

        generatedfiles="$generatedfiles  ${basename}.ll  ${basename}.s  ${basename}.exe
${basename}.out" &&
    Run "$GRACL" "$1" ">" "${basename}.ll" &&
    Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">" "${basename}.s" &&
     Run "$CC" "-o" "${basename}.exe" "${basename}.s" "-pthread graclstdlib.o" &&#link
in standard library in C
    Run "./${basename}.exe" > "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
    if [ $keep -eq 0 ] ; then
        rm -f $generatedfiles
```

```
   fi
   echo "OK"
   echo "###### SUCCESS" 1>&2
   else
   echo "###### FAILED" 1>&2
   globalerror=$error
   fi
}

CheckFail() {
   error=0
   basename=`echo $1 | sed 's/.*\\///
                            s/.grc//'`
   reffile=`echo $1 | sed 's/.grc$//'`
   basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

   echo -n "$basename..."

   echo 1>&2
   echo "###### Testing $basename" 1>&2

   generatedfiles=""

   generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
   RunFail "$GRACL" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
   Compare ${basename}.err ${reffile}.err ${basename}.diff

   # Report the status and clean up the generated files

   if [ $error -eq 0 ] ; then
   if [ $keep -eq 0 ] ; then
       rm -f $generatedfiles
   fi
   echo "OK"
   echo "###### SUCCESS" 1>&2
   else
   echo "###### FAILED" 1>&2
   globalerror=$error
   fi
}

while getopts kdpsh c; do
```

```
   case $c in
   k) # Keep intermediate files
       keep=1
       ;;
   h) # Help
       Usage
       ;;
   esac
done

shift `expr $OPTIND - 1`

LLIFail() {
 echo "Could not find the LLVM interpreter \"$LLI\"."
 echo "Check your LLVM installation and/or modify the LLI variable in testall.sh"
 exit 1
}

which "$LLI" >> $globallog || LLIFail


#Check for standard library
#if [ ! -f printbig.o ]
#then
#    echo "Could not find printbig.o"
#    echo "Try \"make printbig.o\""
#    exit 1
#fi

if [ $# -ge 1 ]
then
   files=$@
else
   files="tests/test-*.grc tests/fail-*.grc"
fi

for file in $files
do
   case $file in
   *test-*)
       Check $file 2>> $globallog
       ;;
```

```
   *fail-*)
       CheckFail $file 2>> $globallog
       ;;
   *)
       echo "unknown file type $file"
       globalerror=1
       ;;
   esac
done

exit $globalerror
```

Testall.sh

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

stdlib : graclstdlib.c
   cc -o graclstdlib -pthread graclstdlib.c

gracl.native :
   opam config exec -- \
   ocamlbuild -use-ocamlfind gracl.native

.PHONY : codegen
codegen:
   ocamlc ast.ml
   ocamlc -c sast.ml ast.cmo
    ocamlfind ocamlc -c -w +a-4 -package llvm -package llvm.analysis -o functions.cmo
functions.ml
    ocamlfind ocamlc -c -w +a-4 -package llvm -package llvm.analysis -o codegen.cmo
codegen.ml

.PHONY : semant
semant:
   ocamlc ast.ml
   ocamlc -c sast.ml ast.cmo
```

```
    ocamlfind ocamlc -c -w +a-4 -package llvm -package llvm.analysis -o functions.cmo
functions.ml
   ocamlc -c semant.ml ast.cmo sast.cmo functions.cmo

.PHONY : helloworld
helloworld:
   ./scripts/breakdown.sh helloWorld

# "make clean" removes all generated files

.PHONY : clean
clean :
   ocamlbuild -clean
    rm -rf *.o testall.log ocamllllvm *.diff *.mli graclparser.ml *.output gracl.native
*.cmi *.cmo *.out *.ll *.s *.err *.exe *.breakdown

# removes some generated files while keeping those useful for debugging
.PHONY : debugclean
debugclean:
   ocamlbuild -clean
    rm -rf ocamllllvm *.mli graclparser.ml *.output gracl.native *.cmi *.cmo *.out *.s
*.exe
```

Makefile

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

type op = Add | Sub | Mult | Div | Mod | Equal | Neq | Less | Great | Leq | Geq |
          And | Or

type uop = Neg | Not

type typ = Int | Bool | Void | Double | String | Graph | Node | Edge |
           Inttable | Doubletable | Nodelist | Edgelist

type expr =
   Literal of int
 | Fliteral of string
 | BoolLit of bool
 | Sliteral of string
 | Id of string
 | Binop of expr * op * expr
 | Unop of uop * expr
```

```
 | Assign of string * expr
 | Call of string * expr list
 | Access of string * string
 | Insert of string * string * expr
 | Noexpr
 | Nodexpr

type bind =
   Dec of typ * string
 | Decinit of typ * string * expr

type formal = typ * string

type stmt =
   Block of stmt list
 | BlockEnd
 | Expr of expr
 | Return of expr
 | If of expr * stmt * stmt
 | For of typ * string * expr * stmt
 | While of expr * stmt
 | Hatch of expr * string * expr list * stmt
 | Synch of string * stmt
 | LoclBind of bind

type func_decl = {
   typ : typ;
   fname : string;
   formals : formal list;
   body : stmt list;
 }

type progpart =
   FuncDecl of func_decl
 | GlobBind of bind

type program = progpart list

(* Pretty-printing functions *)

let string_of_op = function
   Add -> "+"
```

```
 | Sub -> "-"
 | Mult -> "*"
 | Div -> "/"
 | Mod -> "%"
 | Equal -> "=="
 | Neq -> "!="
 | Less -> "<"
 | Great -> ">"
 | Leq -> "<="
 | Geq -> ">="
 | And -> "&&"
 | Or -> "||"

let string_of_uop = function
   Neg -> "-"
 | Not -> "!"

let name_of_bind = function
   Dec(_, n) -> n
 | Decinit(_, n, _) -> n
let strip_val = function
   Dec(a, b) -> (a, b)
 | Decinit(a, b, _) -> (a, b)
let rec string_of_expr = function
   Literal(l) -> string_of_int l
 | Fliteral(l) -> l
 | Sliteral(l) -> l
 | BoolLit(true) -> "true"
 | BoolLit(false) -> "false"
 | Id(s) -> s
 | Binop(e1, o, e2) ->
     string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
 | Unop(o, e) -> string_of_uop o ^ string_of_expr e
 | Assign(v, e) -> v ^ " = " ^ string_of_expr e
 | Call(f, el) ->
     f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
 | Access(t, n) -> t ^ "[" ^ n ^ "]"
 | Insert(t, n, e) -> t ^ "[" ^ n ^ "] = " ^ string_of_expr e
 | Noexpr -> ""
 | Nodexpr -> ""

let string_of_typ = function
```

```
   Int -> "int"
 | Bool -> "bool"
 | Void -> "void"
 | Double -> "double"
 | String -> "String"
 | Graph -> "Graph"
 | Node -> "Node"
 | Edge -> "Edge"
 | Inttable -> "IntTable"
 | Doubletable -> "DoubleTable"
 | Nodelist -> "NodeList"
 | Edgelist -> "EdgeList"

let string_of_vdecl = function
| Dec(t, id) -> string_of_typ t ^ " " ^ id ^ ";\n"
| Decinit(t, id, e) -> string_of_typ t ^ " " ^ id ^ " = " ^ string_of_expr e ^ ";\n"
let rec string_of_stmt = function
   Block(stmts) ->
     "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
 | Expr(expr) -> string_of_expr expr ^ ";\n";
 | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
 | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
 | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
     string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
 | For(t, n, e, s) -> "for (" ^ string_of_typ t ^ " " ^ n ^ " in " ^ string_of_expr e
^ ") " ^ string_of_stmt s
 | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s
 | Hatch(nl, f, el, s) -> "hatch " ^ string_of_expr nl ^ " " ^
     f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ") " ^ string_of_stmt
s
 | Synch(l, stmt) -> "synch " ^ l ^ " " ^ string_of_stmt stmt ^ "\n"
 | LoclBind(b) -> string_of_vdecl b
 | BlockEnd -> "BlockEnd\n"


let string_of_fdecl fdecl =
 string_of_typ fdecl.typ ^ " " ^
 fdecl.fname ^ "(" ^ String.concat ", " (List.map snd fdecl.formals) ^
 ")\n{\n" ^
 String.concat "" (List.map string_of_stmt fdecl.body) ^
 "}\n"
```

```
let string_of_progpart = function
| FuncDecl(f) -> string_of_fdecl f
| GlobBind(b) -> string_of_vdecl b
let string_of_program (progparts) =
 String.concat "" (List.map string_of_progpart progparts)
```

AST for GRACL

```
(* Ocamllex scanner for MicroC *)

{ open Graclparser }

let digit = ['0' - '9']
let digits = digit+

rule token = parse
 [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*"     { comment lexbuf }           (* Comments *)
| "//"     { slcomment lexbuf }
| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| '['      { LBRACK }
| ']'      { RBRACK }
| ';'      { SEMI }
| '.'      { DOT }
| ','      { COMMA }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| '%'      { MODULO }
| '='      { ASSIGN }
| "=="     { EQ }
| "!="     { NEQ }
| '<'      { LT }
| '>'      { GT }
| "<="     { LEQ }
| ">="     { GEQ }
| "&&"     { AND }
| "||"     { OR }
| "!"      { NOT }
```

```
| "if"     { IF }
| "else"   { ELSE }
| "for"    { FOR }
| "in"     { IN }
| "while"  { WHILE }
| "return" { RETURN }
| "hatch"  { HATCH }
| "synch"  { SYNCH }
| "int"    { INT }
| "bool"   { BOOL }
| "double" { DOUBLE }
| "void"   { VOID }
| "String" { STRING }
| "Node"   { NODE }
| "Edge"   { EDGE }
| "Graph"  { GRAPH }
| "EdgeList" { EDGELIST }
| "NodeList" { NODELIST }
| "IntTable" { INTTABLE }
| "DoubleTable" { DOUBLETABLE }
| "True"   { BLIT(true)  }
| "False"  { BLIT(false) }
| "infinity" { FLIT("infinity") }
| '\"'[^'\"']*'\"' as str { SLIT(String.sub str 1 ((String.length str) - 2))}
| digits as lxm { LITERAL(int_of_string lxm) }
| digits '.'  digit* ( ['e' 'E'] ['+' '-']? digits )? as lxm { FLIT(lxm) }
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']*     as lxm { ID(lxm) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

and comment = parse
 "*/" { token lexbuf }
| _    { comment lexbuf }

and slcomment = parse
 '\n' { token lexbuf }
| _ { slcomment lexbuf }
```

Scanner

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

open Ast
```

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

type sexpr = typ * sx
and sx =
   SLiteral of int
 | SFliteral of string
 | SBoolLit of bool
 | SSliteral of string
 | SId of string
 | SBinop of sexpr * op * sexpr
 | SUnop of uop * sexpr
 | SAssign of string * sexpr
 | SCall of string * sexpr list
 | SNoexpr

type sstmt =
   SBlock of sstmt list
 | SExpr of sexpr
 | SReturn of sexpr
 | SIf of sexpr * sstmt * sstmt
 | SFor of typ * string * sexpr * sstmt
 | SWhile of sexpr * sstmt
 | SHatch of sexpr * string * sexpr list * sstmt
 | SBlockEnd

type sbind =
   SDec of typ * string
 | SDecinit of typ * string * sexpr

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

type sprogram = sbind list * sfunc_decl list

(* Pretty-printing functions *)
```

```
let strip_sval = function
   SDec(a, b) -> (a, b)
 | SDecinit(a, b, _) -> (a, b)

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

let rec string_of_sstmt = function
   SBlock(stmts) ->
     "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "}\n"
 | SExpr(expr) -> string_of_sexpr expr ^ ";\n";
 | SReturn(expr) -> "return " ^ string_of_sexpr expr ^ ";\n";
 | SIf(e, s, SBlock([])) ->
     "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s
 | SIf(e, s1, s2) ->  "if (" ^ string_of_sexpr e ^ ")\n" ^
     string_of_sstmt s1 ^ "else\n" ^ string_of_sstmt s2
 | SWhile(e, s) -> "while (" ^ string_of_sexpr e ^ ") " ^ string_of_sstmt s
 | SFor(t, n, e, s) -> "for (" ^ string_of_typ t ^ " " ^ n ^ " in " ^ string_of_sexpr
e ^ ") " ^ string_of_sstmt s
 | SHatch(nle, fn, args, s) -> "hatch " ^ string_of_sexpr nle ^ " " ^ fn
         ^ "("  ^  String.concat  ",  "  (List.map  string_of_sexpr  args)  ^  ")  "  ^
string_of_sstmt s
 | SBlockEnd -> "SBlockEnd\n"

let string_of_svdecl = function
| SDec(t, id) -> string_of_typ t ^ " " ^ id ^ ";\n"
| SDecinit(t, id, e) -> string_of_typ t ^ " " ^ id ^ " = " ^ string_of_sexpr e ^ ";\n"
let string_of_sfdecl fdecl =
```

```
 string_of_typ fdecl.styp ^ " " ^
 fdecl.sfname ^ "(" ^ String.concat ", " (List.map snd fdecl.sformals) ^
 ")\n{\n" ^
 String.concat "" (List.map string_of_svdecl fdecl.slocals) ^
 String.concat "" (List.map string_of_sstmt fdecl.sbody) ^
 "}\n"

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

SAST for GRACL

```
(* Semantic checking for the GRACL compiler *)

open Ast
open Sast
module F = Functions
module StringMap = Map.Make(String)
module StringHash = Hashtbl.Make(struct  (* Consider changing? *)
 type t = string
 let equal x y = x = y
 let hash = Hashtbl.hash
end)

type symtable = (typ * string) list
let check (program) =
 (* Verify a symbol table has no void types or duplicate names *)
 let check_symbol_table (kind : string) (table : symtable) =
 List.iter
 (function (Void, b) -> raise (Failure ("illegal void " ^ kind ^ " " ^ b))
     | _ -> ()) table;
   let rec dups = function
       [] -> ()
     | ((_,n1) :: (_,n2) :: _) when n1 = n2 ->
   raise (Failure ("duplicate " ^ kind ^ " " ^ n1))
     | _ :: t -> dups t
   in dups (List.sort (fun (_,a) (_,b) -> compare a b) table)
 in

 (* Add function to the function map *)
 let update_function_table ft fd  =
```

```
  let built_in_err = "function " ^ fd.fname ^ " may not be redefined"
  and dup_err = "duplicate function " ^ fd.fname
  and make_err er = raise (Failure er)
  and n = fd.fname (* Name of the function *)
  in match fd with (* No duplicate functions or redefinitions of built-ins *)
        _ when StringMap.mem n F.function_decls -> make_err built_in_err
      | _ when StringMap.mem n ft -> make_err dup_err
      | _ ->  StringMap.add n fd ft
 in

 (* Add declaration to the global symbol list *)
 let update_global_table gt b = List.rev ((strip_val b):: List.rev gt)

 in

 let find_func s ft =
   try StringMap.find s ft
   with Not_found -> raise (Failure ("unrecognized function " ^ s))
 in

 let checkFunction func ft sl =
   let _ = check_symbol_table "global" sl; check_symbol_table "formal" func.formals in
(* Checks globals and formals when function is entered *)
   (* globals and formals, local / formal / global order of prio *)
   let top_level_symbols = List.fold_left (fun m (ty, name) -> StringMap.add name (ty,
name) m) StringMap.empty ( sl @ func.formals ) in

   let locals = StringHash.create 25 in

   let count = [|0|] in

   let symbol_table =  top_level_symbols :: [] in

   (* Raise an exception if the given rvalue type cannot be assigned to
      the given lvalue type *)
   let check_assign lvaluet rvaluet err =
      if lvaluet = rvaluet then lvaluet else raise (Failure err)
   in

   (* Return a variable from our local symbol table *)
   let rec type_of_identifier s = function
     | ht::st ->
```

```
      (try StringMap.find s ht
      with Not_found -> type_of_identifier s st)
    | [] -> raise (Failure ("undeclared identifier " ^ s))
  in

  (* Return a semantically-checked expression, i.e., with a type *)
  let rec expr st = function
      Literal  l -> (Int, SLiteral l)
    | Nodexpr    -> (Node, SId "_fakeNode")
    | Fliteral l -> (Double, SFliteral l)
    | Sliteral l -> (String, SSliteral l)
    | BoolLit l  -> (Bool, SBoolLit l)
    | Noexpr     -> (Void, SNoexpr)
    | Id s       -> let (typ, name) = type_of_identifier s st in (typ , SId name)
     | Access(table, node) -> let (tabletyp, tablename) = type_of_identifier table st
and (nodetyp, nodename) = type_of_identifier node st in
      let check_access = function
      Node -> let access_call = function
          | Inttable -> (Int, SCall("_getInt", [(Inttable, SId tablename); (Node, SId
nodename)]))
          | Doubletable -> (Double, SCall("_getDouble", [(Doubletable, SId tablename);
(Node, SId nodename)]))
           | _ -> raise(Failure ("Cannot treat " ^ string_of_typ tabletyp ^ " as an
IntTable/DoubleTable"))
        in access_call tabletyp
       | _ -> raise(Failure ("Cannot use " ^ string_of_typ nodetyp ^ " as keys in an
IntTable/DoubleTable"))
        in check_access nodetyp
    | Insert(table, node, ex) -> let (tabletyp, tablename) = type_of_identifier table
st and (nodetyp, nodename) = type_of_identifier node st in
      (match nodetyp with
      Node -> (let (extyp, ex') = expr st ex in
          let err = "illegal insertion, cannot put " ^ string_of_typ extyp ^ " " ^
string_of_expr ex ^ " in " ^ string_of_typ tabletyp  in
        match tabletyp with
          | Inttable -> (Void, SCall("_insertInt", [(Inttable, SId tablename); (Node,
SId nodename); (check_assign Int extyp err, ex')]))
         | Doubletable -> (Void, SCall("_insertDouble", [(Doubletable, SId tablename);
(Node, SId nodename); (check_assign Double extyp err, ex')]))
           | _ -> raise(Failure ("Cannot treat " ^ string_of_typ tabletyp ^ " as an
IntTable/DoubleTable")))
```

```
        | _ -> raise(Failure ("Cannot use " ^ string_of_typ nodetyp ^ " as keys in an
IntTable/DoubleTable")))
    | Assign(var, e) as ex ->
        let (lt, name) = type_of_identifier var st
        and (rt, e') = expr st e in
        let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
          string_of_typ rt ^ " in " ^ string_of_expr ex
        in (check_assign lt rt err, SAssign(name, (rt, e')))
    | Unop(op, e) as ex ->
        let (t, e') = expr st e in
        let ty = match op with
          Neg when t = Int || t = Double -> t
        | Not when t = Bool -> Bool
        | _ -> raise (Failure ("illegal unary operator " ^
                               string_of_uop op ^ string_of_typ t ^
                               " in " ^ string_of_expr ex))
        in (ty, SUnop(op, (t, e')))
    | Binop(e1, op, e2) as e ->
        let (t1, e1') = expr st e1
        and (t2, e2') = expr st e2 in
        (* All binary operators require operands of the same type *)
        let same = t1 = t2 in
        (* Determine expression type based on operator and operand types *)
        let ty = match op with
          Add | Sub | Mult | Div | Mod when same && t1 = Int   -> Int
        | Add | Sub | Mult | Div when same && t1 = Double -> Double
         | Equal | Neq          when same && (t1 = Int || t1 = Double || t1 = Bool) ->
Bool
        | Less | Leq | Great | Geq
                   when same && (t1 = Int || t1 = Double) -> Bool
        | And | Or when same && t1 = Bool -> Bool
        | _ -> raise (
      Failure ("illegal binary operator " ^
                    string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
                    string_of_typ t2 ^ " in " ^ string_of_expr e))
        in (ty, SBinop((t1, e1'), op, (t2, e2')))
    | Call(fname, args) as call ->
        let fd = find_func fname ft in
        let param_length = List.length fd.formals in
        if List.length args != param_length then
          raise (Failure ("expecting " ^ string_of_int param_length ^
                          " arguments in " ^ string_of_expr call))
```

```
        else let check_call (ft, _) e =
          let (et, e') = expr st e in
          let err = "illegal argument found " ^ string_of_typ et ^
            " expected " ^ string_of_typ ft ^ " in " ^ string_of_expr e
          in (check_assign ft et err, e')
        in
        let args' = List.map2 check_call fd.formals args
        in (fd.typ, SCall(fname, args'))
   in

   let check_bool_expr st e =
     let (t', e') = expr st e
     and err = "expected Boolean expression in " ^ string_of_expr e
     in if t' != Bool then raise (Failure err) else (t', e')
   in

   (* Return a semantically-checked statement i.e. containing sexprs *)
   let rec check_stmt (st : (Ast.typ*string) StringMap.t list)  = function
       Expr e -> SExpr (expr st e)
     | If(p, b1, b2) -> SIf(check_bool_expr st p, check_stmt st b1, check_stmt st b2)
     | While(p, s) -> SWhile(check_bool_expr st p, check_stmt st s)
     | Return e -> let (t, e') = expr st e in
       if t = func.typ then SReturn (t, e')
       else raise (
         Failure ("return gives " ^ string_of_typ t ^ " expected " ^
         string_of_typ func.typ ^ " in " ^ string_of_expr e))
     | For(t, _, e, s) -> let (lt, _) as lexpr = expr st e in
       let ss = check_stmt st s in
       let get_name = function
       | SBlock(SExpr(_, SId name)::_) -> name
       | _ -> raise (Failure("internal error: new name for loop not found")) in
       let check_list = function
       | Nodelist when t = Node -> SFor(t, get_name ss, lexpr, ss)
       | Edgelist when t = Edge -> SFor(t, get_name ss, lexpr, ss)
        | Nodelist | Edgelist as ltyp -> raise (Failure ("Cannot use " ^ string_of_typ
t ^ " loop to iterate over " ^ string_of_typ ltyp))
         | _ as badt -> raise (Failure ("Cannot use for loop to iterate over " ^
string_of_typ badt))
       in check_list lt
     | Hatch(nle, fn, args, s) -> let (nlt, _) as nlexpr = expr st nle in
       let check_hatch = function
```

```
        | Nodelist -> let _ = expr st (Call(fn, Nodexpr::args)) in SHatch(nlexpr, fn,
List.map (expr st) args, check_stmt st s)
        | _ -> raise (Failure ("Hatch must given a NodeList, not a " ^ string_of_typ
nlt))
       in check_hatch nlt
     | Block sl -> let st = StringMap.empty::st in
         let rec check_stmt_list st = function
             [Return _ as s] -> [check_stmt st s]
           | Return _ :: _   -> raise (Failure "nothing may follow a return")
             | Block sl :: ss  -> check_stmt_list (StringMap.empty::st) (sl @ ss) (*
Flatten blocks *)
           | LoclBind(b) as lb :: ss -> let add_local typ name = if StringMap.mem name
(List.hd st) then raise (Failure ("Cannot redeclare " ^ name))
              else StringMap.add name (typ, "var" ^ string_of_int (count.(0)) ^ "_" ^
name) (List.hd st) and (t,n) = strip_val b and _ = count.(0) <- 1 + count.(0) in
             let updated_table = (add_local t n)::(List.tl st) in
                   let stm = check_stmt updated_table lb in stm :: check_stmt_list
updated_table ss
           | BlockEnd :: ss -> SBlockEnd::check_stmt_list (List.tl st) ss
           | s :: ss         -> let stm = check_stmt st s in stm :: check_stmt_list st
ss (* stm is VERY important here *)
           | []              -> []
         in SBlock(check_stmt_list st sl)

     | LoclBind(b) ->
     begin match b with
       | Dec(t,n) -> if t = Void then raise (Failure ("illegal void local " ^ n)) else
            let (_, newname) = type_of_identifier n st in StringHash.replace locals
newname (SDec(t,newname)); SExpr(t, SId newname)
       | Decinit(t,n,e) as di ->
       if t = Void then raise (Failure ("illegal void local " ^ n)) else
       let (rt, ex) = expr st e in
         let err = "illegal assignment " ^ string_of_typ t ^ " = " ^
           string_of_typ rt ^ " in " ^ string_of_vdecl di
         in let _ = check_assign t rt err and (_, newname) = type_of_identifier n st
              in StringHash.replace locals newname (SDecinit(t, newname, (rt, ex)));
SExpr(t, SAssign(newname, (rt, ex))) end

     | BlockEnd -> (* Should have been handled by Block *)
       raise (Failure ("internal error: block end mishandled?"))

     | Synch(id, stmt) -> let (typ, name) = type_of_identifier id st in
```

```
      match typ with
                | Node | Edge | Nodelist | Edgelist -> SBlock(SExpr((Int,
SCall(("_synch_start", [(typ, SId(name))])))) :: check_stmt st stmt :: [SExpr(Int,
SCall("_synch_end", [(typ, SId(name))]))])
         | _ -> raise (Failure ("Cannot synch with " ^ string_of_typ typ ^ " type
variable " ^ id))


   in (* body of check_function *)
   { styp = func.typ;
     sfname = func.fname;
     sformals = func.formals;
      slocals  = begin let locallist = (StringHash.fold (fun _ b lt -> b::lt) locals
[]) in
     let _ = check_symbol_table "local" (List.map strip_sval locallist)
       in locallist end;
     sbody = match check_stmt symbol_table (Block func.body) with
 SBlock(sl) -> sl
     | _ -> raise (Failure ("internal error: block didn't become a block?"))
   }
in

let checkGlobal =
   let rec constexpr = function
       Literal  l -> (Int, SLiteral l)
     | Fliteral l -> (Double, SFliteral l)
     | Sliteral l -> (String, SSliteral l)
     | BoolLit l  -> (Bool, SBoolLit l)
     | Unop(op, e) as ex ->
         let (t, e') = constexpr e in
         let ty = match op with
           Neg when t = Int || t = Double -> t
         | Not when t = Bool -> Bool
         | _ -> raise (Failure ("illegal unary operator " ^
                               string_of_uop op ^ string_of_typ t ^
                               " in " ^ string_of_expr ex))
         in (ty, SUnop(op, (t, e')))
     | Binop(e1, op, e2) as e ->
         let (t1, e1') = constexpr e1
         and (t2, e2') = constexpr e2 in
         (* All binary operators require operands of the same type *)
         let same = t1 = t2 in
```

```
         (* Determine expression type based on operator and operand types *)
         let ty = match op with
           Add | Sub | Mult | Div | Mod when same && t1 = Int    -> Int
         | Add | Sub | Mult | Div when same && t1 = Double -> Double
          | Equal | Neq          when same && (t1 = Int || t1 = Double || t1 = Bool) ->
Bool
         | Less | Leq | Great | Geq
                   when same && (t1 = Int || t1 = Double) -> Bool
         | And | Or when same && t1 = Bool -> Bool
         | _ -> raise (
       Failure ("illegal binary operator " ^
                      string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
                      string_of_typ t2 ^ " in " ^ string_of_expr e))
         in (ty, SBinop((t1, e1'), op, (t2, e2')))
      | _ as ex -> raise (Failure ("initializer element " ^ string_of_expr ex ^ " is
not a compile time constant"))
   in function
 | Decinit(t, n, e) as di ->
       let (rt, ex) = constexpr e
       and check_assign lvaluet rvaluet err =
      if lvaluet = rvaluet then lvaluet else raise (Failure err)
       in
         let err = "illegal assignment " ^ string_of_typ t ^ " = " ^
           string_of_typ rt ^ " in " ^ string_of_vdecl di
           in let _ = check_assign t rt err
         in SDecinit(t, n, (rt, ex))
 | Dec(t,n) -> SDec(t,n)
 in

 let rec semant_check progparts ft (sl : symtable) (globs, funcs) =
   match progparts with
     | [] -> ignore(find_func "main" ft); let _ = check_symbol_table "global" sl in
(globs, funcs)
    | FuncDecl(f)::ps -> let new_funcs = (update_function_table ft f) in semant_check
ps new_funcs sl (globs, List.rev (checkFunction f new_funcs sl :: List.rev funcs))
     | GlobBind(b)::ps  -> semant_check ps ft (update_global_table sl b) (List.rev
(checkGlobal b :: List.rev globs), funcs)

in semant_check program F.function_decls [] ([], [])
```

