Strux

Joshua Bartlett
jcb2254
Language Guru

Sophie Stadler
srs2231
Manager

Fredrick Kofi Tam
fkt2105
System Architect

Millie Yang
my2440
Tester

20 December 2017

Contents

1 Introduction **3**

2 Language Tutorial **4**
2.1 Compiling & Running 4
2.1.1 Run tests 4
2.1.2 Run a new program 4
2.2 A simple Strux program 5

3 Language Manual **6**
3.1 Lexical Elements 6
3.1.1 Identifiers 6
3.1.2 Keywords 6
3.1.3 Whitespace 6
3.1.4 Comments 7
3.2 Operators and Expressions 7
3.2.1 Assignment Operator 7
3.2.2 Arithmetic Operators 8
3.2.3 Comparison Operators 8
3.2.4 Logical Operators 8
3.2.5 Operator Precedence 8
3.2.6 Order of Evaluation 9
3.3 Statements 9
3.3.1 Expression Statements 9
3.3.2 Declaration Statements 9
3.3.3 Control Flow Statements 9
3.4 Data Types 11
3.4.1 Primitives 11
3.4.2 Built-In Data Structures 11
3.5 Functions 18
3.5.1 Built-In 18
3.5.2 User-defined Functions 20

4 Project Plan **22**
4.1 Process . . . 22
4.1.1 Planning . . . 22
4.1.2 Specification . . . 22
4.1.3 Development . . . 22
4.1.4 Testing . . . 23
4.2 Style Guide . . . 23
4.3 Timeline . . . 24
4.4 Roles . . . 24
4.5 Development Environment . . . 24
4.5.1 Tools . . . 24
4.5.2 Languages . . . 25
4.6 Project Log . . . 25

5 Architectural Design **27**

6 Test Plan **29**
6.1 Unit Testing . . . 30
6.2 Integration Testing . . . 30
6.3 Error Handling . . . 31
6.4 Representative Programs . . . 31
6.4.1 test-queue-8.strux . . . 31
6.4.2 test-bstree-8.strux . . . 38
6.5 Automation . . . 41

7 Lessons Learned **42**

8 Appendix **45**
8.1 scanner.mll . . . 45
8.2 parser.mly . . . 47
8.3 ast.ml . . . 50
8.4 semant.ml . . . 54
8.5 codegen.ml . . . 63
8.6 strux.ml . . . 84
8.7 bstree.h . . . 85
8.8 bstree.c . . . 86
8.9 linkedlist.c . . . 94
8.10 queue.c . . . 99
8.11 quicksort.c . . . 103
8.12 stack.c . . . 115
8.13 utils.h . . . 117
8.14 testall.sh . . . 118
8.15 test-gcd.ll . . . 123
8.16 git log . . . 126

Chapter 1

Introduction

Data structures are one of the most important concepts in computer science for beginners and seasoned developers alike. For many students, there is a certain hurdle associated with visualizing data structures—that is, connecting the drawings in a textbook to the Java or C++ they are writing. A major problem with drawings is their static nature; there is no way to see how they are affected by code. Strux tackles this issue by providing a link between code and data structures in the form of visualizations. We use this term to refer to an ASCII art rendering of a stack, queue, linked list, tree, or array that is output by Strux. These visualizations, when called via `.show()`, are printed to the console to help programmers become familiar with the key features of each structure, and illuminate the data their objects currently contain.

Why printing to the console versus, say, generating an image? The primary reasons are ease of use and simple visualization of modifications. Users can simply scroll up through the terminal to see how their stack has changed, rather than sift through a series of images. Strux doesn't require leaving the command line to use effectively.

Beyond visualizations, Strux has been designed with ease-of-use in mind. Its syntax combines our favorite features from Python, Java, and C, which will hopefully make it familiar to students first learning data structures. These structures are also built in, eliminating the complexities that come with designing one's own. We've also opted to implement minor features that we felt increased usability, such as the increment operator and single-line variable assignment. These characteristics make it approachable and effective in its goal to improve new programmers' understanding of data structures.

Chapter 2

Language Tutorial

2.1 Compiling & Running

Strux requires LLVM and its development libraries, the m4 macro preprocessor, opam, and clang. Useful instructions for installing these on your operating system can be found in the MicroC README.

Inside Strux's root folder, type `make`. This first creates the Strux to LLVM compiler, called `strux.native`. It then calls a script, `linkStrux.sh`, that converts the C code to LLVM bytecode.

N.B. As of this writing (December 2017), macOS High Sierra introduces a compatibility problem with the LLVM bitreader. We were unable to run Strux on a machine running High Sierra, but earlier versions of macOS should run perfectly. Strux was also tested on Ubuntu 16.04.

2.1.1 Run tests

To run Strux's test suite, simply call the test script from the root directory:

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

This will iterate through all files in the `tests/` directory, indicate whether they passed, and log their output in case of failure. `testall.sh` is based on the MicroC test script.

2.1.2 Run a new program

The easiest way to run a new program is to call it via the `testall.sh` script. Although the test script will expect a `.out` file to compare against, the linking will be handled automatically. You can inspect the `<your_filename>.out` file in the root directory to see what was printed.

```
$ ./testall.sh <your_filename>.strux
```

2.2 A simple Strux program

All basic Strux programs include a `main()` function that accepts no arguments and returns 0. They are named `<filename>.strux`. A simple "Hello World" program therefore looks like this:

```
int main() {
    print("Hello, World!");
    return 0;
}
```

Chapter 3

Language Manual

3.1 Lexical Elements

3.1.1 Identifiers

An identifier is a unique sequence of characters that are used to identify variables and functions. Identifiers can contain letters, numbers, and the underscore character. Additionally, identifiers are case-sensitive. A valid identifier adheres to the following rules:

1. At least 1 character long
2. Begins with a letter
3. Isn't equal to one of the reserved keywords

3.1.2 Keywords

Keywords are reserved words that each have some unique meaning when compiling. Keywords can not be used as identifiers or reassigned.

int	num	string	bool
while	for	not	true
false	void	if	elif
else	new	return	main
and	or	Queue	Stack
LinkedList	BSTree		

3.1.3 Whitespace

Whitespace is largely ignored in Strux. Other than within string literals, whitespace is only used to separate different tokens. Therefore, these two functions actually produce the same result after being compiled:

```
num addTwo(num a, num b) {
    return a + b;
}

num       addTwo    (num a  ,num b) {     return a+  b;}
```

A space is required after:

- The return keyword, before the value that is returned (if any).
- The new keyword, after an instance of an object is initialized.
- The return type of a variable when defined in an expression.
- The return type of a function in a function signature.

Do not put a space between:

- The type of values in an array and the brackets (`[]`) used to instantiate it
  - Example: `int[4] arr = [1, 2, 3, 4];`

3.1.4 Comments

Anything in a comment will be completely ignored by the compiler. Strux does not have a special syntax for single-line comments. All comments are contained within `:(` and `):`.

```
:( This is a comment ):
```

3.2 Operators and Expressions

3.2.1 Assignment Operator

Operator	Function	Associativity
=	Assignment	Right to left

Strux uses the standard assignment operator (=), to store the value of the right operand to the variable of the left operand of the same type. The left operand cannot be a literal (string or num literal) value and variables on the left cannot be named starting with numbers. Example:

```
int myAge = 21;                 :( valid ):
int "myAge" = 21;               :( invalid ):
string myName = "Kennedy";      :( valid ):
bool 1true = true;              :( invalid ):
```

3.2.2 Arithmetic Operators

Assuming `int x = 100` and `int y = 20`

Operator	Function	Associativity	Example
`+`	Addition	Left to right	`x + y = 120`
`-`	Subtraction	Left to right	`x - y = 80`
`*`	Multiplication	Left to right	`x * y = 2000`
`/`	Division	Left to right	`x / y = 50`
`%`	Modulo	Left to right	`x % y = 0`
`++`	Increment	Left to right	`x++ = 101`
`--`	Decrement	Left to right	`x-- = 99`

3.2.3 Comparison Operators

Assuming `int x = 50` and `int y = 20`

Op.	Function	Associativity	Example
`==`	Equal to	Left to right	`(x == y)` returns `false`
`!=`	Not equal to	Left to right	`(x != y)` returns `true`
`>`	Greater than	Left to right	`(x > y)` returns `true`
`>=`	Greater than or equal to	Left to right	`(x >= y)` returns `true`
`<`	Less than	Left to right	`(x > y)` returns `false`
`<=`	Less than or equal to	Left to right	`(x >= y)` returns `false`

3.2.4 Logical Operators

Assuming `bool x = true` and `bool y = false`

Operator	Function	Associativity	Example
`and`	Logical AND	Left to right	`(x and y)` returns `false`
`or`	Logical OR	Left to right	`(x or y)` returns `true`
`not`	Logical NOT	Left to right	`not x` returns `false`

3.2.5 Operator Precedence

Expressions can have multiple operators, for example `(x - y) * (x % y)`. In these situations, operators are executed based on their level of precedence. Theist below arranges operators from highest precedence to lowest.

1. Multiplication and Division expressions
2. Addition and Subtraction expressions
3. Greater Than, Less Than, Greater Than or Equal, and Less Than or Equal To expressions
4. Equal To and Not Equal To expressions

5. Logical NOT expressions
6. Logical AND expressions
7. Logical OR expressions
8. Assignment expressions

3.2.6 Order of Evaluation

If we have a complex expression, it will be evaluated by starting with the leftmost subexpression. For example, in:

```
(( c() \% d() ) * ( d() + z() )
```

where c, d, e and z are functions, `c()` will be called first, followed by `d()`, `e()` and `z()`. Operator precedence will be ignored in this case.

3.3 Statements

3.3.1 Expression Statements

An expression statement is one that can be executed by Strux. Expressions are terminated with a semicolon, and include method invocations, value assignments, and creation of data structures. Some examples:

```
LinkedList::int myList = new LinkedList::int();
print(8.9);
string greeting = "hello world";
```

3.3.2 Declaration Statements

Declaration statements are used to declare a new variable. They are comprised of its type, its name, and, optionally, its value. A value is assigned with the *equals* operator (=). One can declare multiple variables of the same type in one declaration. Declaration statements are terminated with a semicolon. Variables cannot be declared without an initial value.

```
num fivePointOh = 5.0;
int wordCount = 1255;
string missionStatement = "Strux rocks!";
bool isTired = true;
int i; :( Invalid Syntax ):
```

3.3.3 Control Flow Statements

Control flow statements disrupt the linear evaluation of code. Conditionals, loops, and keywords are used by Strux to introduce specific flow.

Loops

Loops are used to execute a section of code multiple times. Strux includes two types of loops: for loops and while loops.

For Loops For loops are used to execute a block of code until a condition is satisfied. The format is as such:

```
for (initialization; termination; increment/decrement) {
    :( Code goes here ):
}
```

The termination expression above must evaluate to a boolean. When the loop is entered, the initialization is called and checked against the termination condition. Then, the code inside the loop is executed and the initialization value incremented on each iteration. The loop finishes when the termination expression returns false.

An example:

```
int i = 1;
for (i = 1; i <= 10; i++) {
    print(i);               :( Prints the numbers 1-10 ):
}
```

While Loops While loops are used to iterate over a block of code until a condition is being evaluated as false. The syntax is such:

```
while (expression) {
     :( execute this code ):
}
```

The expression above must evaluate to a boolean value. The code contained within the braces will execute until the expression returns false. An example:

```
int i = 0;
while (i++ <= 10) {
    print(i);                :( Prints the numbers 1-10 ):
}
```

Conditionals

Strux uses `if-else` and `if-elif-else` expressions to introduce conditional evaluation. In each of these statements, code within the required braces (`{}`) will evaluate only if the given expression is true. Conditional statements must be enclosed in parentheses. Below, an if-else statement:

```
bool october = false;
if (october == true) {
    print("It's October!");
```

```
} else {
    print("It isn't October.");
}
```

An `if-elif-else` statement presents the opportunity to introduce more (infinite, in fact) conditional statements.

```
int temp = 65;
if (temp > 80) {
    print("It's hot!");
} elif (temp < 45) {
    print("It's cold!");
} elif (temp < 10) {
    print("It's freezing!");
} else {
    print("It's nice out.");
}
```

3.4 Data Types

Strux is a typed language. Type must be specified when a variable is declared, and is immutable.

3.4.1 Primitives

int

Strux represents integer values usint `int`. An `int` is a 32-bit value.

num

Strux represents decimal values using `num`.

string

A `string` is a sequence of ASCII characters enclosed by double quotes (`"`). Strings are immutable.

bool

A variable of type `bool` represents the logical value true or false. When printed, `bool` values are displayed as the integers `1` or `0`.

3.4.2 Built-In Data Structures

Stack

A stack is a data structure that represents LIFO (Last-in-first-out) operations on stack of objects.

Initializing an instance of a Stack Stacks can be initialized using one of two constructors. The type is specified after two colons (::); this pattern is adopted by Queues and LinkedLists as well.

```
Stack::type emptyStack = new Stack::type();
```

The next example initializes a stack filled with several values. These values must all be of the same type: either `int`, `num`, `bool` or `string`.

```
Stack::int filledStack = new Stack::int(1, 2, 3);
```

Library Functions There are several builtin functions for manipulating a stack.

Peek To look at the top element of the stack, use `peek()`. This method looks at, but does not remove, the top of the element in the stack. If the stack is empty, this function returns `null`.

```
filledStack.peek();                    :( returns 3 ):
filledStack.peek();                    :( returns 2 ):
filledStack.peek();                    :( returns 1 ):
filledStack.peek();                    :( returns null ):
```

Remove To remove the top element from the stack, use `remove()`. This function retrieves the value of top most element of stack and removes it from stack. It always returns void.

```
Stack::int stack = new Stack::int(1, 2, 3);
stack.remove();                        :( returns void ):
stack.remove();                        :( returns void ):
stack.remove();                        :( returns void ):
stack.remove();                        :( returns void ):
```

Add To add items to the top of the stack, use `add(int`, `num`, `bool` or `string)`. A new node is created and added to the top of the stack. This element has value that was passed in as the parameter. The method does not return anything.

```
stack.add(5);
```

Size Calling `size()` returns an `int` with the number of elements in the stack.

```
Stack::num stack = new Stack::num();
stack.size();                          :( returns 0 ):
Stack::int stackTwo = new Stack::int(1, 2, 3);
stack.size();                          :( returns 3 ):
```

Queue

A queue is a data structure that represents FIFO (first-in-first-out) operations on a list of objects.

Initializing an instance of a Queue Queues can be initialized using one of two constructors.

```
Queue::num emptyQueue = new Queue::num();
```

The next example initializes a queue filled with several values. These values must all be of the same type: either `int`, `num`, `bool` or `string`.

```
Queue::int queue = new Queue::int(1, 2, 3);
```

Library Functions There are several builtin functions for manipulating a queue.

Peek To look at the head of the queue, use `peek()`. This function looks at, but does not remove, the element in the head of the queue. If queue is empty, this function returns `null`.

```
Queue::int queue = new Queue::int(1, 2, 3);
queue.peek();                          :( returns 1 ):
queue.peek();                          :( returns 1 ):
```

Add To add items to the tail of the queue, use `add(int, num, bool or string)`. A new node is created and added to the tail of the queue. This new element contains value that was passed into the parameter. The function does not return anything.

```
Queue::int queue = new Queue::int(1, 2, 3);
queue.add(4);
```

At this moment, the queue contains 4 elements: 1,2,3,4. 1 is the head of the queue, and 4 is the tail of the queue.

Remove To remove items from the head of the queue, use `remove()`. This function looks at the head of the queue. This function always returns `null`.

```
Queue::int queue = new Queue::int(1, 2, 3);
queue.remove();
```

At this moment, the queue contains 2 elements: 2,3. The element with value 1 was removed since it was the head of the queue.

Size Calling `size()` returns the number of elements in the queue.

```
Queue::num queue = new Queue::num();
queue.size();                              :( returns 0 ):
Queue::int queueTwo = new Queue::int(1, 2, 3);
queueTwo.size();                           :( returns 3 ):
```

LinkedList

A LinkedList is comprised of ListNode objects, which contain data (either a num or string), and a reference to the next ListNode.

Initializing an instance of a LinkedList Linked lists can be initialized using one of two constructors. The first produces an empty LinkedList object:

```
LinkedList::type emptyList = new LinkedList::type();
```

The second initializes a LinkedList filled with values of the same type.

```
LinkedList::int numList = new LinkedList::int(1, 2, 3, 4);
```

Library Functions There are several builtin functions for manipulating a Linked List.

Add To append items to the tail of the Linked List, use `add(int, num, bool or string)`. A node is created from the value passed into add, and is appended to the end of the list. Returns void.

```
numList.add(5);                            :( returns true ):
emptyList.add("not empty anymore");        :( returns true ):
```

Delete To delete an item from the list, call .delete(int, num, or string). The first node containing this value is removed. If there are multiple nodes with this value, all but the first remain. Returns true if this list contained the specified element, false otherwise.

```
numList.delete(3);                         :( returns true ):
```

Size Calling `size()` returns the number of elements in the list.

```
numList.size();                            :( returns 4 ):
```

Arrays

An array is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed.

Initializing an Array Array declarations are made by specifying the type, size, and name of the array. Array sizes are indicated at time of array creation and should be specified for the array to be created. Once created, array sizes are immutable. Arrays must also be initialized with elements at the time of creation. The example below illustrates the creation of arrays in Strux.

```
int[2] intArray = [5, 2];
string[2] name = ["hello", "world"];
num[0] numArray = [];       :( creates empty num array ):
```

Accessing an Array Array elements are accessed by their numerical index.

```
int[5] intArray = [2, 4, 6, 8, 10];
print(intArray[2]);         :( prints out 6 ):
```

Array values can also be assigned/modified by doing the following:

```
int[5] numArray = [2, 4, 6, 8, 10];
numArray[1] = 3;
numArray.show();            :( prints out [2, 3, 6, 8, 10] ):
```

QuickSort

QuickSort is sorting algorithm we use to sort arrays in Strux. QuickSort is a Divide and Conquer algorithm. We first consider the first, last, and middle element of the array. From these three elements, we will pick a pivot, which is the median of the three. To sort an array using quicksort, call the function `.quickSort()`. To visualize quicksort, call the function `.showQuickSort()`. An example is shown below:

Using `.quickSort()`:

```
int[7] arr = [10, 100, 30, 90, 40, 50, 70];
arr.quickSort();   :( calls quicksort on array ):
arr.show();
:( prints out sorted arr: [10, 30, 40, 50, 70, 90, 100] ):
```

Using `.showQuickSort()`

```
int[9] arr = [1, 4, 3, 6, 7, 2, 99, 23, 37];
arr.showQuickSort(); :( this will print the below ):
============================================================
At this step:
current array: [ 1 4 3 6 7 2 99 23 37 ]
numbers swapped: 23,7
array after swap: [ 1 4 3 6 23 2 99 7 37 ]
pivot is 7
numbers swapped: 99,7
array after swap: [ 1 4 3 6 23 2 7 99 37 ]
```

```
numbers swapped: 37,99
array after swap: [ 1 4 3 6 23 2 7 37 99 ]
============================================================
At this step:
current array: [ 1 4 3 6 23 2 7 37 99 ]
numbers swapped: 2,6
array after swap: [ 1 4 3 2 23 6 7 37 99 ]
pivot is 6
numbers swapped: 23,6
array after swap: [ 1 4 3 2 6 23 7 37 99 ]
numbers swapped: 7,23
array after swap: [ 1 4 3 2 6 7 23 37 99 ]
============================================================
At this step:
current array: [ 1 4 3 2 6 7 23 37 99 ]
numbers swapped: 2,3
array after swap: [ 1 4 2 3 6 7 23 37 99 ]
pivot is 3
============================================================
At this step:
current array: [ 1 4 2 3 6 7 23 37 99 ]
pivot swapped: 4,3
array after swap: [ 1 3 2 4 6 7 23 37 99 ]
numbers swapped: 2,3
array after swap: [ 1 2 3 4 6 7 23 37 99 ]
pivot is 3
============================================================
At this step:
current array: [ 1 2 3 4 6 7 23 37 99 ]
pivot is 2
============================================================
At this step:
current array: [ 1 2 3 4 6 7 23 37 99 ]
pivot is 1
============================================================
QuickSort complete! Final Result: [ 1 2 3 4 6 7 23 37 99 ]
```

BSTree

A tree is a data structure comprised of BSTreeNode objects, each of which has references to its children. In Strux, the tree is a binary search tree, meaning that it adheres to the following rules:

1. Each node has at most two children

2. All children in the left subtree of a node are less than the value of the parent node
3. All children in the right subtree of a node are greater than the value of the parent node
4. If `add()` is called on a value that already appears in a tree, it will not be added; the tree contains no duplicates.
5. BSTree only supports numbers: `int` or `num`

BSTree Declaration Initializing a binary search tree for integers in Strux is as easy as:

```
BSTree::int tree = new BSTree::int();
```

Additionally, a new binary search tree can be created with the following syntax:

```
BSTree::int tree = new BSTree::int(5,2,6,2,9);
```

This syntax is equivalent to creating a new, empty tree and then calling add to the tree on each of the numbers in the parentheses. Therefore, it is equivalent to:

```
BSTree::int tree = new BSTree::int();
tree.add(5);
tree.add(2);
tree.add(6);
tree.add(2);
tree.add(9);
```

Library Functions

Add Adds a new element to the tree. Because this is a binary search tree, the element is added is added according to its value. If the value is less than the root, the value is then compared to the left child of the root, and if the value is greater than or equal to the root, the value is compared to the right child of the root. This process is done recursively until the child that must be compared is null, at which point, a new TreeNode is created with the value to be added, and the TreeNode is added to the tree. This function returns void.

```
BSTree::int tree = new BSTree::int();    :( tree is empty ):
tree.add(5);                            :( tree now has 5 ):
tree.add(6);                      :( tree now has 5 and 6 ):
```

Delete Deletes a specified value from the tree. When the element is deleted, its children and parent are updated to reflect the change while still maintaining the binary search tree properties. The function returns void.

```
BSTree::int tree = new BSTree::int();
tree.add(5);
tree.add(6);
tree.delete(6);         :( tree now only has 5 ):
tree.delete(1);         :( tree unchanged ):
```

Contains Used to check if a certain value can be found within a tree. Simply returns true if the value is in the tree or false if it isn't.

```
BSTree::int tree = new BSTree::int();
tree.add(5);
tree.add(6);
tree.contains(5);       :( returns true ):
tree.contains(2);       :( returns false ):
```

3.5 Functions

3.5.1 Built-In

main()

A `main()` function that returns an `int` is required for every program to run. The program will not execute without a main method. The main method looks like this:

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

Note that in this main method we have introduced another built-in function, called `print()`.

print()

Print is called on a primitive data type to output its value to the console. We will illustrate its output for Strux's four primitive data types: `int`, `num`, `bool`, and `string`.

int The expressions

```
int x = 6;
print(x);
```

will print this to the console:

```
6
```

num The expressions

```
num y = 3.5;
print(y);
```

will print this to the console:

```
3.500000
```

bool Boolean values are output as the integer `1` for `true` and `0` for `false`. The expressions

```
bool z = true;
print(z);
```

will print this to the console:

```
1
```

string The expression

```
print("Hello World!");
```

will print this to the console:

```
Hello World!
```

.show()

`show()` is called on a data structure (stack, queue, linked list, tree, or array) and visualizes it. In the next examples, we will illustrate how show is used for our different data structures/types. Note that the Queue and LinkedList datatypes include condensed visualizations, which are shown when a data structure includes more than 10 elements.

Array The expressions

```
int[6] arr = [0, 1, 2, 3, 4, 5];
arr.show();
```

will print this to the console:

```
[0, 1, 2, 3, 4, 5]
```

LinkedList The expressions

```
LinkedList::int ll = new LinkedList::int(0, 1, 2, 3, 4, 5);
ll.show();
```

will print this to the console:

```
+---+  +---+  +---+  +---+  +---+  +---+  +------+
| 0 |->| 1 |->| 2 |->| 3 |->| 4 |->| 5 |->| null |
+---+  +---+  +---+  +---+  +---+  +---+  +------+
0      1      2      3      4      5      <- Index
```

Stack The expressions

```
Stack::int s = new Stack::int(1, 2, 3);
s.show();
```

will print this to the console:

```
+---+ <- Top
| 3 |
+---+
| 2 |
+---+
| 1 |
+---+
```

Queue The expressions

```
Queue::int q = new Queue::int(4, 5, 6, 1);
q.show();
```

will print this to the console:

```
+---+---+---+---+
| 4 | 5 | 6 | 1 |
+---+---+---+---+
Head         Tail
```

BSTree The expressions

```
BSTree::int tree = new BSTree::int(5,6,4,9,5,2);
tree.show();
```

will print this to the console:

```
      *--------(5)--------*
 *---(4)              * ---(6)---*
(2)                  (5)        (9)
```

3.5.2 User-defined Functions

The signature for a method includes the return type, function name, and list of formals with their types. For example:

```
bool isTrue(bool x) {
    if (x == true) {
        return true;
    }
    return false;
}
```

returns whether `x` is true or not. The return type is determined to be boolean, the method name is isTrue(), and the formals include one variable `x` of type `bool`. Note that if this method is defined to be

```
num isTrue(bool x) {
    if (x == true) {
        return true;
    }
    return false;
}
```

Strux will throw an error due to incompatible return types.