Semant.ml

```
(* Top-level of the GRACL compiler: scan & parse the input,
```

```
  check the resulting AST and generate an SAST from it, generate LLVM IR,
  and dump the module, borrowed from MicroC *)

type action = Ast | Sast | LLVM_IR | Compile

let () =
 let action = ref Compile in
 let set_action a () = action := a in
 let speclist = [
   ("-a", Arg.Unit (set_action Ast), "Print the AST");
   ("-s", Arg.Unit (set_action Sast), "Print the SAST");
   ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM IR");
   ("-c", Arg.Unit (set_action Compile),
     "Check and print the generated LLVM IR (default)");
 ] in
 let usage_msg = "usage: ./gracl.native [-a|-s|-l|-c] [file.grc]" in
 let channel = ref stdin in
 Arg.parse speclist (fun filename -> channel := open_in filename) usage_msg;
  let lexbuf = Lexing.from_channel !channel in
 let ast = Graclparser.program Scanner.token lexbuf in
 match !action with
   Ast -> print_string (Ast.string_of_program ast)
 | _ -> let sast = Semant.check ast in
   match !action with
     Ast     -> ()
   | Sast    -> print_string (Sast.string_of_sprogram sast)
   | LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate sast))
   | Compile -> let m = Codegen.translate sast in
 Llvm_analysis.assert_valid_module m;
 print_string (Llvm.string_of_llmodule m)
```

Gracl.ml that generates the .native

```
/* Ocamlyacc parser for GRACL */

%{
open Ast
%}

%token  SEMI  LPAREN  RPAREN  LBRACE  RBRACE  LBRACK  RBRACK  COMMA  DOT  PLUS  MINUS  TIMES
DIVIDE MODULO ASSIGN
%token NOT EQ LT LEQ GT GEQ AND OR NEQ
```

```
%token RETURN IF ELSE FOR WHILE IN HATCH SYNCH
%token  INT  BOOL  DOUBLE  VOID  STRING  NODE  EDGE  GRAPH  EDGELIST  NODELIST  INTTABLE
DOUBLETABLE
%token <int> LITERAL
%token <bool> BLIT
%token <string> ID FLIT SLIT
%token EOF

%start program
%type <Ast.program> program

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT LEQ GT GEQ
%nonassoc HATCH SYNCH
%left PLUS MINUS
%left TIMES DIVIDE MODULO
%right NOT
%left DOT

%%

program:
 progparts EOF { List.rev $1 }

progparts:
  /* nothing */ { ([])         }
| progparts vdecl { (GlobBind($2) :: $1) }
| progparts fdecl { (FuncDecl($2) :: $1) }

fdecl:
  typ ID LPAREN formals_opt RPAREN LBRACE func_body RBRACE
    { { typ = $1;
  fname = $2;
  formals = List.rev $4;
  body = List.rev ( $7 ) } }

func_body:
```

```
   /* nothing */     { ([])                }
 | func_body stmt    { ($2 :: $1) }

formals_opt:
   /* nothing */ { [] }
 | formal_list   { $1 }

formal_list:
   typ ID                   { [($1,$2)]     }
 | formal_list COMMA typ ID { ($3,$4) :: $1 }

typ:
   INT   { Int   }
 | BOOL  { Bool  }
 | DOUBLE { Double }
 | VOID  { Void  }
 | STRING { String }
 | GRAPH { Graph }
 | NODE { Node }
 | EDGE { Edge }
 | INTTABLE { Inttable }
 | DOUBLETABLE { Doubletable }
 | NODELIST { Nodelist }
 | EDGELIST { Edgelist }

vdecl:
  typ ID SEMI { Dec($1, $2) }
  | typ ID ASSIGN expr SEMI { Decinit($1, $2, $4) }

stmt_list:
   /* nothing */  { [] }
 | stmt_list stmt { $2 :: $1 }

stmt:
   expr SEMI                                        { Expr $1               }
 | RETURN expr_opt SEMI                             { Return $2             }
  | LBRACE stmt_list RBRACE                                    { Block(List.rev
(BlockEnd::$2))     }
 | IF LPAREN expr RPAREN stmt %prec NOELSE          { If($3, $5, Block([]))  }
 | IF LPAREN expr RPAREN stmt ELSE stmt             { If($3, $5, $7)         }
  | FOR LPAREN typ ID IN expr RPAREN stmt                        { For($3, $4, $6,
Block(LoclBind(Dec($3, $4))::$8::[BlockEnd]))    }
```

```
 | WHILE LPAREN expr RPAREN stmt                          { While($3, $5)          }
 | HATCH expr ID LPAREN args_opt RPAREN stmt             { Hatch($2, $3, $5, $7)  }
 | SYNCH ID stmt                                        { Synch($2, $3)          }
 | vdecl                                                { LoclBind($1)           }

expr_opt:
   /* nothing */ { Noexpr }
 | expr          { $1 }

expr:
   LITERAL          { Literal($1)                    }
 | FLIT             { Fliteral($1)                   }
 | BLIT             { BoolLit($1)                    }
 | SLIT             { Sliteral($1)                   }
 | ID               { Id($1)                         }
 | expr PLUS   expr { Binop($1, Add,   $3)           }
 | expr MINUS  expr { Binop($1, Sub,   $3)           }
 | expr TIMES  expr { Binop($1, Mult,  $3)           }
 | expr DIVIDE expr { Binop($1, Div,   $3)           }
 | expr MODULO expr { Binop($1, Mod,   $3)           }
 | expr EQ     expr { Binop($1, Equal, $3)           }
 | expr NEQ    expr { Binop($1, Neq,   $3)           }
 | expr LT     expr { Binop($1, Less,  $3)           }
 | expr GT     expr { Binop($1, Great, $3)           }
 | expr LEQ    expr { Binop($1, Leq,   $3)           }
 | expr GEQ    expr { Binop($1, Geq,   $3)           }
 | expr AND    expr { Binop($1, And,   $3)           }
 | expr OR     expr { Binop($1, Or,    $3)           }
 | MINUS expr %prec NOT { Unop(Neg, $2)              }
 | NOT expr         { Unop(Not, $2)                  }
 | ID ASSIGN expr   { Assign($1, $3)                 }
 | ID LPAREN args_opt RPAREN { Call($1, $3)          }
 | call_train ID LPAREN args_opt RPAREN { Call($2, $1 @ $4)}
 | ID LBRACK ID RBRACK { Access($1, $3)              }
 | ID LBRACK ID RBRACK ASSIGN expr { Insert($1, $3, $6) }
 | LPAREN expr RPAREN { $2                           }

call_train:
   call_train ID LPAREN args_opt RPAREN DOT { [Call($2, $1 @ $4)] }
 | ID DOT { [Id($1)] }
 | ID LPAREN args_opt RPAREN DOT { [Call($1, $3)]}
```

```
args_opt:
   /* nothing */ { [] }
 | args_list  { List.rev $1 }

args_list:
   expr                    { [$1] }
 | args_list COMMA expr { $3 :: $1 }
```

Graclparser.mly

```
#!/bin/sh

# clean the directory and make the targets again
make clean
make all

out_dir="outputs"

for i in $(find . -perm +111 -type f);
   do $i > "$out_dir"/"$i"_output.txt
   if cmp -s "$out_dir"/"$i"_output.txt "$out_dir"/"$i"_example.txt; then
       echo " $i ....SUCCESS"
   else
       echo " $i ....FAILURE"
   fi
done
```

test-script.sh

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <string.h>
#include <unistd.h>
#define BUILDSTDLIB
#include "./cfiles/lockedobject.h"
#include "./cfiles/node.c"
#include "./cfiles/edge.c"
#include "./cfiles/nodelist.c"
#include "./cfiles/edgelist.c"
#include "./cfiles/graph.c"
#include "./cfiles/concurrency.c"
#include "./cfiles/doubletable.c"
```

```
#include "./cfiles/inttable.c"
```

Graclstdlib.c

```
CC = gcc
CFLAGS = -g -Wall

.PHONY: all
all: node edge edgelist nodelist graph concurrency inttable doubletable

.PHONY: node
node: node.c
	$(CC) $(CFLAGS) -pthread -o node node.c

.PHONY: edge
edge: edge.c
	$(CC) $(CFLAGS) -pthread -o edge edge.c

.PHONY: edgelist
edgelist: edgelist.c
	$(CC) $(CFLAGS) -pthread -o edgelist edgelist.c

.PHONY: nodelist
nodelist: nodelist.c
	$(CC) $(CFLAGS) -pthread -o nodelist nodelist.c

.PHONY: graph
graph: graph.c
	$(CC) $(CFLAGS) -pthread -o graph graph.c

.PHONY: concurrency
concurrency: concurrency.c
	$(CC) $(CFLAGS) -pthread -o concurrency concurrency.c

.PHONY: inttable
inttable: inttable.c
	$(CC) $(CFLAGS) -pthread -o inttable inttable.c

.PHONY: doubletable
doubletable: doubletable.c
	$(CC) $(CFLAGS) -pthread -o doubletable doubletable.c

.PHONY: clean
```

```
clean:
	rm -f *.o node edge edgelist nodelist graph inttable doubletable concurrency
```

Makefile for the C files

```
(* Built in function library helpers *)

open Ast
module StringMap = Map.Make(String)
let function_decls =
  let add_func map (typ, name, (formallist : formal list)) = StringMap.add name {
    typ = typ;
    fname = name;
    formals = formallist;
    body = [] } map in List.fold_left add_func StringMap.empty [
      (* General Functions *)
      (Void, "print", [String, "s"]);
      (Void, "printi", [Int, "i"]);
      (Graph, "createGraph", [Int, "i"]);
      (Nodelist, "createNodeList", []);
      (Edgelist, "createEdgeList", []);
      (Doubletable, "createDoubleTable", [Int, "i"]);
      (Inttable, "createIntTable", [Int, "i"]);

      (* Casting Functions *)
      (String, "doubleToString", [Double, "d"]);
      (Double, "intToDouble", [Int, "i"]);

      (* String Functions *)
      (Int, "stringLength", [String, "s"]);
      (Bool, "stringEquals", [(String, "s1"); (String, "s2")]);

      (* Node Functions *)
      (String, "data", [Node, "n"]);
      (Nodelist, "neighbors", [Node, "n"]);
      (Edgelist, "edges", [Node, "n"]);
      (Bool, "visited", [Node, "n"]);
      (Node, "updateData", [(Node, "n"); (String, "s")]);
      (Node, "updateVisited", [(Node, "n"); (Bool, "b")]);
      (Double, "cost", [Node, "n"]);
      (Double, "incrementCost", [Node, "n"]);
      (Double, "decrementCost", [Node, "n"]);
```

```
        (Double, "updateCost", [(Node, "n1"); (Double, "i")]);
        (Node, "prec", [Node, "n"]);
        (Node, "setPrec", [(Node, "n1"); (Node, "n2")]);
        (Bool, "nodeEquals", [(Node, "n1"); (Node, "n2")]);
        (Edge, "getEdge", [(Node, "n1"); (Node, "n2")]);

        (* Edge Functions *)
        (Void, "updateEdge", [(Edge, "e"); (Double, "d")]);
        (Node, "start", [Edge, "e"]);
        (Node, "end", [Edge, "e"]);
        (Double, "weight", [Edge, "e"]);
        (Bool, "edgeEquals", [(Edge, "e1"); (Edge, "e2")]);

        (* Graph Functions *)
        (Nodelist, "nodes", [Graph, "g"]);
        (Node, "createNode", [(Graph, "g"); (String, "s")]);
        (Int, "removeNodeGraph", [(Graph, "g"); (Node, "n")]);
           (Edge, "addEdge", [(Graph, "g"); (Node, "start"); (Node, "end"); (Double,
"weight")]);
        (Int,  "removeEdgeGraph", [(Graph, "g"); (Edge, "e")]);

        (* Nodelist Functions *)
        (Node, "head_NL", [Nodelist, "nl"]);
        (Node, "tail_NL", [Nodelist, "nl"]);
        (Int, "length_NL", [Nodelist, "nl"]);
        (Bool, "empty_NL", [Nodelist, "nl"]);
        (Int, "removeNode", [(Nodelist, "nl"); (Node, "n")]);
        (Node, "removeFirst_NL", [Nodelist, "nl"]);
        (Node, "removeLast_NL", [Nodelist, "nl"]);
        (Void, "appendNode", [(Nodelist, "nl"); (Node, "n")]);
        (Void, "prependNode", [(Nodelist, "nl"); (Node, "n")]);
        (Bool, "includesNode", [(Nodelist, "nl"); (Node, "n")]);
        (Nodelist, "neighbors", [(Node, "node")]);

        (* Edgelist Functions *)
        (Edge, "head_EL", [Edgelist, "el"]);
        (Edge, "tail_EL", [Edgelist, "el"]);
        (Int, "length_EL", [Edgelist, "el"]);
        (Bool, "empty_EL", [Edgelist, "el"]);
        (Int, "removeEdge", [(Edgelist, "el"); (Edge, "e")]);
        (Edge, "removeFirst_EL", [Edgelist, "el"]);
        (Edge, "removeLast_EL", [Edgelist, "el"]);
```

```
      (Void, "appendEdge", [(Edgelist, "el"); (Edge, "e")]);
      (Void, "prependEdge", [(Edgelist, "el"); (Edge, "e")]);

      (* Inttable Functions *)
      (Int, "hashCode_it", [(Inttable, "it"); (Node, "n")]);
      (Int, "_getInt", [(Inttable, "it"); (Node, "n")]);
      (Void, "_insertInt", [(Inttable, "it"); (Node, "n"); (Int, "i")]);
      (Nodelist, "intKeys", [Inttable, "it"]);
      (Int, "deleteInt", [(Inttable, "it"); (Node, "n")]);
      (Bool, "inInt", [(Inttable, "it"); (Node, "n")]);

      (* Doubletable Functions *)
      (Int, "hashCode_it", [(Doubletable, "dt"); (Node, "n")]);
      (Double, "_getDouble", [(Doubletable, "dt"); (Node, "n")]);
      (Void, "_insertDouble", [(Doubletable, "dt"); (Node, "n"); (Double, "d")]);
      (Nodelist, "doubleKeys", [Doubletable, "dt"]);
      (Int, "deleteDouble", [(Doubletable, "dt"); (Node, "n")]);
      (Bool, "inDouble", [(Doubletable, "dt"); (Node, "n")]);
   ]
```

Functions.ml that codegen calls

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

int graph_id = 1;

struct Node {
   pthread_mutex_t lock;
   int id; // Only used under the hood
   char *data;
   bool visited;
   struct EdgeList* edges;
   struct Node* precursor;
   double cost;
   int parent_graph_id;
   bool deleted;
};
```

```
struct Edge {
   pthread_mutex_t lock;
   double weight;
   struct Node* start;
   struct Node* end;
   bool deleted;
};

struct EdgeListItem {
   struct Edge* edge;
   struct EdgeListItem* next;
   struct EdgeListItem* prev;
};

struct EdgeList {
   pthread_mutex_t lock;
   struct EdgeListItem* head;
   struct EdgeListItem* tail;
};

struct NodeListItem {
   struct Node* node;
   struct NodeListItem* next;
   struct NodeListItem* prev;
};

struct NodeList {
   pthread_mutex_t lock;
   struct NodeListItem* head;
   struct NodeListItem* tail;
};

struct DataItem {
   struct Node* key;
   struct EdgeList* value;
};

struct Graph
{
   struct DataItem* hashArray;
   struct NodeList* nodes;
   int size;
```

```
    int id_num;
    int graph_id_local;
    int occupied;
};

struct IntTableItem {
    struct IntTableItem* next;
    struct IntTableLLItem* entry;
};

struct IntTableLLItem {
    struct Node* key;
    int value;
};

struct DoubleTableItem {
    struct DoubleTableItem* next;
    struct DoubleTableLLItem* entry;
};

struct DoubleTableLLItem {
    struct Node* key;
    double dub;
};

struct IntTable
{
    pthread_mutex_t lock;
    struct IntTableItem* arr;
    struct NodeList* keys;
    int size;
    int graph_id;
};

struct DoubleTable
{
    pthread_mutex_t lock;
    struct DoubleTableItem* arr;
    struct NodeList* keys;
    int size;
    int doubleId;
    int graph_id;
```

```
};
```

Lockedobject.h which has the definitions of all objects written in C

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

struct Lockable{
   pthread_mutex_t pm;
};

pthread_mutex_t* _convert_to_lockable(void* obj){
   struct Lockable* new;
   new = (struct Lockable*) obj;
   return &(new->pm);
}

int synch_start(pthread_mutex_t* lock) {
   if (pthread_mutex_lock(lock) != 0) {
       printf("\n locking failed\n");
       return -1;
   }
   return 0;
}

int _synch_start(void* obj){
   return synch_start(_convert_to_lockable(obj));
}

int synch_end(pthread_mutex_t* lock) {
   if(pthread_mutex_unlock(lock) != 0) {
       printf("\n mutex is not valid, unlock failed\n");
       return -1;
   }
   return 0;
}

int _synch_end(void* obj){
   return synch_end(_convert_to_lockable(obj));
```

```
}

int synch_destroy(pthread_mutex_t* lock) {
   if (pthread_mutex_destroy(lock) != 0) {
       printf("\n mutex is not valid, destroy failed\n");
       return -1;
   }
   return 0;
}

int hatch_end(pthread_t* thread_arr, int len) {
   int j;
   for (j = 0; j < len; j++) {
       pthread_join(thread_arr[j], NULL);
   }
   return 0;
}
```

Concurrency.c

```
#include <pthread.h>
#include <stdlib.h>
#include <string.h>
#ifndef BUILDSTDLIB
#include "graph.h"
#endif

struct DoubleTable* createDoubleTable(int predicted_size);

int hashCode_dt(struct DoubleTable* dt, struct Node* node);

double getDouble(struct DoubleTable* dt, struct Node* n);

struct DoubleTableLLItem* createDoubleTableLLItem(struct Node* n, double data);

struct DoubleTableItem* createDoubleTableItem(struct Node* n, double data);

// technically complexity could improve if we insert in a sorted manner
// TODO
void insertDouble(struct DoubleTable* dt, struct Node* n, double data);

struct NodeList* doubleKeys(struct DoubleTable* dt);
```

```
bool inDouble(struct DoubleTable* dt, struct Node* n);

int deleteDouble(struct DoubleTable* dt, struct Node* n);
```

doubletable.h

```
#include "doubletable.h"

struct DoubleTable* createDoubleTable(int predicted_size) {
   if (predicted_size <= 0) {
       fprintf(stderr, "Error: DoubleTable must have be at least size 1 or larger\n");
       exit(1);
   }
   struct DoubleTable* dt = malloc(sizeof(struct DoubleTable));
        dt->arr   =   (struct   DoubleTableItem   *)calloc(predicted_size,   sizeof(struct
DoubleTableItem));
   dt->size = predicted_size;
   dt->keys = createNodeList();
   dt->graph_id = -1;
   if (pthread_mutex_init(&dt->lock, NULL) !=0) {
       fprintf(stderr, "createDoubleTable: Failure to initialize mutex\n");
       exit(1);
   }
   return dt;
}

int hashCode_dt(struct DoubleTable* dt, struct Node* n) {
   int id_node = n->id;
   return id_node % dt->size;
}

double _getDouble(struct DoubleTable* dt, struct Node* n) {
   int hashIndex = hashCode_dt(dt, n);
   struct DoubleTableItem* start;
   start = &dt->arr[hashIndex];
   while (start) {
       if ((start->entry) && (start->entry->key->id == n->id)) {
           return start->entry->dub;
       }
       else {
           start = start->next;
```

```
        }
    }
    exit(1);
}

struct DoubleTableLLItem* createDoubleTableLLItem(struct Node* n, double data) {
    struct DoubleTableLLItem* dubtab = malloc(sizeof(struct DoubleTableLLItem));
    dubtab->key = n;
    dubtab->dub = data;
    return dubtab;
}

struct DoubleTableItem* createDoubleTableItem(struct Node* n, double data) {
    struct DoubleTableItem* dubtab = malloc(sizeof(struct DoubleTableItem));
    dubtab->entry = createDoubleTableLLItem(n, data);
    dubtab->next = NULL;
    return dubtab;
}

// technically complexity could improve if we insert in a sorted manner
// TODO
void _insertDouble(struct DoubleTable* dt, struct Node* n, double data) {
    if (n->deleted) {
        fprintf(stderr, "_insertInt: Node deleted\n");
        exit(1);
    }
    if ((dt->graph_id != -1) && (dt->graph_id != n->parent_graph_id)) {
         fprintf(stderr, "_insertDouble error: Cannot insert nodes from different graphs
into the same DoubleTable\n");
        exit(1);
    }
    else if (dt->graph_id == -1) {
        dt->graph_id = n->parent_graph_id;
    }
    int hashIndex = hashCode_dt(dt, n);
    struct DoubleTableItem* start = &dt->arr[hashIndex];
    if (start == NULL) {
        dt->arr[hashIndex] = *createDoubleTableItem(n, data);
    }
    else {
        while (start && start->next) {
            start = start->next;
```

```
        }
        start->next = createDoubleTableItem(n, data);
    }
    appendNode(dt->keys, n);
    return;
}

struct NodeList* doubleKeys(struct DoubleTable* dt){
    return dt->keys;
}

bool inDouble(struct DoubleTable* dt, struct Node* n) {
    int hashIndex = hashCode_dt(dt, n);
    struct DoubleTableItem* start;
    start = &dt->arr[hashIndex];
    while (start) {
        if (start->entry && start->entry->key->id == n->id) {
            return true;
        }
        else {
            start = start->next;
        }
    }
    return false;
}

int deleteDouble(struct DoubleTable* dt, struct Node* n) {
    // remove it from keys
    struct NodeList* all_nodes = malloc(sizeof(struct NodeList));
    all_nodes = doubleKeys(dt);
    removeNode(all_nodes, n);

    int hashIndex = hashCode_dt(dt, n);
    struct DoubleTableItem* start;
    struct DoubleTableItem* prev;
    start = &dt->arr[hashIndex];
    prev = NULL;
    while (start) {
        // find node to delete
        if (start->entry && start->entry->key->id == n->id) {
            if (prev == NULL) { // start of list
                dt->arr[hashIndex] = *start->next;
```

```
            }
            else if (start->next == NULL) { // end of list
                prev->next = NULL;
            }
            else {
                prev->next = start->next;
            }
            return 0;
        }
        else {
            prev = start;
            start = start->next;
        }
    }
    return 0;
}
```

Doubletable.c

```
#include <stdlib.h>
#include <stdio.h>
#ifndef BUILDSTDLIB
#endif

void updateEdge(struct Edge* edge, double new_weight);

double weight(struct Edge* edge);

bool edgeEquals(struct Edge* e1, struct Edge* e2);

/* New start and end accessors */
struct Node* start(struct Edge* edge);

struct Node* end(struct Edge* edge);
```

Edge.h

```
#include "edge.h"

void updateEdge(struct Edge* edge, double new_weight) {
    if (edge->deleted) {
        fprintf(stderr, "updateEdge: Edge deleted\n");
```

```
        exit(1);
    }
    edge->weight = new_weight;
    return;
}

double weight(struct Edge* edge) {
    if (edge->deleted) {
        fprintf(stderr, "weight: Edge deleted\n");
        exit(1);
    }
    return edge->weight;
}

bool edgeEquals(struct Edge* e1, struct Edge* e2) {
    if ((e1->deleted) || (e2->deleted)) {
        fprintf(stderr, "edgeEquals: Edge deleted\n");
        exit(1);
    }
      return  (((e1->weight  ==  e2->weight)  &&  nodeEquals(e1->start,  e2->start))  &&
nodeEquals(e1->end, e2->end));
}

/* New start and end accessors */
struct Node* start(struct Edge* edge) {
    if (edge->deleted) {
        fprintf(stderr, "start: Edge deleted\n");
        exit(1);
    }
    return edge->start;
}

struct Node* end(struct Edge* edge) {
    if (edge->deleted) {
        fprintf(stderr, "end: Edge deleted\n");
        exit(1);
    }
    return edge->end;
}
```

Edge.c

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

```
#include <string.h>
#ifndef BUILDSTDLIB
#include "lockedobject.h"
#endif

struct EdgeList* createEdgeList();

struct EdgeListItem* createEdgeListItem(struct Edge* e);

struct Edge* head_EL(struct EdgeList* edge_list);

struct Edge* tail_EL(struct EdgeList* edge_list);

struct Edge* removeFirst_EL(struct EdgeList* edge_list);

struct Edge* removeLast_EL(struct EdgeList* edge_list);

void prependEdge(struct EdgeList* edge_list, struct Edge* e);

bool empty_EL(struct EdgeList* edge_list);

int length_EL(struct EdgeList* edge_list);

int removeEdge(struct EdgeList* edge_list, struct Edge* e);

void appendEdge(struct EdgeList* edge_list, struct Edge* e);
```

edgelist.h

```
#include "edgelist.h"

struct EdgeList* createEdgeList() {
   struct EdgeList* edge_list = malloc(sizeof(struct EdgeList));
   edge_list->head = NULL;
   edge_list->tail = NULL;
   if (pthread_mutex_init(&edge_list->lock, NULL) !=0) {
       fprintf(stderr, "createEdgeList: Failure to initialize mutex\n");
       exit(1);
   }
   return edge_list;
}
```

```
struct EdgeListItem* createEdgeListItem(struct Edge* e) {
   struct EdgeListItem* item = malloc(sizeof(struct EdgeListItem));
   item->edge = e;
   item->next = NULL;
   item->prev = NULL;
   return item;
}

struct Edge* head_EL(struct EdgeList* edge_list) {
   if (!edge_list->head) {
       return NULL;
   }
   return edge_list->head->edge;
}

struct Edge* tail_EL(struct EdgeList* edge_list) {
   if (!edge_list->tail) {
       return NULL;
   }
   return edge_list->tail->edge;
}

struct Edge* removeFirst_EL(struct EdgeList* edge_list) {
   struct EdgeListItem *head;
   head = edge_list->head;
   if (head) {
       edge_list->head = head->next;
       if(edge_list->head){
           edge_list->head->prev = NULL;
       }
       return head->edge;
   } else {
       return NULL;
   }
}

struct Edge* removeLast_EL(struct EdgeList* edge_list) {
   struct EdgeListItem *last = edge_list->tail;
   struct EdgeListItem *prev = NULL;
   if (last) {
       prev = last->prev;
       edge_list->tail = prev;
```

```
        if (prev) {
            prev->next = NULL;
        }
        return last->edge;
    } else {
        return NULL;
    }
}

void prependEdge(struct EdgeList* edge_list, struct Edge* e) {
    if (e->deleted) {
        fprintf(stderr, "prependEdge: Edge deleted\n");
        exit(1);
    }
    struct EdgeListItem *prepend_item = createEdgeListItem(e);
    struct EdgeListItem *head = edge_list->head;
    prepend_item->next = head;
    head->prev = prepend_item;
    edge_list->head = prepend_item;
    if (!head) {
        // if list is empty
        edge_list->head = prepend_item;
        edge_list->tail = prepend_item;
    }
    return;
}

bool empty_EL(struct EdgeList* edge_list) {
    struct EdgeListItem *current;
    current = edge_list->head;
    return (current == NULL);
}

int length_EL(struct EdgeList* edge_list) {
    int length = 0;
    struct EdgeListItem *current;
    current = edge_list->head;
    while (current != NULL) {
        length++;
        current = current->next;
    }
    return length;
```

```
}

int removeEdge(struct EdgeList* edge_list, struct Edge* e) {
   struct EdgeListItem* head = edge_list->head;
   struct EdgeListItem* prev = NULL;
   struct EdgeListItem* next = NULL;
   if(head == NULL) {
       // list is empty
       return -1;
   } else {
       prev = head->prev;
       while (head) {
                bool  equal  =  (e->weight  ==  head->edge->weight)  &&  (e->start->id  ==
head->edge->start->id) && (e->end->id == head->edge->end->id);
           if (equal) {
               if (prev) {
                   next = head->next;
                   prev->next = next;
                   if (next) {
                       next->prev = prev;
                   }
               } else {
                   next = head->next;
                   edge_list->head = next;
                   if (next) {
                       next->prev = NULL;
                   }
               }
               return 0;
           }
           prev = head;
           head = head->next;
       }
       return -1;
   }
}

void appendEdge(struct EdgeList* edge_list, struct Edge* e) {
   if (e->deleted) {
       fprintf(stderr, "appendEdge: Edge deleted\n");
       exit(1);
   }
```

```
    struct EdgeListItem* new_last = createEdgeListItem(e);
    if (!empty_EL(edge_list)) {
        // if list not empty;
        new_last->prev = edge_list->tail;
        struct EdgeListItem *old_last = malloc(sizeof(struct EdgeListItem));
        old_last = edge_list->tail;
        old_last->next = new_last;
        edge_list->tail = new_last;
    } else {
        // if list is empty;
        edge_list->head = new_last;
        edge_list->tail = new_last;
    }
    return;
}
```

Edgelist.c

```
#include <pthread.h>
#include <stdlib.h>
#include <string.h>
#ifndef BUILDSTDLIB
#include "nodelist.h"
#include "edgelist.h"
#include "edge.h"
#endif

/*
* The key of the graph hashtbl is going to be a node
* The value will be a nodelist of nodes having an edge to the key
*/
struct Graph* createGraph(int size);

int hashCode(struct Graph* g, struct Node* node);

void incrementId(struct Graph* g);

void incrementGraphId();

/* Returns a nodelist of nodes in the graph
* sorted by order of creation
*/
```

```
struct NodeList* nodes(struct Graph* g);

struct Node* createNode(struct Graph* g, char* data);

int removeEdgeGraph(struct Graph* g, struct Edge* e);

/*
* Go through nodelist and delete node from:
* overall graph's nodelist
* and remove all edges from the value list
* for all edges of node to delete, delete these from other value lists
*/
int removeNodeGraph(struct Graph* g, struct Node* n);

struct Edge* addEdge(struct Graph* g, struct Node* start_node, struct Node* end_node,
double edge_weight);
```

graph.h

```
#include "graph.h"

/*
* The key of the graph hashtbl is going to be a node
* The value will be a nodelist of nodes having an edge to the key
*/
struct Graph* createGraph(int size) {
   if (size <= 0) {
       fprintf(stderr, "Error: Graph must have be at least size 1 or larger\n");
       exit(1);
   }
   struct DataItem* d = malloc(sizeof(struct DataItem) * size);
   for (int i = 0; i < size; i++) {
       d[i].key = NULL;
       d[i].value = NULL;
   }
   struct Graph* graph = calloc(1, sizeof(struct Graph));
   graph->hashArray = d;
   graph->size = size;
   graph->occupied = 0;
   graph->nodes = createNodeList();
   graph->id_num = 0;
   graph->graph_id_local = graph_id;
   incrementGraphId();
```

```
    return graph;
}

void incrementId(struct Graph* g){
    g->id_num++;
}

void incrementGraphId() {
    graph_id++;
}

/* Returns a nodelist of nodes in the graph
* sorted by order of creation
*/
struct NodeList* nodes(struct Graph* g){
    return g->nodes;
}

struct Node* createNode(struct Graph* g, char* data) {
    struct Node* node = malloc(sizeof(struct Node));
    struct EdgeList* el = createEdgeList();
    node->data = data;
    node->deleted = false;
    node->visited = false;
    node->cost = -1.0;
    node->precursor = malloc(sizeof(struct Node));
    node->id = g->id_num;
    node->parent_graph_id = g->graph_id_local;
    node->edges = createEdgeList();
    incrementId(g);
    int size_old = g->size;
    int id_node = node->id;
    g->hashArray[id_node % size_old].key = node;
    g->hashArray[id_node % size_old].value = el;
    g->occupied++;
    if (g->occupied == g->size) {
        int size_new = 2*size_old;
        g->size = size_new;
        // save old array
        struct DataItem* arr = g->hashArray;
        // now calloc new double array
        struct DataItem* dnew = malloc(sizeof(struct DataItem) * size_new);
```

```
        for (int i = 0; i < size_new; i++) {
            dnew[i].key = NULL;
            dnew[i].value = NULL;
        }
        for (int i = 0; i < size_old; i++) {
            dnew[i].key = arr[i].key;
            dnew[i].value = arr[i].value;
        }
        g->hashArray = dnew;
    }
    if (pthread_mutex_init(&node->lock, NULL) !=0) {
        fprintf(stderr, "createNode: Failure to initialize mutex\n");
        exit(1);
    }
    appendNode(g->nodes, node);
    return node;
}

int removeEdgeGraph(struct Graph* g, struct Edge* e) {
    // remove edge from value list of end_node
    struct EdgeList* values;
    struct Node* end_node = end(e);
    values = g->hashArray[end_node->id % g->size].value;
    removeEdge(values, e);

    // remove edge from node internal edgelist of start
    struct EdgeList* edge_list;
    struct Node* start_node = start(e);
    edge_list = start_node->edges;
    removeEdge(edge_list, e);
    e->deleted = true;
    return 0;
}


/*
* Go through nodelist and delete node from:
* overall graph's nodelist
* and remove all edges from the value list
* for all edges of node to delete, delete these from other value lists
*/
int removeNodeGraph(struct Graph* g, struct Node* n) {
```

```
    struct EdgeList* values;
    struct EdgeListItem* list_item = NULL;

    removeNode(g->nodes, n);

    // get values
    values = g->hashArray[n->id % g->size].value;
    if (values) {
        list_item = values->head;
    }

    // iterate through values and removeEdgeGraph for each
    while (list_item) {
        removeEdgeGraph(g, list_item->edge);
        list_item = list_item->next;
    }
    // null out the values
    g->hashArray[n->id % g->size].value = NULL;

    // Now, take care of the key
    // iterate through edgelist and remove
    // edge from every valuelist where it occurs
    struct EdgeList* node_edges;
    node_edges = n->edges;

    if (node_edges) {
        list_item = node_edges->head;
    }

    while (list_item) {
        struct Edge* e = list_item->edge;
        removeEdgeGraph(g, e);
        list_item = list_item->next;
    }

    // null out key
    g->hashArray[n->id % g->size].key = NULL;
    n->deleted = true;
    return 0;
}
```

```
struct Edge* addEdge(struct Graph* g, struct Node* start_node, struct Node* end_node,
double edge_weight) {
   if ((start_node->deleted) || (end_node->deleted)) {
       fprintf(stderr, "addEdge: Start or end node deleted\n");
       exit(1);
   }
   struct Edge* edge = malloc(sizeof(struct Edge));
   edge->start = start_node;
   edge->end = end_node;
   edge->weight = edge_weight;
   if (pthread_mutex_init(&edge->lock, NULL) !=0) {
       fprintf(stderr, "addEdge: Failure to initialize mutex\n");
       exit(1);
   }
   else {
       appendEdge(start_node->edges, edge);
       struct EdgeList* values;
       values = g->hashArray[end_node->id % g->size].value;
       appendEdge(values, edge);
       return edge;
   }
}
```

Graph.c

```
#include <pthread.h>
#include <stdlib.h>
#include <string.h>
#ifndef BUILDSTDLIB
#include "graph.h"
#endif

struct IntTable* createIntTable(int predicted_size);

int hashCode_it(struct IntTable* it, struct Node* node);

int getInt(struct IntTable* it, struct Node* n);

struct IntTableLLItem* createIntTableLLItem(struct Node* n, int data);

struct IntTableItem* createIntTableItem(struct Node* n, int data);

// technically complexity could improve if we insert in a sorted manner
```

```
// TODO
void insertInt(struct IntTable* it, struct Node* n, int data);

bool inInt(struct IntTable* it, struct Node* n);

int deleteInt(struct IntTable* it, struct Node* n);
```

inttable.h

```
#include "inttable.h"

struct IntTable* createIntTable(int predicted_size) {
   if (predicted_size <= 0) {
       fprintf(stderr, "Error: IntTable must have be at least size 1 or larger\n");
       exit(1);
   }
   struct IntTable* it = malloc(sizeof(struct IntTable));
        it->arr   =   (struct   IntTableItem   *)calloc(predicted_size,   sizeof(struct
IntTableItem));
   it->size = predicted_size;
   it->keys = createNodeList();
   it->graph_id = -1;
   if (pthread_mutex_init(&it->lock, NULL) !=0) {
       fprintf(stderr, "createIntTable: Failure to initialize mutex\n");
       exit(1);
   }
   return it;
}

int hashCode_it(struct IntTable* it, struct Node* n) {
   int id_node = n->id;
   return id_node % it->size;
}

int _getInt(struct IntTable* it, struct Node* n) {
   int hashIndex = hashCode_it(it, n);
   struct IntTableItem* start;
   start = &it->arr[hashIndex];
   while (start) {
       if (start->entry && (start->entry->key->id == n->id)) {
           return start->entry->value;
       }
       else {
```

```
            start = start->next;
        }
    }
    exit(1);
}

struct IntTableLLItem* createIntTableLLItem(struct Node* n, int data) {
    struct IntTableLLItem* inttab = malloc(sizeof(struct IntTableLLItem));
    inttab->key = n;
    inttab->value = data;
    return inttab;
}

struct IntTableItem* createIntTableItem(struct Node* n, int data) {
    struct IntTableItem* inttab = malloc(sizeof(struct IntTableItem));
    inttab->entry = createIntTableLLItem(n, data);
    inttab->next = NULL;
    return inttab;
}

// technically complexity could improve if we insert in a sorted manner
// TODO
void _insertInt(struct IntTable* it, struct Node* n, int data) {
    if (n->deleted) {
        fprintf(stderr, "_insertInt: Node deleted\n");
        exit(1);
    }
    if ((it->graph_id != -1) && (it->graph_id != n->parent_graph_id)) {
          fprintf(stderr, "_insertInt error: Cannot insert nodes from different graphs
into the same IntTable\n");
        exit(1);
    }
    else if (it->graph_id == -1) {
        it->graph_id = n->parent_graph_id;
    }
    int hashIndex = hashCode_it(it, n);
    struct IntTableItem* start = &it->arr[hashIndex];
    if (start == NULL) {
        it->arr[hashIndex] = *createIntTableItem(n, data);
    }
    else {
        while (start && start->next) {
```