Chapter 4

Project Plan

4.1 Process

4.1.1 Planning

Our group met approximately twice a week throughout the semester. We spent at least half an hour on Fridays with our TA, Chang Liu, to ask questions that we had made note of, gauge our progress, and solicit advice on the upcoming work. We met again on Tuesday or Wednesday nights, and often over the weekend as well. In the beginning of the semester, these sessions often involved looking at code together, first combing through MicroC and later implementing the foundation of our language. This time spent together gave everyone a very good idea of what Strux would look like. Later in the semester, these meetings functioned as an opportunity to check in, assign tasks and deadlines, and debug tricky issues.

4.1.2 Specification

The features we wanted to implement in Strux were specified first in our Language Reference Manual. Before development began, we were able to specify an MVP set of features that we felt were necessary in Strux, which mostly included data structures and the `.show()` functionality. During development, we realized that some previously specified features were no longer relevant to Strux (like a `forEach` loop), while others no longer made sense (like a single data type to represent both integers and floats). A quick meeting or conversation over chat was able to resolve these issues and prioritize language features. The LRM was updated throughout this process.

4.1.3 Development

Development was done in the same order as compiler architecture: scanner, parser, AST, semantic checking, and code generation. Some features that we

had deprioritized and shelved for later were implemented in December, and the scanner, parser, and other files were updated accordingly.

4.1.4 Testing

Tests were always included with the addition of a new feature to the language. When writing tests, we strove to include tests that both addressed basic functionality (unit tests), and combined multiple features (integration tests). Additionally, we spent time towards the end of development increasing our test coverage. We added tests to stress-test our data structures when many elements were added, address the precision of floats, and check edge cases, like calling a function on an empty structure.

4.2 Style Guide

The following are suggestions for keeping your Strux code clean and readable:

- Use camelCase for variable and function names (e.g. `addTwoNumbers()`, `medianValue`).
- Use spaces instead of the tab character. This ensures uniformity across all devices and text editors.
- The suggested indentation is 4 spaces.
- Limit line length to 80 characters.
- Keep comments concise, but descriptive. Below is the recommended styling for single- and multi-line comments, though any formatting within the comment symbols is valid.

```
:( This comment is one line. ):
:(
    Here's a comment that takes up a few lines.
    It's a text block. Describe a function and
    its return values like this.
):
```

4.3 Timeline

Date	Activities
September 17	First meeting
September 25	Proposal submitted
October 6	Proposal feedback from Chang; begin work on LRM
October 16	LRM submitted
Oct 23 - Nov 11	Group meetings about MicroC and Hello World
November 10	Hello World demo with initial test suite
November 11-17	Implementation and testing of standard language features done
December 8	Final meeting with Chang
December 11	All data structures implemented
December 11 - 20	Adding last minute features, cleaning up code, presentation and report preparation

4.4 Roles

Our group assumed roles at the beginning of the project, and while we tried to take these into account when relevant, everyone naturally assumed a variety of responsibilities. Work on the scanner, parser, and AST was done together in person for the submission of Hello World. After that, we decided to have each member implement one data structure, and share the work on code generation and semantic checking. Our team would generally pick tasks to work on for the week based on availability and knowledge of relevant areas of the codebase.

Name	Role	Data Structures
Josh	Language Guru	Tree
Kofi	System Architect	Linked list, quick sort
Millie	Tester	Stack, queue
Sophie	Manager	Array

4.5 Development Environment

4.5.1 Tools

One major roadblock that our group encountered involved a problem with the LLVM Bitreader module on macOS High Sierra. The bug prevented us from linking C files to our Strux executable on machines with this OS. Since Millie and Josh, who had not yet updated to High Sierra, experienced no problems, we decided to create a Digital Ocean droplet running Ubuntu 16.04 and Sophie and Kofi worked there. Since Millie and Josh were able to compile locally, they continued to do so.

On the Ubuntu platform, we developed using vim. On the local machines, members chose between vim, Sublime Text, and Atom. We also made use of an

Atom plugin, `atom-pair`, which provided a Google Docs-like collaborative real-time environment to program in. This was particularly useful in the beginning when we were developing the scanner and parser.

We established a GitHub-hosted Git repository for maintaining our code. It contains all source code, as well as tests and scripts. Google Docs was used heavily to develop the proposal, LRM, and final report. It was also home to a constantly updated To Do list. General, frequent communication took place on Facebook Messenger.

4.5.2 Languages

- **Ocaml** (4.2.3) for compiler implementation
- **C** for data structure source code
- **clang** for linking our compiler and C code

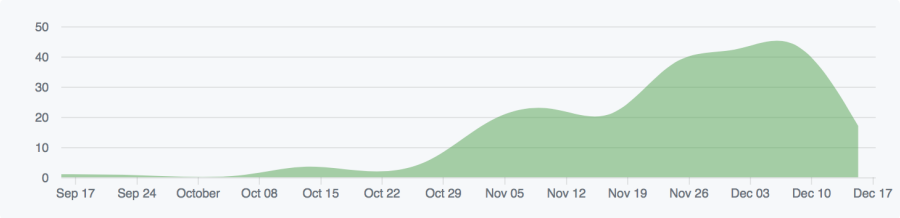
4.6 Project Log

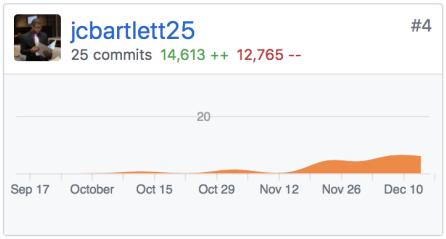
GitHub reported the following statistics for project development. We noticed that the line count additions are skewed high for Millie and Josh as a result of some long stress tests that they pushed and their output files. We've also included our git log for the master branch of the project, which can be found in the Appendix on page 126.

Sep 17, 2017 – Dec 19, 2017

Contributions: **Commits**

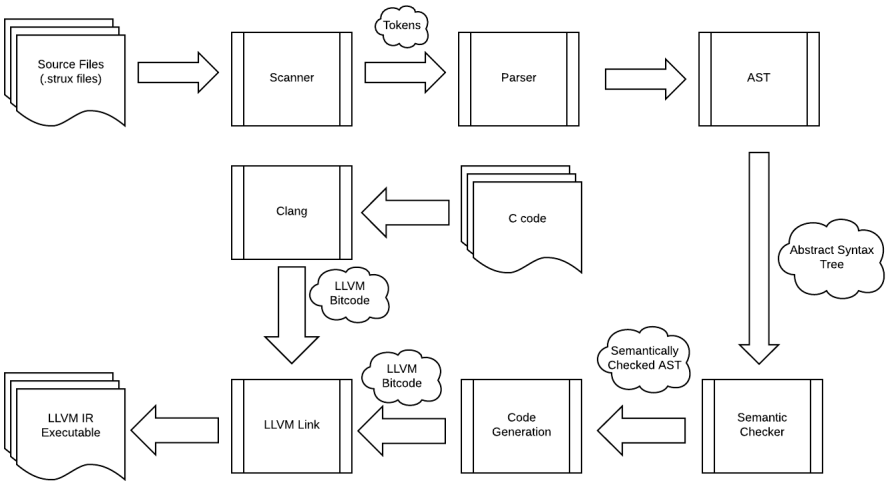
Contributions to master, excluding merge commits





Chapter 5

Architectural Design



The architectural design of Strux is very similar to that of MicroC, with the exception of Strux having additional C modules that are linked with codegen LLVM bitcode to form the LLVM IR.

Source files, which are in the `.strux` format, are taken in by the scanner, where characters in the `.strux` files are converted to tokens. Tokens are generated based on the various regular expression and CFG rules we have in the scanner file for catching and identifying keywords, values and so on. These generated tokens are then passed on the parser (`parser.mly`) where they are used to build an Abstract Syntax Tree (AST) of the tokens generated.

The AST is then checked for semantic errors by `semant.ml`. If errors are found, exceptions are thrown to handle the caught errors. If none are found, the semantically checked AST is passed on to the code generator (`codegen.ml`)

for Ocaml-LLVM binding.

In code generator (`codegen.ml`), the syntactically checked AST undergoes postorder traversal to produce LLVM bitcode using Ocaml-LLVM bindings. C files for our data structures are also compiled with clang to produce bitcode. The LLVM Linker links bitcode from the C files with the bitcode from `codegen.ml` to produce the LLVM IR. This is done upon execution of a program, most frequently in our test script (`testall.sh`).

The scanner, parser, and AST were implemented by the whole team together in person. We also implemented basic language features in codegen and semant during these sessions. Later, C code was developed separately: Josh handled BSTree files, Millie developed Stack and Queue, Kofi worked on LinkedLists and quicksort, and Sophie did arrays, as well as the printing for LinkedList, Stack, Queue, and array. Development of these features also involved making the appropriate updates to the semantic checker and codegen file.

Chapter 6

Test Plan

We wrote both unit and integration tests to test all our main features as well as language features. Specifically, we decided to test for the following:

1. All operations that we have created (`+`, `-`, `%`, `++`, `-`, `<=`, `>=`, `*`, `and`, `or`, `>`, `<`, `/`, `!=`, `==`)
2. All functions for our specific data structures
3. Initialization of objects
4. Static vs. dynamic scoping
5. Edge cases
   (a) Null cases, such as calling remove when queue is empty
   (b) Printing data structures with many elements or no elements
   (c) Having very large numbers, and very small numbers
   (d) Adding 1000+ values to data structures
6. Number/string variations
   (a) Random characters
   (b) Negative numbers
   (c) Float precision
   (d) Numerical operations on floats/integers
7. Calling functions that do not exist
8. Terminating gracefully and throwing meaningful error
9. Duplication (duplicate variable names)

10. Incompatible types (e.g. trying to initialize 3 as a `num` instead of an `int`, multiplying an integer by a float, or trying to put a stack inside a constructor of a queue all return function type mismatch, especially in main function, and so on)

Whenever a teammate implemented a feature, they included tests for this (specific data structure implementation is specified in section 4.4). As tester, Millie also wrote many stress tests at the end of the project addressing the above areas.

6.1 Unit Testing

Here is an example of one of our unit tests:

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

And another example of a unit test:

```
int main() {
    if (true) {
        print("it's true!");
    }
    return 0;
}
```

6.2 Integration Testing

An example of our integration test to run a greatest common denominator algorithm (`test-gcd.strux`). This test makes use of several Strux features, including formal arguments, loops, conditionals, and modulo operators. It returns 9 as output.

```
int main() {
    print(gcd(81, 153));
    return 0;
}

int gcd(int n1, int n2) {
    int gcd = 1;

    for (int i = 1; i <= n1 and i <= n2; i++) {
      if (n1 % i == 0 and n2 % i == 0) {
          gcd = i;
      }
```

```
        }
        return gcd;
    }
```

Tthe target language LLVM code can be found in the Appendix section 8.15 on page 123.

6.3 Error Handling

Here is an example of how we terminated gracefully:

```
int main() {
    int[2] myArr = [2, 5, 6, 7];
    return 0;
}
```

Returns:

```
Fatal error: exception Failure("Invalid length declaration")
```

Another example of error handling:

```
void hello() {

}
```

Returns:

```
Fatal error: exception Failure("unrecognized function main")
```

6.4 Representative Programs

6.4.1 test-queue-8.strux

We decided to test that all functions for our data structures operate as intended, on different primitive types we have declared in our language. Specifically in this program we decided to add and remove objects and test that the queue data structure still preserves its FILO quality.

```
1  int main() {
2      Queue::num qn = new Queue::num();
3      qn.add(3.6);
4      qn.add(4.216);
5      qn.add(-5.25);
6      qn.add(6.4);
7      qn.show();
8
```

```
    Queue::int qi = new Queue::int();
    qi.add(23623);
    qi.add(-39632);
    qi.add(4 * 1);
    qi.add(356);
    qi.remove();
    qi.add(89494);
    qi.add(22);
    qi.show();

    Queue::string qs = new Queue::string();
    qs.add("Josh");
    qs.add("Kofi");
    qs.add("Millie");
    qs.add("Sophie");
    int qsSize = qs.size();
    print(qsSize);
    qs.show();

    Queue::bool qb = new Queue::bool();
    qb.add(true);
    qb.add(false);
    qb.show();

    return 0;
}
```

Target code for this program:

```
; ModuleID = 'Strux'

%struct.Queue = type { i32, %struct.Node*, %struct.Node* }
%struct.Node = type { %struct.Node*, i8* }
%struct.LinkedList = type { %struct.ListNode*, i32 }
%struct.ListNode = type { i8*, %struct.ListNode* }
%struct.Stack = type { i32, %struct.Node.0* }
%struct.Node.0 = type { %struct.Node.0*, i8* }
%struct.BSTree = type { %struct.BSTreeNode* }
%struct.BSTreeNode = type { i8*, %struct.BSTreeNode*,
 ↪  %struct.BSTreeNode*, %struct.BSTreeNode* }

@string = private unnamed_addr constant [5 x i8] c"Josh\00"
@string.1 = private unnamed_addr constant [5 x i8] c"Kofi\00"
@string.2 = private unnamed_addr constant [7 x i8] c"Millie\00"
@string.3 = private unnamed_addr constant [7 x i8] c"Sophie\00"
@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
```

```
17
18  declare i32 @printf(i8*, ...)
19
20  declare %struct.Queue* @initQueue()
21
22  declare void @enqueue(%struct.Queue*, i8*)
23
24  declare void @dequeue(%struct.Queue*)
25
26  declare i8* @peek(%struct.Queue*)
27
28  declare i32 @queue_size(%struct.Queue*)
29
30  declare void @queue_show_int(%struct.Queue*)
31
32  declare void @queue_show_float(%struct.Queue*)
33
34  declare void @queue_show_string(%struct.Queue*)
35
36  declare %struct.LinkedList* @initList()
37
38  declare void @add(%struct.LinkedList*, i8*)
39
40  declare void @delete(%struct.LinkedList*, i32)
41
42  declare i8* @get(%struct.LinkedList*, i32)
43
44  declare i32 @size(%struct.LinkedList*)
45
46  declare void @ll_show_int(%struct.LinkedList*)
47
48  declare void @ll_show_float(%struct.LinkedList*)
49
50  declare void @ll_show_string(%struct.LinkedList*)
51
52  declare %struct.Stack* @initStack()
53
54  declare void @push(%struct.Stack*, i8*)
55
56  declare void @pop(%struct.Stack*)
57
58  declare i8* @top(%struct.Stack*)
59
60  declare i32 @stack_size(%struct.Stack*)
61
62  declare void @stack_show_int(%struct.Stack*)
```

```
63
64  declare void @stack_show_float(%struct.Stack*)
65
66  declare void @stack_show_string(%struct.Stack*)
67
68  declare i32* @cQuickSort(i32*, i32)
69
70  declare void @cShowQuickSort(i32*, i32)
71
72  declare double* @cQuickfSort(double*, i32)
73
74  declare void @cShowfQuickSort(double*, i32)
75
76  declare i8** @cQuicksSort(i8**, i32)
77
78  declare void @cShowsQuickSort(i8**, i32)
79
80  declare %struct.BSTree* @initBSTree()
81
82  declare void @addIntToTree(%struct.BSTree*, i8*)
83
84  declare void @addNumToTree(%struct.BSTree*, i8*)
85
86  declare void @deleteIntFromTree(%struct.BSTree*, i32)
87
88  declare void @deleteNumFromTree(%struct.BSTree*, double)
89
90  declare i1 @treeContainsInt(%struct.BSTree*, i32)
91
92  declare i1 @treeContainsFloat(%struct.BSTree*, double)
93
94  declare void @showIntTree(%struct.BSTree*)
95
96  declare void @showNumTree(%struct.BSTree*)
97
98  declare i32 @printbig(i32)
99
100 define i32 @main() {
101 entry:
102   %qn = alloca %struct.Queue*
103   %init = call %struct.Queue* @initQueue()
104   store %struct.Queue* %init, %struct.Queue** %qn
105   %qn1 = load %struct.Queue*, %struct.Queue** %qn
106   %malloccall = tail call i8* @malloc(i32 ptrtoint (double*
      ↪  getelementptr (double, double* null, i32 1) to i32))
107   %tmp = bitcast i8* %malloccall to double*
```

```
  store double 3.600000e+00, double* %tmp
  %ptr = bitcast double* %tmp to i8*
  call void @enqueue(%struct.Queue* %qn1, i8* %ptr)
  %qn2 = load %struct.Queue*, %struct.Queue** %qn
  %malloccall.3 = tail call i8* @malloc(i32 ptrtoint (double*
  ↪  getelementptr (double, double* null, i32 1) to i32))
  %tmp4 = bitcast i8* %malloccall.3 to double*
  store double 4.216000e+00, double* %tmp4
  %ptr5 = bitcast double* %tmp4 to i8*
  call void @enqueue(%struct.Queue* %qn2, i8* %ptr5)
  %qn6 = load %struct.Queue*, %struct.Queue** %qn
  %malloccall.7 = tail call i8* @malloc(i32 ptrtoint (double*
  ↪  getelementptr (double, double* null, i32 1) to i32))
  %tmp8 = bitcast i8* %malloccall.7 to double*
  store double -5.250000e+00, double* %tmp8
  %ptr9 = bitcast double* %tmp8 to i8*
  call void @enqueue(%struct.Queue* %qn6, i8* %ptr9)
  %qn10 = load %struct.Queue*, %struct.Queue** %qn
  %malloccall.11 = tail call i8* @malloc(i32 ptrtoint (double*
  ↪  getelementptr (double, double* null, i32 1) to i32))
  %tmp12 = bitcast i8* %malloccall.11 to double*
  store double 6.400000e+00, double* %tmp12
  %ptr13 = bitcast double* %tmp12 to i8*
  call void @enqueue(%struct.Queue* %qn10, i8* %ptr13)
  %qn14 = load %struct.Queue*, %struct.Queue** %qn
  call void @queue_show_float(%struct.Queue* %qn14)
  %qi = alloca %struct.Queue*
  %init15 = call %struct.Queue* @initQueue()
  store %struct.Queue* %init15, %struct.Queue** %qi
  %qi16 = load %struct.Queue*, %struct.Queue** %qi
  %malloccall.17 = tail call i8* @malloc(i32 ptrtoint (i32*
  ↪  getelementptr (i32, i32* null, i32 1) to i32))
  %tmp18 = bitcast i8* %malloccall.17 to i32*
  store i32 23623, i32* %tmp18
  %ptr19 = bitcast i32* %tmp18 to i8*
  call void @enqueue(%struct.Queue* %qi16, i8* %ptr19)
  %qi20 = load %struct.Queue*, %struct.Queue** %qi
  %malloccall.21 = tail call i8* @malloc(i32 ptrtoint (i32*
  ↪  getelementptr (i32, i32* null, i32 1) to i32))
  %tmp22 = bitcast i8* %malloccall.21 to i32*
  store i32 -39632, i32* %tmp22
  %ptr23 = bitcast i32* %tmp22 to i8*
  call void @enqueue(%struct.Queue* %qi20, i8* %ptr23)
  %qi24 = load %struct.Queue*, %struct.Queue** %qi
  %malloccall.25 = tail call i8* @malloc(i32 ptrtoint (i32*
  ↪  getelementptr (i32, i32* null, i32 1) to i32))
```

```
148   %tmp26 = bitcast i8* %malloccall.25 to i32*
149   store i32 4, i32* %tmp26
150   %ptr27 = bitcast i32* %tmp26 to i8*
151   call void @enqueue(%struct.Queue* %qi24, i8* %ptr27)
152   %qi28 = load %struct.Queue*, %struct.Queue** %qi
153   %malloccall.29 = tail call i8* @malloc(i32 ptrtoint (i32*
      ↪  getelementptr (i32, i32* null, i32 1) to i32))
154   %tmp30 = bitcast i8* %malloccall.29 to i32*
155   store i32 356, i32* %tmp30
156   %ptr31 = bitcast i32* %tmp30 to i8*
157   call void @enqueue(%struct.Queue* %qi28, i8* %ptr31)
158   %qi32 = load %struct.Queue*, %struct.Queue** %qi
159   call void @dequeue(%struct.Queue* %qi32)
160   %qi33 = load %struct.Queue*, %struct.Queue** %qi
161   %malloccall.34 = tail call i8* @malloc(i32 ptrtoint (i32*
      ↪  getelementptr (i32, i32* null, i32 1) to i32))
162   %tmp35 = bitcast i8* %malloccall.34 to i32*
163   store i32 89494, i32* %tmp35
164   %ptr36 = bitcast i32* %tmp35 to i8*
165   call void @enqueue(%struct.Queue* %qi33, i8* %ptr36)
166   %qi37 = load %struct.Queue*, %struct.Queue** %qi
167   %malloccall.38 = tail call i8* @malloc(i32 ptrtoint (i32*
      ↪  getelementptr (i32, i32* null, i32 1) to i32))
168   %tmp39 = bitcast i8* %malloccall.38 to i32*
169   store i32 22, i32* %tmp39
170   %ptr40 = bitcast i32* %tmp39 to i8*
171   call void @enqueue(%struct.Queue* %qi37, i8* %ptr40)
172   %qi41 = load %struct.Queue*, %struct.Queue** %qi
173   call void @queue_show_int(%struct.Queue* %qi41)
174   %qs = alloca %struct.Queue*
175   %init42 = call %struct.Queue* @initQueue()
176   store %struct.Queue* %init42, %struct.Queue** %qs
177   %qs43 = load %struct.Queue*, %struct.Queue** %qs
178   %malloccall.44 = tail call i8* @malloc(i32 ptrtoint (i1**
      ↪  getelementptr (i1*, i1** null, i32 1) to i32))
179   %tmp45 = bitcast i8* %malloccall.44 to i8**
180   store i8* getelementptr inbounds ([5 x i8], [5 x i8]* @string,
      ↪  i32 0, i32 0), i8** %tmp45
181   %ptr46 = bitcast i8** %tmp45 to i8*
182   call void @enqueue(%struct.Queue* %qs43, i8* %ptr46)
183   %qs47 = load %struct.Queue*, %struct.Queue** %qs
184   %malloccall.48 = tail call i8* @malloc(i32 ptrtoint (i1**
      ↪  getelementptr (i1*, i1** null, i32 1) to i32))
185   %tmp49 = bitcast i8* %malloccall.48 to i8**
186   store i8* getelementptr inbounds ([5 x i8], [5 x i8]*
      ↪  @string.1, i32 0, i32 0), i8** %tmp49
```

```
187   %ptr50 = bitcast i8** %tmp49 to i8*
188   call void @enqueue(%struct.Queue* %qs47, i8* %ptr50)
189   %qs51 = load %struct.Queue*, %struct.Queue** %qs
190   %malloccall.52 = tail call i8* @malloc(i32 ptrtoint (i1**
      ↪  getelementptr (i1*, i1** null, i32 1) to i32))
191   %tmp53 = bitcast i8* %malloccall.52 to i8**
192   store i8* getelementptr inbounds ([7 x i8], [7 x i8]*
      ↪  @string.2, i32 0, i32 0), i8** %tmp53
193   %ptr54 = bitcast i8** %tmp53 to i8*
194   call void @enqueue(%struct.Queue* %qs51, i8* %ptr54)
195   %qs55 = load %struct.Queue*, %struct.Queue** %qs
196   %malloccall.56 = tail call i8* @malloc(i32 ptrtoint (i1**
      ↪  getelementptr (i1*, i1** null, i32 1) to i32))
197   %tmp57 = bitcast i8* %malloccall.56 to i8**
198   store i8* getelementptr inbounds ([7 x i8], [7 x i8]*
      ↪  @string.3, i32 0, i32 0), i8** %tmp57
199   %ptr58 = bitcast i8** %tmp57 to i8*
200   call void @enqueue(%struct.Queue* %qs55, i8* %ptr58)
201   %qsSize = alloca i32
202   %qs59 = load %struct.Queue*, %struct.Queue** %qs
203   %isEmpty = call i32 @queue_size(%struct.Queue* %qs59)
204   store i32 %isEmpty, i32* %qsSize
205   %qsSize60 = load i32, i32* %qsSize
206   %printf = call i32 (i8*, ...) @printf(i8* getelementptr
      ↪  inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32 0), i32
      ↪  %qsSize60)
207   %qs61 = load %struct.Queue*, %struct.Queue** %qs
208   call void @queue_show_string(%struct.Queue* %qs61)
209   %qb = alloca %struct.Queue*
210   %init62 = call %struct.Queue* @initQueue()
211   store %struct.Queue* %init62, %struct.Queue** %qb
212   %qb63 = load %struct.Queue*, %struct.Queue** %qb
213   %malloccall.64 = tail call i8* @malloc(i32 ptrtoint (i1*
      ↪  getelementptr (i1, i1* null, i32 1) to i32))
214   %tmp65 = bitcast i8* %malloccall.64 to i1*
215   store i1 true, i1* %tmp65
216   %ptr66 = bitcast i1* %tmp65 to i8*
217   call void @enqueue(%struct.Queue* %qb63, i8* %ptr66)
218   %qb67 = load %struct.Queue*, %struct.Queue** %qb
219   %malloccall.68 = tail call i8* @malloc(i32 ptrtoint (i1*
      ↪  getelementptr (i1, i1* null, i32 1) to i32))
220   %tmp69 = bitcast i8* %malloccall.68 to i1*
221   store i1 false, i1* %tmp69
222   %ptr70 = bitcast i1* %tmp69 to i8*
223   call void @enqueue(%struct.Queue* %qb67, i8* %ptr70)
224   %qb71 = load %struct.Queue*, %struct.Queue** %qb
```

```
  call void @queue_show_int(%struct.Queue* %qb71)
  ret i32 0
}

declare noalias i8* @malloc(i32)
```

The program outputs the following to the console:

```
+----------+----------+-----------+----------+
| 3.600000 | 4.216000 | -5.250000 | 6.400000 |
+----------+----------+-----------+----------+
Head                                    Tail
+--------+---+-----+-------+----+
| -39632 | 4 | 356 | 89494 | 22 |
+--------+---+-----+-------+----+
Head                         Tail
4
+------+------+--------+--------+
| Josh | Kofi | Millie | Sophie |
+------+------+--------+--------+
Head                         Tail
+---+---+
| 1 | 0 |
+---+---+
Head Tail
```

6.4.2 test-bstree-8.strux

Another example of a test we created that represents our language is a test on whether a BSTree is empty. This is an edge case test to ensure that the user understands that the tree is empty, and it is not that there is no output.

```
int main() {
    BSTree::int tree1 = new BSTree::int();
    tree1.show();
    return 0;
}
```

Target code for this program:

```
; ModuleID = 'Strux'

%struct.Queue = type { i32, %struct.Node*, %struct.Node* }
%struct.Node = type { %struct.Node*, i8* }
%struct.LinkedList = type { %struct.ListNode*, i32 }
%struct.ListNode = type { i8*, %struct.ListNode* }
```

```
%struct.Stack = type { i32, %struct.Node.0* }
%struct.Node.0 = type { %struct.Node.0*, i8* }
%struct.BSTree = type { %struct.BSTreeNode* }
%struct.BSTreeNode = type { i8*, %struct.BSTreeNode*,
 ↪  %struct.BSTreeNode*, %struct.BSTreeNode* }

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

declare %struct.Queue* @initQueue()

declare void @enqueue(%struct.Queue*, i8*)

declare void @dequeue(%struct.Queue*)

declare i8* @peek(%struct.Queue*)

declare i32 @queue_size(%struct.Queue*)

declare void @queue_show_int(%struct.Queue*)

declare void @queue_show_float(%struct.Queue*)

declare void @queue_show_string(%struct.Queue*)

declare %struct.LinkedList* @initList()

declare void @add(%struct.LinkedList*, i8*)

declare void @delete(%struct.LinkedList*, i32)

declare i8* @get(%struct.LinkedList*, i32)

declare i32 @size(%struct.LinkedList*)

declare void @ll_show_int(%struct.LinkedList*)

declare void @ll_show_float(%struct.LinkedList*)

declare void @ll_show_string(%struct.LinkedList*)

declare %struct.Stack* @initStack()

declare void @push(%struct.Stack*, i8*)

declare void @pop(%struct.Stack*)

```

```
52  declare i8* @top(%struct.Stack*)
53
54  declare i32 @stack_size(%struct.Stack*)
55
56  declare void @stack_show_int(%struct.Stack*)
57
58  declare void @stack_show_float(%struct.Stack*)
59
60  declare void @stack_show_string(%struct.Stack*)
61
62  declare i32* @cQuickSort(i32*, i32)
63
64  declare void @cShowQuickSort(i32*, i32)
65
66  declare double* @cQuickfSort(double*, i32)
67
68  declare void @cShowfQuickSort(double*, i32)
69
70  declare i8** @cQuicksSort(i8**, i32)
71
72  declare void @cShowsQuickSort(i8**, i32)
73
74  declare %struct.BSTree* @initBSTree()
75
76  declare void @addIntToTree(%struct.BSTree*, i8*)
77
78  declare void @addNumToTree(%struct.BSTree*, i8*)
79
80  declare void @deleteIntFromTree(%struct.BSTree*, i32)
81
82  declare void @deleteNumFromTree(%struct.BSTree*, double)
83
84  declare i1 @treeContainsInt(%struct.BSTree*, i32)
85
86  declare i1 @treeContainsFloat(%struct.BSTree*, double)
87
88  declare void @showIntTree(%struct.BSTree*)
89
90  declare void @showNumTree(%struct.BSTree*)
91
92  declare i32 @printbig(i32)
93
94  define i32 @main() {
95  entry:
96    %tree1 = alloca %struct.BSTree*
97    %init = call %struct.BSTree* @initBSTree()
```

```
98    store %struct.BSTree* %init, %struct.BSTree** %tree1
99    %tree11 = load %struct.BSTree*, %struct.BSTree** %tree1
100   call void @showIntTree(%struct.BSTree* %tree11)
101   ret i32 0
102 }
```

The program outputs the following to the console:

```
Tree is empty!
```

6.5 Automation

The code of our test script, `testall.sh`, can be found in the Appendix on page 118. We compile C code for our data structures into bitcode and use bitcode as dependencies for our program. All our C code is under the folder `c/`. Other than that our test script is similar to MicroC's test script, looping through our `tests/` directory, running each program, and comparing its output to an expected `.out` or `.err` file.

Chapter 7

Lessons Learned

Josh

As cliché as it may sound, I would say that the most important lesson I learned from this project would definitely be the teamwork aspect of the assignment. While I've worked on software in teams before for various internships, those experiences didn't compare to this project at all. During my internships, I had a manager that would set deadlines for me and made sure that I wasn't struggling, but for this project, it was the first time when everyone on the team was on a similar level of experience. Due to this, we had to set our own deadlines and hold each other accountable for the entirety of the semester. Thanks to our TA giving us a very honest talk about how much work we would need to put into this project at the beginning of the semester, we were all pretty motivated to start early and make sure we remained consistent throughout the whole semester. I believe this advice was key for us and the main reason why we never had to pull any crazy long nights. Since there is only one midpoint official deadline for the project, it is very easy to fall behind and forget about it until a couple weeks before the final deadline, but we kept setting internal deadlines for ourselves for when we wanted different features/data structures done. That is the reason we were able to get our project done early with all the features we planned. Starting early and holding each other accountable throughout the year is the best advice I can give.

Fredrick

This was the first time I delved into functional programming and it indeed took a while to get used to. I personally had to take a step back and really read about how Ocaml worked and the O'Reilly book series which is my usual go-to, has a very useful book on Ocaml. Working in teams, especially one in which our functional programming strengths had not been evident yet, was also very interesting. We assigned roles in the beginning based on our assumed levels of

expertise, but as time went on and our experience with the Ocaml and LLVM stack deepened, we started to develop new skills that were not necessarily in line with our initial roles. Having a team such as ours, which had a lot of people good in many areas, fostered a lot of overlap and encouraged us to help one another when a blocker was reached. This collaborative and friendly environment is one that helped me to become a better team member and more empathetic towards other people's struggles and work/code. My advice on the project is to start early and be in constant communication with the TA about the state of your project. We asked TA a lot of questions initially and that helped to keep us on track so that we did not have to rush in the end. Having teammates that are also respectful of your time and effort is essential, as it helps to foster a great team dynamic. If you start early and have proactive teammates, you'll be on the right track.

Millie

This is actually the first computer science class I took that has such a huge project portion. Writing a programming language is not an easy task at all, and doing it in a team has proven to be difficult at times because of varying schedules and other inconveniences. There were times when all of us could work together, but there were also times when some of us were busier than others. I think the main takeaway from this class is that scheduling and communication between teammates are very important. Without setting strict deadlines and schedules way before the actual deadline, we would not have been able to complete our task. Following a SCRUM-like schedule has proven to be very useful in that every week becomes goal-oriented. Having a Messenger group chat where we ping each other about our progress updates meant that we would push each other to work by demonstrating what we have done. Other than what has been mentioned above, this project has also taught me that titles do not mean much. I think that all of us have taken on different roles throughout this project, and if anything, focusing on contribution based on talent is more important than mere titles we established earlier in the class. Since a programming language is hard to do on your own, it is important to work with others and use your strengths to cater for others' weaknesses, or vice versa. If we had merely focused on working on what we were originally assigned, this project would not have been done. It is very interesting to see how specialization in tasks might not necessarily connote to higher productivity, especially as it hits the turning point of diminishing returns.

Sophie

Before the semester, and honestly throughout most of it, I was terrified by this project. However, having completed it successfully without a single all-nighter, I think I can safely say that it was not worth stressing over because of the diligence

of our team. We heeded the advice to start early, which was probably the main reason for our success. We also were careful with setting our priorities for the project, which meant that we had a limited working version of the language weeks before the deadline. This even allowed us to implement features not originally included in our language proposal, like trees. All of that is the result of constant communication. Chatting on Facebook and providing encouragement to teammates meant that we were working on the project constantly, and even made it fun. Everyone felt a sense of accountability to one another, and I think that allowed us to submit something we are all proud of. My advice to other teams is to take this project, but not yourselves, too seriously, and things will be fine.

Chapter 8

Appendix

8.1 scanner.mll

Josh, Kofi, Millie, Sophie

```
(* Ocamllex scanner for Strux *)

{ open Parser }

let whitespace = [' ' '\t' '\r' '\n']
let digits = ['0'-'9']
let integer = digits+
let decimal = ['.']
let esc = '\\' ['\\' ''' '"' 'n' 'r' 't']
let ascii = ([' '-'!' '#'-'[' ']'-'~'])
let string = '"' ( (ascii | esc)* as s) '"'
let float = digits* decimal digits+ | digits+ decimal digits*
let alphabet = ['a'-'z' 'A'-'Z']
let alphanumund = alphabet | digits | '_'
let id = alphabet alphanumund*

rule token = parse
    whitespace  { token lexbuf } (* Whitespace *)
  | ":("        { comment lexbuf } (* Comments *)
  | '('         { LPAREN }
  | ')'         { RPAREN }
  | '{'         { LBRACE }
  | '}'         { RBRACE }
  | '['         { LBRACK }
  | ']'         { RBRACK }
  | ';'         { SEMI }
  | ','         { COMMA }
```

```
28  | '+'         { PLUS }
29  | '-'         { MINUS }
30  | '*'         { TIMES }
31  | '/'         { DIVIDE }
32  | '%'         { MOD }
33  | '.'         { DOT }
34  | "++"        { INCR }
35  | "--"        { DECR }
36  | '='         { ASSIGN }
37  | "::"        { DOUBLECOL }
38  | "=="        { EQ }
39  | "!="        { NEQ }
40  | '<'         { LT }
41  | "<="        { LEQ }
42  | ">"         { GT }
43  | ">="        { GEQ }
44  | "and"       { AND }
45  | "or"        { OR }
46  | "not"       { NOT }
47  | "if"        { IF }
48  | "elif"      { ELIF }
49  | "else"      { ELSE }
50  | "for"       { FOR }
51  | "while"     { WHILE }
52  | "return"        { RETURN }
53  | "num"           { NUM }
54  | "int"           { INT }
55  | "bool"          { BOOL }
56  | "string"        { STRING }
57  | "void"          { VOID }
58  | "true"          { TRUE }
59  | "false"         { FALSE }
60  | "new"           { NEW }
61  | "null"          { NULL }
62  | "Queue"         { QUEUE }
63  | "LinkedList"    { LINKEDLIST }
64  | "BSTree"        { BSTREE }
65  | "Stack"         { STACK }
66  | integer as lxm      { INT_LITERAL(int_of_string lxm) }
67  | id as lxm             { ID(lxm) }
68  | float as fltlit       { NUM_LITERAL(float_of_string fltlit)
    ↪  }
69  | digits+ as intlit     { INT_LITERAL(int_of_string intlit) }
70  | string                { STRING_LITERAL(s) }
71  | eof                   { EOF }
```

```
72    | _ as char { raise (Failure("illegal character " ^
      ↪  Char.escaped char)) }
73
74  and comment = parse
75    "):" { token lexbuf }
76    | _    { comment lexbuf }
```

8.2 parser.mly

Josh, Kofi, Millie, Sophie

```
1   %{ open Ast %}
2
3   %token SEMI LPAREN RPAREN LBRACE RBRACE LBRACK RBRACK COMMA
    ↪  DOUBLECOL
4   %token PLUS MINUS TIMES DIVIDE INCR DECR MOD ASSIGN NOT
5   %token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
6   %token RETURN NULL IF ELIF ELSE NEW FOR WHILE NUM INT BOOL
    ↪  STRING VOID DOT
7   %token QUEUE LINKEDLIST STACK BSTREE
8   %token <float> NUM_LITERAL
9   %token <int> INT_LITERAL
10  %token <string> STRING_LITERAL
11  %token <string> ID
12  %token EOF
13
14  %nonassoc NOELSE
15  %nonassoc ELSE
16  %nonassoc ELIF
17  %left INCR DECR
18  %right ASSIGN
19  %left OR
20  %left AND DOT
21  %left EQ NEQ
22  %left LT GT LEQ GEQ
23  %left PLUS MINUS
24  %left TIMES DIVIDE
25  %right MOD
26  %right NOT NEG
27
28  %start program
29  %type <Ast.program> program
30
31  %%
32
```

```
33  program:
34    decls EOF { $1 }
35
36  decls:
37      /* nothing */ { [] }
38    | decls fdecl   { $2 :: $1 }
39
40  fdecl:
41    typ ID LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
42       { { typ = $1;
43        fname = $2;
44        formals = $4;
45        body = List.rev $7 } }
46
47  formals_opt:
48      /* nothing */ { [] }
49    | formal_list   { List.rev $1 }
50
51  formal_list:
52      typ ID                   { [($1,$2)] }
53    | formal_list COMMA typ ID { ($3,$4) :: $1 }
54
55  primitive:
56      NUM          { Num }
57    | INT          { Int }
58    | STRING       { String }
59    | BOOL         { Bool }
60    | VOID         { Void }
61
62  ds_type:
63      primitive LBRACK INT_LITERAL RBRACK { Arraytype($1, $3) }
64    | QUEUE DOUBLECOL primitive           { QueueType($3) }
65    | LINKEDLIST DOUBLECOL primitive      { LinkedListType($3) }
66    | STACK DOUBLECOL primitive           { StackType($3) }
67    | BSTREE DOUBLECOL primitive          { BSTreeType($3) }
68
69  typ:
70      primitive   { $1 }
71    | ds_type     { $1 }
72
73  stmt_list:
74      /* nothing */  { [] }
75    | stmt_list stmt { $2 :: $1 }
76
77  stmt:
78      expr SEMI { Expr $1 }
```

```
79    | RETURN SEMI { Return Noexpr }
80    | RETURN expr SEMI { Return $2 }
81    | LBRACE stmt_list RBRACE { Block(List.rev $2) }
82    | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5,
      ↪  Block([])) }
83    | IF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7) }
84    | IF LPAREN expr RPAREN stmt else_stmt    { If($3, $5, $6) }
85    | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt
86      { For($3, $5, $7, $9) }
87    | WHILE LPAREN expr RPAREN stmt { While($3, $5) }
88
89  else_stmt:
90      ELIF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5,
        ↪  Block([])) }
91    | ELIF LPAREN expr RPAREN stmt else_stmt    { If($3, $5, $6) }
92    | ELIF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7) }
93
94  expr_opt:
95      /* nothing */ { Noexpr }
96    | expr          { $1 }
97
98  expr:
99      literal          { $1 }
100   | expr PLUS   expr { Binop($1, Add,   $3) }
101   | expr MINUS  expr { Binop($1, Sub,   $3) }
102   | expr TIMES  expr { Binop($1, Mult,  $3) }
103   | expr DIVIDE expr { Binop($1, Div,   $3) }
104   | expr MOD    expr { Binop($1, Mod,   $3) }
105   | expr EQ     expr { Binop($1, Equal, $3) }
106   | expr NEQ    expr { Binop($1, Neq,   $3) }
107   | expr LT     expr { Binop($1, Less,  $3) }
108   | expr LEQ    expr { Binop($1, Leq,   $3) }
109   | expr GT     expr { Binop($1, Greater, $3) }
110   | expr GEQ    expr { Binop($1, Geq,   $3) }
111   | expr AND    expr { Binop($1, And,   $3) }
112   | expr OR     expr { Binop($1, Or,    $3) }
113   | MINUS expr %prec NEG  { Unop(Neg, $2) }
114   | NOT expr              { Unop(Not, $2) }
115   | expr INCR             { Postop($1, Incr) }
116   | expr DECR             { Postop($1, Decr) }
117   | typ ID                { Assign($1, $2, Noexpr) }
118   | typ ID ASSIGN expr    { Assign($1, $2, $4) }
119   | expr DOT ID LPAREN actuals_opt RPAREN    { ObjectCall($1,
      ↪  $3, $5) }
120   | ID ASSIGN expr                          { Reassign($1, $3)
      ↪  }
```

```
121   | ID LPAREN actuals_opt RPAREN             { FuncCall($1, $3)
      ↪ }
122   | LBRACK actuals_opt RBRACK                { ArrayLit($2) }
123   | ID LBRACK expr RBRACK                    { ArrayAccess($1,
      ↪ $3) }
124   | ID LBRACK expr RBRACK ASSIGN expr        {
      ↪ ArrayElementAssign($1, $3, $6) }
125   | LPAREN expr RPAREN                       { $2 }
126   | NEW QUEUE DOUBLECOL typ LPAREN actuals_opt RPAREN       {
      ↪ QueueLit($4, $6) }
127   | NEW LINKEDLIST DOUBLECOL typ LPAREN actuals_opt RPAREN  {
      ↪ LinkedListLit($4, $6) }
128   | NEW STACK DOUBLECOL typ LPAREN actuals_opt RPAREN       {
      ↪ StackLit($4, $6) }
129   | NEW BSTREE DOUBLECOL typ LPAREN actuals_opt RPAREN      {
      ↪ BSTreeLit($4, $6) }
130
131 literal:
132     STRING_LITERAL   { StringLit($1) }
133   | ID               { Id($1) }
134   | INT_LITERAL      { IntLit($1) }
135   | NUM_LITERAL      { NumLit($1) }
136   | TRUE             { BoolLit(true) }
137   | FALSE            { BoolLit(false) }
138   | NULL             { Null }
139
140 actuals_opt:
141     /* nothing */ { [] }
142   | actuals_list  { List.rev $1 }
143
144 actuals_list:
145     expr                    { [$1] }
146   | actuals_list COMMA expr { $3 :: $1 }
```

8.3 ast.ml

Josh, Kofi, Millie, Sophie

```
1 (* Abstract Syntax Tree and functions for printing it *)
2
3 type op = Add | Sub | Mult | Div | Mod | Equal | Neq | Less |
  ↪ Leq | Greater | Geq |
4           And | Or | Incr | Decr
5
6 type uop = Neg | Not
```

```
7
8  type typ = Num | Int | String | Bool | Void | AnyType |
   ↪  NumberType |
9             Arraytype of typ * int | QueueType of typ |
               ↪  LinkedListType of typ |
10            StackType of typ | BSTreeType of typ
11
12 type bind = typ * string
13
14 type expr =
15     NumLit of float
16   | IntLit of int
17   | StringLit of string
18   | BoolLit of bool
19   | Null
20   | Id of string
21   | Binop of expr * op * expr
22   | Unop of uop * expr
23   | Postop of expr * op
24   | Assign of typ * string * expr
25   | Reassign of string * expr
26   | FuncCall of string * expr list
27   | QueueLit of typ * expr list
28   | BSTreeLit of typ * expr list
29   | LinkedListLit of typ * expr list
30   | StackLit of typ * expr list
31   | ObjectCall of expr * string * expr list
32   | Noexpr
33   | ArrayLit of expr list
34   | ArrayAccess of string * expr
35   | ArrayElementAssign of string * expr * expr
36
37 type stmt =
38     Block of stmt list
39   | Expr of expr
40   | Return of expr
41   | If of expr * stmt * stmt
42   | For of expr * expr * expr * stmt
43   | While of expr * stmt
44
45 type func_decl = {
46     typ : typ;
47     fname : string;
48     formals : bind list;
49     body : stmt list;
50   }
```

```
51
52  type program = func_decl list
53
54  (* Pretty-printing functions *)
55
56  let string_of_op = function
57      Add -> "+"
58    | Sub -> "-"
59    | Mult -> "*"
60    | Div -> "/"
61    | Mod -> "%"
62    | Equal -> "=="
63    | Neq -> "!="
64    | Less -> "<"
65    | Leq -> "<="
66    | Greater -> ">"
67    | Geq -> ">="
68    | And -> "and"
69    | Or -> "or"
70    | Incr -> "++"
71    | Decr -> "--"
72
73  let string_of_uop = function
74      Neg -> "-"
75    | Not -> "not"
76
77  let rec string_of_typ = function
78      Num -> "num"
79    | Int -> "int"
80    | String -> "string"
81    | Bool -> "bool"
82    | Void -> "void"
83    | Arraytype(typ, len) -> string_of_typ typ ^ "[" ^
      ↪  string_of_int len ^ "]"
84    | QueueType(typ) -> "Queue " ^ string_of_typ typ
85    | BSTreeType(typ) -> "BSTree " ^ string_of_typ typ
86    | LinkedListType(typ) -> "LinkedList " ^ string_of_typ typ
87    | StackType(typ) -> "Stack " ^ string_of_typ typ
88    | AnyType -> "AnyType"
89    | NumberType -> "NumberType"
90
91  let rec string_of_expr = function
92      StringLit(s) -> s
93    | NumLit(f) -> string_of_float f
94    | IntLit(i) -> string_of_int i
95    | BoolLit(true) -> "true"
```

```
96    | BoolLit(false) -> "false"
97    | Null -> "null"
98    | Id(s) -> s
99    | Binop(e1, o, e2) ->
100       string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^
          ↪ string_of_expr e2
101   | Unop(o, e) -> string_of_uop o ^ string_of_expr e
102   | Postop(e, o) -> string_of_expr e ^ string_of_op o
103   | Assign(t, v, e) -> string_of_typ t ^ " " ^ v ^ " = " ^
      ↪ string_of_expr e
104   | Reassign(v, e) -> v ^ "=" ^ string_of_expr e
105   | FuncCall(f, el) ->
106       f ^ "(" ^ String.concat ", " (List.map string_of_expr el)
          ↪ ^ ")"
107   | ObjectCall(o, f, e1) -> string_of_expr o ^ "." ^ f ^ "(" ^
      ↪ String.concat ", " (List.map string_of_expr e1) ^ ")"
108   | ArrayLit a -> "[" ^ String.concat " " (List.map
      ↪ string_of_expr a) ^ "]"
109   | ArrayAccess(v, i) -> v ^ "[" ^ string_of_expr i ^ "]"
110   | ArrayElementAssign(s, i, e) -> s ^ "[" ^ string_of_expr i ^
      ↪ "]" ^ " = " ^ string_of_expr e
111   | Noexpr -> ""
112   | QueueLit(typ, e1) -> "new " ^ "Queue" ^ "::" ^ string_of_typ
      ↪ typ ^ "(" ^ String.concat ", " (List.map string_of_expr
      ↪ e1) ^ ")"
113   | LinkedListLit(typ, e1) -> "new " ^ "LinkedList" ^ "::" ^
      ↪ string_of_typ typ ^ "(" ^ String.concat ", " (List.map
      ↪ string_of_expr e1) ^ ")"
114   | BSTreeLit(typ, e1) -> "new " ^ "BSTree" ^ "::" ^
      ↪ string_of_typ typ ^ "(" ^ String.concat ", " (List.map
      ↪ string_of_expr e1) ^ ")"
115   | StackLit(typ, e1) -> "new " ^ "Stack" ^ "::" ^ string_of_typ
      ↪ typ ^ "(" ^ String.concat ", " (List.map string_of_expr
      ↪ e1) ^ ")"
116
117 let rec string_of_stmt = function
118     Block(stmts) ->
119       "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^
          ↪ "}\n"
120   | Expr(expr) -> string_of_expr expr ^ ";\n";
121   | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
122   | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^
      ↪ string_of_stmt s
123   | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
124       string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
125   | For(e1, e2, e3, s) ->
```

```
126       "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2 ^
          ↪  " ; " ^
127       string_of_expr e3  ^ ") " ^ string_of_stmt s
128   | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^
      ↪  string_of_stmt s
129
130 let string_of_vdecl (t, id) = string_of_typ t ^ " " ^ id ^ ";\n"
131
132 let string_of_fdecl fdecl =
133   string_of_typ fdecl.typ ^ " " ^
134   fdecl.fname ^ "(" ^ String.concat ", " (List.map snd
      ↪  fdecl.formals) ^
135   ")\n{\n" ^
136   String.concat "" (List.map string_of_stmt fdecl.body) ^
137   "}\n"
138
139 let string_of_program (funcs) =
140   String.concat "\n" (List.map string_of_fdecl funcs)
```