```
            start = start->next;
        }
        start->next = createIntTableItem(n, data);
    }
    appendNode(it->keys, n);
    return;
}

struct NodeList* intKeys(struct IntTable* it){
    return it->keys;
}

bool inInt(struct IntTable* it, struct Node* n) {
    int hashIndex = hashCode_it(it, n);
    struct IntTableItem* start = &it->arr[hashIndex];
    while (start) {
        if (start->entry && (start->entry->key->id == n->id)) {
            return true;
        }
        else {
            start = start->next;
        }
    }
    return false;
}

int deleteInt(struct IntTable* it, struct Node* n) {
    // remove it from keys
    struct NodeList* all_nodes = malloc(sizeof(struct NodeList));
    all_nodes = intKeys(it);
    removeNode(all_nodes, n);

    int hashIndex = hashCode_it(it, n);
    struct IntTableItem* start;
    struct IntTableItem* prev;
    start = &it->arr[hashIndex];
    prev = NULL;
    while (start) {
        // find node to delete
        if (start->entry && (start->entry->key->id == n->id)) {
            if (prev == NULL) { // start of list
                it->arr[hashIndex] = *start->next;
```

```
            }
            else if (start->next == NULL) { // end of list
                prev->next = NULL;
            }
            else {
                prev->next = start->next;
            }
            return 0;
        }
        else {
            prev = start;
            start = start->next;
        }
    }
    return 0;
}
```

Inttable.c

```
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
#ifndef BUILDSTDLIB
#include "lockedobject.h"
#include "edge.h"
#endif

const char* data(struct Node* node);

/* Return the edgelist of surrounding edges */
struct EdgeList* edges(struct Node* node);

/* Returns a boolean representing
* if the node has already been visited. */
bool visited(struct Node* node);

struct Node* updateData(struct Node* node, char* new_data);

/* Updates the visited field on the node to be the inputted bool */
struct Node* updateVisited(struct Node* node, bool tf);

double cost(struct Node* node);
```

```
double incrementCost(struct Node* node);

double decrementCost(struct Node* node);

double updateCost(struct Node* node, double new_cost);

struct Node* prec(struct Node* node);

struct Node* setPrec(struct Node* node, struct Node* precursor);

bool nodeEquals(struct Node* node1, struct Node* node2);

/* TODO: NEEDS TESTING */
struct Edge* getEdge(struct Node* node1, struct Node* node2);
```

node.h

```
#include "node.h"

const char* data(struct Node* node)
{
   if (node->deleted) {
       fprintf(stderr, "data: Node deleted\n");
       exit(1);
   }
   return node->data;
}

/* Return the edgelist of surrounding edges */
struct EdgeList* edges(struct Node* node)
{
   if (node->deleted) {
       fprintf(stderr, "edges: Node deleted\n");
       exit(1);
   }
   return node->edges;
}

/* Returns a boolean representing
* if the node has already been visited. */
bool visited(struct Node* node)
{
```

```
    if (node->deleted) {
        fprintf(stderr, "visited: Node deleted\n");
        exit(1);
    }
    return node->visited;
}

struct Node* updateData(struct Node* node, char* new_data)
{
    if (node->deleted) {
        fprintf(stderr, "updateData: Node deleted\n");
        exit(1);
    }
    node->data = new_data;
    return node;
}

/* Updates the visited field on the node to be the inputted bool */
struct Node* updateVisited(struct Node* node, bool tf)
{
    if (node->deleted) {
        fprintf(stderr, "updateVisited: Node deleted\n");
        exit(1);
    }
    node->visited = tf;
    return node;
}

double cost(struct Node* node)
{
    if (node->deleted) {
        fprintf(stderr, "cost: Node deleted\n");
        exit(1);
    }
    return node->cost;
}

double incrementCost(struct Node* node)
{
    if (node->deleted) {
        fprintf(stderr, "incrementCost: Node deleted\n");
        exit(1);
```

```
    }
    node->cost = node->cost + 1.0;
    return node->cost;
}

double decrementCost(struct Node* node)
{
    if (node->deleted) {
        fprintf(stderr, "decrementCost: Node deleted\n");
        exit(1);
    }
    node->cost = node->cost - 1.0;
    return node->cost;
}

double updateCost(struct Node* node, double new_cost)
{
    if (node->deleted) {
        fprintf(stderr, "updateCost: Node deleted\n");
        exit(1);
    }
    node->cost = new_cost;
    return node->cost;
}

struct Node* prec(struct Node* node)
{
    if (node->deleted) {
        fprintf(stderr, "prec: Node deleted\n");
        exit(1);
    }
    return node->precursor;
}

struct Node* setPrec(struct Node* node, struct Node* precursor)
{
    if ((node->deleted) || (precursor->deleted)) {
        fprintf(stderr, "setPrec: Node deleted\n");
        exit(1);
    }
    node->precursor = precursor;
    return node;
```

```
}

bool nodeEquals(struct Node* node1, struct Node* node2)
{
   if ((node1->deleted) || (node2->deleted)) {
       fprintf(stderr, "nodeEquals: Node deleted\n");
       exit(1);
   }
   return (node1->id == node2->id);
}

struct Edge* getEdge(struct Node* node1, struct Node* node2)
{
   if ((node1->deleted) || (node2->deleted)) {
       fprintf(stderr, "getEdge: Node deleted\n");
       exit(1);
   }
   struct EdgeListItem *current;
   current = node1->edges->head;
   while (current != NULL) {
       struct Node* s_node = current->edge->start;
       struct Node* e_node = current->edge->end;
       if (nodeEquals(s_node, node1) && nodeEquals(e_node, node2)) {
           return current->edge;
       } else {
           current = current->next;
       }
   }
   return NULL;
}
```

Node.c

```
#include <stdlib.h>
#include <string.h>
#ifndef BUILDSTDLIB
#include "lockedobject.h"
#include "node.h"
#endif

struct NodeListItem* createNodeListItem(struct Node* e);
```

```
struct NodeList* createNodeList();

struct Node* head_NL(struct NodeList* node_list);

struct Node* tail_NL(struct NodeList* node_list);

struct Node* removeFirst_NL(struct NodeList* node_list);

struct Node* removeLast_NL(struct NodeList* node_list);

void prependNode(struct NodeList* node_list, struct Node* e);

bool empty_NL(struct NodeList* node_list);

int length_NL(struct NodeList* node_list);

void appendNode(struct NodeList* node_list, struct Node* n);

int removeNode(struct NodeList* node_list, struct Node* e);

bool includesNode(struct NodeList* nl, struct Node* n);

struct NodeList* neighbors(struct Node* node);
```

nodelist.h

```
#include "nodelist.h"

struct NodeListItem* createNodeListItem(struct Node* e) {
   struct NodeListItem* item = malloc(sizeof(struct NodeListItem));
   item->node = e;
   item->next = NULL;
   item->prev = NULL;
   return item;
}

struct NodeList* createNodeList() {
   struct NodeList* node_list = malloc(sizeof(struct NodeList));
   node_list->head = NULL;
   node_list->tail = NULL;
   if (pthread_mutex_init(&node_list->lock, NULL) !=0) {
```

```
        fprintf(stderr, "createNodeList: Failure to initialize mutex\n");
        exit(1);
    }
    return node_list;
}

struct Node* head_NL(struct NodeList* node_list) {
    if (!node_list->head) {
        return NULL;
    }
    return node_list->head->node;
}

struct Node* tail_NL(struct NodeList* node_list) {
    if (!node_list->tail) {
        return NULL;
    }
    return node_list->tail->node;
}

struct Node* removeFirst_NL(struct NodeList* node_list) {
    struct NodeListItem* head;
    head = node_list->head;
    if (head) {
        node_list->head = head->next;
        if(node_list->head){
            node_list->head->prev = NULL;
        }
        return head->node;
    } else {
        return NULL;
    }
}

struct Node* removeLast_NL(struct NodeList* node_list) {
    struct NodeListItem* last = node_list->tail;
    struct NodeListItem* prev = NULL;
    if (last) {
        prev = last->prev;
        node_list->tail = prev;
        if (prev) {
            prev->next = NULL;
```

```
        }
        return last->node;
    } else {
        return NULL;
    }
}

void prependNode(struct NodeList* node_list, struct Node* e) {
    if (e->deleted) {
        fprintf(stderr, "prependNode: Node deleted\n");
        exit(1);
    }
    struct NodeListItem* prepend_item = createNodeListItem(e);
    if (!node_list->head) {
        // if list is empty
        node_list->head = prepend_item;
        node_list->tail = prepend_item;
        return;
    }
    struct NodeListItem* head = node_list->head;
    prepend_item->next = head;
    head->prev = prepend_item;
    node_list->head = prepend_item;
    return;
}

bool empty_NL(struct NodeList* node_list) {
    struct NodeListItem* current;
    current = node_list->head;
    return (current == NULL);
}

int length_NL(struct NodeList* node_list) {
    int length = 0;
    struct NodeListItem *current;
    current = node_list->head;
    while (current != NULL) {
        length++;
        current = current->next;
    }
    return length;
}
```

```
void appendNode(struct NodeList* node_list, struct Node* n) {
   if (n->deleted) {
       fprintf(stderr, "appendNode: Node deleted\n");
       exit(1);
   }
   struct NodeListItem* new_last = createNodeListItem(n);
   if (!empty_NL(node_list)) {
       // if list not empty;
       new_last->prev = node_list->tail;
       struct NodeListItem* old_last = malloc(sizeof(struct NodeListItem));
       old_last = node_list->tail;
       old_last->next = new_last;
       node_list->tail = new_last;
   } else {
       // if list is empty;
       node_list->head = new_last;
       node_list->tail = new_last;
   }
   return;
}

int removeNode(struct NodeList* node_list, struct Node* e) {
   struct NodeListItem* head = node_list->head;
   struct NodeListItem* prev = NULL;
   struct NodeListItem* next = NULL;
   if(head == NULL) {
       // list is empty
       return -1;
   } else {
       prev = head->prev;
       while (head) {
           if (e->id == head->node->id) {
               if (prev) {
                   next = head->next;
                   prev->next = next;
                   if (next) {
                       next->prev = prev;
                   }
               } else {
                   next = head->next;
                   node_list->head = next;
```

```
                    if (next) {
                        next->prev = NULL;
                    }
                }
                return 0;
            }
            prev = head;
            head = head->next;
        }
        return -1;
    }
}

bool includesNode(struct NodeList* nl, struct Node* n){
    struct NodeListItem* temp = malloc(sizeof(struct NodeListItem));
    if (nl != NULL){
        temp = nl->head;
    } else {
        return false;
    }
    while (temp) {
        if (n->id == temp->node->id){
            return true;
        }
        temp = temp->next;
    }
    return false;
}

struct NodeList* neighbors(struct Node* node){
    if (node->deleted) {
        fprintf(stderr, "neighbors: Node deleted\n");
        exit(1);
    }
    struct EdgeList* edges = node->edges;
    struct NodeList* neighbors = createNodeList();
    struct EdgeListItem* current = edges->head;
    while (current != NULL) {
        appendNode(neighbors, current->edge->end);
        current = current->next;
    }
    return neighbors;
```

```
}
```

Nodelist.c

SMALLER TEST FILES WRITTEN IN GRACL

```
int main()
{
 int i;
 bool b;

 i = 42;
 i = 10;
 b = True;
 b = False;
 i = False; /* Fail: assigning a bool to an integer */
}
```

Fail-assign1.grc

```
int main()
{
 int i;
 bool b;

 b = 48; /* Fail: assigning an integer to a bool */
}
```

Fail-assign2.grc

```
void myvoid()
{
 return;
}

int main()
{
 int i;

 i = myvoid(); /* Fail: assigning a void to an integer */
}
```

Fail-assign3.grc

```
int main(){
   Graph g = createGraph(10);
   Node n = g.createNode("test");
   g[n] = 5;
}
```

Fail-brackfuncs.grc

```
int main() {
   IntTable it = createIntTable(5);
   String node = "Node";
   it[node] = 10;
}
```

Fail-brackfuncs2.grc

```
int main() {
   IntTable it = createIntTable(5);
   String node = "Node";
   printi(it[node]);
}
```

Fail-brackfuncs3.grc

```
int main(){
   Graph inttable = createGraph(10);
   Node n = inttable.createNode("test");
   printi(inttable[n]);
}
```

Fail-brackfuncs4.grc

```
int main()
{
 int i;

 i = 15;
 return i;
 i = 32; /* Error: code after a return */
}
```

Fail-dead1.grc

```
int main()
{
 int i;
```

```
 {
   i = 15;
   return i;
 }
 i = 32; /* Error: code after a return */
}
```

Fail-dead2.grc

```
int a;
bool b;

void foo(int c, bool d)
{
 int dd;
 bool e;
 a + c;
 c - a;
 a * 3;
 c / 2;
 d + a; /* Error: bool + int */
}

int main()
{
 return 0;
}
```

Fail-expr1.grc

```
int a;
bool b;

void foo(int c, bool d)
{
 int d;
 bool e;
 b + a; /* Error: bool + int */
}

int main()
{
```

```
 return 0;
}
```

Fail-expr2.grc

```
int a;
double b;

void foo(int c, double d)
{
 int d;
 double e;
 b + a; /* Error: double + int */
}

int main()
{
 return 0;
}
```

Fail-expr3.grc

```
int main()
{
 -3.5 && 1; /* Float with AND? */
 return 0;
}
```

Fail-float1.grc

```
int main()
{
 -3.5 && 2.5; /* Float with AND? */
 return 0;
}
```

Fail-float2.grc

```
int main(){
   Graph g = createGraph(10);
   g.createNode("yo");
   for (Node n in g){
       //Can't iterate over graph
```

```
    }
}
```

Fail-for1.grc

```
int main(){
    Graph g = createGraph(10);
    NodeList nl = createNodeList();
    nl.appendNode(g.createNode("yo"));
    for (Edge e in nl){
        //Can't use Edges to iterate over NodeList
    }
}
```

Fail-for2.grc

```
int main(){
    Graph g = createGraph(10);
    EdgeList el = createEdgeList();
    el.appendEdge(g.addEdge(g.createNode("start"), g.createNode("end"), 3.14));
    for (Edge e in el){
        print(e.data()); //e is an Edge, data looks for Node
    }
}
```

Fail-for3.grc

```
int foo() {}

int bar() {}

int baz() {}

void bar() {} /* Error: duplicate function bar */

int main()
{
 return 0;
}
```

Fail-func1.grc

```
int foo(int a, bool b, int c) { }

void bar(int a, bool b, int a) {} /* Error: duplicate formal a in bar */
```

```
int main()
{
 return 0;
}
```

Fail-func2.grc

```
int foo(int a, bool b, int c) { }

void bar(int a, void b, int c) {} /* Error: illegal void formal b */

int main()
{
 return 0;
}
```

Fail-func3.grc

```
int foo() {}

void bar() {}

int print() {} /* Should not be able to define print */

void baz() {}

int main()
{
 return 0;
}
```

Fail-func4.grc

```
int foo() {}

int bar() {
 int a;
 void b; /* Error: illegal void local b */
 bool c;

 return 0;
}
```

```
int main()
{
 return 0;
}
```

Fail-func5.grc

```
void foo(int a, bool b)
{
}

int main()
{
 foo(42, True);
 foo(42); /* Wrong number of arguments */
}
```

Fail-func6.grc

```
void foo(int a, bool b)
{
}

int main()
{
 foo(42, True);
 foo(42, True, False); /* Wrong number of arguments */
}
```

Fail-func7.grc

```
void foo(int a, bool b)
{
}

void bar()
{
}

int main()
{
 foo(42, True);
 foo(42, bar()); /* int and void, not int and bool */
```

```
}
```

Fail-func8.grc

```
void foo(int a, bool b)
{
}

int main()
{
 foo(42, True);
 foo(42, 42); /* Fail: int, not bool */
}
```

Fail-func9.grc

```
int main() {
   int foo() {
       //Can't declare a function in a function
   }
   return 0;
}
```

Fail-funcinfunc.grc

```
int c;
bool b;
void a; /* global variables should not be void */


int main()
{
 return 0;
}
```

Fail-global1.grc

```
int b;
bool c;
int a;
int b; /* Duplicate global variable */

int main()
{
```

```
 return 0;
}
```

Fail-global2.grc

```
void foo () {
  a = 5;
}
int a; //Should not be found by foo
int main() {
   foo();
   return 0;
}
```

Fail-global3.grc

```
int main()
{
 if (True) {}
 if (False) {} else {}
 if (42) {} /* Error: non-bool predicate */
}
```

Fail-if1.grc

```
int main()
{
 if (true) {
   foo; /* Error: undeclared variable */
 }
}
```

Fail-if2.grc

```
int main()
{
 if (true) {
   42;
 } else {
   bar; /* Error: undeclared variable */
 }
}
```

Fail-if3.grc

```
int main() {
   test = 3; //Cannot use before declaring
   int test = 4;
}
```

Fail-init.grc

```
int i = 6;
int k = i; //Cannot initialize a global to a nonconstant expression
int main()
{
 return 0;
}
```

Fail-initconst.grc

```
int main() {
   double e = 3.142;
   double pi = 2.718;
   double res = e % pi; //mod doesn't work with doubles
}
```

Fail-mod.grc

```

```

fail-nomain.grc

```
/* Should be illegal to redefine */
void print() {}

```

Fail-print.grc

```
int main()
{
 return True; /* Should return int */
}

```

Fail-return1.grc

```
void foo()
{
 if (true) return 42; /* Should return void */
 else return;
}

int main()
```

```
{
 return 42;
}
```

Fail-return2.grc

```
int main() {
   Graph g = createGraph(10);
   synch g { //Cannot synch on graph
       Node n = g.createNode("Node data");
       print(n.data());
   }
   return 0;
}
```

fail-synch.grc

```
int main()
{
 int i;

 while (True) {
   i = i + 1;
 }

 while (42) { /* Should be boolean */
   i = i + 1;
 }

}
```

Fail-while1.grc

```
int main()
{
 int i;

 while (True) {
   i = i + 1;
 }

 while (True) {
   foo(); /* foo undefined */
 }
```

```
}
```

Fail-while2.grc

```
int add(int x, int y)
{
 return x + y;
}

int main()
{
 printi( add(17, 25) );
 return 0;
}
```

Test-add1.grc

```
int main()
{
 printi(39 + 3);
 return 0;
}
```

Test-arith1.grc

```
int main()
{
 printi(1 + 2 * 3 + 4);
 return 0;
}
```

Test-arith2.grc

```
int foo(int a)
{
 return a;
}

int main()
{
 int a;
 a = 42;
```

```
 a = a + 5;
 printi(a);
 return 0;
}
```

Test-arith3.grc

```
int main() {
   int a = 1;
   int b = 2;
   {
       int b = 2;
       {
           int a = 3;
           printi(a); //3
           printi(b); //2
       }
       {
           int b = 4;
           printi(a); //1
           printi(b); //4
       }
       printi(a); //1
       printi(b); //2
   }
   printi(a); //1
   printi(b); //2
}
```

Test-blockscope.grc

```
int main() {
   IntTable it = createIntTable(10);
   Graph g = createGraph(10);
   Node n1 = g.createNode("Hello");
   it[n1] = 42;
   printi(it[n1]);
   DoubleTable dt = createDoubleTable(10);
   Node n2 = g.createNode("Goodbye");
   dt[n2] = 5.5;
   print(doubleToString(dt[n2]));
}
```

Test-brackfuncs.grc

```
int main() {
   int i = 5;
   double d = intToDouble(i);
   printi(i);
   print(doubleToString(d));
   print(doubleToString(intToDouble(23)));
   return 0;
}
```

Test-casting.grc

```
int glob = 120 - -3;
String me = "Are you";
double e = 2.718;
int main()
{
 int a = 42;
 int b = a + 5;
 String you = "me?";
 printi(glob);
 print(me);
 print(you);
 printi(a);
 printi(b);
 print(doubleToString(e));
 return 0;
}
```

Test-decinit.grc

```
int main()
{
 double a;
 a = 3.14159267;
 print(doubleToString(a));
 return 0;
}
```

Test-double1.grc

```
int main()
{
 double a;
 double b;
 double c;
```

```
 a = 3.14159267;
 b = -2.71828;
 c = a + b;
 print(doubleToString(c));
 return 0;
}
```

Test-double2.grc

```
void printf(double d){
 print(doubleToString(d));
}
void printb(bool b){
 if (b){
   print("True");
 }
 else {
   print("False");
 }
}
void testdouble(double a, double b)
{
 printf(a + b);
 printf(a - b);
 printf(a * b);
 printf(a / b);
 printb(a == b);
 printb(a == a);
 printb(!(a == b));
 printb(!(a == a));
 printb(b < a);
 printb(b <= a);
 printb(a < b);
 printb(a <= b);
}

int main()
{
 double c;
 double d;

 c = 42.0;
 d = 3.14159;
```

```
 testdouble(c, d);

 testdouble(d, d);

 return 0;
}
```

Test-double3.grc

```
int main(){
   bool success = True;

   Graph g = createGraph(10);
   String nodedata1 = "noded1";
   String nodedata2 = "noded2";
   String nodedata3 = "noded3";
   String nodedata4 = "noded4";
   String nodedata5 = "noded5";
   String nodedata6 = "noded6";
   String nodedata7 = "noded7";
   String nodedata8 = "noded8";
   String nodedata9 = "noded9";
   String nodedata10 = "noded10";
   Node node1 = g.createNode(nodedata1);
   Node node2 = g.createNode(nodedata2);
   Node node3 = g.createNode(nodedata3);
   Node node4 = g.createNode(nodedata4);
   Node node5 = g.createNode(nodedata5);
   Node node6 = g.createNode(nodedata6);
   Node node7 = g.createNode(nodedata7);
   Node node8 = g.createNode(nodedata8);
   Node node9 = g.createNode(nodedata9);
   Node node10 = g.createNode(nodedata10);

   DoubleTable dt = createDoubleTable(5);