8.4 semant.ml

Josh, Kofi, Millie, Sophie

```
1  (* Semantic checking for the MicroC compiler *)
2
3  open Ast
4
5  module StringMap = Map.Make(String)
6
7  (* Semantic checking of a program. Returns void if successful,
8     throws an exception if something is wrong. Checks each
   ↪  function *)
9
10 let check (functions) =
11
12   (* Raise an exception if the given list has a duplicate *)
13   let report_duplicate exceptf list =
14     let rec helper = function
15         n1 :: n2 :: _ when n1 = n2 -> raise (Failure (exceptf
           ↪  n1))
16       | _ :: t -> helper t
17       | [] -> ()
18     in helper (List.sort compare list)
19   in
20
```

```
21  (* Raise an exception if a given binding is to a void type *)
22  let check_not_void exceptf = function
23      (Void, n) -> raise (Failure (exceptf n))
24    | _ -> ()
25  in
26
27  (* Raise an exception of the given rvalue type cannot be
    ↪  assigned to
28     the given lvalue type *)
29  let check_assign lvaluet rvaluet err =
30    if lvaluet = rvaluet then rvaluet
31    else if lvaluet = Num && rvaluet = AnyType then lvaluet
32    else if lvaluet = Int && rvaluet = AnyType then lvaluet
33    else if lvaluet = String && rvaluet = AnyType then lvaluet
34    else if lvaluet = Bool && rvaluet = AnyType then lvaluet
35    else if lvaluet = Num && rvaluet = NumberType then lvaluet
36    else if lvaluet = Int && rvaluet = NumberType then lvaluet
37    else raise err
38  in
39
40  (**** Checking Functions ****)
41  if List.mem "print" (List.map (fun fd -> fd.fname) functions)
42  then raise (Failure ("function print may not be defined"))
    ↪  else ();
43
44  if List.mem "delete" (List.map (fun fd -> fd.fname) functions)
45  then raise (Failure ("function delete may not be defined"))
    ↪  else ();
46
47  report_duplicate (fun n -> "duplicate function " ^ n)
48    (List.map (fun fd -> fd.fname) functions);
49
50  (* Function declaration for a named function *)
51  let built_in_decls =  StringMap.add "print"
52     { typ = Void; fname = "print"; formals = [(Num, "x")];
53       body = [] }
54
55       (StringMap.add "printb"
56     { typ = Void; fname = "printb"; formals = [(Bool, "x")];
57       body = [] }
58
59        (StringMap.add "add"
60    { typ = Void; fname = "add"; formals = [(AnyType, "x")];
61        body = [] }
62
63        (StringMap.add "show"
```

```
64      { typ = Void; fname = "show"; formals = [];
65          body = [] }
66
67          (StringMap.add "peek"
68      { typ = AnyType; fname = "peek"; formals = [];
69          body = [] }
70
71          (StringMap.add "remove"
72      { typ = Void; fname = "remove"; formals = [];
73          body = [] }
74
75          (StringMap.add "get"
76      { typ = AnyType; fname = "get"; formals = [(Int, "x")];
77          body = [] }
78
79          (StringMap.add "size"
80       { typ = Int; fname = "size"; formals = [];
81          body = [] }
82
83          (StringMap.add "delete"
84       { typ = Void; fname = "delete"; formals = [(NumberType,
         ↪  "x")];
85          body = [] }
86
87          (StringMap.add "contains"
88       { typ = Bool; fname = "contains"; formals = [(NumberType,
         ↪  "x")];
89          body = [] }
90
91          (StringMap.add "fquickSort"
92       { typ = Void; fname = "fquickSort"; formals = [];
93         body = [] }
94
95          (StringMap.add "fshowQuickSort"
96       { typ = Void; fname = "fshowQuickSort"; formals = [];
97         body = [] }
98
99          (StringMap.add "quickSort"
100      { typ = Void; fname = "quickSort"; formals = [];
101        body = [] }
102
103         (StringMap.singleton "showQuickSort"
104       { typ = Void; fname = "showQuickSort"; formals = [];
105         body = [] }
106
107     ))))))))))))
```

```
108
109     in
110
111     let function_decls = List.fold_left (fun m fd -> StringMap.add
        ↪  fd.fname fd m)
112                          built_in_decls functions
113     in
114
115     let function_decl s = try StringMap.find s function_decls
116          with Not_found -> raise (Failure ("unrecognized function
             ↪  " ^ s))
117     in
118
119     let _ = function_decl "main" in (* Ensure "main" is defined *)
120
121     let check_function func =
122
123       List.iter (check_not_void (fun n -> "illegal void formal " ^
          ↪  n ^
124         " in " ^ func.fname)) func.formals;
125
126       report_duplicate (fun n -> "duplicate formal " ^ n ^ " in "
          ↪  ^ func.fname)
127         (List.map snd func.formals);
128
129       (* Store type and names of variables - formal *)
130       let variables = ref (List.fold_left (fun m (t, n) ->
          ↪  StringMap.add n t m)
131       StringMap.empty (func.formals))
132       in
133
134       (* Helper: Returns type of identifier *)
135       let type_of_identifier name =
136         try StringMap.find name (!variables)
137         with Not_found -> raise (Failure ("undeclared identifier "
            ↪  ^ name))
138       in
139
140       (* Helper: Check if variable is already declared *)
141       let check_var_decl var_name err =
142         if StringMap.mem var_name (!variables)
143         then raise err
144       in
145
146       let array_typ = function
147           Arraytype(typ, _) -> typ
```

```
148       | _ -> raise(Failure("Expecting an array and was not an
          ↪  array"))
149     in
150
151     let arr_lit_len = function
152         ArrayLit(el) -> List.length el
153       | _ -> -1
154     in
155
156     let invalid_arr_size s e =
157       let literal_len = arr_lit_len e in
158       if literal_len == -1
159       then false
160       else
161         let decl_len = (match s with
162                           Arraytype(_, len) -> len) in
163         if decl_len == literal_len then false else true
164     in
165
166     let get_type = function
167         QueueType(typ) -> typ
168       | LinkedListType(typ) -> typ
169       | StackType(typ) -> typ
170       | BSTreeType(typ) -> typ
171       | _ -> Void
172     in
173
174     (* Return the type of an expression or throw an exception *)
175     let rec expr = function
176         NumLit _ -> Num
177       | IntLit _ -> Int
178       | StringLit _ -> String
179       | QueueLit (t, _) -> QueueType(t)
180       | LinkedListLit (t, _) -> LinkedListType(t)
181       | StackLit (t, _) -> StackType(t)
182       | BSTreeLit (t, _) -> BSTreeType(t)
183       | BoolLit _ -> Bool
184       | Id s -> type_of_identifier s
185       | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 = expr
          ↪  e2 in
186         (match op with
187           Add | Sub | Mult | Div | Mod when t1 = Int && t2 = Int
              ↪  -> Int
188         | Add | Sub | Mult | Div | Mod when t1 = Num && t2 = Num
              ↪  -> Num
189         | Equal | Neq when t1 = t2 -> Bool
```

```
190           | Less | Leq | Greater | Geq when t1 = Int && t2 = Int
              ↪  -> Bool
191           | Less | Leq | Greater | Geq when t1 = Num && t2 = Num
              ↪  -> Bool
192           | And | Or when t1 = Bool && t2 = Bool -> Bool
193           | _ -> raise (Failure ("illegal binary operator " ^
194                 string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
195                 string_of_typ t2 ^ " in " ^ string_of_expr e))
196         )
197       | Postop (e, op) ->
198           let t1 = expr e in
199           (match op with
200             Incr when t1 = Int -> Int
201           | Decr when t1 = Int -> Int
202           | Incr when t1 = Num -> Num
203           | Decr when t1 = Num -> Num
204           | _ -> raise (Failure("illegal unary operator " ^
205                 string_of_op op ^ " on " ^ string_of_expr e)))
206       | Unop(op, e) as ex -> let t = expr e in
207         (match op with
208           Neg when t = Num -> Num
209         | Neg when t = Int -> Int
210         | Not when t = Bool -> Bool
211         | _ -> raise (Failure ("illegal unary operator " ^
              ↪  string_of_uop op ^
212               string_of_typ t ^ " in " ^ string_of_expr ex)))
213       | Noexpr -> Void
214       | Assign(typ, var, e) as ex ->
215           let lt = (match typ with
216             Arraytype(t, _) -> t
217           | _ -> typ
218           ) in
219           let rt = expr e
220           and invalid_arr = invalid_arr_size typ e in
221           if invalid_arr then raise (Failure ("Invalid length
              ↪  declaration"))
222           else
223           if rt == Void then raise (Failure("Must initialize
              ↪  variable with a value."))
224         else
225           ignore (check_assign lt rt (Failure ("illegal
              ↪  assignment " ^ string_of_typ typ ^ " = " ^
              ↪  string_of_typ rt ^ " in " ^ string_of_expr ex)));
226           check_var_decl var (Failure ("duplicate declaration of
              ↪  variable " ^ var));
227           let _ =
```

```
228             (match func.fname with
229               | _ -> variables := StringMap.add var typ
                  ↪  (!variables)
230             )
231           in rt
232
233       | Reassign(var, e) as ex ->
234           let rt = expr e and lt = type_of_identifier var in
235           check_assign lt rt (Failure ("illegal assignment " ^
              ↪  string_of_typ lt ^
236                                        " = " ^ string_of_typ rt ^ " in "
                                          ↪  ^ string_of_expr ex))
237       | FuncCall(fname, actuals) as call ->
238         if fname = "print"
239         then (if List.length actuals == 1
240               then let arg_type = string_of_typ (expr (List.hd
                      ↪  actuals)) in
241                    if arg_type = string_of_typ (Num) ||
242                       arg_type = string_of_typ (Int) ||
243                       arg_type = string_of_typ (String) ||
244                       arg_type = string_of_typ (Bool) ||
245                       arg_type = string_of_typ (AnyType)
246                    then Void
247                    else raise (Failure ("illegal actual
                       ↪  argument found in print " ^
248                                         string_of_typ (expr (List.hd
                                           ↪  actuals)) ^
249                                         " in " ^ string_of_expr (List.hd
                                           ↪  actuals)))
250               else raise (Failure ("expecting 1 argument in " ^
                    ↪  string_of_expr call)))
251         else let fd = function_decl fname in
252            if List.length actuals != List.length fd.formals
253            then raise (Failure ("expecting " ^ string_of_int
254                (List.length fd.formals) ^ " arguments in " ^
                    ↪  string_of_expr call))
255            else
256              List.iter2 (fun (ft, _) e -> let et = expr e in
257                 ignore (check_assign ft et
258                   (Failure ("illegal actual argument found " ^
                      ↪  string_of_typ et ^
259                   " expected " ^ string_of_typ ft ^ " in " ^
                      ↪  string_of_expr e))))
260                fd.formals actuals;
261              fd.typ
262       | ArrayLit(el) -> expr (List.nth el 0)
```

```
263         | ArrayAccess(var, el) as element->
264             if expr el != Int
265             then raise (Failure ("Invalid element access in " ^
                ↪  string_of_expr element))
266             else array_typ (type_of_identifier var)
267         | ArrayElementAssign (s, i, e) as ex ->
268             let lt =
269               if expr i != Int
270               then raise (Failure ("invalid element access in " ^
                  ↪  string_of_expr ex))
271               else array_typ (type_of_identifier s)
272             in
273             let rt = expr e in
274             check_assign lt rt (Failure ("illegal assignment " ^
                ↪  string_of_typ lt ^ " = " ^ string_of_typ rt ^ " in
                ↪  " ^ string_of_expr ex));
275         | ObjectCall(oname, fname, actuals) as objectcall ->
276             let fd = function_decl fname in
277             let returntype = ref (fd.typ) in
278             if List.length actuals != List.length fd.formals then
279               raise (Failure ("expecting " ^ string_of_int
                  ↪  (List.length fd.formals) ^
280                     " arguments in " ^ string_of_expr objectcall))
281             else
282                List.iter2 (fun (ft, _) e -> let et = expr e in
283                   if fname = "add" then
284                      let acttype = expr oname in
285                      let actqtype = get_type acttype in
286                     ignore(check_assign actqtype et (Failure
                        ↪  ("illegal actual add argument found " ^
                        ↪  string_of_typ et ^
287                       " expected " ^ string_of_typ actqtype ^ " in "
                        ↪  ^ string_of_expr e)))
288                   else if fname = "remove" then
289                      let acttype = expr oname in
290                      let actqtype = get_type acttype in
291                     ignore(check_assign actqtype et (Failure
                        ↪  ("illegal actual remove argument found " ^
                        ↪  string_of_typ et ^
292                       " expected " ^ string_of_typ actqtype ^ " in "
                        ↪  ^ string_of_expr e)))
293                   else if fname = "quickSort" then
294                     let acttype = expr oname in
295                     let actatype = array_typ acttype in
```

```
296                   ignore(check_assign actatype et (Failure
                      ↪  ("illegal actual size argument found " ^
                      ↪  string_of_typ et ^
297                   " expected " ^ string_of_typ actatype ^ " in "
                      ↪  ^ string_of_expr e)))
298                 else if fname = "delete" then
299                    let acttype = expr oname in
300                    match acttype with
301                    | BSTreeType(actqtype) -> ignore(check_assign
                       ↪  actqtype et (Failure ("illegal actual
                       ↪  delete argument found " ^ string_of_typ
                       ↪  et ^
302                   " expected " ^ string_of_typ actqtype ^ " type
                      ↪  " ^ " in " ^ string_of_expr e)))
303                    | LinkedListType(_) -> ignore(check_assign
                       ↪  Int et (Failure ("illegal actual delete
                       ↪  argument found " ^ string_of_typ et ^
304                   " expected int type " ^ " in " ^
                      ↪  string_of_expr e)))
305
306                 else if fname = "contains" then
307                    let acttype = expr oname in
308                    match acttype with
309                    | BSTreeType(actqtype) -> ignore(check_assign
                       ↪  actqtype et (Failure ("illegal actual
                       ↪  contains argument found " ^ string_of_typ
                       ↪  et ^
310                   " expected " ^ string_of_typ actqtype ^ " type
                      ↪  " ^ " in " ^ string_of_expr e)))
311                    | _ -> raise (Failure (".contains() is only
                       ↪  applicable to the the tree data type"))
312
313                 else ignore (check_assign ft et (Failure
                    ↪  ("illegal actual argument found " ^
                    ↪  string_of_typ et ^
314                       " expected " ^ string_of_typ ft ^ " in " ^
                          ↪  string_of_expr e)))) fd.formals
                          ↪  actuals;
315                  !returntype
316
317     in
318
319     let check_bool_expr e = if expr e != Bool
320      then raise (Failure ("expected Boolean expression in " ^
         ↪  string_of_expr e))
321      else () in
```

```
322
323     (* Verify a statement or throw an exception *)
324     let rec stmt = function
325         Block sl -> let rec check_block = function
326            [Return _ as s] -> stmt s
327          | Return _ :: _ -> raise (Failure "nothing may follow a
             ↪  return")
328          | Block sl :: ss -> check_block (sl @ ss)
329          | s :: ss -> stmt s ; check_block ss
330          | [] -> ()
331         in check_block sl
332       | Expr e -> ignore (expr e)
333       | Return e -> let t = expr e in if t = func.typ then ()
          ↪  else
334           raise (Failure ("return gives " ^ string_of_typ t ^ "
             ↪  expected " ^
335                           string_of_typ func.typ ^ " in " ^
                              ↪  string_of_expr e))
336
337       | If(p, b1, b2) -> check_bool_expr p; stmt b1; stmt b2
338       | For(e1, e2, e3, st) -> ignore (expr e1); check_bool_expr
          ↪  e2;
339                                ignore (expr e3); stmt st
340       | While(p, s) -> check_bool_expr p; stmt s
341     in
342
343     stmt (Block func.body)
344
345   in
346   List.iter check_function functions
```

8.5 codegen.ml

Josh, Kofi, Millie, Sophie

```
1   (* Code generation: translate takes a semantically checked AST
    ↪  and
2   produces LLVM IR
3   *)
4   module L = Llvm
5   module A = Ast
6
7   module StringMap = Map.Make(String)
8
9   let context = L.global_context ()
```

```
10
11  let queuem = L.MemoryBuffer.of_file "queue.bc"
12  let listm = L.MemoryBuffer.of_file "linkedlist.bc"
13  let stackm = L.MemoryBuffer.of_file "stack.bc"
14  let quicksortm = L.MemoryBuffer.of_file "quicksort.bc"
15  let bstreem = L.MemoryBuffer.of_file "bstree.bc"
16  let qqm = Llvm_bitreader.parse_bitcode context queuem
17  let list_qm = Llvm_bitreader.parse_bitcode context listm
18  let stack_qm = Llvm_bitreader.parse_bitcode context stackm
19  let bstree_qm = Llvm_bitreader.parse_bitcode context bstreem
20  let quicksort_qm = Llvm_bitreader.parse_bitcode context
    ↪  quicksortm
21
22  let the_module = L.create_module context "Strux"
23  and f_t    = L.double_type context  (* float *)
24  and i8_t   = L.i8_type   context    (* print type *)
25  and i1_t   = L.i1_type   context    (* bool type *)
26  and void_t = L.void_type context    (* void type *)
27  and str_t  = L.pointer_type (L.i8_type context) (* string *)
28  and i32_t  = L.i32_type  context
29  and queue_t = L.pointer_type (match L.type_by_name qqm
    ↪  "struct.Queue" with
30      None -> raise (Invalid_argument "Option.get queue") | Some x
        ↪  -> x)
31  and stack_t = L.pointer_type (match L.type_by_name stack_qm
    ↪  "struct.Stack" with
32      None -> raise (Invalid_argument "Option.get stack") | Some x
        ↪  -> x)
33  and linkedlist_t = L.pointer_type (match L.type_by_name list_qm
    ↪  "struct.LinkedList" with
34      None -> raise (Invalid_argument "Option.get linkedlmist") |
        ↪  Some x -> x)
35  and bstree_t = L.pointer_type (match L.type_by_name bstree_qm
    ↪  "struct.BSTree" with
36      None -> raise (Invalid_argument "Option.get bstree") | Some
        ↪  x -> x);;
37
38  let rec ltype_of_typ = function (* LLVM type for AST type *)
39      A.Num -> f_t
40    | A.Int -> i32_t
41    | A.String -> str_t
42    | A.Bool -> i1_t
43    | A.Void -> void_t
44    | A.Arraytype(t, _) -> L.pointer_type (ltype_of_typ t)
45    | A.QueueType _ -> queue_t
46    | A.LinkedListType _ -> linkedlist_t
```

```
47    | A.BSTreeType _ -> bstree_t
48    | A.StackType _ -> stack_t
49    | A.AnyType -> str_t
50    | A.NumberType -> str_t
51    | _ -> raise(Failure("Invalid Data Type"))
52
53  and translate (functions) =
54    (* -------BUILT IN FUNCTIONS----------- *)
55
56    (* print *)
57    let printf_t = L.var_arg_function_type i32_t [| L.pointer_type
      ↪  i8_t |] in
58    let printf_func = L.declare_function "printf" printf_t
      ↪  the_module in
59
60    (* built-in queue functions *)
61    let initQueue_t = L.function_type queue_t [| |] in
62    let initQueue_f = L.declare_function "initQueue" initQueue_t
      ↪  the_module in
63    let enqueue_t = L.function_type void_t [| queue_t;
      ↪  L.pointer_type i8_t|] in
64    let enqueue_f = L.declare_function "enqueue" enqueue_t
      ↪  the_module in
65    let dequeue_t = L.function_type void_t [| queue_t |] in
66    let dequeue_f = L.declare_function "dequeue" dequeue_t
      ↪  the_module in
67    let peek_t = L.function_type (L.pointer_type i8_t) [| queue_t
      ↪  |] in
68    let peek_f = L.declare_function "peek" peek_t the_module in
69    let sizeQ_t = L.function_type i32_t [| queue_t |] in
70    let sizeQ_f = L.declare_function "queue_size" sizeQ_t
      ↪  the_module in
71    let q_show_t = L.function_type void_t [| queue_t |] in
72    let q_show_int = L.declare_function "queue_show_int" q_show_t
      ↪  the_module in
73    let q_show_float = L.declare_function "queue_show_float"
      ↪  q_show_t the_module in
74    let q_show_string = L.declare_function "queue_show_string"
      ↪  q_show_t the_module in
75
76    (*built-in linkedlist functions*)
77    let initList_t = L.function_type linkedlist_t [| |] in
78    let initList_f = L.declare_function "initList" initList_t
      ↪  the_module in
79    let add_t = L.function_type void_t [| linkedlist_t;
      ↪  L.pointer_type i8_t|] in
```

```
80  let add_f = L.declare_function "add" add_t the_module in
81  let delete_t = L.function_type void_t [| linkedlist_t; i32_t
    ↪  |] in
82  let delete_f = L.declare_function "delete" delete_t the_module
    ↪  in
83  let get_t = L.function_type (L.pointer_type i8_t) [|
    ↪  linkedlist_t; i32_t |] in
84  let get_f = L.declare_function "get" get_t the_module in
85  let sizeList_t = L.function_type i32_t [| linkedlist_t |] in
86  let sizeList_f = L.declare_function "size" sizeList_t
    ↪  the_module in
87  let l_show_t = L.function_type void_t [| linkedlist_t |] in
88  let l_show_int = L.declare_function "ll_show_int" l_show_t
    ↪  the_module in
89  let l_show_float = L.declare_function "ll_show_float" l_show_t
    ↪  the_module in
90  let l_show_string = L.declare_function "ll_show_string"
    ↪  l_show_t the_module in
91
92  (*built-in stack functions*)
93  let initStack_t = L.function_type stack_t [| |] in
94  let initStack_f = L.declare_function "initStack" initStack_t
    ↪  the_module in
95  let push_t = L.function_type void_t [| stack_t; L.pointer_type
    ↪  i8_t|] in
96  let push_f = L.declare_function "push" push_t the_module in
97  let pop_t = L.function_type void_t [| stack_t |] in
98  let pop_f = L.declare_function "pop" pop_t the_module in
99  let top_t = L.function_type (L.pointer_type i8_t) [| stack_t
    ↪  |] in
100 let top_f = L.declare_function "top" top_t the_module in
101 let sizeS_t = L.function_type i32_t [| stack_t |] in
102 let sizeS_f = L.declare_function "stack_size" sizeS_t
    ↪  the_module in
103 let s_show_t = L.function_type void_t [| stack_t |] in
104 let s_show_int = L.declare_function "stack_show_int" s_show_t
    ↪  the_module in
105 let s_show_float = L.declare_function "stack_show_float"
    ↪  s_show_t the_module in
106 let s_show_string = L.declare_function "stack_show_string"
    ↪  s_show_t the_module in
107
108
109 (* quicksort array functions *)
110 (*takes in int[] to do quick sort *)
```

```
111  let int_quickSort_t = L.function_type (L.pointer_type
     ↪  (ltype_of_typ A.Int)) [| L.pointer_type (ltype_of_typ
     ↪  A.Int); i32_t |] in
112  let int_quickSort_f = L.declare_function "cQuickSort"
     ↪  int_quickSort_t the_module in
113  let int_show_quickSort_t = L.function_type void_t [|
     ↪  L.pointer_type (ltype_of_typ A.Int); i32_t |] in
114  let int_show_quickSort_f = L.declare_function "cShowQuickSort"
     ↪  int_show_quickSort_t the_module in
115
116  (*takes in num[] to do quick sort *)
117  let num_quickSort_t = L.function_type (L.pointer_type
     ↪  (ltype_of_typ A.Num)) [| L.pointer_type (ltype_of_typ
     ↪  A.Num); i32_t |] in
118  let num_quickSort_f = L.declare_function "cQuickfSort"
     ↪  num_quickSort_t the_module in
119  let num_show_quickSort_t = L.function_type void_t [|
     ↪  L.pointer_type (ltype_of_typ A.Num); i32_t |] in
120  let num_show_quickSort_f = L.declare_function
     ↪  "cShowfQuickSort" num_show_quickSort_t the_module in
121
122  (*takes in string[] to do quick sort *)
123  let string_quickSort_t = L.function_type (L.pointer_type
     ↪  (ltype_of_typ A.String)) [| L.pointer_type (ltype_of_typ
     ↪  A.String); i32_t |] in
124  let string_quickSort_f = L.declare_function "cQuicksSort"
     ↪  string_quickSort_t the_module in
125  let string_show_quickSort_t = L.function_type void_t [|
     ↪  L.pointer_type (ltype_of_typ A.String); i32_t |] in
126  let string_show_quickSort_f = L.declare_function
     ↪  "cShowsQuickSort" string_show_quickSort_t the_module in
127
128  (*built-in bstree functions*)
129  let initBSTree_t = L.function_type bstree_t [| |] in
130  let initBSTree_f = L.declare_function "initBSTree"
     ↪  initBSTree_t the_module in
131  let bstreeadd_t = L.function_type void_t [| bstree_t;
     ↪  L.pointer_type i8_t|] in
132  let bstreeadd_int_f = L.declare_function "addIntToTree"
     ↪  bstreeadd_t the_module in
133  let bstreeadd_float_f = L.declare_function "addNumToTree"
     ↪  bstreeadd_t the_module in
134  let bstreedelete_int_t = L.function_type void_t [| bstree_t;
     ↪  i32_t|] in
135  let bstreedelete_int_f = L.declare_function
     ↪  "deleteIntFromTree" bstreedelete_int_t the_module in
```

```
136   let bstreedelete_float_t = L.function_type void_t [| bstree_t;
      ↪  f_t|] in
137   let bstreedelete_float_f = L.declare_function
      ↪  "deleteNumFromTree" bstreedelete_float_t the_module in
138   let bstreecontains_int_t = L.function_type i1_t [| bstree_t;
      ↪  i32_t|] in
139   let bstreecontains_int_f = L.declare_function
      ↪  "treeContainsInt" bstreecontains_int_t the_module in
140   let bstreecontains_float_t = L.function_type i1_t [| bstree_t;
      ↪  f_t|] in
141   let bstreecontains_float_f = L.declare_function
      ↪  "treeContainsFloat" bstreecontains_float_t the_module in
142   let bstree_show_t = L.function_type void_t [| bstree_t |] in
143   let bstree_show_int = L.declare_function "showIntTree"
      ↪  bstree_show_t the_module in
144   let bstree_show_float = L.declare_function "showNumTree"
      ↪  bstree_show_t the_module in
145
146   (*print big *)
147   let printbig_t = L.function_type i32_t [| i32_t |] in
148   let printbig_func = L.declare_function "printbig" printbig_t
      ↪  the_module in
149
150   (* Define each function (arguments and return type) so we can
      ↪  call it *)
151   let function_decls =
152     let function_decl m func_decl =
153       let name = func_decl.A.fname
154       and formal_types =
155         Array.of_list (List.map (fun (t,_) -> ltype_of_typ t)
          ↪  func_decl.A.formals)
156       in let ftype = L.function_type (ltype_of_typ
          ↪  func_decl.A.typ) formal_types in
157       StringMap.add name (L.define_function name ftype
          ↪  the_module, func_decl) m in
158     List.fold_left function_decl StringMap.empty functions in
159
160   (* Format str for printf *)
161   let string_format_str b = L.build_global_stringptr "%s\n"
      ↪  "fmt" b
162   and int_format_str    b = L.build_global_stringptr "%d\n"
      ↪  "fmt" b
163   and float_format_str  b = L.build_global_stringptr "%f\n"
      ↪  "fmt" b in
164
165   let format_str x_type builder =
```

```
166     let b = builder in
167       match x_type with
168         A.Int      -> int_format_str b
169       | A.Num    -> float_format_str b
170       | A.String  -> string_format_str b
171       | A.Bool     -> int_format_str b
172       | _ -> raise (Failure ("Invalid printf type"))
173   in
174
175   let int_format_zeroth     b = L.build_global_stringptr "[%d"
      ↪  "fmt" b
176   and num_format_zeroth     b = L.build_global_stringptr "[%f"
      ↪  "fmt" b
177   and string_format_zeroth  b = L.build_global_stringptr "[%s"
      ↪  "fmt" b in
178
179   let int_format_arr      b = L.build_global_stringptr ", %d"
      ↪  "fmt" b
180   and num_format_arr      b = L.build_global_stringptr ", %f"
      ↪  "fmt" b
181   and string_format_arr   b = L.build_global_stringptr ", %s"
      ↪  "fmt" b in
182
183   let int_format_last     b = L.build_global_stringptr ", %d]\n"
      ↪  "fmt" b
184   and num_format_last     b = L.build_global_stringptr ", %f]\n"
      ↪  "fmt" b
185   and string_format_last  b = L.build_global_stringptr ", %s]\n"
      ↪  "fmt" b in
186
187   let format_arr_print x_type index length builder =
188     let b = builder in
189     if index == 0 then
190       match x_type with
191         A.Int -> int_format_zeroth b
192       | A.Num -> num_format_zeroth b
193       | A.String -> string_format_zeroth b
194       | A.Bool -> int_format_zeroth b
195       | _ -> raise (Failure ("Invalid array print type"))
196     else if index == length - 1 then
197       match x_type with
198         A.Int -> int_format_last b
199       | A.Num -> num_format_last b
200       | A.String -> string_format_last b
201       | A.Bool -> int_format_last b
202       | _ -> raise (Failure ("Invalid array print type"))
```

```
203     else
204       match x_type with
205         A.Int -> int_format_arr b
206       | A.Num -> num_format_arr b
207       | A.String -> string_format_arr b
208       | A.Bool -> int_format_arr b
209       | _ -> raise (Failure ("Invalid array print type"))
210     in
211
212     (* Fill in the body of the given function *)
213     let build_function_body func_decl =
214       let (the_function, _) = StringMap.find func_decl.A.fname
          ↪  function_decls in
215       let llbuilder = L.builder_at_end context (L.entry_block
          ↪  the_function) in
216
217       (* Construct the function's "locals": formal arguments and
          ↪  locally
218          declared variables.  Allocate each on the stack,
    ↪  initialize their
219          value, if appropriate, and remember their values in the
    ↪  "locals" map *)
220
221       (* function to add local variables to a map *)
222     let local_vars =
223         let add_formal m (t,n) p = L.set_value_name n p;
224         let local = L.build_alloca (ltype_of_typ t) n llbuilder
            ↪  in
225         ignore (L.build_store p local llbuilder);
226         StringMap.add n (t,local) m in
227
228         (* add formals to the map *)
229         ref (List.fold_left2 add_formal StringMap.empty
            ↪  func_decl.A.formals
230           (Array.to_list (L.params the_function))) in
231
232       (* Return the value or the type for a variable or formal
          ↪  argument *)
233       (* All the tables have the structure (type, llvalue) *)
234       let lookup n : L.llvalue =
235         try (snd (StringMap.find n !local_vars))
236         with Not_found -> (raise ( Failure ("Variable not
            ↪  found!")))
237       in
238
239       let name_to_type n : A.typ =
```

```
240         try (fst (StringMap.find n !local_vars))
241         with Not_found -> (raise ( Failure ("Variable not
            ↪ found!")))
242       in
243     (* Array creation, initialization, access *)
244     let create_array t len builder =
245       let ltype = ltype_of_typ t in
246       let size_t = L.build_intcast (L.size_of ltype) i32_t "tmp"
            ↪ builder in
247       let total_size = L.build_mul size_t len "tmp" builder in
248       let total_size = L.build_add total_size (L.const_int i32_t
            ↪ 1) "tmp" builder in
249       let arr_malloc = L.build_array_malloc ltype total_size
            ↪ "tmp" builder in
250       let arr = L.build_pointercast arr_malloc (L.pointer_type
            ↪ ltype) "tmp" builder in
251       arr
252     in
253
254     let initialize_array t el builder =
255       let len = L.const_int i32_t (List.length el) in
256       let arr = create_array t len builder in
257       let _ =
258         let assign_value i =
259           let index = L.const_int i32_t i in
260           let index = L.build_add index (L.const_int i32_t 1)
                ↪ "tmp" builder in
261           let _val = L.build_gep arr [| index |] "tmp" builder
                ↪ in
262           L.build_store (List.nth el i) _val builder
263         in
264         for i = 0 to (List.length el)-1 do
265           ignore (assign_value i)
266         done
267       in
268       arr
269     in
270
271     let access_array arr index assign builder =
272       let index = L.build_add index (L.const_int i32_t 1) "tmp"
            ↪ builder in
273       let arr = L.build_load arr "tmp" builder in
274       let _val = L.build_gep arr [| index |] "tmp" builder in
275       if assign then _val else L.build_load _val "tmp" builder
276     in
277
```

```
278     let get_array_len = function
279       A.Id name -> (match (name_to_type name) with
280                      A.Arraytype(_, len) -> len
281                    | _ -> raise (Failure ("Can't get the length
                       ↪  of this object")))
282     in
283
284     let is_array = function
285       A.Id name -> (match (name_to_type name) with
286         A.Arraytype(_, _) -> true
287       | _ -> false)
288     in
289
290     let rec get_array_index e =
291       match e with
292         A.IntLit x -> x
293       | A.Binop (e1, op, e2) -> (match op with
294               A.Add -> (get_array_index e1) + (get_array_index
                         ↪  e2)
295             | A.Sub -> (get_array_index e1) - (get_array_index
                       ↪  e2)
296             | A.Mult -> (get_array_index e1) * (get_array_index
                       ↪  e2)
297             | A.Div -> (get_array_index e1) / (get_array_index
                       ↪  e2)
298             | _ -> 0)
299       | _ -> 0 (* If index is a variable we can't check, so
                  ↪  default to 0 *)
300     in
301
302     let get_type = function
303       A.Id name -> (match (name_to_type name) with
304         A.QueueType(typ) -> typ
305       | A.LinkedListType(typ) -> typ
306       | A.BSTreeType(typ) -> typ
307       | A.StackType(typ) -> typ
308       | A.Arraytype(typ, _) -> typ
309       | _ as typ -> typ)
310     in
311
312     let rec gen_type = function
313         A.IntLit _ -> A.Int
314       | A.NumLit _ -> A.Num
315       | A.StringLit _ -> A.String
316       | A.BoolLit _ -> A.Bool
```

```
317       | A.ArrayLit el -> A.Arraytype (gen_type (List.nth el 0),
          ↪  List.length el)
318       | A.ArrayElementAssign (_, _, el) -> gen_type el
319       | A.Id name -> (match (name_to_type name) with
320                     A.Arraytype(t, _) -> t
321                   | _ as ty -> ty)
322       | A.Unop(_,e) -> gen_type e
323       | A.Binop(e1,_,_) -> gen_type e1
324       | A.Postop(e, _) -> gen_type e
325       | A.Assign(_,var,_) -> gen_type (A.Id(var))
326       | A.Reassign(var,_) -> gen_type (A.Id(var))
327       | A.FuncCall(var,_) -> let fdecl =
328                              List.find (fun x -> x.A.fname = var)
                               ↪  functions in
329                              fdecl.A.typ
330       | A.ArrayAccess(id,_) -> gen_type (A.Id(id))
331       | A.ObjectCall(_, "size", _) -> A.Int
332       | A.ObjectCall(obj, "peek", _) -> get_type obj
333       | A.ObjectCall(obj, "get", _) -> get_type obj
334       | A.ObjectCall(obj, "contains", _) -> A.Bool
335       | A.Noexpr -> A.Void
336     in
337
338     let call_size_ptr = function
339       A.Id name -> (match (name_to_type name) with
340         A.QueueType _ -> sizeQ_f
341       | A.LinkedListType _ -> sizeList_f
342       | A.StackType _ -> sizeS_f
343       | _ -> raise (Failure ("Invalid data structure type - size
          ↪  function")))
344     in
345
346     let call_add_ptr ds_type = function
347       A.Id name -> (match (name_to_type name) with
348         A.QueueType _ -> enqueue_f
349       | A.LinkedListType _ -> add_f
350       | A.StackType _ -> push_f
351       | A.BSTreeType _ -> (match ds_type with
352             A.Int -> bstreeadd_int_f
353           | A.Num -> bstreeadd_float_f)
354       | _ -> raise (Failure ("Invalid data structure type - add
          ↪  function")))
355     in
356
357     let call_pop_ptr = function
358       A.Id name -> (match (name_to_type name) with
```

```
359         A.QueueType _ -> dequeue_f
360       | A.StackType _ -> pop_f
361       | _ -> raise (Failure ("Invalid data structure type -
          ↪  delete function")))
362     in
363
364     let call_show_ptr data_type = function
365       A.Id name -> (match (name_to_type name) with
366         A.QueueType _ -> (match data_type with
367             A.Int -> q_show_int
368           | A.Num -> q_show_float
369           | A.Bool -> q_show_int
370           | A.String -> q_show_string)
371       | A.StackType _ -> (match data_type with
372             A.Int -> s_show_int
373           | A.Num -> s_show_float
374           | A.Bool -> s_show_int
375           | A.String -> s_show_string)
376       | A.LinkedListType _ -> (match data_type with
377             A.Int -> l_show_int
378           | A.Num -> l_show_float
379           | A.Bool -> l_show_int
380           | A.String -> l_show_string)
381       | A.BSTreeType _ -> (match data_type with
382             A.Int -> bstree_show_int
383           | A.Num -> bstree_show_float)
384       | _ -> raise (Failure ("Invalid data structure type - show
          ↪  function")))
385     in
386
387     let call_delete_ptr ds_type = function
388       A.Id name -> (match (name_to_type name) with
389         A.LinkedListType _ -> delete_f
390       | A.BSTreeType _ -> (match ds_type with
391             A.Int -> bstreedelete_int_f
392           | A.Num -> bstreedelete_float_f)
393       | _ -> raise (Failure ("Invalid data structure type -
          ↪  delete function")))
394     in
395
396     let call_contains_ptr ds_type = function
397       A.Id name -> (match (name_to_type name) with
398         A.BSTreeType _ -> (match ds_type with
399             A.Int -> bstreecontains_int_f
400           | A.Num -> bstreecontains_float_f)
```

```
401       | _ -> raise (Failure ("Invalid data structure type -
          ↪  contains function")))
402     in
403
404     let call_quicksort_ptr data_type = function
405       A.Id name -> (match (name_to_type name) with
406         A.Arraytype(_, _) -> (match data_type with
407           A.Int -> int_quickSort_f
408         | A.Num -> num_quickSort_f
409         | A.String -> string_quickSort_f)
410       | _ -> raise (Failure ("Cannot perform quicksort on this
          ↪  datatype")))
411     in
412
413     let call_show_quicksort_ptr data_type = function
414       A.Id name -> (match (name_to_type name) with
415         A.Arraytype(_, _) -> (match data_type with
416           A.Int -> int_show_quickSort_f
417         | A.Num -> num_show_quickSort_f
418         | A.String -> string_show_quickSort_f)
419       | _ -> raise (Failure ("Cannot perform quicksort on this
          ↪  datatype")))
420     in
421
422     let call_peek_ptr = function
423       A.Id name -> (match (name_to_type name) with
424         A.QueueType _ -> peek_f
425       | A.StackType _ -> top_f
426       | _ -> raise (Failure ("Invalid data structure type - peek
          ↪  function")))
427     in
428
429     let init_bstree_add typ = match typ with
430       | A.Int -> bstreeadd_int_f
431       | A.Num -> bstreeadd_float_f
432       | _ -> raise (Failure ("Invalid tree constructor"))
433     in
434
435     let rec expr_generator llbuilder = function
436         A.NumLit(n) -> L.const_float f_t n
437       | A.IntLit(i) -> L.const_int i32_t i
438       | A.BoolLit(b) -> L.const_int i1_t (if b then 1 else 0)
439       | A.StringLit(s) -> L.build_global_stringptr s "string"
          ↪  llbuilder
440       | A.Id s -> L.build_load (lookup s) s llbuilder
441       | A.QueueLit (typ, act) ->
```

```
442           let d_ltyp = ltype_of_typ typ in
443           let queue_ptr = L.build_call initQueue_f [| |] "init"
              ↪  llbuilder in
444           let add_element elem =
445             let d_ptr = match typ with
446             | A.QueueType _ -> expr_generator llbuilder elem
447             | _ ->
448               let element = expr_generator llbuilder elem in
449               let d_ptr = L.build_malloc d_ltyp "tmp" llbuilder in
450               ignore (L.build_store element d_ptr llbuilder);
                  ↪  d_ptr in
451             let void_d_ptr = L.build_bitcast d_ptr (L.pointer_type
                  ↪  i8_t) "ptr" llbuilder in
452             ignore (L.build_call enqueue_f [| queue_ptr;
                  ↪  void_d_ptr |] "" llbuilder)
453           in ignore (List.map add_element act);
454           queue_ptr
455         | A.LinkedListLit (typ, act) ->
456           let d_ltyp = ltype_of_typ typ in
457           let list_ptr = L.build_call initList_f [| |] "init"
              ↪  llbuilder in
458           let add_element elem =
459             let d_ptr = match typ with
460             | A.LinkedListType _ -> expr_generator llbuilder elem
461             | _ ->
462               let element = expr_generator llbuilder elem in
463               let d_ptr = L.build_malloc d_ltyp "tmp" llbuilder in
464               ignore (L.build_store element d_ptr llbuilder);
                  ↪  d_ptr in
465             let void_d_ptr = L.build_bitcast d_ptr (L.pointer_type
                  ↪  i8_t) "ptr" llbuilder in
466             ignore (L.build_call add_f [| list_ptr; void_d_ptr |]
                  ↪  "" llbuilder)
467           in ignore (List.map add_element act);
468           list_ptr
469         | A.StackLit (typ, act) ->
470           let d_ltyp = ltype_of_typ typ in
471           let stack_ptr = L.build_call initStack_f [| |] "init"
              ↪  llbuilder in
472           let add_element elem =
473             let d_ptr = match typ with
474             | A.StackType _ -> expr_generator llbuilder elem
475             | _ ->
476               let element = expr_generator llbuilder elem in
477               let d_ptr = L.build_malloc d_ltyp "tmp" llbuilder in
```

```
478             ignore (L.build_store element d_ptr llbuilder);
                ↪  d_ptr in
479           let void_d_ptr = L.build_bitcast d_ptr (L.pointer_type
              ↪  i8_t) "ptr" llbuilder in
480           ignore (L.build_call push_f [| stack_ptr; void_d_ptr
              ↪  |] "" llbuilder)
481         in ignore (List.map add_element act);
482         stack_ptr
483       | A.BSTreeLit (typ, act) ->
484         let d_ltyp = ltype_of_typ typ in
485         let bstree_ptr = L.build_call initBSTree_f [| |] "init"
            ↪  llbuilder in
486         let obj_method = init_bstree_add typ in
487         let add_element elem =
488           let d_ptr = match typ with
489           | A.BSTreeType _ -> expr_generator llbuilder elem
490           | _ ->
491             let element = expr_generator llbuilder elem in
492             let d_ptr = L.build_malloc d_ltyp "tmp" llbuilder in
493             ignore (L.build_store element d_ptr llbuilder);
                ↪  d_ptr in
494           let void_d_ptr = L.build_bitcast d_ptr (L.pointer_type
              ↪  i8_t) "ptr" llbuilder in
495           ignore (L.build_call obj_method [| bstree_ptr;
              ↪  void_d_ptr |] "" llbuilder)
496         in ignore (List.map add_element act);
497         bstree_ptr
498       | A.Binop (e1, op, e2) ->
499         let e1' = expr_generator llbuilder e1 in
500         let e2' = expr_generator llbuilder e2
501         and num_ops = (match op with
502             A.Add     -> L.build_fadd
503           | A.Sub     -> L.build_fsub
504           | A.Mult    -> L.build_fmul
505           | A.Div     -> L.build_fdiv
506           | A.Mod     -> L.build_frem
507           | A.And     -> L.build_and
508           | A.Or      -> L.build_or
509           | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
510           | A.Neq     -> L.build_fcmp L.Fcmp.One
511           | A.Less    -> L.build_fcmp L.Fcmp.Ult
512           | A.Leq     -> L.build_fcmp L.Fcmp.Ole
513           | A.Greater -> L.build_fcmp L.Fcmp.Ogt
514           | A.Geq     -> L.build_fcmp L.Fcmp.Oge
515           | _ -> L.build_fcmp L.Fcmp.Oeq
516         )
```

```
517           and int_ops = (match op with
518               A.Add     -> L.build_add
519             | A.Sub   -> L.build_sub
520             | A.Mult   -> L.build_mul
521             | A.Div  -> L.build_sdiv
522             | A.Mod     -> L.build_srem
523             | A.And     -> L.build_and
524             | A.Or      -> L.build_or
525             | A.Equal   -> L.build_icmp L.Icmp.Eq
526             | A.Neq     -> L.build_icmp L.Icmp.Ne
527             | A.Less    -> L.build_icmp L.Icmp.Slt
528             | A.Leq     -> L.build_icmp L.Icmp.Sle
529             | A.Greater -> L.build_icmp L.Icmp.Sgt
530             | A.Geq     -> L.build_icmp L.Icmp.Sge
531             | _ -> L.build_icmp L.Icmp.Eq
532           )
533           in
534
535           if ((L.type_of e1' = f_t) && (L.type_of e2' = f_t)) then
              ↪  num_ops e1' e2' "tmp" llbuilder
536           else int_ops e1' e2' "tmp" llbuilder
537         | A.Unop (op, e) ->
538             let e' = expr_generator llbuilder e in
539             let int_unops op e' = (match op with
540                 A.Neg     -> L.build_neg e' "tmp" llbuilder
541               | A.Not     -> L.build_not e' "tmp" llbuilder
542             )
543             and num_unops op e' = (match op with
544                 A.Neg     -> L.build_fneg e' "tmp" llbuilder )
545             in
546             if ((L.type_of e' = f_t))
547             then num_unops op e'
548             else int_unops op e'
549
550         | A.Postop (e, op) ->
551             let e' = expr_generator llbuilder e in
552             let llval = (match e with
553               A.Id(s) -> s
554             | _ -> raise (Failure("This input type cannot be
                  ↪  incremented/decremented"))
555             )
556             and op_typ = (match op with
557               A.Incr -> A.Add
558             | A.Decr -> A.Sub
559             )
560             and num_typ = if ((L.type_of e' = f_t))
```

```
561             then A.NumLit(1.0)
562             else A.IntLit(1) in
563
564             expr_generator llbuilder (A.Reassign(llval, A.Binop(e,
                ↪  op_typ, num_typ)))
565         | A.Assign (t, s, e) ->
566             let _ = (local_vars := StringMap.add s (t,
                ↪  (L.build_alloca (ltype_of_typ t)) s llbuilder)
                ↪  !local_vars) in
567             let e' = expr_generator llbuilder e and llval = lookup
                ↪  s in
568             ignore (L.build_store e' llval llbuilder); e'
569         | A.Reassign (s, e) ->
570             let e' = expr_generator llbuilder e and llval = lookup
                ↪  s in
571             ignore (L.build_store e' llval llbuilder); e'
572         | A.ArrayLit el -> let t = gen_type (List.nth el 0) in
573             initialize_array t (List.map (expr_generator
                ↪  llbuilder) el) llbuilder
574         | A.ArrayAccess (s, i) ->
575             let index = expr_generator llbuilder i
576             and llval = lookup s in
577             let index_int = get_array_index i
578             and len = match (name_to_type s) with
579                       A.Arraytype(_, len) -> len
580                     | _ -> raise (Failure ("Can't get the length
                        ↪  of this object")) in
581
582             if (len < index_int) || (index_int < 0)
583             then raise (Failure ("Array index out of bounds"))
584             else access_array llval index false llbuilder
585         | A.ArrayElementAssign (s, i, e) ->
586             let e' = expr_generator llbuilder e in
587             let index = expr_generator llbuilder i in
588             let llval = lookup s in
589             let var = access_array llval index true llbuilder in
590             ignore (L.build_store e' var llbuilder); e'
591         | A.FuncCall("print", [e]) ->
592             let e' = expr_generator llbuilder e in
593             let e_type = gen_type e in
594             L.build_call printf_func [| (format_str e_type
                ↪  llbuilder) ; e' |] "printf" llbuilder
595         | A.FuncCall ("printbig", [e]) ->
596             L.build_call printbig_func [| (expr_generator
                ↪  llbuilder e) |] "printbig" llbuilder
597         | A.FuncCall (f, act) ->
```

```
598             let (fdef, func_decl) = StringMap.find f
                ↪ function_decls in
599             let actuals = List.rev (List.map (expr_generator
                ↪ llbuilder) (List.rev act)) in
600             let result =
601               (match func_decl.A.typ with
602                 A.Void -> ""
603               | _ -> f ^ "_result")
604             in
605             L.build_call fdef (Array.of_list actuals) result
                ↪ llbuilder
606
607         | A.ObjectCall (obj, "remove", []) ->
608           let obj_val = expr_generator llbuilder obj in
609           let obj_method = call_pop_ptr obj in
610           ignore (L.build_call obj_method [| obj_val|] ""
                  ↪ llbuilder); obj_val
611         | A.ObjectCall (obj, "peek", []) ->
612           let obj_val = expr_generator llbuilder obj in
613           let obj_type = get_type obj in
614           let obj_method = call_peek_ptr obj in
615           let val_ptr = L.build_call obj_method [| obj_val |]
                  ↪ "val_ptr" llbuilder in
616           let dtyp = ltype_of_typ obj_type in
617           let ptr = L.build_bitcast val_ptr (L.pointer_type dtyp)
                  ↪ "d_ptr" llbuilder in
618           (L.build_load ptr "d_ptr" llbuilder)
619         | A.ObjectCall (obj, "show", []) ->
620           let obj_val = expr_generator llbuilder obj in
621           let obj_type = get_type obj in
622           let print_array arr builder =
623             let len = get_array_len obj in
624             let llval = lookup (match arr with
625                                   A.Id(s) -> s
626                                 | _ -> raise (Failure("variable
                                  ↪ not found"))) in
627             for index = 0 to len - 1 do
628               let item = access_array llval (L.const_int i32_t
                      ↪ index) false builder in
629               (L.build_call printf_func [| (format_arr_print
                      ↪ obj_type index len llbuilder) ; item |] "printf"
                      ↪ llbuilder)
630             done
631           in
632
633           if is_array obj then print_array obj llbuilder
```

```
634           else (let obj_method = call_show_ptr obj_type obj in
635             ignore (L.build_call obj_method [| obj_val |] ""
                ↪  llbuilder));
636           obj_val
637         | A.ObjectCall (obj, "size", []) ->
638           let e = expr_generator llbuilder obj in
639           let obj_size = call_size_ptr obj in
640           let size_ptr = L.build_call obj_size [| e |] "isEmpty"
                ↪  llbuilder in
641           size_ptr
642         | A.ObjectCall (obj, "add", [e]) ->
643           let obj_val = expr_generator llbuilder obj in
644           let obj_type = get_type obj in
645           let e_val = expr_generator llbuilder e in
646           let d_ltyp = L.type_of e_val in
647           let d_ptr = L.build_malloc d_ltyp "tmp" llbuilder in
648           ignore(L.build_store e_val d_ptr llbuilder);
649           let obj_method = call_add_ptr obj_type obj in
650           let void_e_ptr = L.build_bitcast d_ptr (L.pointer_type
                ↪  i8_t) "ptr" llbuilder in
651           ignore (L.build_call obj_method [| obj_val; void_e_ptr|]
                ↪  "" llbuilder); obj_val
652         | A.ObjectCall (obj, "delete", [e]) ->
653           let obj_val = expr_generator llbuilder obj in
654           let obj_type = get_type obj in
655           let e_val = expr_generator llbuilder e in
656           let obj_method = call_delete_ptr obj_type obj in
657           ignore (L.build_call obj_method [| obj_val; e_val |] ""
                ↪  llbuilder);
658           obj_val
659         | A.ObjectCall (obj, "contains", [e]) ->
660           let obj_val = expr_generator llbuilder obj in
661           let obj_type = get_type obj in
662           let e_val = expr_generator llbuilder e in
663           let obj_method = call_contains_ptr obj_type obj in
664           let result = L.build_call obj_method [| obj_val; e_val
                ↪  |] "res" llbuilder in
665           result
666         | A.ObjectCall (l, "get", [e]) ->
667           let l_ptr = expr_generator llbuilder l in
668           let e_val = expr_generator llbuilder e in
669           let l_type = get_type l in
670           let val_ptr = L.build_call get_f [| l_ptr; e_val |]
                ↪  "val_ptr" llbuilder in
671           let l_dtyp = ltype_of_typ l_type in
```

```
672         let d_ptr = L.build_bitcast val_ptr (L.pointer_type
            ↪ l_dtyp) "d_ptr" llbuilder in
673         (L.build_load d_ptr "d_ptr" llbuilder)
674       | A.ObjectCall(a, "quickSort", []) ->
675         let a_val = expr_generator llbuilder a in
676         let len = L.const_int i32_t (get_array_len a) in
677         let obj_method = call_quicksort_ptr (get_type a) a in
678         ignore (L.build_call obj_method [| a_val; len |] ""
            ↪ llbuilder); a_val
679       | A.ObjectCall(a, "showQuickSort", []) ->
680         let a_val = expr_generator llbuilder a in
681         let len = L.const_int i32_t (get_array_len a) in
682         let obj_method = call_show_quicksort_ptr (get_type a) a
            ↪ in
683         ignore (L.build_call obj_method [| a_val; len|] ""
            ↪ llbuilder); a_val in
684
685       (* Invoke "f llbuilder" if the current block doesn't
          ↪ already
686          have a terminal (e.g., a branch). *)
687       let add_terminal llbuilder f =
688         match L.block_terminator (L.insertion_block llbuilder)
            ↪ with
689           Some _ -> ()
690         | None -> ignore (f llbuilder) in
691
692
693     (* Statement generator *)
694     let rec stmt_generator llbuilder = function
695       A.Block stmtlist -> List.fold_left stmt_generator
          ↪ llbuilder stmtlist
696     | A.Return e -> ignore (match func_decl.A.typ with
697                           A.Void -> L.build_ret_void llbuilder
698                         | _      -> L.build_ret
                          ↪ (expr_generator llbuilder e)
                          ↪ llbuilder); llbuilder
699     | A.Expr se -> ignore (expr_generator llbuilder se);
        ↪ llbuilder
700     | A.If (predicate, s1, s2) -> generate_if predicate s1 s2
        ↪ llbuilder
701     | A.While (predicate, body) -> generate_while  predicate
        ↪ body llbuilder
702     | A.For (e1, e2, e3, s) -> stmt_generator llbuilder (
        ↪ A.Block [A.Expr e1 ; A.While (e2, A.Block [s ; A.Expr
        ↪ e3]) ] )
703
```

```
704     and generate_if predicate s1 s2 llbuilder =
705       let bool_val = expr_generator llbuilder predicate in
706       let merge_bb = L.append_block context "merge" the_function
          ↪  in
707
708       let then_bb = L.append_block context "then" the_function
          ↪  in
709       add_terminal (stmt_generator (L.builder_at_end context
          ↪  then_bb) s1)
710        (L.build_br merge_bb);
711
712       let else_bb = L.append_block context "else" the_function
          ↪  in
713       add_terminal (stmt_generator  (L.builder_at_end context
          ↪  else_bb) s2)
714        (L.build_br merge_bb);
715
716       ignore (L.build_cond_br bool_val then_bb else_bb
          ↪  llbuilder);
717       L.builder_at_end context merge_bb
718
719     and generate_while predicate body llbuilder =
720       let pred_bb = L.append_block context "while" the_function
          ↪  in
721         ignore (L.build_br pred_bb llbuilder);
722
723         let body_bb = L.append_block context "while_body"
            ↪  the_function in
724         add_terminal (stmt_generator (L.builder_at_end context
            ↪  body_bb) body)
725           (L.build_br pred_bb);
726
727         let pred_builder = L.builder_at_end context pred_bb in
728         let bool_val = expr_generator pred_builder predicate in
729
730         let merge_bb = L.append_block context "merge"
            ↪  the_function in
731         ignore (L.build_cond_br bool_val body_bb merge_bb
            ↪  pred_builder);
732         L.builder_at_end context merge_bb
733     in
734
735       (* Build the code for each statement in the function *)
736       let llbuilder = stmt_generator llbuilder (A.Block
          ↪  func_decl.A.body) in
737
```

```
738       (* Add a return if the last block falls off the end *)
739       add_terminal llbuilder (match func_decl.A.typ with
740           A.String -> L.build_ret (L.build_global_stringptr ""
              ↪  "string" llbuilder)
741         | A.Void -> L.build_ret_void
742         | A.Num -> L.build_ret (L.const_float f_t 0.)
743         | A.Bool -> L.build_ret (L.const_int i1_t 0)
744         | _ -> L.build_ret_void)
745     in
746     List.iter build_function_body functions;
747     the_module
```