   // test insert with collisions
   dt[node1] = 1.0;
   dt[node2] = 2.0;
   dt[node3] = 3.0;
   dt[node4] = 4.0;
```

```
   dt[node5] = 5.0;
   dt[node6] = 6.0;
   dt[node7] = 7.0;
   dt[node8] = 8.0;
   dt[node9] = 9.0;
   dt[node10] = 10.0;
     dt[node10] = 10.0; // insert same node twice -- node10 appended to end twice
without issue


   // test includes
   if (!dt.inDouble(node1) || !dt.inDouble(node7) || !dt.inDouble(node10)) {
       success = False;
   }

   // test get
   double get1 = dt[node1];
   double get8 = dt[node8];
   double get3 = dt[node3];
   if ((get1 != 1.0) || (get8 != 8.0) || (get3 != 3.0)) {
       success = False;
   }

   // test delete
   dt.deleteDouble(node1);
   dt.deleteDouble(node2);
   dt.deleteDouble(node3);
   dt.deleteDouble(node4);
   dt.deleteDouble(node5);
   dt.deleteDouble(node6);
   dt.deleteDouble(node7);
   dt.deleteDouble(node8);
   dt.deleteDouble(node9);
   dt.deleteDouble(node10);
   dt.deleteDouble(node10); // try deleting same node (added to dt twice) twice
   dt.deleteDouble(node1); // try deleting same node (added to dt once) twice

   if (dt.inDouble(node1)) {
       success = False;
   }
   NodeList keys = dt.doubleKeys();
   if (keys.length_NL() != 0) {
```

```
        success = False;
    }

    if(success) {
        print("SUCCESS");
    }
    else {
        print("FAILURE");
    }

    return 0;
}
```

Test-doubletable.grc

```
int main() {
  // create graph with 2 nodes and test the functions
    Graph g = createGraph(100);
    String nodedata1 = "noded1";
    String nodedata2 = "noded2";
    String nodedata3 = "noded3";

    Node node1 = g.createNode(nodedata1);
    Node node2 = g.createNode(nodedata2);
    Node node3 = g.createNode(nodedata3);

    Edge edge1 = g.addEdge(node1, node2, 1.7);
    Edge edge2 = g.addEdge(node2, node3, 6.8);
    Edge edge3 = g.addEdge(node1, node2, 1.7);

    bool success = True;

    // test updateEdge and weight functions
    edge2.updateEdge(16.3);
    if (edge2.weight() != 16.300000) {
        success = False;
    }

    edge2.updateEdge(90.1);
    if (edge2.weight() != 90.100000) {
        success = False;
    }
```

```
    edge2.updateEdge(-90.1);
    if (edge2.weight() != -90.100000) {
        success = False;
    }

    edge2.updateEdge(infinity);
    if (edge2.weight() != infinity) {
        success = False;
    }

    // test edgeEquals
    if(edge1.edgeEquals(edge2)) {
        success = False;
    }
    if(!edge1.edgeEquals(edge1)) {
        success = False;
    }

    // test end and start functions
    if(!edge1.end().data().stringEquals(edge2.start().data())) {
        success = False;
    }
    if(!edge1.start().data().stringEquals(edge3.start().data())) {
        success = False;
    }
    if(edge1.start().data().stringEquals(edge1.end().data())) {
        success = False;
    }

    if (success) {
        print("SUCCESS");
    } else {
        print("FAILURE");
    }
}
```

Test-edge.grc

```
int main() {
    Graph g = createGraph(100);
    EdgeList el = createEdgeList();
    el.appendEdge(g.addEdge(g.createNode("1"), g.createNode("2"), infinity));
```

```
    el.appendEdge(g.addEdge(g.createNode("3"), g.createNode("4"), 3.14));
    for (Edge e in el){
        print(doubleToString(e.weight()));
        print(e.start().data());
        print(e.end().data());
    }
}
```

Test-edgefor.grc

```
void printEdgeList(EdgeList el) {
    int i = el.length_EL();
    while (i > 0){
        Edge e = el.removeFirst_EL();
        print(e.start().data());
        el.appendEdge(e);
        i = i - 1;
    }
}

int main() {
    Graph g = createGraph(5);

    String hello = "Hello";
    String goodbye = "Goodbye";

    Node n1 = g.createNode(hello);
    Node n2 = g.createNode(goodbye);

    Edge e1 = g.addEdge(n1, n2, 15.3);
    Edge e2 = g.addEdge(n2, n1, 3.7);
    EdgeList el = createEdgeList();

    print("Empty should be true:");
    if (el.empty_EL()) {
        print("true");
    } else {
        print("false");
    }

    print("Append edge with start 'hello'");
    el.appendEdge(e1);
    print("Prepend edge with start 'goodbye'");
```

```
    el.prependEdge(e2);
    print("List should be goodbye hello:");
    printEdgeList(el);
}
```

Test-edgelist1.grc

```
void printEdgeList(EdgeList el) {
    int i = el.length_EL();
    while (i > 0){
        Edge e = el.removeFirst_EL();
        print(e.start().data());
        el.appendEdge(e);
        i = i - 1;
    }
}

int main() {
    Graph g = createGraph(5);

    String hello = "Hello";
    String goodbye = "Goodbye";

    Node n1 = g.createNode(hello);
    Node n2 = g.createNode(goodbye);

    Edge e1 = g.addEdge(n1, n2, 15.3);
    Edge e2 = g.addEdge(n2, n1, 3.7);
    EdgeList el = createEdgeList();

    print("Empty should be true:");
    if (el.empty_EL()) {
        print("true");
    } else {
        print("false");
    }

    print("Append edge with start 'hello'");
    el.appendEdge(e1);
    print("Prepend edge with start 'goodbye'");
    el.prependEdge(e2);
    print("List should be goodbye hello:");
```

```
    printEdgeList(el);



    print("Weight of head should be 3.7:");
    Edge h = el.head_EL();
    double hw = h.weight();
    print(doubleToString(hw));

    print("Weight of tail should be 15.3:");
    Edge t = el.tail_EL();
    double tw = t.weight();
    print(doubleToString(tw));

    print("Length should be 2:");
    printi(el.length_EL());
}
```

Test-edgelist2.grc

```
void printEdgeList(EdgeList el) {
    /*for (Edge e in el) {
        print(e.data());
    }*/

    int i = el.length_EL();
    while (i > 0){
        Edge e = el.removeFirst_EL();  //This call segfaults
        print(e.start().data());
        el.appendEdge(e);
        i = i - 1;
    }
}

int main() {
    //Make two nodes to be start and end for edge
    String hello = "Hello";
    String goodbye = "Goodbye";
    String yo = "Yo";
    Graph g = createGraph(100);
    Node n1 = g.createNode(hello);
    Node n2 = g.createNode(goodbye);
    Node n3 = g.createNode(yo);
    Edge e1 = g.addEdge(n1, n2, 15.3);
```

```
Edge e2 = g.addEdge(n2, n1, 3.7);
Edge e3 = g.addEdge(n3, n2, 4.8);
bool success = True;

EdgeList el = createEdgeList();
if (!el.empty_EL()) {
    success = False;
}
el.appendEdge(e1);
el.prependEdge(e2);
if (!(el.length_EL()==2)) {
    success = False;
}
el.prependEdge(e3);
if (!(el.length_EL()==3)) {
    success = False;
}

el.removeFirst_EL();
print("After remove first (list should be goodbye hello): ");
printEdgeList(el);
if (!(el.length_EL()==2)) {
    success = False;
}

el.prependEdge(e3);
el.removeLast_EL();
print("After remove last (list should be yo goodbye): ");
printEdgeList(el);
if (!(el.length_EL()==2)) {
    success = False;
}

el.removeEdge(e1);
print("After remove edge (list should be yo goodbye): ");
printEdgeList(el);
if (!(el.length_EL()==2)) {
    success = False;
}

el.removeEdge(e2);
if (!(el.length_EL()==1)) {
```

```
        success = False;
    }

    el.removeEdge(e3);
    if (!el.empty_EL()) {
        success = False;
    }

    if (success) {
        print("SUCCESS");
    }
    else {
        print("FAILURE");
    }

    return 0;
}
```

Test-edgelistfull.grc

```
int fib(int x)
{
 if (x < 2) return 1;
 return fib(x-1) + fib(x-2);
}

int main()
{
 printi(fib(0));
 printi(fib(1));
 printi(fib(2));
 printi(fib(3));
 printi(fib(4));
 printi(fib(5));
 return 0;
}
```

Test-fib.grc

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

```
int main()
{
 int a;
 a = add(39, 3);
 printi(a);
 return 0;
}
```

Test-func1.grc

```
/* Bug noticed by Pin-Chin Huang */

int fun(int x, int y)
{
 return 0;
}

int main()
{
 int i;
 i = 1;

 fun(i = 2, i = i+1);

 printi(i);
 return 0;
}
```

Test-func2.grc

```
void printem(int a, int b, int c, int d)
{
 printi(a);
 printi(b);
 printi(c);
 printi(d);
}

int main()
{
 printem(42,17,192,8);
 return 0;
```

```
}
```

Test-func3.grc

```
int add(int a, int b)
{
 int c;
 c = a + b;
 return c;
}

int main()
{
 int d;
 d = add(52, 10);
 printi(d);
 return 0;
}
```

Test-func4.grc

```
int foo(int a)
{
 return a;
}

int main()
{
 return 0;
}
```

Test-func5.grc

```
void foo() {}

int bar(int a, bool b, int c) { return a + c; }

int main()
{
 printi(bar(17, False, 25));
 return 0;
}
```

Test-func6.grc

```
int a;

void foo(int c)
{
 a = c + 42;
}

int main()
{
 foo(73);
 printi(a);
 return 0;
}
```

Test-func7.grc

```
void foo(int a)
{
 printi(a + 3);
}

int main()
{
 foo(40);
 return 0;
}
```

Test-func8.grc

```
void foo(int a)
{
 printi(a + 3);
 return;
}

int main()
{
 foo(40);
 return 0;
}
```

Test-func9.grc

```
//Tests how functions and methods interact
int main() {
   Graph g = createGraph(5);
   Node n = g.createNode(doubleToString(intToDouble(42)));
   print(n.data());
}
```

Test-funcmethod.grc

```
//Tests having a function and variables with the same name
int foo(int foo){
   printi(foo);
   foo = foo + 1;
   return foo;
}
int foo = 5;
int main() {
   foo = foo(foo);
   printi(foo);
}
```

Test-funcvar.grc

```
int gcd(int a, int b) {
 while (!(a == b)) {
   if (b < a) a = a - b;
   else b = b - a;
 }
 return a;
}

int main()
{
 printi(gcd(2,14));
 printi(gcd(3,15));
 printi(gcd(99,121));
 return 0;
}
```

Test-gcd.grc

```
int gcd(int a, int b) {
 while (!(a == b))
   if (b < a) a = a - b;
```

```
   else b = b - a;
 return a;
}

int main()
{
 printi(gcd(14,21));
 printi(gcd(8,36));
 printi(gcd(99,121));
 return 0;
}
```

Test-gcd2.grc

```
bool one = 3 >= 4;
bool two = 5.0 >= 4.0;
bool three = 0.00 > -0.;
bool four = 0.00 <= -1.0;
bool five = 0 > -1;
bool six = 526 > (10 / 2);
bool seven = 546 != 2;

int intone = 1;
int inttwo = 4;
double dubone = 1.0;
double dubtwo = -3.0;
double dubthree = 0.;

void printb(bool b){
 if (b){
   printi(1);
 }
 else {
   printi(0);
 }
}

int main()
{
 printb(one);
 printb(two);
 printb(three);
```

```
 printb(four);
 printb(five);
 printb(six);
 printb(seven);
 printi(99);
 printb(intone == inttwo);
 printb(intone != inttwo);
 printb(dubone != dubthree);
 printi(99);

 printb(dubtwo > dubone);
 printb(dubone > dubthree);
 printb(dubtwo > dubtwo);
 printb(intone >= inttwo);
 printb(dubone <= dubthree);

 return 0;
}
```

Test-glob-ops1.grc

```
int a;
int b;

void printa()
{
 printi(a);
}

void printbb()
{
 printi(b);
}

void incab()
{
 a = a + 1;
 b = b + 1;
}

int main()
{
```

```
 a = 42;
 b = 21;
 printa();
 printbb();
 incab();
 printa();
 printbb();
 return 0;
}
```

Test-global1.grc

```
bool i;

int main()
{
 int i; /* Should hide the global i */

 i = 42;
 printi(i + i);
 return 0;
}
```

Test-global2.grc

```
int i;
bool b;
int j;

int main()
{
 i = 42;
 j = 10;
 printi(i + j);
 return 0;
}
```

Test-global3.grc

```
void printNodeList(NodeList nl) {
   for (Node n in nl) {
       print(n.data());
   }
```

```
}

int main() {
   Graph g;
   Node node1;
   Node node2;
   Node node3;
   Node node4;
   Node node5;
   Node node6;
   Node node7;
   Node node8;
   Edge e1;
   Edge e2;
   Edge e3;
   Edge e4;
   Edge e5;
   Edge e6;
   Edge e7;
   Edge e8;
   Edge e9;
   Edge e10;
   Edge e11;
   NodeList nl1;
   NodeList nl2;
   bool success = True;

   g = createGraph(3);
   node1 = g.createNode("A");
   node2 = g.createNode("B");
   node3 = g.createNode("C");
   node4 = g.createNode("D");
   node5 = g.createNode("E");
   node6 = g.createNode("F");
   node7 = g.createNode("G");
   node8 = g.createNode("H");

   nl1 = g.nodes();
   if (!(nl1.length_NL() == 8)) {
       success = False;
   }
```

```
e1 = g.addEdge(node1, node2, 14.0);
e2 = g.addEdge(node2, node1, 13.0);
e3 = g.addEdge(node1, node3, 43.0);
e4 = g.addEdge(node2, node3, 43.0);
e5 = g.addEdge(node3, node2, 43.0);
e6 = g.addEdge(node3, node1, 43.0);
e7 = g.addEdge(node1, node8, 23.0);
e8 = g.addEdge(node1, node7, 23.0);
e9 = g.addEdge(node1, node6, 23.0);
e10 = g.addEdge(node8, node4, 23.0);
e11 = g.addEdge(node5, node4, 23.0);

EdgeList el1 = node1.edges();
if (!(el1.length_EL() == 5)) {
    success = False;
}

g.removeNodeGraph(node3);

nl2 = node1.neighbors();
if (!(nl2.length_NL() == 4)) {
    success = False;
}

el1 = node1.edges();
if (!(el1.length_EL() == 4)) {
    success = False;
}

nl2 = g.nodes();
if (!(nl2.length_NL() == 7)) {
    success = False;
}

g.removeEdgeGraph(e11);
el1 = node5.edges();
if (!el1.empty_EL()) {
    success = False;
}

g.removeNodeGraph(node1);
g.removeNodeGraph(node2);
```

```
    g.removeNodeGraph(node4);
    g.removeNodeGraph(node5);
    g.removeNodeGraph(node6);
    g.removeNodeGraph(node7);
    g.removeNodeGraph(node8);
    nl2 = g.nodes();
    if (!(empty_NL(nl2))) {
        success = False;
    }

    node8 = g.createNode("C");
    node7 = g.createNode("H");
    e11 = g.addEdge(node8, node7, 23.0);

    print(e11.start().data());
    g.removeEdgeGraph(e11);
    print(node7.data());

    if (success) {
        print("SUCCESS");
    } else {
        print("FAILURE");
    }

    return 0;
}
```

Test-graph.grc

```
int start_routine(Node n){
    print(n.data());
    return 0;
}

int main(){
    Graph g;
    Node node1;
    Node node2;
    Node node3;
    Node node4;
    Node node5;
    Node node6;
```

```
   Node node7;
   Node node8;
   Node node9;
   NodeList nl1;
   g = createGraph(10);
   node1 = g.createNode("A");
   node2 = g.createNode("B");
   node3 = g.createNode("C");
   node4 = g.createNode("D");
   node5 = g.createNode("E");
   node6 = g.createNode("F");
   node7 = g.createNode("G");
   node8 = g.createNode("H");
   node9 = g.createNode("I");
   nl1 = g.nodes();

   hatch nl1 start_routine(){
       print("Code executed by the parent thread");
   }
   return 0;
}
```

Test-hatch.grc

```
int main() {
   Node n;
   {
       Graph g = createGraph(5);
       n = g.createNode("Hello there");
   }
   int g = 5; //Graph g out of scope
   print(n.data()); //Even though graph cannot be referenced, node (in graph) persists
}
```

Test-heap.grc

```
int main()
{
 printi(42);
 printi(71);
 printi(1);
 return 0;
}
```

Test-hello.grc

```
int main() {
   print("Hello, World!");
   return 0;
}
```

test-helloWorld.grc

```
int main()
{
 if (True) printi(42);
 printi(17);
 return 0;
}
```

Test-if1.grc

```
int main()
{
 if (True) printi(42); else printi(8);
 printi(17);
 return 0;
}
```

Test-if2.grc

```
int main()
{
 if (False) printi(42);
 printi(17);
 return 0;
}
```

Test-if3.grc

```
int main()
{
 if (False) printi(42); else printi(8);
 printi(17);
 return 0;
}
```

Test-if4.grc

```
int cond(bool b)
{
```

```
 int x;
 if (b)
   x = 42;
 else
   x = 17;
 return x;
}

int main()
{
printi(cond(True));
printi(cond(False));
return 0;
}
```

Test-if5.grc

```
int cond(bool b)
{
 int x;
 x = 10;
 if (b)
   if (x == 10)
     x = 42;
 else
   x = 17;
 return x;
}

int main()
{
printi(cond(True));
printi(cond(False));
return 0;
}
```

Test-if6.grc

```
void printf(double d){
   print(doubleToString(d));
}
void printb(bool b){
   if (b) printi(1);
```

```
   else printi(0);
}
double inf = infinity;
int main() {
   double inity = infinity;
   printf(inf);
   printf(inity);
   printf(inf + 4.0);
   printf(inf - 2.5);
   printf(inf * 3.9);
   printf(inf / 1.8);
   printf(inf + inity);
   printf(inf - inity);
   printf(inf * inity);
   printf(inf / inity);
   printf(1.0 / inf);
   printf(1.0 / 0.0);
   printf(1.0 - inf);
   printf(-inf);
   printb(inf == 4.0);
   printb(inf == inity);
   printb(4.0 == inf);
   printb(inf != 42.0);
   printb(inf != inity);
   printb(42.0 != inf);
   printb(inf > 10.0);
   printb(inf > inity);
   printb(10.0 > inf);
   printb(inf < 10.0);
   printb(inf < inity);
   printb(10.0 < inf);
   printb(inf >= 10.0);
   printb(inf >= inity);
   printb(10.0 >= inf);
   printb(inf <= 10.0);
   printb(inf <= inity);
   printb(10.0 <= inf);
   return 0;
}
```

Test-infinity.grc

```
int main(){
```

```
  bool success = True;

  Graph g = createGraph(10);
  String nodedata1 = "noded1";
  String nodedata2 = "noded2";
  String nodedata3 = "noded3";
  String nodedata4 = "noded4";
  String nodedata5 = "noded5";
  String nodedata6 = "noded6";
  String nodedata7 = "noded7";
  String nodedata8 = "noded8";
  String nodedata9 = "noded9";
  String nodedata10 = "noded10";
  Node node1 = g.createNode(nodedata1);
  Node node2 = g.createNode(nodedata2);
  Node node3 = g.createNode(nodedata3);
  Node node4 = g.createNode(nodedata4);
  Node node5 = g.createNode(nodedata5);
  Node node6 = g.createNode(nodedata6);
  Node node7 = g.createNode(nodedata7);
  Node node8 = g.createNode(nodedata8);
  Node node9 = g.createNode(nodedata9);
  Node node10 = g.createNode(nodedata10);

  IntTable it = createIntTable(5);

  // test insert with collisions
  it[node1] = 1;
  it[node2] = 2;
  it[node3] = 3;
  it[node4] = 4;
  it[node5] = 5;
  it[node6] = 6;
  it[node7] = 7;
  it[node8] = 8;
  it[node9] = 9;
  it[node10] = 10;
   it[node10] = 10; // insert same node twice -- node10 appended to end twice without
issue

  // test includes
  if (!it.inInt(node1) || !it.inInt(node7) || !it.inInt(node10)) {
```

```
        success = False;
    }

    // test get
    int get1 = it[node1];
    int get8 = it[node8];
    int get3 = it[node3];
    if ((get1 != 1) || (get8 != 8) || (get3 != 3)) {
        success = False;
    }

    // test delete
    it.deleteInt(node1);
    it.deleteInt(node2);
    it.deleteInt(node3);
    it.deleteInt(node4);
    it.deleteInt(node5);
    it.deleteInt(node6);
    it.deleteInt(node7);
    it.deleteInt(node8);
    it.deleteInt(node9);
    it.deleteInt(node10);
    it.deleteInt(node10); // try deleting same node (added to it twice) twice
    it.deleteInt(node1); // try deleting same node (added to it once) twice

    if (it.inInt(node1)) {
        success = False;
    }
    NodeList keys = it.intKeys();
    if (keys.length_NL() != 0) {
        success = False;
    }

    if (success) {
        print("SUCCESS");
    }
    else {
        print("FAILURE");
    }

    return 0;
}
```

Test-inttable.c

```
void foo(bool i)
{
 int i; /* Should hide the formal i */

 i = 42;
 printi(i + i);
}

int main()
{
 foo(True);
 return 0;
}
```

Test-local1.grc

```
int foo(int a, bool b)
{
 int c;
 bool d;

 c = a;

 return c + 10;
}

int main() {
printi(foo(37, False));
return 0;
}
```

Test-local2.grc

```
int main() {
   Node n = createGraph(5).createNode("blah");
   n.updateData(doubleToString(3.14));
   double d = 33.2;
   d.doubleToString().print();
   return 0;
}
```

Test-method.grc

```
int main() {
   int i = 10;
   printi(i % 2);
   printi(i % 3);
   printi(i % 7);
   printi(-15 % i);
}
```

Test-mod.grc