8.6 strux.ml

Josh, Kofi, Millie, Sophie

```
1  (* Top-level of the Strux compiler: scan & parse the input,
2     check the resulting AST, generate LLVM IR, and dump the
   ↪  module *)
3
4  module StringMap = Map.Make(String)
5
6  type action = Ast | LLVM_IR | Compile
7
8  let _ =
9    let action = ref Compile in
10   let set_action a () = action := a in
11   let speclist = [
12     ("-a", Arg.Unit (set_action Ast), "Print the AST");
13     ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated
       ↪  LLVM IR");
14     ("-c", Arg.Unit (set_action Compile),
15       "Check and print the generated LLVM IR (default)");
16   ] in
17   let usage_msg = "usage: ./strux.native [-a|-l|-c]
      ↪  [file.strux]" in
18   let channel = ref stdin in
19   Arg.parse speclist (fun filename -> channel := open_in
      ↪  filename) usage_msg;
20   let lexbuf = Lexing.from_channel !channel in
21   let ast = Parser.program Scanner.token lexbuf in
22   Semant.check ast;
23   match !action with
24     Ast -> print_string (Ast.string_of_program ast)
```

```
  | LLVM_IR -> print_string (Llvm.string_of_llmodule
  ↪  (Codegen.translate ast))
  | Compile -> let m = Codegen.translate ast in
    Llvm_analysis.assert_valid_module m;
    print_string (Llvm.string_of_llmodule m)
```

8.7 bstree.h

Josh

```
#ifndef __BSTREE_H__
#define __BSTREE_H__

int DATA_WIDTH = 6;

struct BSTreeNode
{
    void *data;
    struct BSTreeNode *left;
    struct BSTreeNode *right;
    struct BSTreeNode *parent;
};

struct BSTree
{
    struct BSTreeNode *root;
};

struct BSTree *initBSTree();
struct BSTreeNode *createNode(void *data);
struct BSTreeNode *addIntToTreeHelper(struct BSTreeNode *node,
↪  void *data);
void addIntToTree(struct BSTree *tree, void *data);
struct BSTreeNode *addNumToTreeHelper(struct BSTreeNode *node,
↪  void *data);
void addNumToTree(struct BSTree *tree, void *data);
struct BSTreeNode *getMin(struct BSTreeNode *node);
struct BSTreeNode *deleteIntFromTreeHelper(struct BSTreeNode
↪  *node, int data);
void deleteIntFromTree(struct BSTree *tree, int data);
struct BSTreeNode *deleteNumFromTreeHelper(struct BSTreeNode
↪  *node, double data);
void deleteNumFromTree(struct BSTree *tree, double data);
int treeContainsInt(struct BSTree *tree, int data);
int treeContainsNum(struct BSTree *tree, double data);
```

```
int treeHeight(struct BSTreeNode* node);
int printLeftChild(struct BSTreeNode *tree, int offset, int depth, char s[50][255], int typ);
int printRightChild(struct BSTreeNode *tree, int offset, int depth, char s[50][255], int typ);
void showTree(struct BSTreeNode *tree, int typ);
void showIntTree(struct BSTree *tree);
void showNumTree(struct BSTree *tree);

#endif
```

8.8 bstree.c

Josh

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "utils.h"
#include "bstree.h"

struct BSTree *initBSTree()
{
    struct BSTree *tree = (struct BSTree*) malloc(sizeof(struct BSTree));
    tree->root = NULL;
    return tree;
}

/* Allocates memory for a new BSTreeNode and returns its pointer */
struct BSTreeNode *createNode(void *data)
{
    struct BSTreeNode *newNode = (struct BSTreeNode*) malloc(sizeof(struct BSTreeNode));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    newNode->parent = NULL;
    return newNode;
}

/* Adds the given data to the tree */
struct BSTreeNode *addIntToTreeHelper(struct BSTreeNode *node, void *data)
```

```
27  {
28      /* Node is empty, place the new node here */
29      if (node == NULL)
30          return createNode(data);
31
32      if (*(int *) data < *(int *) node->data)
33          node->left = addIntToTreeHelper(node->left, data);
34      else if (*(int *) data > *(int *) node->data)
35          node->right = addIntToTreeHelper(node->right, data);
36
37      return node;
38  }
39
40  void addIntToTree(struct BSTree *tree, void *data)
41  {
42      if (tree->root == NULL) {
43          tree->root = createNode(data);
44      }
45      else {
46          addIntToTreeHelper(tree->root, data);
47      }
48  }
49
50  /* Adds the given data to the tree */
51  struct BSTreeNode *addNumToTreeHelper(struct BSTreeNode *node,
    ↪  void *data)
52  {
53      /* Node is empty, place the new node here */
54      if (node == NULL)
55          return createNode(data);
56
57      if (*(double *) data < *(double *) node->data)
58          node->left = addNumToTreeHelper(node->left, data);
59      else if (*(double *) data > *(double *) node->data)
60          node->right = addNumToTreeHelper(node->right, data);
61
62      return node;
63  }
64
65  void addNumToTree(struct BSTree *tree, void *data)
66  {
67      if (tree->root == NULL) {
68          tree->root = createNode(data);
69      }
70      else {
71          addNumToTreeHelper(tree->root, data);
```

```
72      }
73  }
74
75  struct BSTreeNode *getMin(struct BSTreeNode *node)
76  {
77      /* Invalid node given */
78      if (node == NULL)
79          return NULL;
80
81      if (node->left)
82          return getMin(node->left);
83      else
84          return node;
85  }
86
87  struct BSTreeNode *deleteIntFromTreeHelper(struct BSTreeNode
    ↪  *node, int data)
88  {
89      struct BSTreeNode *temp;
90
91      /* element not found */
92      if (node == NULL)
93          return NULL;
94
95      if (data < *(int *) node->data)
96          node->left = deleteIntFromTreeHelper(node->left, data);
97      else if (data > *(int *) node->data)
98          node->right = deleteIntFromTreeHelper(node->right,
            ↪  data);
99
100     /* element found */
101     else{
102
103         /* two children present, handle accordingly */
104         if (node->left && node->right) {
105
106             temp = getMin(node->right);
107             node->data = temp->data;
108             node->right = deleteIntFromTreeHelper(node->right,
                ↪  *(int *) temp->data);
109         }
110
111         /* zero or one child present, handle accordingly */
112         else {
113
114             temp = node;
```

```
115
116             if (node->left == NULL)
117                 node = node->right;
118             else if (node->right == NULL)
119                 node = node->left;
120
121             free(temp);
122         }
123     }
124
125     return node;
126
127 }
128
129 void deleteIntFromTree(struct BSTree *tree, int data)
130 {
131     if (tree->root->left == NULL && tree->root->right == NULL) {
132
133         if ( *(int *) tree->root->data == data) {
134             struct BSTreeNode *temp = tree->root;
135             tree->root = NULL;
136             free(temp);
137         }
138
139         return ;
140     }
141
142     deleteIntFromTreeHelper(tree->root, data);
143 }
144
145 struct BSTreeNode *deleteNumFromTreeHelper(struct BSTreeNode
    ↪  *node, double data)
146 {
147     struct BSTreeNode *temp;
148
149     /* element not found */
150     if (node == NULL)
151         return NULL;
152
153     if (data < *(double *) node->data)
154         node->left = deleteNumFromTreeHelper(node->left, data);
155     else if (data > *(double *) node->data)
156         node->right = deleteNumFromTreeHelper(node->right,
            ↪  data);
157
158     /* element found */
```

```
159     else{
160 
161         /* two children present, handle accordingly */
162         if (node->left && node->right) {
163 
164             temp = getMin(node->right);
165             node->data = temp->data;
166             node->right = deleteNumFromTreeHelper(node->right,
                ↪  *(int *) temp->data);
167         }
168 
169         /* zero or one child present, handle accordingly */
170         else {
171 
172             temp = node;
173 
174             if (node->left == NULL)
175                 node = node->right;
176             else if (node->right == NULL)
177                 node = node->left;
178 
179             free(temp);
180         }
181     }
182 
183     return node;
184 
185 }
186 
187 void deleteNumFromTree(struct BSTree *tree, double data)
188 {
189     if (tree->root->left == NULL && tree->root->right == NULL) {
190 
191         if ( *(double *) tree->root->data == data) {
192             struct BSTreeNode *temp = tree->root;
193             tree->root = NULL;
194             free(temp);
195         }
196 
197         return ;
198     }
199 
200 
201     deleteNumFromTreeHelper(tree->root, data);
202 }
203 
```

```
204
205
206 int treeContainsIntHelper(struct BSTreeNode *node, int data)
207 {
208     /* Not found */
209     if (node == NULL)
210         return 0;
211
212     /* Keep searching */
213     if (data < *(int *) node->data)
214         return treeContainsIntHelper(node->left, data);
215     else if (data > *(int *) node->data)
216         return treeContainsIntHelper(node->right, data);
217
218     /* Element found */
219     else
220         return 1;
221 }
222
223 int treeContainsNumHelper(struct BSTreeNode *node, double data)
224 {
225     /* Not found */
226     if (node == NULL)
227         return 0;
228
229     /* Keep searching */
230     if (data < *(double *) node->data)
231         return treeContainsNumHelper(node->left, data);
232     else if (data > *(double *) node->data)
233         return treeContainsNumHelper(node->right, data);
234
235     /* Element found */
236     else
237         return 1;
238 }
239
240 int treeContainsNum(struct BSTree *tree, double data)
241 {
242     return treeContainsNumHelper(tree->root, data);
243 }
244
245 int treeContainsInt(struct BSTree *tree, int data)
246 {
247     return treeContainsIntHelper(tree->root, data);
248 }
249
```

```
250 int treeHeight(struct BSTreeNode* node)
251 {
252     if (node == NULL)
253         return 0;
254 
255     else {
256 
257         /* get the height of both children */
258         int leftHeight = treeHeight(node->left);
259         int rightHeight = treeHeight(node->right);
260 
261         /* take the larger of the two */
262         if (leftHeight > rightHeight)
263             return (leftHeight + 1);
264         else
265             return (rightHeight + 1);
266    }
267 }
268 
269 
270 int printLeftChild(struct BSTreeNode *tree, int offset, int
    ↪  depth, char s[50][255], int typ)
271 {
272     char b[20];
273 
274     if (!tree) return 0;
275 
276     if (typ == FLOATING)
277         sprintf(b, "(%.2lf)", *(double *)tree->data);
278     else
279         sprintf(b, "(%04d)", *(int *)tree->data);
280 
281     int left  = printLeftChild(tree->left, offset, depth + 1, s,
    ↪  typ);
282     int right = printRightChild(tree->right, offset + left +
    ↪  DATA_WIDTH, depth + 1, s, typ);
283 
284     for (int i = 0; i < DATA_WIDTH; i++)
285         s[2 * depth][offset + left + i] = b[i];
286 
287     if (depth) {
288 
289         for (int i = 0; i < DATA_WIDTH + right; i++)
290             s[2 * depth - 1][offset + left + DATA_WIDTH/2 + i] =
                ↪  '-';
291 
```

```
292         s[2 * depth - 1][offset + left + DATA_WIDTH/2] = '*';
293         s[2 * depth - 1][offset + left + DATA_WIDTH + right +
            ↪  DATA_WIDTH/2] = '*';
294
295     }
296
297     return left + DATA_WIDTH + right;
298 }
299
300 int printRightChild(struct BSTreeNode *tree, int offset, int
    ↪  depth, char s[50][255], int typ)
301 {
302     char b[20];
303
304     if (!tree) return 0;
305
306     if (typ == FLOATING)
307         sprintf(b, "(%.2lf)", *(double *)tree->data);
308     else
309         sprintf(b, "(%04d)", *(int *)tree->data);
310
311     int left  = printLeftChild(tree->left, offset, depth + 1, s,
        ↪  typ);
312     int right = printRightChild(tree->right, offset + left +
        ↪  DATA_WIDTH, depth + 1, s, typ);
313
314     for (int i = 0; i < DATA_WIDTH; i++)
315         s[2 * depth][offset + left + i] = b[i];
316
317     if (depth) {
318
319         for (int i = 0; i < left + DATA_WIDTH; i++)
320             s[2 * depth - 1][offset - DATA_WIDTH/2 + i] = '-';
321
322         s[2 * depth - 1][offset + left + DATA_WIDTH/2] = '*';
323         s[2 * depth - 1][offset - DATA_WIDTH/2 - 1] = '*';
324     }
325
326     return left + DATA_WIDTH + right;
327 }
328
329 void showTree(struct BSTreeNode *root, int typ)
330 {
331     int maxDepth = treeHeight(root);
332
333     char s[50][255];
```

```
334
335     for (int i = 0; i < 50; i++)
336         sprintf(s[i], "%80s", " ");
337
338     printLeftChild(root, 0, 0, s, typ);
339
340     int until;
341
342     if (maxDepth * 2 - 1 > 50)
343         until = 50;
344     else
345         until = maxDepth * 2 - 1;
346
347     for (int i = 0; i < maxDepth*2-1; i++)
348         printf("%s\n", s[i]);
349 }
350
351 void showIntTree(struct BSTree *tree)
352 {
353     if (tree->root) {
354         showTree(tree -> root, INTEGER);
355         printf("\n%s\n",
             ↪  "------------------------------------------------------------------------------
356     }
357     else {
358         printf("%s\n", "Tree is empty!");
359     }
360 }
361
362 void showNumTree(struct BSTree *tree)
363 {
364     if (tree->root) {
365         showTree(tree -> root, FLOATING);
366         printf("\n%s\n",
             ↪  "------------------------------------------------------------------------------
367     }
368     else {
369         printf("%s\n", "Tree is empty!");
370     }
371 }
```

8.9 linkedlist.c

Kofi, Sophie

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdlib.h>
#include <string.h>
#include "utils.h"

struct LinkedList {
        struct ListNode *head;
        int size;
};

struct ListNode {
        void *data;
        struct ListNode *next;
};

struct LinkedList* initList() {
        struct LinkedList* list = (struct LinkedList*)
        ↪  malloc(sizeof(struct LinkedList));
        list->size = 0;
        list->head = NULL;
        return list;
}

void add(struct LinkedList *list, void *data) {
        struct ListNode* newNode = (struct ListNode*)
        ↪  malloc(sizeof(struct ListNode));
        if (newNode == NULL)
                return;

        newNode->data = data;
        newNode->next = NULL;

        if (list->head == NULL) {
                list->head = newNode;
                list->size++;
        } else {
                struct ListNode* temp = list->head;
                while (temp->next != NULL) {
                        temp = temp->next;
                }
                temp->next = newNode;
                list->size++;
        }
}
```

```
44
45  int isEmpty(struct LinkedList *list) {
46          return (list->head == NULL);
47  }
48
49  void delete(struct LinkedList* list, int index) {
50      if (isEmpty(list)) {
51          return;
52      }
53
54      if (index == 0) {
55          struct ListNode* temp = list->head;
56          list->head = list->head->next;
57          list->size--;
58          free(temp);
59          return;
60      }
61
62      int pointer = 0;
63      struct ListNode* temp = list->head;
64      while(pointer < index-1) {
65          temp = temp->next;
66          pointer++;
67      }
68      temp->next = temp->next->next;
69      temp = temp->next;
70      free(temp);
71      list->size--;
72  }
73
74  void* get(struct LinkedList* list, int index) {
75          struct ListNode* node = list->head;
76          while (index > 0) {
77                  node = node->next;
78                  index--;
79          }
80          return node->data;
81  }
82
83  struct ListNode* access(struct LinkedList* list, int index) {
84          struct ListNode* temp = list->head;
85          while (index > 0) {
86                  temp = temp->next;
87                  index--;
88          }
89          return temp;
```

```
90  }
91
92  int size(struct LinkedList *list) {
93          return (list->size);
94  }
95
96  void printLlBorder(struct LinkedList *list, int typ) {
97      int i;
98      for (i = 0; i < list->size; i++) {
99          int len = 0;
100         char tmp[256];
101         if (typ == INTEGER) {
102             len = sprintf(tmp, " %d ", *(int *) get(list, i));
103         } else if (typ == FLOATING) {
104             len = sprintf(tmp, " %f ", *(double *) get(list,
                ↪  i));
105         } else if (typ == STRING) {
106             len = sprintf(tmp, " %s ", *(char **) get(list, i));
107         }
108
109         int j;
110         printf("+");
111         for (j = 0; j < len; j++) {
112             printf("-");
113         }
114         printf("+  ");
115     }
116     printf("+------+\n");
117 }
118
119 void printIndexes(struct LinkedList *list, int typ) {
120     int i;
121     for (i = 0; i < list->size; i++) {
122         int len = 0;
123         char tmp[256];
124         if (typ == INTEGER) {
125             len = sprintf(tmp, "| %d |->", *(int *) get(list,
                ↪  i));
126         } else if (typ == FLOATING) {
127             len = sprintf(tmp, "| %f |->", *(double *) get(list,
                ↪  i));
128         } else if (typ == STRING) {
129             len = sprintf(tmp, "| %s |->", *(char **) get(list,
                ↪  i));
130         }
131         printf("%-*d", len, i);
```

```
132     }
133     printf("<- Index\n");
134 }
135 
136 void ll_simple_show(struct LinkedList *list, int typ) {
137     int i;
138     for (i = 0; i < list->size; i++) {
139         printf("[ ");
140         if (typ == INTEGER) {
141             printf("%d", *(int *) get(list, i));
142         } else if (typ == FLOATING) {
143             printf("%f", *(double *) get(list, i));
144         } else if (typ == STRING) {
145             printf("%s", *(char **) get(list, i));
146         }
147         printf(" ]");
148 
149         if (access(list, i) -> next != NULL) {
150             printf("%s", " -> ");
151         }
152     }
153     printf(" -> [ NULL ]\n");
154 }
155 
156 void ll_show(struct LinkedList *list, int typ) {
157     if (list->size == 0) {
158         printf("LinkedList is empty!");
159         return;
160     }
161 
162     if (list->size > 10) {
163         ll_simple_show(list, typ);
164         return;
165     }
166 
167     printLlBorder(list, typ);
168 
169     int i;
170     for (i = 0; i < list->size; i++) {
171         if (typ == INTEGER) {
172             printf("| %d |->", *(int *) get(list, i));
173         } else if (typ == FLOATING) {
174             printf("| %f |->", *(double *) get(list, i));
175         } else if (typ == STRING) {
176             printf("| %s |->", *(char **) get(list, i));
177         }
```

```
    }
    printf("| NULL |\n");
    printLlBorder(list, typ);
    printIndexes(list, typ);
}

void ll_show_int(struct LinkedList* list)
{
    ll_show(list, INTEGER);
}

void ll_show_string(struct LinkedList* list)
{
    ll_show(list, STRING);
}

void ll_show_float(struct LinkedList* list)
{
    ll_show(list, FLOATING);
}
```