```
int main (){
   Graph g = createGraph(5);
   String greeting = "Hello";

   Node n = g.createNode(greeting);
   print(n.data());
}
```

Test-node1.grc

```
int main (){
   //this is for testing createNode
   Graph g = createGraph(5);
   String greeting = "Hello";

   Node n = g.createNode(greeting);
   print(n.data());

   //this is for testing updateData
   String hsm_forever = "zac efron";
   n.updateData(hsm_forever);
   print(n.data());
}
```

Test-node2.grc

```
int main() {
   Graph g = createGraph(10);
   g.createNode("1");
   g.createNode("2");
   g.createNode("3");
   g.createNode("4");
   g.createNode("5");
   for (Node n in g.nodes()) print(n.data());
   print("Here");
```

```
    return 0;
}
```

Test-nodefor1.grc

```
int main() {
    Graph g = createGraph(10);
    NodeList list = createNodeList();
    list.appendNode(g.createNode("one"));
    list.appendNode(g.createNode("two"));
    list.appendNode(g.createNode("three"));
    for (Node n in list){
        print(n.data());
    }
}
```

Test-nodefor2.grc

```
int main() {
    // NEED TO TEST NODE NEIGHBORS

    // create graph with 2 nodes and test the functions
    Graph g = createGraph(100);
    String greeting = "Hello";
    String nodedata1 = "noded1";
    String nodedata2 = "noded2";
    Node node1 = g.createNode(nodedata1);
    Node node2 = g.createNode(nodedata2);
    bool success = False;

    // test data function
    if ((node1.data().stringEquals("noded1")) && (node2.data().stringEquals("noded2")))
{
        success = True;
    }

    // test update data function
    node1 = node1.updateData(greeting);
    if (!node1.data().stringEquals("Hello")) {
        success = False;
    }

    // test visited function, that nodes are initialized to false
    if (node1.visited()) {
        success = False;
```

```
}

// test update visited
node1 = node1.updateVisited(True);
if (!node1.visited()) {
    success = False;
}

// test nodeequals
if (!node1.nodeEquals(node1)) {
    success = False;
}

if (node1.nodeEquals(node2)) {
    success = False;
}
// test the 4 cost functions
if (node1.cost() != -1.000000) {
    success = False;
}
node1.incrementCost();
if (node1.cost() != 0.000000) {
    success = False;
}
node1.decrementCost();
if (node1.cost() != -1.000000) {
    success = False;
}
node1.updateCost(5.0);
if (node1.cost() != 5.000000) {
    success = False;
}

// test setPrec and prec
node1 = node1.setPrec(node2);
if (!node1.prec().data().stringEquals(node2.data())) {
    success = False;
}
String data1 = node1.prec().data();
if (!data1.stringEquals("noded2")) {
    success = False;
}
```

```
    // test getEdge
    Edge edge1 = g.addEdge(node1, node2, 1.7);
    Edge edge2 = g.addEdge(node2, node1, 6.8);
    if (!edge1.edgeEquals(node1.getEdge(node2))) {
        success = False;
    }
    if (!edge2.edgeEquals(node2.getEdge(node1))) {
        success = False;
    }

    if (success) {
        print("SUCCESS");
    } else {
        print("FAILURE");
    }
}
```

Test-nodefull.grc

```
void printNodeList(NodeList nl) {
    /*
    for (Node n in nl) {
        print(n.data());
    }*/
    int i = nl.length_NL();
    while (i > 0){
        Node n = nl.removeFirst_NL();
        print(n.data());
        nl.appendNode(n);
        i = i - 1;
    }
}


int main()
{
    Graph g = createGraph(5);
    String greeting = "Hello";
    String nodedata1 = "noded1";
    String nodedata2 = "noded2";
    Node n1 = createNode(g, nodedata1);
```

```
Node n2 = createNode(g, nodedata2);
Node n3 = createNode(g, greeting);
bool success = True;

NodeList nl = createNodeList();
if (!nl.empty_NL()) {
    success = False;
}

nl.appendNode(n1);
if (!(nl.length_NL() == 1)) {
    success = False;
}
nl.prependNode(n2);
if (!(nl.length_NL() == 2)) {
    success = False;
}
if (!nl.includesNode(n2)) {
    success = False;
}
if (!nl.includesNode(n1)) {
    success = False;
}

nl.removeFirst_NL();
if (!nl.includesNode(n1)) {
    success = False;
}
if (nl.includesNode(n2)) {
    success = False;
}

nl.prependNode(n2);
nl.removeLast_NL();
if (!nl.includesNode(n2)) {
    success = False;
}
if (nl.includesNode(n1)) {
    success = False;
}

nl.appendNode(n1);
```

```
   nl.appendNode(n3);
   nl.removeNode(n1);
   nl.removeNode(n3);
   nl.removeNode(n2);
   if (!nl.empty_NL()) {
       success = False;
   }

   if (success) {
       print("SUCCESS");
   }
   else {
       print("FAILURE");
   }

   return 0;
}
```

Test-nodelistfull.grc

```
void printb(bool b){
 if (b){
   printi(1);
 }
 else {
   printi(0);
 }
}
int main()
{
 printi(1 + 2);
 printi(1 - 2);
 printi(1 * 2);
 printi(100 / 2);
 printi(99);
 printb(1 == 2);
 printb(1 == 1);
 printi(99);
 printb(!(1 == 2));
 printb(!(1 == 1));
 printi(99);
 printb(1 < 2);
 printb(2 < 1);
```

```
 printi(99);
 printb(1 <= 2);
 printb(1 <= 1);
 printb(2 <= 1);
 printi(99);
 printb(2 < 1);
 printb(1 < 2);
 printi(99);
 printb(2 <= 1);
 printb(1 <= 1);
 printb(1 <= 2);
 return 0;
}
```

Test-ops1.grc

```
void printb(bool b){
 if (b){
   printi(1);
 }
 else {
   printi(0);
 }
}
int main()
{
 printb(True);
 printb(False);
 printb(True && True);
 printb(True && False);
 printb(False && True);
 printb(False && False);
 printb(True || True);
 printb(True || False);
 printb(False || True);
 printb(False || False);
 printb(!False);
 printb(!True);
 printi(-10);
 printi(--42);
}
```

Test-ops2.grc

```
void printb(bool b){
 if (b){
   printi(1);
 }
 else {
   printi(0);
 }
}
int main()
{
 printi(1 + 2);
 printi(1 - 2);
 printi(1 * 2);
 printi(100 / 2);
 printi(99);
 printb(1 == 2);
 printb(1 == 1);
 printi(99);
 printb(1 != 2);
 printb(1 != 1);
 printi(99);
 printb(1 < 2);
 printb(2 < 1);
 printi(99);
 printb(1 <= 2);
 printb(1 <= 1);
 printb(2 <= 1);
 printi(99);
 printb(2 < 1);
 printb(1 < 2);
 printi(99);
 printb(2 <= 1);
 printb(1 <= 1);
 printb(1 <= 2);

 printb(1.0 == 3.0);
 printb(1.0 == 1.00);
 printb(1.0 != 1.00);
 printb(1.0 != 3.0);
 printb(1 != -1);
```

```
 printb(1.0 != -1.0);

 printb(1 > 4);
 printb(4 > 1);
 printb(4.0 > 1.0);
 printb(1.0 > 4.0);
 printb(1 > -1);
 printb(1.0 > 1.0);
 printb(4.0 > -4.0);

 return 0;
}
```

Test-ops3.grc

```
void printb(bool b){
 if (b){
   printi(1);
 }
 else {
   printi(0);
 }
}
int main()
{
 printb(1 >= 2);
 printb(1 >= 1);
 printb(2 >= 1);
 printi(99);

 printb(4.0 >= 1.0);
 printb(1.0 >= 4.0);
 printb(2. >= -1.0);
 printb(1 >= -1);
 printb(1.0 >= 1.0);
 printb(4.0 >= -4.0);
 printb(0.00 >= -0.);
 printb(4 >= -0);
 printb(-4 >= -0);
 printb(-1.0 >= -0.00000);

 return 0;
```

```
}
```

Test-ops4.grc

```
int main() {
   int big = 2147483647;
   int small = 1;
   printi(big);
   printi(big+small); //This results in overflow
   return 0;
}
```

Test-overflow.grc

```
int main(){
   int i = 2;
   if (i > 0){
       double i = 3.14;
       print(doubleToString(i));
   }
}
```

Test-scope1.grc

```
int a = 5;
void foo (int a) {
   printi(a); //hides global
   int a = 10;
   printi(a); //hides formal
}
int main() {
   printi(a);
   foo(7);
   foo(a);
}
```

Test-scope2.grc

```
int main() {
   if (True) {
       int a = 42; //checks variable declaration/initialization in blocks
       printi(a);
       int b;
       b = 13;
       printi(b);
```

```
    }
    else {
        int a = 43;
    }

}
```

Test-scope3.grc

```
int main() {
    String s = "Hello";
    String t = "hello";
    String empty = "";
    printi(s.stringLength());
    printi(empty.stringLength());
    if (s.stringEquals(t)) print("Equals");
    else print("Not Equals");
    if (s.stringEquals("Hello")) print("Equals");
    else print("Not Equals");
}
```

Test-string.grc

```
int main() {
    Graph g = createGraph(1);
    Node n = g.createNode("Node data");
    print(n.data());
    synch n {
        n.updateData("Yo");
        print(n.data());
    }
    return 0;
}
```

Test-synch.grc

```
int main() {
    Graph g = createGraph(10);
    g.createNode("A");
    g.createNode("B");
    g.createNode("C");
    IntTable it = createIntTable(10);
    int i = 1;
    for (Node n in g.nodes()){
        it[n] = i;
        i = i + 1;
```

```
   }
   for (Node n in g.nodes()){
       printi(it[n]);
   }

}
```

Test-tableloop.grc

```
int main()
{
 printi(42);
 return 0;
}
```

Test-var1.grc

```
int a;

void foo(int c)
{
 a = c + 42;
}

int main()
{
 foo(73);
 printi(a);
 return 0;
}
```

Test-var2.grc

```
int main()
{
 int i;
 i = 5;
 while (0 < i) {
   printi(i);
   i = i - 1;
 }
 printi(42);
 return 0;
}
```

Test-while1.grc

```
int foo(int a)
{
 int j;
 j = 0;
 while (0 < a) {
   j = j + 2;
   a = a - 1;
 }
 return j;
}

int main()
{
 printi(foo(7));
 return 0;
}
```

Test-while2.grc

TESTS IN C

```
#!/bin/sh

# clean the directory and make the targets again
make clean
make all

out_dir="outputs"

for i in $(find . -perm +111 -type f);
   do $i > "$out_dir"/"$i"_output.txt
   if cmp -s "$out_dir"/"$i"_output.txt "$out_dir"/"$i"_example.txt; then
       echo " $i ....SUCCESS"
   else
       echo " $i ....FAILURE"
   fi
done
```

Test_script.sh

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "../../graclstdlib.c"
#include "print-functions.c"

int counter = 0;

void* testSync(void *arg)
{
   struct Node* node1 = (struct Node*) arg;
   _synch_start(arg);

   unsigned long i = 0;
   counter += 1;
   printf("Thread %d started\n", counter);
   printf("Printing the node: %s\n", node1->data);
   updateData(node1, "i'm an updated node\n");
   printf("Printing the node after data has been updated: %s\n", node1->data);

   for(i=0; i<(0xFFFFFFFF);i++);

   printf("Thread %d finished\n", counter);

   _synch_end(arg);

   return NULL;
}

int main(){
   // testing of synch
   struct Node* node1;
   struct Graph* g;
   int err;
   int i = 0;
   char hello[] = "Hello\n";
   pthread_t tid[2];

   g = createGraph(10);
   node1 = createNode(g, hello);
```

```
    pthread_mutex_t* pm = _convert_to_lockable((void*)node1);

    while(i < 2)
    {
        err = pthread_create(&(tid[i]), NULL, &testSync, (void*)pm);
        if (err != 0)
            printf("\ncan't create thread :[%s]", strerror(err));
        i++;
    }

    hatch_end(tid, 2);
    synch_destroy(pm);

    return 0;
}
```

Concurrency-test.c

```
#include <stdlib.h>
#include <stdio.h>
#include "../../graclstdlib.c"
#include "print-functions.c"

//      Loosely      based      off      of      the      Java      implementation      here:
https://www.baeldung.com/java-dijkstra

// Find the unsettled node the shortest distance away
struct Node* getCheapestNode(struct NodeList* unsettledNodes) {
    struct Node* cheapestNode = NULL;
     double cheapestCost = 10000000000000000000000.0; // had trouble with infinity so
going with this as max
    struct NodeListItem* current = unsettledNodes->head;
    while (current) {
        if (current->node->cost < cheapestCost) {
            cheapestCost = current->node->cost;
            cheapestNode = current->node;
        }
        current = current->next;
    }
    return cheapestNode;
}
```

```
// Recurse through goal node's precursors to start node to get full path
struct NodeList* getShortestPath(struct Node* source, struct Node* goal) {
   struct NodeList* path = createNodeList();
   struct Node* current = goal;
   if(nodeEquals(source, goal)){
       prependNode(path, current);
       return path;
   }
   while (current->precursor) {
       //printf("current%s\n", current->data);
       //printf("precursor%s\n", current->precursor->data);
       prependNode(path, current);
       if(nodeEquals(current, source)){
           //printf("inside if block %s\n", current->data);
           return path;
       }
       current = current->precursor;

   }
   return path;
}

// Main Dijkstra method
struct NodeList* dijkstra(struct Graph* g, struct Node* source, struct Node* goal) {
   updateCost(source, 0.0);
   struct NodeList* settledNodes = createNodeList(); // Nodes with known cheapest cost
    struct NodeList* unsettledNodes = createNodeList(); // Frontier nodes with unknown
cheapest cost
   appendNode(unsettledNodes, source);
   while (!empty_NL(unsettledNodes)) {
       struct Node* currentNode = getCheapestNode(unsettledNodes);
       if (includesNode(unsettledNodes, currentNode)) {
           removeNode(unsettledNodes, currentNode);
       }
       // Iterate through neighbors to find minimum distance and add them to unsettled
       struct EdgeListItem* currentEdgeItem = currentNode->edges->head;
       while (currentEdgeItem) {
                                  //printf("printing   current   explored   node   %s\n",
currentEdgeItem->edge->end->data);
           struct Edge* currentEdge = currentEdgeItem->edge;
           struct Node* adjacentNode = currentEdge->end;
```

```
            if (!includesNode(settledNodes, adjacentNode)) {
                 // If it's a shorter path to adjacentNode, update adjacentNode cost and
prec
                double newCost = currentNode->cost + currentEdge->weight;
                //printf("current node is ");
                //printf("%s\n", currentNode->data);
                /*
                printf("current node cost\n");
                printf("%f\n", cost(currentNode));
                printf("edgeweight");
                printf("%f\n", newCost);
                printf("adjacent node cost");
                printf("%f\n", cost(adjacentNode)); */

                if (adjacentNode->cost == -1.0 || newCost < adjacentNode->cost) {
                    updateCost(adjacentNode, newCost);
                    setPrec(adjacentNode, currentNode);
                }
                // Add neighbors to unsettled
                if (!includesNode(unsettledNodes, adjacentNode)) {
                    appendNode(unsettledNodes, adjacentNode);
                }
            }
            currentEdgeItem = currentEdgeItem->next;
        }
        appendNode(settledNodes, currentNode);
    }
    //return nothing if goal node is unsettled, i.e. not seen
    //printf("settled\n");
    //printNodeList(settledNodes);
    //printf("\nunsettled\n");
    //printNodeList(unsettledNodes);
    if (!includesNode(settledNodes, goal)){
        //return empty nodelist
        return createNodeList();
    }
    //else return shortest path
    return getShortestPath(source, goal);
}

// Print data of each node (start to finish) for testing
void printPathNodeData(struct NodeList* path){
```

```
    //printf("FINAL\n");
    struct NodeListItem* current = path->head;
    while (current) {
        printf("%s ", current->node->data);
        current = current->next;
    }
    printf("\n");
}

// TEST!
int main() {
    struct Graph* g = createGraph(10);

    struct Node* nodeA = createNode(g, "A");
    struct Node* nodeB = createNode(g, "B");
    struct Node* nodeC = createNode(g, "C");
    struct Node* nodeD = createNode(g, "D");
    struct Node* nodeE = createNode(g, "E");
    struct Node* nodeF = createNode(g, "F");

    addEdge(g, nodeA, nodeB, 10.0);
    addEdge(g, nodeA, nodeC, 15.0);
    addEdge(g, nodeB, nodeD, 12.0);
    addEdge(g, nodeB, nodeF, 15.0);
    addEdge(g, nodeC, nodeE, 10.0);
    addEdge(g, nodeD, nodeE, 2.0);
    addEdge(g, nodeD, nodeF, 1.0);
    addEdge(g, nodeF, nodeE, 5.0);

    printf("Path should be A-B-D-F\n");
    printPathNodeData(dijkstra(g, nodeA, nodeF));
    printf("\n");

    printf("Path should be A-B-D-E\n");
    printPathNodeData(dijkstra(g, nodeA, nodeE));
    printf("\n");

    printf("Path should be E\n");
    printPathNodeData(dijkstra(g, nodeE, nodeE));
    printf("\n");

    printf("Path should not exist\n");
```

```
    printPathNodeData(dijkstra(g, nodeB, nodeC));
    printf("\n");

    printf("Path should be A-B\n");
    printPathNodeData(dijkstra(g, nodeA, nodeB));
    printf("\n");

    printf("Path should not exist\n");
    printPathNodeData(dijkstra(g, nodeF, nodeA));
    printf("\n");

    printf("Path should be D-E\n");
    printPathNodeData(dijkstra(g, nodeD, nodeE));
    printf("\n");

    printf("Path should be B-D-F\n");
    printPathNodeData(dijkstra(g, nodeB, nodeF));
    printf("\n");

    printf("Path should be B-D-E\n");
    printPathNodeData(dijkstra(g, nodeB, nodeE));
    printf("\n");


    printf("Path should be D\n");
    printPathNodeData(dijkstra(g, nodeD, nodeD));
    printf("\n");

    printf("Path should be C\n");
    printPathNodeData(dijkstra(g, nodeC, nodeC));
    printf("\n");

    return 0;
}
```

Dijkstra-test.c

```
#include <stdlib.h>
#include <stdio.h>
#include "../../graclstdlib.c"
#include "print-functions.c"

int main(){
```

```
bool success = true;

struct Graph* g = createGraph(10);
char nodedata1[] = "noded1";
char nodedata2[] = "noded2";
char nodedata3[] = "noded3";
char nodedata4[] = "noded4";
char nodedata5[] = "noded5";
char nodedata6[] = "noded6";
char nodedata7[] = "noded7";
char nodedata8[] = "noded8";
char nodedata9[] = "noded9";
char nodedata10[] = "noded10";
struct Node* node1 = createNode(g, nodedata1);
struct Node* node2 = createNode(g, nodedata2);
struct Node* node3 = createNode(g, nodedata3);
struct Node* node4 = createNode(g, nodedata4);
struct Node* node5 = createNode(g, nodedata5);
struct Node* node6 = createNode(g, nodedata6);
struct Node* node7 = createNode(g, nodedata7);
struct Node* node8 = createNode(g, nodedata8);
struct Node* node9 = createNode(g, nodedata9);
struct Node* node10 = createNode(g, nodedata10);

struct DoubleTable* dt = createDoubleTable(5);

// test hashcode
if(hashCode_dt(dt, node1) != (node1->id % dt->size)){
    success = false;
}
if(hashCode_dt(dt, node2) != (node2->id % dt->size)){
    success = false;
}
if(hashCode_dt(dt, node3) != (node3->id % dt->size)){
    success = false;
}

// test insert with collisions
_insertDouble(dt, node1, 1.0);
_insertDouble(dt, node2, 2.0);
_insertDouble(dt, node3, 3.0);
_insertDouble(dt, node4, 4.0);
```

```
   _insertDouble(dt, node5, 5.0);
   _insertDouble(dt, node6, 6.0);
   _insertDouble(dt, node7, 7.0);
   _insertDouble(dt, node8, 8.0);
   _insertDouble(dt, node9, 9.0);
   _insertDouble(dt, node10, 10.0);
    _insertDouble(dt, node10, 10.0); // insert same node twice -- node10 appended to
end twice without issue

   // test inserting a node from a different graph
   /*
   struct Graph* g2 = createGraph(10);
   char nodedata11[] = "noded11";
   struct Node* node11 = createNode(g2, nodedata11);
   _insertDouble(dt, node11, 11.0); // GIVES AN ERROR (as it should)
   */

   // test includes
   if (!inDouble(dt, node1) || !inDouble(dt, node7) || !inDouble(dt, node10)) {
       success = false;
   }

   // test get
   double get1 = _getDouble(dt, node1);
   double get8 = _getDouble(dt, node8);
   double get3 = _getDouble(dt, node3);
   if ((get1 != 1.0) || (get8 != 8.0) || (get3 != 3.0)) {
       success = false;
   }

   removeNodeGraph(g, node4);
   if(!inDouble(dt, node4)){
       success = false;
   }
   if (_getDouble(dt, node4) != 4.0) {
       success = false;
   }
   // line should fault out
   //_insertDouble(dt, node4, 18.0);
   deleteDouble(dt, node4);
   if (inDouble(dt, node4)) {
       success = false;
```

```
    }

    // test delete
    deleteDouble(dt, node1);
    deleteDouble(dt, node2);
    deleteDouble(dt, node3);
    deleteDouble(dt, node4);
    deleteDouble(dt, node5);
    deleteDouble(dt, node6);
    deleteDouble(dt, node7);
    deleteDouble(dt, node8);
    deleteDouble(dt, node9);
    deleteDouble(dt, node10);
    deleteDouble(dt, node10); // try deleting same node (added to dt twice) twice
    deleteDouble(dt, node1); // try deleting same node (added to dt once) twice

    if (inDouble(dt, node1)) {
        success = false;
    }
    if (length_NL(doubleKeys(dt)) != 0) {
        success = false;
    }

    // test making an dt with a size of 0 or smaller
    //struct DoubleTable* dt2 = createDoubleTable(0);
    //struct DoubleTable* dt3 = createDoubleTable(-1);

    if (success) {
        printf("SUCCESS\n");
    }
    else {
        printf("FAILURE\n");
    }

    return 0;
}
```