8.10 queue.c

Millie, Sophie

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "utils.h"

struct Queue {
        int size;
        struct Node *front;
        struct Node *rear;
};

struct Node {
        struct Node *next;
        void *data;
};

struct Queue *initQueue() {
        struct Queue *q = (struct Queue*) malloc(sizeof(struct
        ↪  Queue));
        q->size = 0;
```

```
20          q->front = 0;
21          q->rear = 0;
22          return q;
23  }
24
25  int queue_size(struct Queue *queue) {
26          return queue->size;
27  }
28
29  void enqueue(struct Queue *q, void *data) {
30          struct Node *node = (struct Node*)malloc(sizeof(struct
            ↪  Node));
31          node->data = data;
32          node->next = NULL;
33          if (q->size == 0) {
34                  q->front = q->rear = node;
35                  q->size++;
36                  return;
37          }
38          q->rear->next = node;
39          q->rear = node;
40          q->size++;
41  }
42
43  void dequeue(struct Queue *q) {
44          if (q->size == 0) {
45                  return;
46          }
47          struct Node *node = q->front;
48          if (q->size == 1) {
49                  q->front = NULL;
50                  q->rear = NULL;
51                  q->size--;
52          }
53          else {
54                  q->front = q->front->next;
55                  q->size--;
56          }
57          free(node);
58  }
59
60  void *peek(struct Queue *q) {
61          if(q->size == 0) {
62                  return NULL;
63          }
64          return q->front->data;
```

```
65  }
66
67  int printBorder(struct Queue *q, int typ) {
68          int totalChars = 0;
69          int size = q->size;
70
71          int i;
72
73          totalChars += printf("%s", "+");
74          struct Node *curr = q->front;
75          for (i = 0; i < size; i++) {
76              int len = 0;
77              char tmp[1000];
78              if (typ == INTEGER) {
79                  len = sprintf(tmp, " %d ", *(int *) curr->data);
80              } else if (typ == FLOATING) {
81                  len = sprintf(tmp, " %f ", *(double *)
                        ↪  curr->data);
82              } else if (typ == STRING) {
83                  len = sprintf(tmp, " %s ", *(char **)
                        ↪  curr->data);
84              }
85
86              int j;
87              for (j = 0; j < len; j++) {
88                  totalChars += printf("-");
89              }
90              totalChars += printf("+");
91
92              curr = curr->next;
93          }
94          printf("\n");
95
96          return totalChars;
97  }
98
99  void printHeadTail(int len) {
100         int padding = len - strlen("Tail");
101         printf("%-*s%s\n", padding, "Head", "Tail");
102 }
103
104 void queue_simple_show(struct Queue *q, int typ) {
105     int i;
106     struct Node *curr = q->front;
107     printf("Head->");
108     for (i = 0; i < q->size; i++) {
```

```
109         printf("[ ");
110         if (typ == INTEGER) {
111             printf("%d", *(int *) curr->data);
112         } else if (typ == FLOATING) {
113             printf("%f", *(double *) curr->data);
114         } else if (typ == STRING) {
115             printf("%s", *(char **) curr->data);
116         }
117         curr = curr-> next;
118         printf(" ]");
119     }
120     printf("<-Tail\n");
121 }
122 
123 void queue_show(struct Queue *q, int typ) {
124         if (q->size == 0) {
125             printf("Queue is empty!");
126             return;
127         }
128 
129         if (q-> size > 10) {
130             queue_simple_show(q, typ);
131             return;
132         }
133 
134         // Print top border
135         printBorder(q, typ);
136 
137         int i;
138         int size = q->size;
139         struct Node *curr = q->front;
140         printf("%s", "|");
141         for (i = 0; i < size; i++) {
142             if (typ == INTEGER) {
143                 printf(" %d |", *(int *) curr->data);
144             } else if (typ == FLOATING) {
145                 printf(" %f |", *(double *) curr->data);
146             } else if (typ == STRING) {
147                 printf(" %s |", *(char **) curr->data);
148             }
149             curr = curr-> next;
150         }
151         printf("\n");
152 
153         int totalChars = printBorder(q, typ);
154         printHeadTail(totalChars);
```

```
}

void queue_show_int(struct Queue *q)
{
        queue_show(q, INTEGER);
}

void queue_show_float(struct Queue *q)
{
        queue_show(q, FLOATING);
}

void queue_show_string(struct Queue *q)
{
        queue_show(q, STRING);
}
```

8.11 quicksort.c

Kofi

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <stdbool.h>

// for swapping values in array
void swap(int *x, int *y)
{
    // intermediate place holder variable
    int z;

    // allocate value of x to z, set x to y and y to z
    z = *x;
    *x = *y;
    *y = z;
    return ;
}

// for swapping values in array
void fswap(double *x, double *y)
{
    // intermediate place holder variable
    double z;
```

```
25
26      // allocate value of x to z, set x to y and y to z
27      z = *x;
28      *x = *y;
29      *y = z;
30      return ;
31  }
32
33  void sswap(char **x, char **y)
34  {
35      char *temp = *x;
36      *x = *y;
37      *y = temp;
38  }
39
40  // prints out int list
41  void display(int a[], int length) {
42      int i;
43      printf("[ ");
44
45      for(i = 1;i<=length;i++) {
46        printf("%d ",a[i]);
47    }
48
49    printf("]\n");
50  }
51
52  // prints out float list
53  void fdisplay(double a[], int length) {
54      int i;
55      printf("[ ");
56
57      for(i = 1;i<=length;i++) {
58        printf("%f ",a[i]);
59    }
60
61    printf("]\n");
62  }
63
64  // prints out string list
65  void sdisplay(char* a[], int length) {
66      int i;
67      printf("[ ");
68
69      for(i = 1;i<=length;i++) {
70        printf("%s ",a[i]);
```

```
71    }
72
73    printf("]\n");
74  }
75
76  // Function that uses median of three partitioning
77  int median_of_3(int a[], int left, int right)
78  {
79      // get middle of array
80      int middle = (left + right)/2;
81
82      // rearrange arrange now with pivot known
83      if (a[middle] < a[left]) {
84          swap(&a[left],&a[middle]);
85      }
86      if (a[right] < a[left]) {
87          swap(&a[left],&a[right]);
88      }
89      if (a[right]< a[middle]) {
90          swap(&a[middle],&a[right]);
91      }
92
93      swap(&a[middle], &a[right - 1]);
94      return a[right - 1];
95  }
96
97  double fmedian_of_3(double a[], int left, int right)
98  {
99  // get middle of array
100     int middle = (left + right)/2;
101
102 // rearrange arrange now with pivot known
103     if (a[middle] < a[left]) {
104         fswap(&a[left],&a[middle]);
105     }
106     if (a[right] < a[left]) {
107         fswap(&a[left],&a[right]);
108     }
109     if (a[right] < a[middle]) {
110         fswap(&a[middle],&a[right]);
111     }
112
113     fswap(&a[middle], &a[right - 1]);
114     return a[right - 1];
115 }
116
```

```
117 char *smedian_of_3(char *a[], int left, int right)
118 {
119 // get middle of array
120     int middle = (left + right)/2;
121 
122 // rearrange arrange now with pivot known
123     if (strcmp(a[middle], a[left]) < 0) {
124         sswap(&a[left],&a[middle]);
125     }
126     if (strcmp(a[right], a[left]) < 0) {
127         sswap(&a[left],&a[right]);
128     }
129     if (strcmp(a[right], a[middle]) < 0) {
130         sswap(&a[middle],&a[right]);
131     }
132 
133     sswap(&a[middle], &a[right - 1]);
134     return a[right - 1];
135 }
136 
137 void quickSort(int a[], int left, int right)
138 {
139   if (left < right) {
140     int pivot = median_of_3(a,left,right);
141     if (left == right - 1) return;
142     int i = left;
143     int j = right - 1;
144     for ( ; ;) {
145         while(a[++i]<pivot) {}
146             while(pivot<a[--j]) {}
147                 if ( i < j) {
148                     swap(&a[i],&a[j]);
149                 } else {
150                     break ;
151                 }
152             }
153             swap(&a[i],& a[right -1]);
154             quickSort(a,left,i-1);
155             quickSort(a,i+1,right);
156         }
157 
158         return ;
159     }
160 
161 void fquickSort(double a[], int left, int right)
162 {
```

```
163   if (left < right) {
164     double pivot = fmedian_of_3(a,left,right);
165 
166     if (left == right - 1) return;
167     int i = left;
168     int j = right - 1;
169     for ( ; ;) {
170         while(a[++i]<pivot) {}
171             while(pivot<a[--j]) {}
172                 if ( i < j) {
173                     fswap(&a[i],&a[j]);
174                 }
175                 else {
176                     break ;
177                 }
178             }
179             fswap(&a[i], &a[right -1]);
180             fquickSort(a,left,i-1);
181             fquickSort(a,i+1,right);
182         }
183 
184         return ;
185     }
186 
187 void squickSort(char *a[], int left, int right)
188 {
189   if (left < right) {
190     char *pivot = smedian_of_3(a,left,right);
191     //printf("%s\n", pivot);}
192 
193     if (left == right - 1) return;
194     int i = left;
195     int j = right - 1;
196     for ( ; ;) {
197         while(strcmp(a[++i],pivot)<0) {}
198             while(strcmp(pivot,a[--j])<0) {}
199                 if ( i < j) {
200                     sswap(&a[i],&a[j]);
201                 }
202                 else {
203                     break ;
204                 }
205             }
206             sswap(&a[i], &a[right -1]);
207             squickSort(a,left,i-1);
208             squickSort(a,i+1,right);
```

```
209         }
210
211         return ;
212     }
213 // ======== show quicksorts =============
214
215 int partition(int a[], int left, int right, int length)
216 {
217   int x = a[right];
218   int i = left-1;
219   for(int j=left; j<right; j++)
220   {
221     if(a[j] <= x)
222     {
223       i++;
224       swap(&a[i], &a[j]);
225       if (a[i] != a[j] && i != j) {
226         printf("numbers swapped: %d,%d\n", a[j],a[i]);
227         printf("array after swap: ");
228         display(a, length);
229       }
230     }
231   }
232   swap(&a[i+1], &a[right]);
233   if (a[i+1] != a[right] && ((i+1) != right)) {
234     printf("numbers swapped: %d,%d\n", a[i+1],a[right]);
235     printf("array after swap: ");
236     display(a, length);
237   }
238   return i+1;
239 }
240
241 int fpartition(double a[], int left, int right, int length)
242 {
243   double x = a[right];
244   int i = left-1;
245   for(int j=left; j<right; j++)
246   {
247     if(a[j] <= x)
248     {
249       i++;
250       fswap(&a[i], &a[j]);
251       if (a[i] != a[j] && i != j) {
252         printf("numbers swapped: %f,%f\n", a[j],a[i]);
253         printf("array after swap: ");
254         fdisplay(a, length);
```

```
      }
    }
  }
  fswap(&a[i+1], &a[right]);
  if (a[i+1] != a[right] && ((i+1) != right)) {
    printf("numbers swapped: %f,%f\n", a[i+1],a[right]);
    printf("array after swap: ");
    fdisplay(a, length);
  }
  return i+1;
}

int spartition(char *a[], int left, int right, int length)
{
  char *x = a[right];
  int i = left-1;
  for(int j=left; j<right; j++)
  {
    if(strcmp(a[j], x) <= 0)
    {
      i++;
      sswap(&a[i], &a[j]);
      if (strcmp(a[i], a[j]) != 0 && i != j) {
        printf("strings swapped: %s,%s\n", a[j],a[i]);
        printf("array after swap: ");
        sdisplay(a, length);
      }
    }
  }
  sswap(&a[i+1], &a[right]);
  if (strcmp(a[i+1], a[right]) != 0 && ((i+1) != right)) {
    printf("strings swapped: %s,%s\n", a[i+1],a[right]);
    printf("array after swap: ");
    sdisplay(a, length);
  }
  return i+1;
}

int show_median_of_3(int a[], int left, int right, int length)
{
    // get middle of array
    int middle = (left + right)/2;

    ↪  printf("=========================================================================
    printf("At this step:\n");
    printf("current array: ");
```

```
300     display(a, length);
301
302     // rearrange arrange now with pivot known
303     if (a[middle] < a[left]) {
304         swap(&a[left],&a[middle]);
305         printf("pivot swapped: %d,%d\n", a[middle],a[left]);
306         printf("array after swap: ");
307         display(a, length);
308     }
309     if (a[right] < a[left]) {
310         swap(&a[left],&a[right]);
311         printf("pivot swapped: %d,%d\n", a[right],a[left]);
312         printf("array after swap: ");
313         display(a, length);
314     }
315     if (a[right]< a[middle]) {
316         swap(&a[middle],&a[right]);
317         printf("pivot swapped: %d,%d\n", a[right],a[middle]);
318         printf("array after swap: ");
319         display(a, length);
320     }
321
322     swap(&a[middle], &a[right - 1]);
323     if (a[middle] != a[right-1] && (middle != (right-1))) {
324       printf("numbers swapped: %d,%d\n", a[middle],a[right-1]);
325       printf("array after swap: ");
326       display(a, length);
327     }
328     printf("pivot is %d\n", a[right-1]);
329     return a[right - 1];
330 }
331
332 double show_fmedian_of_3(double a[], int left, int right, int
    ↪  length)
333 {
334 // get middle of array
335     int middle = (left + right)/2;
336
    ↪  printf("==========================================================================
337     printf("At this step:\n");
338     printf("current array: ");
339     fdisplay(a, length);
340
341 // rearrange arrange now with pivot known
342     if (a[middle] < a[left]) {
343         fswap(&a[left],&a[middle]);
```

```
344         printf("pivot swapped: %f,%f\n", a[middle],a[left]);
345         printf("array after swap: ");
346         fdisplay(a, length);
347     }
348     if (a[right] < a[left]) {
349         fswap(&a[left],&a[right]);
350         printf("pivot swapped: %f,%f\n", a[right],a[left]);
351         printf("array after swap: ");
352         fdisplay(a, length);
353     }
354     if (a[right] < a[middle]) {
355         fswap(&a[middle],&a[right]);
356         printf("pivot swapped: %f,%f\n", a[right],a[middle]);
357         printf("array after swap: ");
358         fdisplay(a, length);
359     }
360 
361     fswap(&a[middle], &a[right - 1]);
362     if (a[middle] != a[right-1] && (middle != (right-1))) {
363       printf("numbers swapped: %f,%f\n", a[middle],a[right-1]);
364       printf("array after swap: ");
365       fdisplay(a, length);
366     }
367     printf("pivot is %f\n", a[right-1]);
368     return a[right - 1];
369 }
370 
371 char *show_smedian_of_3(char *a[], int left, int right, int
    ↪  length)
372 {
373 // get middle of array
374     int middle = (left + right)/2;
375 
    ↪  printf("=============================================================================
376     printf("At this step:\n");
377     printf("current array: ");
378     sdisplay(a, length);
379 
380 // rearrange arrange now with pivot known
381     if (strcmp(a[middle], a[left]) < 0) {
382         sswap(&a[left],&a[middle]);
383         printf("pivot swapped: %s,%s\n", a[middle],a[left]);
384         printf("array after swap: ");
385         sdisplay(a, length);
386     }
387     if (strcmp(a[right], a[left]) < 0) {
```

```
388         sswap(&a[left],&a[right]);
389         printf("pivot swapped: %s,%s\n", a[right],a[left]);
390         printf("array after swap: ");
391         sdisplay(a, length);
392     }
393     if (strcmp(a[right], a[middle]) < 0) {
394         sswap(&a[middle],&a[right]);
395         printf("pivot swapped: %s,%s\n", a[right],a[middle]);
396         printf("array after swap: ");
397         sdisplay(a, length);
398     }
399 
400     sswap(&a[middle], &a[right - 1]);
401     if (strcmp(a[middle], a[right-1]) != 0 && (middle !=
    ↪  (right-1))) {
402       printf("strings swapped: %s,%s\n", a[middle],a[right-1]);
403       printf("array after swap: ");
404       sdisplay(a, length);
405     }
406     printf("pivot is %s\n", a[right-1]);
407     return a[right - 1];
408 }
409 
410 void showQuickSort(int a[], int left, int right, int length)
411 {
412   if(left < right)
413   {
414     show_median_of_3(a, left, right, length);
415 
416     int q = partition(a, left, right, length);
417     showQuickSort(a, left, q-1, length);
418     showQuickSort(a, q+1, right, length);
419   }
420 }
421 
422 void showfQuickSort(double a[], int left, int right, int length)
423 {
424   if(left < right)
425   {
426     show_fmedian_of_3(a, left, right, length);
427 
428     int q = fpartition(a, left, right, length);
429     showfQuickSort(a, left, q-1, length);
430     showfQuickSort(a, q+1, right, length);
431   }
432 }
```

```
433
434 void showsQuickSort(char *a[], int left, int right, int length)
435 {
436   if(left < right)
437   {
438     show_smedian_of_3(a, left, right, length);
439
440     int q = spartition(a, left, right, length);
441     showsQuickSort(a, left, q-1, length);
442     showsQuickSort(a, q+1, right, length);
443   }
444 }
445
446 void cShowQuickSort(int a[], int length) {
447   int right = length;
448   length = length;
449   showQuickSort(a, 1, right, length);
450   printf("%s\n",
      ↪  "=============================================================================
451   printf("%s", "QuickSort complete! Final Result: ");
452   display(a, length);
453 }
454
455 void cShowfQuickSort(double a[], int length) {
456   int right = length;
457   length = length;
458   showfQuickSort(a, 1, right, length);
459   printf("%s\n",
      ↪  "=============================================================================
460   printf("%s", "QuickSort complete! Final Result: ");
461   fdisplay(a, length);
462 }
463
464 void cShowsQuickSort(char *a[], int length) {
465   int right = length;
466   length = length;
467   showsQuickSort(a, 1, right, length);
468   printf("%s\n",
      ↪  "=============================================================================
469   printf("%s", "QuickSort complete! Final Result: ");
470   sdisplay(a, length);
471 }
472
473 void cQuickSort(int a[], int length) {
474   length = length;
475   quickSort(a, 1, length);
```

```
}

void cQuickfSort(double a[], int length) {
  length = length;
  fquickSort(a, 1, length);
}

void cQuicksSort(char *a[], int length) {
  length = length;
  squickSort(a, 1, length);
}

// int main()
// {
//     int a[] = {10, 35, 25, 56, 2002, 100, 90, 86, 40, 50, 7};
//     cShowQuickSort(a, 11);
//     cQuickSort(a, 11);

//     int b[] = {85,331,234,46,4,3,22,89,7,12, 33,
↪  44,55,66,77};
//     cShowQuickSort(b, 15);
//     cQuickSort(b, 15);

//     double c[] = { 3.4, 56.2, 5.3, 6.0, 2.6, 566.7, 778.3};
//     cShowfQuickSort(c, 7);
//     cQuickfSort(c, 7);

//     double d[] = { 55.4, 36.2, 23.3, 16.0, 52.6, 26.7, 78.3};
//     cShowfQuickSort(d, 7);
//     cQuickfSort(d, 7);

//     int f[] = {10, 100, 30, 90, 40, 50, 70};
//     cShowQuickSort(f, 7);
//     cQuickSort(f, 7);

//     int g[] = {4,6,3,2,1,9,7};
//     cShowQuickSort(g, 7);
//     cQuickSort(g, 7);
//     char *a[] = {"abx", "cdf", "eadfaer", "baweaw", "dwaw"};
//     cQuicksSort(a, 4);
//     sswap(&a[0], &a[1]);
//     sdisplay(a, 5);

// }
```

8.12 stack.c

Millie, Sophie

```c
#include <stdio.h>
#include <stdlib.h>
#include "utils.h"

struct Stack {
        int size;
        struct Node *top;
};

struct Node {
        struct Node *next;
        void *data;
};

struct Stack* initStack(unsigned capacity) {
        struct Stack* stack = (struct Stack*)
        ↪  malloc(sizeof(struct Stack));
        stack->size = 0;
        stack->top = NULL;
        return stack;
}

int stack_size(struct Stack* stack) {
        return stack->size;
}

void push(struct Stack *stack, void *data) {
        struct Node* node = (struct Node*)malloc(sizeof(struct
        ↪  Node));
        node->data = data;
        node->next = stack->top;
        stack->top = node;
        stack->size++;
}

void pop(struct Stack *stack) {
        if (stack->size == 0) {
                return;
        }

        struct Node* node = stack->top;
        if (stack->size == 1) {
```

```
41                  stack->top = NULL;
42                  stack->size--;
43          } else {
44                  stack->top = stack->top->next;
45                  stack->size--;
46          }
47          free(node);
48  }
49
50  void *top(struct Stack *stack) {
51          if (stack->size == 0) {
52                  return NULL;
53          }
54          return stack->top->data;
55  }
56
57  void printLine(int size) {
58          printf("+");
59          int i;
60          for (i = 0; i < size; i++) {
61              printf("-");
62          }
63          printf("+");
64  }
65
66  void stack_show(struct Stack *stack, int typ) {
67          if (stack->size == 0) {
68              printf("Stack is empty!");
69              return;
70          }
71
72          int INTEGER_MAX = 13;
73          int FLOATING_MAX = 20;
74          int STRING_MAX = 48;
75
76          if (typ == INTEGER) {
77              printLine(INTEGER_MAX);
78          } else if (typ == FLOATING) {
79              printLine(FLOATING_MAX);
80          } else if (typ == STRING) {
81              printLine(STRING_MAX);
82          }
83          printf(" <- Top\n");
84
85          int i;
86          int size = stack->size;
```

```
87          struct Node *curr = stack->top;
88          for (i = 0; i < size; i++) {
89              if (typ == INTEGER) {
90                  printf("| %-*d |\n", INTEGER_MAX - 2, *(int *)
                    ↪  curr->data);
91                  printLine(INTEGER_MAX);
92              } else if (typ == FLOATING) {
93                  printf("| %-*f |\n", FLOATING_MAX - 2, *(double
                    ↪  *) curr->data);
94                  printLine(FLOATING_MAX);
95              } else if (typ == STRING) {
96                  printf("| %-*s |\n", STRING_MAX - 2, *(char **)
                    ↪  curr->data);
97                  printLine(STRING_MAX);
98              }
99              printf("\n");
100             curr = curr->next;
101         }
102 }
103
104 void stack_show_int(struct Stack* stack)
105 {
106         stack_show(stack, INTEGER);
107 }
108
109 void stack_show_float(struct Stack* stack)
110 {
111         stack_show(stack, FLOATING);
112 }
113
114 void stack_show_string(struct Stack* stack)
115 {
116         stack_show(stack, STRING);
117 }
```

8.13 utils.h

Sophie

```
1 #ifndef type_consts
2 #define type_consts
3   #define INTEGER 0
4   #define FLOATING 1
5   #define STRING 2
```

```
#endif
```