Doubletable-test.c

```
#include <stdlib.h>
#include <stdio.h>
#include "../../graclstdlib.c"
#include "print-functions.c"
```

```
int main() {
  // create graph with 2 nodes and test the functions
   struct Graph* g = createGraph(100);
   char nodedata1[] = "noded1";
   char nodedata2[] = "noded2";
   char nodedata3[] = "noded3";

   struct Node* node1 = createNode(g, nodedata1);
   struct Node* node2 = createNode(g, nodedata2);
   struct Node* node3 = createNode(g, nodedata3);

   struct Edge* edge1 = addEdge(g, node1, node2, 1.7);
   struct Edge* edge2 = addEdge(g, node2, node3, 6.8);
   struct Edge* edge3 = addEdge(g, node1, node2, 1.7);

   bool success = true;

   char bits[] = {0, 0, 0, 0, 0, 0, 0xf0, 0x7f};
   double infinity = *(double*)bits;

   // test updateEdge and weight functions
   updateEdge(edge2, 16.3);
   if (edge2->weight != 16.300000) {
       success = false;
   }
   updateEdge(edge2, 90.1);
   if (edge2->weight != 90.100000) {
       success = false;
   }
   updateEdge(edge2, -90.1);
   if (edge2->weight != -90.100000) {
       success = false;
   }
   updateEdge(edge2, infinity);
   if (edge2->weight != infinity) {
       success = false;
   }

   // test edgeEquals
   if(edgeEquals(edge1, edge2)) {
       success = false;
   }
```

```
    if(!edgeEquals(edge1, edge1)) {
        success = false;
    }

    // test end and start functions
    if(strcmp(end(edge1)->data, start(edge2)->data) != 0) {
        success = false;
    }
    if(strcmp(start(edge1)->data, start(edge3)->data) != 0) {
        success = false;
    }
    if(strcmp(start(edge1)->data, end(edge1)->data) == 0) {
        success = false;
    }

    removeNodeGraph(g, node1);
    //start(edge1);
    //end(edge3);
    //updateEdge(edge1, 15.0);
    //double bad_dub = weight(edge3);
    double good_dub = weight(edge2);
    printf("%f\n", good_dub);
    removeNodeGraph(g, node3);
    //start(edge2);
    //double bad_dub2 = weight(edge2);
    //edgeEquals(edge1, edge2);

    if (success) {
        printf("SUCCESS\n");
    } else {
        printf("FAILURE\n");
    }
}
```

Edge-test.c

```
#include <stdlib.h>
#include <stdio.h>
#include "../../graclstdlib.c"
#include "print-functions.c"

int main() {
    //Make two nodes to be start and end for edge
```

```
char hello[] = "Hello\n";
char goodbye[] = "Goodbye\n";
char yo[] = "Yo\n";
struct Graph* g = createGraph(2);
struct Node* n1 = createNode(g, hello);
struct Node* n2 = createNode(g, goodbye);
struct Node* n3 = createNode(g, yo);
struct Edge* e1 = addEdge(g, n1, n2, 15.3);
struct Edge* e2 = addEdge(g, n2, n1, 3.7);
struct Edge* e3 = addEdge(g, n3, n2, 4.8);
bool success = true;

struct EdgeList* el = createEdgeList();
if (!empty_EL(el)) {
    success = false;
}

appendEdge(el, e1);
prependEdge(el, e2);
if (!(length_EL(el)==2)) {
    success = false;
}

prependEdge(el, e3);
if (!(length_EL(el)==3)) {
    success = false;
}

removeFirst_EL(el);
printf("After remove first (list should be goodbye hello): ");
printEdgeList(el);
printf("\n");
if (!(length_EL(el)==2)) {
    success = false;
}

prependEdge(el, e3);
removeLast_EL(el);
printf("After remove last (list should be yo goodbye): ");
printEdgeList(el);
printf("\n");
if (!(length_EL(el)==2)) {
```

```
    success = false;
}

removeEdge(el, e1);
printf("After remove edge (list should be yo goodbye): ");
printEdgeList(el);
printf("\n");
if (!(length_EL(el)==2)) {
    success = false;
}

removeEdge(el, e2);
if (!(length_EL(el)==1)) {
    success = false;
}

removeEdge(el, e3);
if (!empty_EL(el)) {
    success = false;
}

struct Edge* e4 = addEdge(g, n3, n2, 4.8);
struct Edge* e5 = addEdge(g, n2, n3, 4.8);
struct Edge* e6 = addEdge(g, n1, n2, 4.8);
appendEdge(el, e4);
appendEdge(el, e5);
appendEdge(el, e6);
removeEdgeGraph(g, e4);
printEdge(head_EL(el));
printf("\n");

removeEdge(el, e4);
removeEdgeGraph(g, e5);
removeFirst_EL(el);
//Breaks the program, should print prependEdge: Edge  deleted
//prependEdge(el, e4);

//Breaks the program, should print appendEdge: Edge  deleted
//appendEdge(el, e4);
printEdge(head_EL(el));
printf("\n");
```

```
    //removeEdge(el, e5);
    printEdge(tail_EL(el));
    printf("\n");

    removeEdge(el, e6);

    //this will give an error because it's empty
    //printEdge(head_EL(el));

    if (success) {
        printf("SUCCESS\n");
    }
    else {
        printf("FAILURE\n");
    }

    return 0;
}
```

Edgelist-test.c

```
#include <stdlib.h>
#include <stdio.h>
#include "../../graclstdlib.c"
#include "print-functions.c"


int main() {
    struct Node* node1;
    struct Node* node2;
    struct Node* node3;
    struct Edge* e1;
    struct Edge* e2;
    struct Edge* e3;
    struct Edge* e4;
    struct Edge* e5;
    struct Edge* e6;
    struct Edge* e7;
    struct Edge* e8;
    struct Edge* e9;
    struct Edge* e10;
    struct Edge* e11;
```

```
struct NodeList* nl1;
struct Graph* g;
struct NodeList* nl2;
bool success = true;

g = createGraph(3);
node1 = createNode(g, "A");
node2 = createNode(g, "B");
node3 = createNode(g, "C");

// add more than the size of the graph to make
// sure the size of the graph is doubled
struct Node* node4 = createNode(g, "D");
struct Node* node5 = createNode(g, "E");
struct Node* node6 = createNode(g, "F");
struct Node* node7 = createNode(g, "G");
struct Node* node8 = createNode(g, "H");

nl1 = nodes(g);
if (!(length_NL(nl1) == 8)) {
    success = false;
}

e1 = addEdge(g, node1, node2, 14.0);
e2 = addEdge(g, node2, node1, 13.0);
e3 = addEdge(g, node1, node3, 43.0);
e4 = addEdge(g, node2, node3, 43.0);
e5 = addEdge(g, node3, node2, 43.0);
e6 = addEdge(g, node3, node1, 43.0);
e7 = addEdge(g, node1, node8, 23.0);
e8 = addEdge(g, node1, node7, 23.0);
e9 = addEdge(g, node1, node6, 23.0);
e10 = addEdge(g, node8, node4, 23.0);
e11 = addEdge(g, node5, node4, 23.0);

if (!(length_EL(edges(node1)) == 5)) {
    success = false;
}

removeNodeGraph(g, node3);
//Breaks the program, should print edges: Node deleted
//struct EdgeList* el1 = edges(node3);
```

```
//printEdgeList(el1);

//Breaks the program, should print weight:Edge deleted
//double weight_e3 = weight(e3);
//printf("%f", weight_e3);

//Breaks the program, should print weight:Edge deleted
//double weight_e5 = weight(e5);
//printf("%f", weight_e5);

if (!(length_NL(neighbors(node1)) == 4)) {
    success = false;
}
if (!(length_EL(edges(node1)) == 4)) {
    success = false;
}
nl2 = nodes(g);
if (!(length_NL(nl2) == 7)) {
    success = false;
}

removeEdgeGraph(g, e11);
if (!empty_EL(edges(node5))) {
    success = false;
}

//addEdge(g, node1, node2, 5.0); // this line causes a MEM ERROR (CHECK FOR THIS)
removeNodeGraph(g, node1);
if (node1->id != 0) {
    success = false;
}
// Should print: edges: Node deleted
//if (!empty_EL(edges(node1))) {
    //success = false;
//}

printNodeList(nodes(g));

// what happens to node / edge after removed from graph
removeNodeGraph(g, node2);
removeNodeGraph(g, node4);
removeNodeGraph(g, node5);
```

```
removeNodeGraph(g, node6);
removeNodeGraph(g, node7);
removeNodeGraph(g, node8);
if (!empty_NL(nodes(g))) {
    success = false;
}

// Check that the user can't add back node1
node8 = createNode(g, "C");
if (node8->id != 8) {
    success = false;
}
node7 = createNode(g, "H");
e11 = addEdge(g, node8, node7, 23.0);

//should print C
printf("%s\n", data(start(e11)));

// Remove the edge and make sure nothing happened to the node
removeEdgeGraph(g, e11);

//Breaks the program, should print start: Edge deleted
//printf("%s\n", data(start(e11)));
printf("%s\n", data(node7));

//Breaks the program, should print addEdge: Start or end node deleted
//e11 = addEdge(g, node5, node4, 23.0);

printNodeList(nodes(g));

//Breaks the program, should print nodeEquals: Node deleted
//removeNodeGraph(g, node3);

// test making an graph with a size of 0 or smaller
// Breaks the program, Error: Graph must have be at least size 1 or larger
//struct Graph* g2 = createGraph(0);
//struct Graph* g3 = createGraph(-1);

if (success) {
    printf("SUCCESS\n");
} else {
    printf("FAILURE\n");
```

```
    }

    return 0;
}
```

Graph-test.c

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include "../../graclstdlib.c"
#include "print-functions.c"

struct hatchArgs {
    struct Node* n;
};

int start_routine(struct Node* n){
    printf("%s", data(n));
    return 0;
}

void *hatch_unwrapper(void *args) {
    struct hatchArgs *fargs = (struct hatchArgs *) args;

    start_routine(fargs->n);
    return NULL;
}

int main(){
    struct Graph* g;
    struct Node* node1;
    struct Node* node2;
    struct Node* node3;
    struct Node* node4;
    struct Node* node5;
    struct Node* node6;
    struct Node* node7;
    struct Node* node8;
    struct Node* node9;
    struct NodeList* nl;
    int i = 0;
```

```
    g = createGraph(10);
    node1 = createNode(g, "A");
    node2 = createNode(g, "B");
    node3 = createNode(g, "C");
    node4 = createNode(g, "D");
    node5 = createNode(g, "E");
    node6 = createNode(g, "F");
    node7 = createNode(g, "G");
    node8 = createNode(g, "H");
    node9 = createNode(g, "I");
    nl = nodes(g);
    int len_possible_start = length_NL(nl);

    pthread_t* threads = malloc(sizeof(pthread_t) * len_possible_start);

    struct NodeListItem* currentNode = nl->head;

    while(currentNode){
        pthread_create(threads + i, NULL, hatch_unwrapper, currentNode);
        currentNode = currentNode->next;
        i++;
    }

    hatch_end(threads, len_possible_start);
}
```

Hatch-test.c

```
#include <stdlib.h>
#include <stdio.h>
#include "../../graclstdlib.c"
#include "print-functions.c"

int main(){
    bool success = true;

    struct Graph* g = createGraph(10);
    char nodedata1[] = "noded1";
    char nodedata2[] = "noded2";
    char nodedata3[] = "noded3";
    char nodedata4[] = "noded4";
    char nodedata5[] = "noded5";
    char nodedata6[] = "noded6";
```

```
   char nodedata7[] = "noded7";
   char nodedata8[] = "noded8";
   char nodedata9[] = "noded9";
   char nodedata10[] = "noded10";
   struct Node* node1 = createNode(g, nodedata1);
   struct Node* node2 = createNode(g, nodedata2);
   struct Node* node3 = createNode(g, nodedata3);
   struct Node* node4 = createNode(g, nodedata4);
   struct Node* node5 = createNode(g, nodedata5);
   struct Node* node6 = createNode(g, nodedata6);
   struct Node* node7 = createNode(g, nodedata7);
   struct Node* node8 = createNode(g, nodedata8);
   struct Node* node9 = createNode(g, nodedata9);
   struct Node* node10 = createNode(g, nodedata10);

   struct IntTable* it = createIntTable(5);

   // test hashcode
   if(hashCode_it(it, node1) != (node1->id % it->size)){
       success = false;
   }
   if(hashCode_it(it, node2) != (node2->id % it->size)){
       success = false;
   }
   if(hashCode_it(it, node3) != (node3->id % it->size)){
       success = false;
   }

   // test insert with collisions
   _insertInt(it, node1, 1);
   _insertInt(it, node2, 2);
   _insertInt(it, node3, 3);
   _insertInt(it, node4, 4);
   _insertInt(it, node5, 5);
   _insertInt(it, node6, 6);
   _insertInt(it, node7, 7);
   _insertInt(it, node8, 8);
   _insertInt(it, node9, 9);
   _insertInt(it, node10, 10);
    _insertInt(it, node10, 10); // insert same node twice -- node10 appended to end
twice without issue
```

```
// test inserting a node from a different graph
/*
struct Graph* g2 = createGraph(10);
char nodedata11[] = "noded11";
struct Node* node11 = createNode(g2, nodedata11);
_insertInt(it, node11, 11.0); // GIVES AN ERROR (as it should)
*/

// test includes
if (!inInt(it, node1) || !inInt(it, node7) || !inInt(it, node10)) {
    success = false;
}

// test get
double get1 = _getInt(it, node1);
double get8 = _getInt(it, node8);
double get3 = _getInt(it, node3);
if ((get1 != 1) || (get8 != 8) || (get3 != 3)) {
    success = false;
}

// test delete
deleteInt(it, node1);
deleteInt(it, node2);
deleteInt(it, node3);
deleteInt(it, node4);
deleteInt(it, node5);
deleteInt(it, node6);
deleteInt(it, node7);
deleteInt(it, node9);
deleteInt(it, node10);
deleteInt(it, node10); // try deleting same node (added to it twice) twice
deleteInt(it, node1); // try deleting same node (added to it once) twice

if (inInt(it, node1)) {
    success = false;
}
if (length_NL(intKeys(it)) != 1) {
    success = false;
}

// test making an dt with a size of 0 or smaller
```

```
    //struct IntTable* it2 = createIntTable(0);
    //struct IntTable* it3 = createIntTable(-1);

    removeNodeGraph(g, node8);
    if (!inInt(it, node8)) {
        success = false;
    }
    if (_getInt(it, node8) != 8) {
        success = false;
    }
    // line should fault out
    //_insertInt(it, node8, 18);
    deleteInt(it, node8);
    if (inInt(it, node8)) {
        success = false;
    }

    if (success) {
        printf("SUCCESS\n");
    }
    else {
        printf("FAILURE\n");
    }

    return 0;
}
```

Inttable-test.c

```
#include <stdlib.h>
#include <stdio.h>
#include "../../graclstdlib.c"
#include "print-functions.c"

int main() {
    // NEED TO TEST NODE NEIGHBORS

    // create graph with 2 nodes and test the functions
    struct Graph* g = createGraph(100);
    char greeting[] = "Hello\n";
    char nodedata1[] = "noded1";
    char nodedata2[] = "noded2";
    struct Node* node1 = createNode(g, nodedata1);
```

```
struct Node* node2 = createNode(g, nodedata2);
bool success = false;

// test data function
if (!(strcmp(data(node1), "noded1") && strcmp(data(node2), "noded2"))) {
    success = true;
}

// test update data function
node1 = updateData(node1, greeting);
if ((strcmp(node1->data, "Hello\n") != 0)) {
    success = false;
}

// test visited function, that nodes are initialized to false
if (visited(node1)) {
    success = false;
}

// test update visited
if (!visited(updateVisited(node1, true))) {
    success = false;
}

// test nodeequals
if (!nodeEquals(node1, node1)) {
    success = false;
}

if (nodeEquals(node1, node2)) {
    success = false;
}
// test the 4 cost functions
if (cost(node1) != -1) {
    success = false;
}
incrementCost(node1);
if (cost(node1) != 0) {
    success = false;
}
decrementCost(node1);
if (cost(node1) != -1) {
```

```
    success = false;
}
updateCost(node1, 5);
if (cost(node1) != 5) {
    success = false;
}

// test the two precursors
if (!prec(node1)) { // should be null when inited
    success = false;
}
node1 = setPrec(node1, node2);
if (prec(node1) == NULL) {
    success = false;
}
const char* data1;
if (prec(node1) == NULL) {
    success = false;
}
data1 = data(prec(node1));
if (data1 == NULL) {
    success = false;
}
if (strcmp(data1, "noded2") != 0) {
    success = false;
}

// test getEdge
struct Edge* edge1 = addEdge(g, node1, node2, 1.7);
struct Edge* edge2 = addEdge(g, node2, node1, 6.8);
if (!edgeEquals(edge1, getEdge(node1, node2))) {
    success = false;
}
if (!edgeEquals(edge2, getEdge(node2, node1))) {
    success = false;
}
if (getEdge(node1, node1)) {
    success = false;
}

removeNodeGraph(g, node1);
removeNodeGraph(g, node2);
```

```
    // these (as desired, error out)
    //struct Node* parent = prec(node1);
    //setPrec(node1, node2);
    //char* dat = data(node1);
    //bool vis = visited(node2);
    //struct Edgelist* edge = edges(node1);
    //double cost1 = cost(node2);
    //incrementCost(node2);
    //decrementCost(node1);
    //updateCost(node1, 3.0);
    //struct Edge* e = getEdge(node1, node2);
    //bool bad_test = nodeEquals(node1, node2);
    //updateVisited(node1, true);
    //updateData(node2, "hi");


    if (success) {
        printf("SUCCESS\n");
    } else {
        printf("FAILURE\n");
    }
}
```

Node-test.c

```
#include <stdlib.h>
#include <stdio.h>
#include "../../graclstdlib.c"
#include "print-functions.c"

int main()
{
    //Make two nodes
    struct Graph* g = createGraph(100);
    char greeting[] = "Hello\n";
    char nodedata1[] = "noded1";
    char nodedata2[] = "noded2";
    //char goodbye[] = "Goodbye\n";
    struct Node* node1 = createNode(g, nodedata1);
    struct Node* node2 = createNode(g, nodedata2);
    struct Node* nodeh = createNode(g, greeting);
```

```
bool success = true;

struct NodeList* nl = createNodeList();
if (!empty_NL(nl)) {
    success = false;
}

appendNode(nl, node1);
if (!(length_NL(nl) == 1)) {
    success = false;
}
prependNode(nl, node2);
if (!(length_NL(nl) == 2)) {
    success = false;
}
if (!includesNode(nl, node2)) {
    success = false;
}
if (!includesNode(nl, node1)) {
    success = false;
}

removeFirst_NL(nl);
if (!includesNode(nl, node1)) {
    success = false;
}
if (includesNode(nl, node2)) {
    success = false;
}

prependNode(nl, node2);
removeLast_NL(nl);
if (!includesNode(nl, node2)) {
    success = false;
}
if (includesNode(nl, node1)) {
    success = false;
}

appendNode(nl, node1);
appendNode(nl, nodeh);
removeNode(nl, node1);
```

```
    removeNode(nl, nodeh);
    removeNode(nl, node2);
    if (!empty_NL(nl)) {
        success = false;
    }

    struct Node* node3 = createNode(g, "A");
    struct Node* node4 = createNode(g, "B");
    struct Node* node5 = createNode(g, "C");
    appendNode(nl, node3);
    appendNode(nl, node4);
    appendNode(nl, node5);

    removeNodeGraph(g, node4);
    removeNode(nl, node4);
    //appendNode(nl, node4);
    //prependNode(nl, node4);
    removeNodeGraph(g, node3);
    //neighbors(node3);
    if (!includesNode(nl, node3)) {
        success = false;
    }

    if (success) {
        printf("SUCCESS\n");
    }
    else {
        printf("FAILURE\n");
    }

    return 0;
}
```

Nodelist-test.c

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <stdbool.h>
#include "../../graclstdlib.c"
#include "../tests/print-functions.c"
```

```
bool goalTest(struct Node* goal, struct Node* current){
   return nodeEquals(current, goal);
}

int normalDFS(struct Node* current, struct Node* goal, struct NodeList* myPath, struct
NodeList* path){
   bool done = false;
   int bad = 0;
   _synch_start((void*)path);
   if(!empty_NL(path)){
       _synch_end((void*)path);
       printf("break out\n");
       done = true;
   }

   if (!done) {
       _synch_end((void*)path);
       appendNode(myPath, current);

       if (goalTest(goal, current)){
           _synch_start((void*)path);
           // use forloop in grc
           path->head = myPath->head;
           path->tail = myPath->tail;
           printNodeList(path);
           _synch_end((void*)path);
           done = true;
       } else {
           _synch_start((void*)current);
           if (updateVisited(current, true) < 0){
               _synch_end((void*)current);
               bad = -1;
               done = true;
           }
           if (!done) {
               _synch_end((void*)current);
               struct EdgeListItem* currentEdgeItem = current->edges->head;
               while (currentEdgeItem) {
                   struct Edge* currentEdge = currentEdgeItem->edge;
                   struct Node* adjacentNode = end(currentEdge);

                   _synch_start((void*)adjacentNode);
```

```
                    if (!visited(adjacentNode)) {
                        _synch_end((void*)adjacentNode);
                        normalDFS(adjacentNode, goal, myPath, path);
                    } else {
                        _synch_end((void*)adjacentNode);
                    }
                    currentEdgeItem = currentEdgeItem->next;
                }
            }
        }
    }
    return bad;
}

void normalDFS_start(struct Node* current, struct Node* goal, struct NodeList* myPath,
struct NodeList* path){
    printf("Enters dfs_start:\n");
    struct NodeList* individual_nl = createNodeList();
    struct Node* node = myPath->head->node;
    appendNode(individual_nl, node);
    normalDFS(current, goal, individual_nl, path);
}

/* COMPILER START */
struct normalDFSArgs {
    struct Node* current;
    struct Node* goal;
    struct NodeList* myPath;
    struct NodeList* path;
};

void *normalDFS_unwrapper(void *args) {
    printf("In unwrapper\n");
    struct normalDFSArgs *fargs = (struct normalDFSArgs *) args;

    normalDFS_start(fargs->current, fargs->goal, fargs->myPath, fargs->path);
    return NULL;
}
/* COMPILER END */

int main(){
    struct Graph* g;
```

```
struct Node* node1;
struct Node* node2;
struct Node* node3;
struct Node* node4;
struct Node* node5;
struct Node* node6;
struct Node* node7;
struct Node* node8;
struct Node* node9;
struct Node* node10;
struct Edge* e1;
struct Edge* e2;
struct Edge* e3;
struct Edge* e4;
struct Edge* e5;
struct Edge* e6;
struct Edge* e7;
struct Edge* e8;
struct Edge* e9;
g = createGraph(4);
node1 = createNode(g, "A");
node2 = createNode(g, "B");
node3 = createNode(g, "C");
node4 = createNode(g, "D");
node5 = createNode(g, "E");
node6 = createNode(g, "F");
node7 = createNode(g, "G");
node8 = createNode(g, "H");
node9 = createNode(g, "I");
node10 = createNode(g, "J");
e1 = addEdge(g, node1, node2, 14.0);
e2 = addEdge(g, node2, node5, 13.0);
e3 = addEdge(g, node5, node9, 43.0);
e4 = addEdge(g, node1, node3, 13.0);
e5 = addEdge(g, node3, node6, 43.0);
e6 = addEdge(g, node3, node7, 13.0);
e7 = addEdge(g, node6, node10, 43.0);
e8 = addEdge(g, node1, node4, 13.0);
e9 = addEdge(g, node4, node8, 43.0);

struct Node* parent = node1;
struct Node* goal = node10;
```

```
   struct NodeList* children = createNodeList();
   struct NodeList* myPath = createNodeList();
   appendNode(myPath, parent);
   if (nodeEquals(parent, node10)) {
       return 0;
   }
   children = neighbors(parent);
   struct NodeList* path = createNodeList();

   int len_possible_start = length_NL(children);

   /* user passes list of nodes, functions, arguments to compiler */

   /* COMPILER START */
   pthread_t* threads = malloc(sizeof(pthread_t) * len_possible_start);

   // done by the compiler
        struct  normalDFSArgs*  dfsargs  =  malloc(sizeof(struct  normalDFSArgs)  *
len_possible_start);

   struct NodeListItem* currentNode = children->head;
   int i = 0;
   printf("Before while\n");
   while (currentNode) {
       // done by the compiler
       dfsargs[i].current = currentNode->node;
       dfsargs[i].goal = goal;
       dfsargs[i].myPath = myPath;
       dfsargs[i].path = path;

       pthread_create(threads + i, NULL, normalDFS_unwrapper, dfsargs + i);
       printf("PARENT AFTER\n");
       currentNode = currentNode->next;
       i++;
   }
   hatch_end(threads, len_possible_start);
   /* COMPILER END */

   /* C FOR HATCH */
   printf("Finishes join\n");

}
```

normalDFS-conc.c

```c
#include <stdlib.h>
#include <string.h>

void printNodeList(struct NodeList* node_list) {
   struct NodeListItem *current = malloc(sizeof(struct NodeListItem));
   current = node_list->head;
   while (current != NULL) {
       printf("id: %d data: %s\n", current->node->id, current->node->data);
       current = current->next;
   }
}

void printNode (struct Node* node) {
   printf("%s\n", node->visited ? "true" : "false");
   printf("%d\n", node->id);
   printf("%s", node->data);
   return;
}

void printEdge (struct Edge* edge) {
   printf("Start node: %s", edge->start->data);
   printf("End node: %s", edge->end->data);
   printf("Weight: %f", edge->weight);
}

void printEdgeList (struct EdgeList* edge_list) {
   struct EdgeListItem* current;
   current = edge_list->head;
   while (current != NULL) {
       printf("%s\n", current->edge->start->data); // Check each start node's data
       current = current->next;
   }
   return;
}
```

Print-functions.c

LARGER TEST FILES FOR GRAPH ALGORITHMS

```c
bool goalTest(Node goal, Node current) {
```

```
   return current.nodeEquals(goal);
}

bool goal_found = False;

int normalDFS(Node current, Node goal, NodeList myPath, NodeList path) {
   bool done = False;
   if (goal_found) {
       done = True;
   }
   if (!done && !goal_found) {
       myPath.appendNode(current);
       if (goalTest(goal, current)) {
           //print("Found goal, goal path:");
           for (Node n in myPath) {
               print(n.data());
           }
           goal_found = True;
           done = True;
       } else {
           if (!done && !goal_found) {
               NodeList neighbors = current.neighbors();
               for (Node n in neighbors) {
                   bool recurse = False;
                   synch n {
                       if (!n.visited() && !goal_found) {
                           recurse = True;
                           n.updateVisited(True);
                       }
                   }
                   if (recurse && !goal_found) {
                       normalDFS(n, goal, myPath, path);
                   }
               }
           }
       }
   }
   return 0;
}

void normalDFS_start(Node current, Node goal, NodeList myPath, NodeList path){
   //print("Thread hatched: ");
```

```
    NodeList individual_nl = createNodeList();
    for (Node n in myPath) {
        individual_nl.appendNode(n);
    }
    updateVisited(current, True);
    normalDFS(current, goal, individual_nl, path);
    //print("Thread complete.");
}

int main() {
    Graph g;
    bool done = False;
    Node node1;
    Node node2;
    Node node3;
    Node node4;
    Node node5;
    Node node6;
    Node node7;
    Node node8;
    Node node9;
    Node node10;
    Edge e1;
    Edge e2;
    Edge e3;
    Edge e4;
    Edge e5;
    Edge e6;
    Edge e7;
    Edge e8;
    Edge e9;
    g = createGraph(12);
    node1 = g.createNode("A");
    node2 = g.createNode("B");
    node3 = g.createNode("C");
    node4 = g.createNode("D");
    node5 = g.createNode("E");
    node6 = g.createNode("F");
    node7 = g.createNode("G");
    node8 = g.createNode("H");
    node9 = g.createNode("I");
    node10 = g.createNode("J");
```

```
Node node11 = g.createNode("K");
Node node12 = g.createNode("L");
Node node13 = g.createNode("M");
Node node14 = g.createNode("N");
Node node15 = g.createNode("O");
Node node16 = g.createNode("P");
Node node17 = g.createNode("Q");
Node node18 = g.createNode("R");
Node node19 = g.createNode("S");
Node node20 = g.createNode("T");
Node node21 = g.createNode("U");
Node node22 = g.createNode("V");
Node node23 = g.createNode("W");
Node node24 = g.createNode("X");
Node node25 = g.createNode("Y");

e1 = g.addEdge(node1, node2, 14.0);
e2 = g.addEdge(node2, node5, 13.0);
e3 = g.addEdge(node5, node9, 43.0);
Edge e10 = g.addEdge(node9, node11, 5.0);
Edge e11 = g.addEdge(node11, node12, 2.0);
Edge e12 = g.addEdge(node12, node13, 3.333);
Edge e13 = g.addEdge(node13, node14, 3.333);
Edge e14 = g.addEdge(node13, node15, 3.333);
Edge e15 = g.addEdge(node14, node15, 0.45);

e4 = g.addEdge(node1, node3, 13.0);
e5 = g.addEdge(node3, node6, 43.0);
e6 = g.addEdge(node3, node7, 13.0);
Edge e16 = g.addEdge(node7, node16, 8.0);
Edge e17 = g.addEdge(node7, node17, 1.35);
Edge e18 = g.addEdge(node17, node18, 5.77777);
Edge e19 = g.addEdge(node18, node19, 5.5);
Edge e20 = g.addEdge(node19, node17, -45.0);

e7 = g.addEdge(node6, node10, 43.0);
Edge e21 = g.addEdge(node10, node20, -5.0);
Edge e22 = g.addEdge(node10, node21, -3.0);
Edge e23 = g.addEdge(node21, node22, 100.);
Edge e24 = g.addEdge(node20, node23, 66.0);

e8 = g.addEdge(node1, node4, 13.0);
```

```
   e9 = g.addEdge(node4, node8, 43.0);
   Edge e25 = g.addEdge(node8, node24, 1000.);
   Edge e26 = g.addEdge(node24, node25, 66.0);
    Node parent = node1;
   Node goal = node23;
   NodeList children = createNodeList();

   NodeList myPath = createNodeList();
   myPath.appendNode(parent);
   updateVisited(parent, True);
   if (parent.nodeEquals(goal)) {
       for (Node n in myPath) {
           print(n.data());
       }

   } else {
       children = neighbors(parent);
       NodeList path = createNodeList();

       hatch children normalDFS_start(goal, myPath, path) {
           // parent doesnt want to do anything before join in this case
       }
   }

}
```

Test-concdfs.grc

```
bool alreadyVisited;
Node source;

void printNodeList(NodeList nl){
   for (Node n in nl){
       print(n.data());
   }
}

bool goalTest(Node goal, Node current){
   return current.nodeEquals(goal);
}

int normalDFS(Node current, Node goal, NodeList myPath){
   if (goalTest(goal, current)){
```

```
        bool notfound = True;
        while (notfound) {
            myPath.prependNode(current);
            if(current.nodeEquals(source)){
                notfound = False;
            }
            current = current.prec();
        }
        for (Node n in myPath) {
            print(n.data());
        }
    } else {
        current.updateVisited(True);
        for (Node n in current.neighbors()) {
            alreadyVisited = n.visited();
            if (!alreadyVisited){
                n.setPrec(current);
                normalDFS(n, goal, myPath);
            }
        }
    }

    return 0;
}

int main(){
    Graph g = createGraph(12);
    Node node1 = g.createNode("A");
    Node node2 = g.createNode("B");
    Node node3 = g.createNode("C");
    Node node4 = g.createNode("D");
    Node node5 = g.createNode("E");
    Node node6 = g.createNode("F");
    Node node7 = g.createNode("G");
    Node node8 = g.createNode("H");
    Node node9 = g.createNode("I");
    Node node10 = g.createNode("J");
    Edge e1 = g.addEdge(node1, node2, 14.0);
    Edge e2 = g.addEdge(node2, node5, 13.0);
    Edge e3 = g.addEdge(node5, node9, 43.0);
    Edge e4 = g.addEdge(node1, node3, 13.0);
    Edge e5 = g.addEdge(node3, node6, 43.0);
```

```
Edge e6 = g.addEdge(node3, node7, 13.0);
Edge e7 = g.addEdge(node6, node10, 43.0);
Edge e8 = g.addEdge(node1, node4, 13.0);
Edge e9 = g.addEdge(node4, node8, 43.0);

Node node11 = g.createNode("K");
Node node12 = g.createNode("L");
Node node13 = g.createNode("M");
Node node14 = g.createNode("N");
Node node15 = g.createNode("O");
Node node16 = g.createNode("P");
Node node17 = g.createNode("Q");
Node node18 = g.createNode("R");
Node node19 = g.createNode("S");
Node node20 = g.createNode("T");
Node node21 = g.createNode("U");
Node node22 = g.createNode("V");
Node node23 = g.createNode("W");
Node node24 = g.createNode("X");
Node node25 = g.createNode("Y");

Edge e10 = g.addEdge(node9, node11, 5.0);
Edge e11 = g.addEdge(node11, node12, 2.0);
Edge e12 = g.addEdge(node12, node13, 3.333);
Edge e13 = g.addEdge(node13, node14, 3.333);
Edge e14 = g.addEdge(node13, node15, 3.333);
Edge e15 = g.addEdge(node14, node15, 0.45);

e4 = g.addEdge(node1, node3, 13.0);
e5 = g.addEdge(node3, node6, 43.0);
e6 = g.addEdge(node3, node7, 13.0);
Edge e16 = g.addEdge(node7, node16, 8.0);
Edge e17 = g.addEdge(node7, node17, 1.35);
Edge e18 = g.addEdge(node17, node18, 5.77777);
Edge e19 = g.addEdge(node18, node19, 5.5);
Edge e20 = g.addEdge(node19, node17, -45.0);

e7 = g.addEdge(node6, node10, 43.0);
Edge e21 = g.addEdge(node10, node20, -5.0);
Edge e22 = g.addEdge(node10, node21, -3.0);
Edge e23 = g.addEdge(node21, node22, 100.);
Edge e24 = g.addEdge(node20, node23, 66.0);
```

```
    e8 = g.addEdge(node1, node4, 13.0);
    e9 = g.addEdge(node4, node8, 43.0);
    Edge e25 = g.addEdge(node8, node24, 1000.);
    Edge e26 = g.addEdge(node24, node25, 66.0);

    NodeList myPath = createNodeList();
    source = node1;
    normalDFS(node1, node23, myPath);
}
```

test-dfs.grc

```
// Based off of the Java implementation here: https://www.baeldung.com/java-dijkstra
Node getCheapestNode(NodeList unsettledNodes) {
    Node cheapestNode;
    double cheapestCost = infinity;
    for (Node n in unsettledNodes) {
        if (n.cost() < cheapestCost) {
            cheapestCost = n.cost();
            cheapestNode = n;
        }
    }
    return cheapestNode;
}


NodeList getShortestPath(Node source, Node goal, NodeList nl) {
    NodeList path = createNodeList();
    Node current = goal;
    if(source.nodeEquals(goal)){
        path.prependNode(current);
    }
    else if(!nl.includesNode(goal)){
        path = createNodeList();
    }
    else{
        bool notfound = True;
        while (notfound) {
            path.prependNode(current);
            if(current.nodeEquals(source)){
```

```
                notfound = False;
            }
            current = current.prec();
        }
    }
    return path;
}


NodeList dijkstra(Graph g, Node source, Node goal) {
    source.updateCost(0.0);

    // nodes with known cheapest cost
    NodeList settledNodes = createNodeList();
    // frontier nodes with unknown cheapest cost
    NodeList unsettledNodes = createNodeList();
    unsettledNodes.appendNode(source);

    while (unsettledNodes.length_NL()>0) {
        Node currentNode = getCheapestNode(unsettledNodes);
        if (unsettledNodes.includesNode(currentNode)){
            unsettledNodes.removeNode(currentNode);
        }

        // Iterate through neighbors to find minimum distance and add them to unsettled
        EdgeList edges = currentNode.edges();
        for (Edge edge in edges) {
            Node adjacentNode = edge.end();

            if (!settledNodes.includesNode(adjacentNode)) {
                double newCost = currentNode.cost()+edge.weight();
                 // If it's a shorter path to adjacentNode, update adjacentNode cost and
prec

                if (cost(adjacentNode) == -1.0 || newCost < adjacentNode.cost()) {
                    adjacentNode.updateCost(newCost);
                    adjacentNode.setPrec(currentNode);

                }
                // Add neighbors to unsettled
                if (!unsettledNodes.includesNode(adjacentNode)) {
                    unsettledNodes.appendNode(adjacentNode);
```

```
                }
            }
        }
        settledNodes.appendNode(currentNode);
    }
    return getShortestPath(source, goal, settledNodes);
}


int printNl(NodeList nl){
    for (Node n in nl) {
        print(n.data());
    }
    return 0;
}

// TEST!
int main(){
    Graph g = createGraph(6);

    Node nodeA = g.createNode("A");
    Node nodeB = g.createNode("B");
    Node nodeC = g.createNode("C");
    Node nodeD = g.createNode("D");
    Node nodeE = g.createNode("E");
    Node nodeF = g.createNode("F");

    Edge e1 = g.addEdge(nodeA, nodeB, 10.0);
    Edge e2 = g.addEdge(nodeA, nodeC, 15.0);
    Edge e3 = g.addEdge(nodeB, nodeD, 12.0);
    Edge e4 = g.addEdge(nodeB, nodeF, 15.0);
    Edge e5 = g.addEdge(nodeC, nodeE, 10.0);
    Edge e6 = g.addEdge(nodeD, nodeE, 2.0);
    Edge e7 = g.addEdge(nodeD, nodeF, 1.0);
    Edge e8 = g.addEdge(nodeF, nodeE, 5.0);


    //print("Path should be A-B-D-F");
    NodeList path = dijkstra(g, nodeA, nodeF);
    printNl(path);

    //print("Path should be A-B-D-E");
```

```
    path = dijkstra(g, nodeA, nodeE);
    printNl(path);

    //print("Path should be E");
    path = dijkstra(g, nodeE, nodeE);
    printNl(path);

    //print("Path should be A-B");
    path = dijkstra(g, nodeA, nodeB);
    printNl(path);

    //print("Path should be D-E");
    path = dijkstra(g, nodeD, nodeE);
    printNl(path);

    //print("Path should be B-D-F");
    path = dijkstra(g, nodeB, nodeF);
    printNl(path);

    //print("Path should be B-D-E");
    path = dijkstra(g, nodeB, nodeE);
    printNl(path);

    //print("Path should be D");
    path = dijkstra(g, nodeD, nodeD);
    printNl(path);

    //print("Path should be C");
    path = dijkstra(g, nodeC, nodeC);
    printNl(path);

    //print("Path should not exist");
    path = dijkstra(g, nodeB, nodeC);
    printNl(path);

    //print("Path should not exist");
    path = dijkstra(g, nodeF, nodeA);
    printNl(path);

    return 0;
}
```

Test-dijkstra.grc

```
bool threads_done = False;

void bfs(Node start, NodeList visited_list, NodeList to_visit) {
   NodeList path = createNodeList();
   bool final_path_found = False;
   for (Node n in visited_list) {
       path.appendNode(n);
   }
   for (Node n in start.neighbors()) {
       if (!threads_done && !visited(n)) {
           to_visit.appendNode(n);
       }
       else {
           threads_done = True;
           final_path_found = True;
           path.appendNode(n);
           for (Node n in path) {
               print(n.data());
           }
       }
   }
   if (!threads_done) {
       Node next = removeFirst_NL(to_visit);
       synch next {
           updateVisited(next, True);
       }
       path.appendNode(next);
       if(!threads_done){
           bfs(next, path, to_visit);
       }
   }
   return;
}

void bidirectional_search(Node start) {
   NodeList path = createNodeList();
   path.appendNode(start);
   NodeList to_visit = createNodeList();
   bfs(start, path, to_visit);
   print("Threads collide");
```

```
    return;
}

int main() {
    Graph g = createGraph(12);

    Node node1 = g.createNode("A");
    Node node2 = g.createNode("B");
    Node node3 = g.createNode("C");
    Node node4 = g.createNode("D");
    Node node5 = g.createNode("E");
    Node node6 = g.createNode("F");
    Node node7 = g.createNode("G");
    Node node8 = g.createNode("H");
    Node node9 = g.createNode("I");
    Node node10 = g.createNode("J");
    Node node15 = g.createNode("O");

    //A->B, A->C, A->D
    Edge e1 = g.addEdge(node1, node2, 14.0);
    Edge e2 = g.addEdge(node1, node3, 14.0);
    Edge e3 = g.addEdge(node1, node4, 14.0);


    Edge e4 = g.addEdge(node2, node1, 14.0);
    Edge e5 = g.addEdge(node3, node1, 14.0);
    Edge e6 = g.addEdge(node4, node1, 14.0);
    Edge e7 = g.addEdge(node2, node5, 14.0);
    Edge e8 = g.addEdge(node5, node2, 14.0);
    Edge e9 = g.addEdge(node3, node6, 14.0);
    Edge e10 = g.addEdge(node6, node3, 14.0);
    Edge e11 = g.addEdge(node4, node7, 14.0);
    Edge e12 = g.addEdge(node7, node4, 14.0);

    // E->H, H->E
    Edge e13 = g.addEdge(node5, node8, 14.0);
    Edge e14 = g.addEdge(node8, node5, 14.0);

    //f->i, i->f
    Edge e15 = g.addEdge(node6, node9, 14.0);
    Edge e16 = g.addEdge(node9, node6, 14.0);
```

```
    // G->J, J->G
    Edge e17 = g.addEdge(node7, node10, 14.0);
    Edge e18 = g.addEdge(node10, node7, 14.0);

    //H->O, O->H
    Edge e19 = g.addEdge(node8, node15, 14.0);
    Edge e20 = g.addEdge(node15, node8, 14.0);

    // I-->O
    // O->I
    Edge e21 = g.addEdge(node9, node15, 14.0);
    Edge e22 = g.addEdge(node15, node9, 14.0);

    // j->o, o->j
    Edge e23 = g.addEdge(node10, node15, 14.0);
    Edge e24 = g.addEdge(node15, node10, 14.0);

    Node start = node1;
    Node goal = node15;
    updateVisited(goal, True);
    updateVisited(start, True);
    NodeList starters = createNodeList();
    starters.appendNode(start);
    starters.appendNode(goal);

    hatch starters bidirectional_search() {
        print("Parent process waiting for threads to join...");
    }
    return 0;
}
```

test-bidirection.grc

```
#!/bin/sh

make clean
make

for i in {1..5}
    # get the time in miliseconds for normaldfs
    do ts="$((`date '+%s%N'`/1000000))";

    ./scripts/testone.sh dfs;
```

```
    tt="$((`date '+%s%N'`/1000000))";
    diff=`expr $tt - $ts`;
    a=`expr $a + $diff`;

    # get the time in miliseconds for conc dfs
    conc_ts="$((`date '+%s%N'`/1000000))";
    ./scripts/testone.sh concdfs;

    conc_tt="$((`date '+%s%N'`/1000000))";
    diff=`expr $conc_tt - $conc_ts`;
    b=`expr $b + $diff`;
done


# echo the difference between normaldfs and concdfs
echo "Ran the tests five times. Here are the results: ";
echo "Time taken to run normaldfs five times is: "$a" milliseconds";
echo "Time taken to run concdfs five times is: "$b" milliseconds";
res1=`expr $a / 5`;
res2=`expr $b / 5`;
echo "On average, normaldfs ran in: "$res1" milliseconds";
echo "On average, concdfs ran in: "$res2" milliseconds";
```

./time-dfs.sh