8.14 testall.sh

Josh, Kofi, Millie, Sophie

```
#!/bin/sh

# Regression testing script for Strux
# Step through a list of files
#  Compile, run, and check the output of each expected-to-work test
#  Compile and check the error of each expected-to-fail test

# Path to the LLVM interpreter
LLI="lli"
#LLI="/usr/local/opt/llvm/bin/lli"
# Path to the LLVM compiler
LLC="llc"
#LLC="/usr/local/opt/llvm@3.7/bin/llc-3.7"

# Path to the C compiler
CC="clang"

# Path to the strux compiler.  Usually "./strux.native"
# Try "_build/strux.native" if ocamlbuild was unable to create a symbolic link.
STRUX="./strux.native"
#STRUX="_build/strux.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] [.strux files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
```

```
38  }
39
40  SignalError() {
41      if [ $error -eq 0 ] ; then
42          echo "FAILED"
43          error=1
44      fi
45      echo "  $1"
46  }
47
48  # Compare <outfile> <reffile> <difffile>
49  # Compares the outfile with reffile.  Differences, if any,
    ↪  written to difffile
50  Compare() {
51      generatedfiles="$generatedfiles $3"
52      echo diff -b $1 $2 ">" $3 1>&2
53      diff -b "$1" "$2" > "$3" 2>&1 || {
54          SignalError "$1 differs"
55          echo "FAILED $1 differs from $2" 1>&2
56      }
57  }
58
59  # Run <args>
60  # Report the command, run it, and report any errors
61  Run() {
62      echo $* 1>&2
63      eval $* || {
64          SignalError "$1 failed on $*"
65          return 1
66      }
67  }
68
69  # RunFail <args>
70  # Report the command, run it, and expect an error
71  RunFail() {
72      echo $* 1>&2
73      eval $* && {
74          SignalError "failed: $* did not report an error"
75          return 1
76      }
77      return 0
78  }
79
80  Check() {
81      error=0
82      basename=`echo $1 | sed 's/.*\\///
```

```
                                s/.strux//'`
    reffile=`echo $1 | sed 's/.strux$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

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

    generatedfiles=""

    # generatedfiles="$generatedfiles ${basename}.ll
    ↪  ${basename}.out" &&
    generatedfiles="$generatedfiles ${basename}.ll ${basename}.s
    ↪  ${basename}.exe ${basename}.out" &&
    Run "$STRUX" "<" "$1" ">" "${basename}.ll" &&
    Run "$LLC" "${basename}.ll" ">" "${basename}.s" &&
    Run "$CC" "-o" "${basename}.exe" "${basename}.s" "queue.bc"
    ↪  "bstree.bc" "linkedlist.bc" "stack.bc" "quicksort.bc" &&
    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/.strux//'`
    reffile=`echo $1 | sed 's/.strux$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
```

```
126     echo "###### Testing $basename" 1>&2
127
128     generatedfiles=""
129
130     generatedfiles="$generatedfiles ${basename}.err
        ↪  ${basename}.diff" &&
131     RunFail "$STRUX" "<" $1 "2>" "${basename}.err" ">>"
        ↪  $globallog &&
132     Compare ${basename}.err ${reffile}.err ${basename}.diff
133
134     # Report the status and clean up the generated files
135
136     if [ $error -eq 0 ] ; then
137         if [ $keep -eq 0 ] ; then
138             rm -f $generatedfiles
139         fi
140         echo "OK"
141         echo "###### SUCCESS" 1>&2
142     else
143         echo "###### FAILED" 1>&2
144         globalerror=$error
145     fi
146 }
147
148 while getopts kdpsh c; do
149     case $c in
150         k) # Keep intermediate files
151             keep=1
152             ;;
153         h) # Help
154             Usage
155             ;;
156     esac
157 done
158
159 shift `expr $OPTIND - 1`
160
161 LLIFail() {
162   echo "Could not find the LLVM interpreter \"$LLI\"."
163   echo "Check your LLVM installation and/or modify the LLI
      ↪  variable in testall.sh"
164   exit 1
165 }
166
167 which "$LLI" >> $globallog || LLIFail
168
```

```
169 if [ ! -f linkedlist.bc ]
170     then
171         echo "Could not find linkedlist.bc"
172         echo "Try \"./linkStrux.sh\""
173         exit 1
174 fi
175
176 if [ ! -f queue.bc ]
177     then
178         echo "Could not find queue.bc"
179         echo "Try \"./linkStrux.sh\""
180         exit 1
181 fi
182
183 if [ ! -f bstree.bc ]
184     then
185         echo "Could not find bstree.bc"
186         echo "Try \"./linkStrux.sh\""
187         exit 1
188 fi
189
190 if [ ! -f stack.bc ]
191     then
192         echo "Could not find stack.bc"
193         echo "Try \"./linkStrux.sh\""
194         exit 1
195 fi
196
197 if [ $# -ge 1 ]
198 then
199     files=$@
200 else
201     files="tests/test-*.strux tests/fail-*.strux"
202 fi
203
204 for file in $files
205 do
206     case $file in
207         *test-*)
208             Check $file 2>> $globallog
209             ;;
210         *fail-*)
211             CheckFail $file 2>> $globallog
212             ;;
213         *)
214             echo "unknown file type $file"
```

```
            globalerror=1
            ;;
    esac
done

exit $globalerror
```

8.15 test-gcd.ll

As referenced in section 6.2, Integration Testing.

```
; ModuleID = 'Strux'

%struct.Queue = type { i32, %struct.Node*, %struct.Node* }
%struct.Node = type { %struct.Node*, i8* }
%struct.LinkedList = type { %struct.ListNode*, i32 }
%struct.ListNode = type { i8*, %struct.ListNode* }
%struct.Stack = type { i32, %struct.Node.0* }
%struct.Node.0 = type { %struct.Node.0*, i8* }
%struct.BSTree = type { %struct.BSTreeNode* }
%struct.BSTreeNode = type { i8*, %struct.BSTreeNode*,
↪  %struct.BSTreeNode*, %struct.BSTreeNode* }

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"

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

declare %struct.Queue* @initQueue()

declare void @enqueue(%struct.Queue*, i8*)

declare void @dequeue(%struct.Queue*)

declare i8* @peek(%struct.Queue*)

declare i32 @queue_size(%struct.Queue*)

declare void @queue_show_int(%struct.Queue*)

declare void @queue_show_float(%struct.Queue*)

declare void @queue_show_string(%struct.Queue*)

declare %struct.LinkedList* @initList()

```

```
34 declare void @add(%struct.LinkedList*, i8*)
35
36 declare void @delete(%struct.LinkedList*, i32)
37
38 declare i8* @get(%struct.LinkedList*, i32)
39
40 declare i32 @size(%struct.LinkedList*)
41
42 declare void @ll_show_int(%struct.LinkedList*)
43
44 declare void @ll_show_float(%struct.LinkedList*)
45
46 declare void @ll_show_string(%struct.LinkedList*)
47
48 declare %struct.Stack* @initStack()
49
50 declare void @push(%struct.Stack*, i8*)
51
52 declare void @pop(%struct.Stack*)
53
54 declare i8* @top(%struct.Stack*)
55
56 declare i32 @stack_size(%struct.Stack*)
57
58 declare void @stack_show_int(%struct.Stack*)
59
60 declare void @stack_show_float(%struct.Stack*)
61
62 declare void @stack_show_string(%struct.Stack*)
63
64 declare i32* @cQuickSort(i32*, i32)
65
66 declare void @cShowQuickSort(i32*, i32)
67
68 declare double* @cQuickfSort(double*, i32)
69
70 declare void @cShowfQuickSort(double*, i32)
71
72 declare i8** @cQuicksSort(i8**, i32)
73
74 declare void @cShowsQuickSort(i8**, i32)
75
76 declare %struct.BSTree* @initBSTree()
77
78 declare void @addIntToTree(%struct.BSTree*, i8*)
79
```

```
80  declare void @addNumToTree(%struct.BSTree*, i8*)
81
82  declare void @deleteIntFromTree(%struct.BSTree*, i32)
83
84  declare void @deleteNumFromTree(%struct.BSTree*, double)
85
86  declare i1 @treeContainsInt(%struct.BSTree*, i32)
87
88  declare i1 @treeContainsFloat(%struct.BSTree*, double)
89
90  declare void @showIntTree(%struct.BSTree*)
91
92  declare void @showNumTree(%struct.BSTree*)
93
94  declare i32 @printbig(i32)
95
96  define i32 @gcd(i32 %n1, i32 %n2) {
97  entry:
98    %n11 = alloca i32
99    store i32 %n1, i32* %n11
100   %n22 = alloca i32
101   store i32 %n2, i32* %n22
102   %gcd = alloca i32
103   store i32 1, i32* %gcd
104   %i = alloca i32
105   store i32 1, i32* %i
106   br label %while
107
108 while:                                            ; preds =
    ↪  %merge, %entry
109   %i15 = load i32, i32* %i
110   %n116 = load i32, i32* %n11
111   %tmp17 = icmp sle i32 %i15, %n116
112   %i18 = load i32, i32* %i
113   %n219 = load i32, i32* %n22
114   %tmp20 = icmp sle i32 %i18, %n219
115   %tmp21 = and i1 %tmp17, %tmp20
116   br i1 %tmp21, label %while_body, label %merge22
117
118 while_body:                                       ; preds =
    ↪  %while
119   %n13 = load i32, i32* %n11
120   %i4 = load i32, i32* %i
121   %tmp = srem i32 %n13, %i4
122   %tmp5 = icmp eq i32 %tmp, 0
123   %n26 = load i32, i32* %n22
```

```
  %i7 = load i32, i32* %i
  %tmp8 = srem i32 %n26, %i7
  %tmp9 = icmp eq i32 %tmp8, 0
  %tmp10 = and i1 %tmp5, %tmp9
  br i1 %tmp10, label %then, label %else

merge:                                            ; preds =
↪  %else, %then
  %i12 = load i32, i32* %i
  %i13 = load i32, i32* %i
  %tmp14 = add i32 %i13, 1
  store i32 %tmp14, i32* %i
  br label %while

then:                                             ; preds =
↪  %while_body
  %i11 = load i32, i32* %i
  store i32 %i11, i32* %gcd
  br label %merge

else:                                             ; preds =
↪  %while_body
  br label %merge

merge22:                                          ; preds =
↪  %while
  %gcd23 = load i32, i32* %gcd
  ret i32 %gcd23
}

define i32 @main() {
entry:
  %gcd_result = call i32 @gcd(i32 81, i32 153)
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr
   ↪  inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32 0), i32
   ↪  %gcd_result)
  ret i32 0
}
```

8.16 git log

```
commit 834aba2192df3838f571d2db2f773064d29853e7
Author: sophstad <srs2231@columbia.edu>
Date:   Wed Dec 20 21:17:34 2017 +0000
```

```
    Add more fail tests

commit 8dfaeb5d3c3da1dd2a64f2f5f81b53c731a54993
Author: sophstad <srs2231@columbia.edu>
Date:   Wed Dec 20 21:09:23 2017 +0000

    Remove clang incompatible tests

commit a414f6774ff94ece3afa17fdb32161f24f658be2
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Wed Dec 20 15:41:03 2017 -0500

    final touches to bstree

commit 4d1391bfb8ec03795e7d9adf6ff11e4a4db5b788
Author: sophstad <srs2231@columbia.edu>
Date:   Wed Dec 20 18:56:05 2017 +0000

    Remove reduce/reduce conflict in parser

commit 06f426b76e6a9842fd190237941d53e2115ac019
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Wed Dec 20 13:23:56 2017 -0500

    Update Makefile to create tarball

commit 0fbf2ec875a5430311ec1585e5e8bfb9ac27a5bc
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Wed Dec 20 11:05:15 2017 -0500

    Update strux.ml

commit 3f9c220dca8c545e8a8fec65e73deeaa46a64dac
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Tue Dec 19 21:40:01 2017 -0500

    Update README.md

commit 01bde1c864323b89d7c1a4f03281e266e793c98b
Author: sophstad <srs2231@columbia.edu>
Date:   Wed Dec 20 01:23:18 2017 +0000

    Delete bad test

commit bb29d187f576391455a7c91cc0ba2abda7b82b09
```

```
Merge: b575c58 c80ff79
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Tue Dec 19 19:00:46 2017 -0500

    Merge pull request #39 from sophstad/llreorg

    few changes here and there

commit c80ff79327e77234da2e9afbca91b0642794b038
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Tue Dec 19 18:49:18 2017 -0500

    few changes here and there

commit b575c58c08052c0e54f2b502b8e3ab15ea6caf1c
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 19 15:25:24 2017 -0500

    adding tests

commit ae81717a8397afef86eb6c5167d8f92ef87f430d
Author: sophstad <srs2231@columbia.edu>
Date:   Tue Dec 19 05:37:49 2017 +0000

    Remove strux.sh file

commit e9ea07be8d9fd13a1c32aa20c525b3c833cf095b
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Tue Dec 19 00:31:47 2017 -0500

    Update scanner.mll

commit 6bf03bc933b879efdb711b68dd96182009946da2
Author: sophstad <srs2231@columbia.edu>
Date:   Tue Dec 19 04:22:59 2017 +0000

    Remove all traces of globals; clean up code

commit 9645b66e7bcc7c468b341205c5b70c593f04c6d3
Merge: 7452b1d 5a5310f
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Mon Dec 18 23:15:12 2017 -0500

    Merge pull request #38 from sophstad/ss/reorganize-c

    Reorganize Directory
```

```
commit 5a5310f13866ade0262d6664715a8d934c73c8f9
Author: sophstad <srs2231@columbia.edu>
Date:   Tue Dec 19 04:06:06 2017 +0000

    Move c files; make calls linkStrux.sh

commit 7452b1db0642131318d3e0c36012a87799ab2cf0
Merge: 198d8a0 d9dc6a1
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Mon Dec 18 13:46:04 2017 -0500

    Merge pull request #37 from sophstad/demo-update

    updating demo

commit d9dc6a14e49b9cdbffad3f5e94231aa54dd09dc7
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Mon Dec 18 13:37:38 2017 -0500

    updating demo

commit 198d8a08a863ce098aa657f45558bbbdcebcc072
Author: sophstad <srs2231@columbia.edu>
Date:   Mon Dec 18 17:32:56 2017 +0000

    generateModules.sh -> linkStrux.sh

commit 3d81693bc13cd13aef3fd107eef8a4b52cd13e8b
Merge: 697e3c8 94f63b5
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Mon Dec 18 12:21:37 2017 -0500

    Merge pull request #36 from sophstad/ss/print-object-call

    Allow object calls inside of print

commit 697e3c80aa72ea1df29f73c8e6686d440cff4448
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Mon Dec 18 12:14:09 2017 -0500

    fixed small bug that wouldn't let us delete the root of a tree when it was the very last

commit 94f63b5b81b156a7d14cbae99f9a2dd1e16eb97f
Author: sophstad <srs2231@columbia.edu>
Date:   Fri Dec 15 04:18:28 2017 +0000
```

```
    Allow object calls inside of print

commit a7f6fe8f56557d20290f279999eaf645e3bda09b
Merge: 794443d 1ddcdc9
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Mon Dec 18 11:55:56 2017 -0500

    Merge pull request #35 from sophstad/josh/tree2

    Adding Demo and Contains

commit 1ddcdc9a608807c33a96ef2c9560969f99f254dc
Author: sophstad <srs2231@columbia.edu>
Date:   Mon Dec 18 16:53:11 2017 +0000

    .contains() returns bool

commit 0b352627c13f6d1be314c4bca47cdca86de3463d
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Mon Dec 18 02:06:59 2017 -0500

    adding demo test and updating quicksort

commit 59944e43ad9a50f45e6b6b74effc2c0d05d83c14
Merge: 83c7243 794443d
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Mon Dec 18 01:05:20 2017 -0500

    merge conflicts

commit 83c7243a5ace064da6a47ccd7e33d978ffa211b5
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Mon Dec 18 01:02:06 2017 -0500

    minor changes

commit 794443dc413eda8b1185f46a56b202de793d4af8
Merge: 84581d8 f95f21b
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Sun Dec 17 16:20:32 2017 -0500

    Merge pull request #34 from sophstad/quicksort-str

    QuickSort Now Works on Strings!
```

```
commit f95f21b48f5caac4e5b9d3d8f495c0000485f180
Merge: b8b4e49 84581d8
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Sun Dec 17 16:11:21 2017 -0500

    Merge remote-tracking branch 'origin/master' into quicksort-str

commit b8b4e499e8251bdcbd4576694a9e2395a737ae63
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Sun Dec 17 16:11:02 2017 -0500

    quicksort fully working on strings and showquicksort as well

commit 4538a41d897c3695c14a17f55d8431e754810dcc
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Sun Dec 17 15:51:12 2017 -0500

    .quickSort() for strings now works

commit 84581d81569a787c1925cb9ec1e45162fb3a98af
Author: sophstad <srs2231@columbia.edu>
Date:   Sun Dec 17 05:42:26 2017 +0000

    Fix shift/reduce conflict; remove unused code

commit 37bc34b97642d8f2c8e32e072566f6c8cc281923
Merge: 2359172 8a9ccd1
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Sat Dec 16 17:33:55 2017 -0500

    Merge pull request #33 from sophstad/ss/simplify-show

    Queue and LinkedList simplified printing

commit 8a9ccd18b728dea357f576d91c5797c2134b6d73
Author: sophstad <srs2231@columbia.edu>
Date:   Sat Dec 16 21:45:42 2017 +0000

    Queue and linkedlist simplified printing

commit 235917210c5380d01a90acf21e6ab87f37336883
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Dec 16 16:34:30 2017 -0500

    clean up
```

```
commit 24995488f339df01f65c51bfa0d1b1d18d4b42c3
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Sat Dec 16 16:32:23 2017 -0500

    only printing up to the height of the tree and updating tests

commit 861d20d50c156b87f3f6bfae8078e9a78731ec23
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Dec 16 16:20:57 2017 -0500

    tests

commit 6fdf1b7518f6a0caee30d5b0feeb64ecfdf9f3ac
Merge: 2a68cb2 e1df61c
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Dec 16 15:53:07 2017 -0500

    Merge branch 'master' of github.com:sophstad/strux

commit 2a68cb246ac0f0f8629365d39f51613a5e8829dc
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Dec 16 15:52:31 2017 -0500

    updated test

commit e1df61c896f42fb11f7240508b8d3d7737dd03fc
Merge: 4c12ffd 2704ce1
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Sat Dec 16 15:34:20 2017 -0500

    Merge pull request #32 from sophstad/ss/show-empty

    Don't show empty structures, add tests

commit 2704ce122ec817faec61159e50c9ae98b5e69b61
Author: sophstad <srs2231@columbia.edu>
Date:   Sat Dec 16 20:27:22 2017 +0000

    Add check and test for empty structures

commit e506c8ac33a721f2fb79f0c89c293f624b45be5e
Author: sophstad <srs2231@columbia.edu>
Date:   Sat Dec 16 20:24:02 2017 +0000

    Add gcd tests
```

```
commit c04fda3e2bd3cee9915f132a5392c0c0acdea96d
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Sat Dec 16 15:20:14 2017 -0500

    can now init a tree with a list of values

commit 4c12ffdc39f9b66f5f0df36cc9d4918643f190df
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Dec 16 15:04:14 2017 -0500

    modified test a bit

commit 5cd8c7750c55dd82ecb516e7eaaf0678ce3799e0
Merge: 7d26214 0e83bc6
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Fri Dec 15 18:28:25 2017 -0500

    Merge remote-tracking branch 'origin/master' into josh/tree2

commit 7d26214e37a9d6e035e064905ed5bacc480b3b93
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Fri Dec 15 18:27:20 2017 -0500

    added contains and prettified code a bit

commit d3ac0869623387753486fa9ec00a3dfe7bb0d1f9
Author: sophstad <srs2231@columbia.edu>
Date:   Sun Dec 10 19:59:14 2017 +0000

    Improve unop testing

commit b4699f24248d8447cd2463b0d1904c01faaa0cd8
Author: sophstad <srs2231@columbia.edu>
Date:   Sun Dec 10 19:50:26 2017 +0000

    Improve add test

commit 0e83bc691915e229207e52cd0d19f6e04f3b4628
Merge: 8ec06e7 84e3ea8
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Fri Dec 15 12:57:12 2017 -0500

    Merge pull request #31 from sophstad/ss/array-improvements

    Array Improvements
```

```
commit 84e3ea8da625d9928523e66e326cb295443b8fed
Author: sophstad <srs2231@columbia.edu>
Date:   Fri Dec 15 05:58:38 2017 +0000

    Check array length at instantiation

commit 1fffa44e69ba212c82bf37a8801ec977b9736fee
Author: sophstad <srs2231@columbia.edu>
Date:   Fri Dec 15 05:11:37 2017 +0000

    Add limited array out of bounds handling

commit 8ec06e7a994bdfa2a3c83bcd85a3217d1aacd66a
Merge: a458191 eab54cf
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Thu Dec 14 22:47:49 2017 -0500

    Merge pull request #30 from sophstad/ss/prettify-linkedlist

    Prettify linked list

commit eab54cf6e85c72031a97f519a21c3dba87fd6d42
Author: sophstad <srs2231@columbia.edu>
Date:   Fri Dec 15 03:28:21 2017 +0000

    Prettify linked list

commit a4581912d7741c5999f0a18b964cf914d1d2acad
Merge: 34ee230 7edd816
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Thu Dec 14 21:59:48 2017 -0500

    Merge pull request #29 from sophstad/josh/tree

    Strux BSTree

commit 7edd8162044fde6d81e7a58941248e0417cc7261
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Thu Dec 14 20:18:15 2017 -0500

    Remove duplicate token declaration

commit 80ab7a828c6b12157bc37ad09f27dbf42b7c7461
Merge: c09d362 34ee230
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Thu Dec 14 20:06:35 2017 -0500
```

```
    merge conflicts 2.0

commit c09d362d3f8d3cd7c7496e8408c1b5271d08479c
Merge: 74851e3 70f6e4c
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Thu Dec 14 20:01:51 2017 -0500

    fixing merge conflicts

commit 74851e30b0c8fdf2e26eca12317ca576f05cb37a
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Thu Dec 14 19:39:16 2017 -0500

    finished tree implementation

commit 34ee230296053b3523c9cb3a92802e665845a1c8
Merge: a858321 1eb5124
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Thu Dec 14 16:53:03 2017 -0500

    Merge pull request #28 from sophstad/ss/quicksort-hood

    Unify quicksort functions

commit 1eb512465e5410bf0e3cb6e532498e720ea1dc18
Author: sophstad <srs2231@columbia.edu>
Date:   Thu Dec 14 21:49:45 2017 +0000

    Unify quicksort functions

commit a858321c687852af2ea20d17a977d5d06f3b2196
Merge: 70f6e4c 4d82fc3
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Thu Dec 14 15:48:39 2017 -0500

    Merge pull request #27 from sophstad/quicksort-2

    Quicksort works now, shifted indices to match ocaml array

commit 4d82fc386e9ff4ea79adc137a06939ea95970a51
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Thu Dec 14 15:46:21 2017 -0500

    fixing quicksort bug
```

```
commit bef318e0a76e6b962241b3b8ebc33d39ee7c5bb0
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Thu Dec 14 15:37:37 2017 -0500

    integrating delete() and making tree work with nums

commit 4a0463929335be0e4821ed6ee907fa535b9c9426
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 14 15:32:53 2017 -0500

    fix let's see 2

commit 54c7b0083dff7ea07fe7fca0a0d5909dbc61b260
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 14 15:27:32 2017 -0500

    fix let's see

commit d0c0ad4eb881dbe9b96070b6517e8eff1ce6afa9
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 14 15:24:06 2017 -0500

    trial to see if this fixes ocaml array

commit 4eec4f045bd1ce8d410ec5ea6fd7f6f585510211
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 14 14:04:56 2017 -0500

    forgot to comment out main :(

commit 44a0141dc3c650dae677ef29bd43eca2dac27789
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 14 14:02:44 2017 -0500

    changing type to double in quicksort c

commit 1742af0f115a1f982143fcb4b2af25124c10603b
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 14 13:55:16 2017 -0500

    adding size to arrays on initialization in tests

commit 508e14754bcdb16e77dbe4d24003cded7f3e66d4
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 14 13:42:30 2017 -0500
```

```
    changing quicksort tests

commit 8f8319de0632401eb6e8d5d01663db2786d13bd8
Merge: dcec61c 70f6e4c
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Thu Dec 14 13:40:56 2017 -0500

    Merge pull request #26 from sophstad/master

    quick sort-2

commit 70f6e4ca0bc2cc5f912268c2088f6b9bbf3eef02
Merge: f329d46 dcec61c
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Thu Dec 14 13:34:14 2017 -0500

    Merge branch 'quicksort-2' into master

commit 3912cb9ab6a903082ad160ceaea013051c1a48d0
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Wed Dec 13 15:12:05 2017 -0500

    add and init are working now for trees

commit f329d46a2c263c8bb8241d39a884a43940d1a02e
Merge: a49339e 760515a
Author: Millie Yang <millie.yang@columbia.edu>
Date:   Wed Dec 13 14:40:49 2017 -0500

    Merge pull request #25 from sophstad/ss/array-length

    Array Updates

commit ae32dbd9b5c964c9ca0796a179060b18fd930420
Merge: 2db4f9c a49339e
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Wed Dec 13 14:26:14 2017 -0500

    beginning to implement tree into codegen

commit 760515ac9071d7efe1ae7c9b32cc450770af3e75
Author: sophstad <srs2231@columbia.edu>
Date:   Tue Dec 12 21:57:01 2017 +0000

    Show array works
```

```
commit 18abc710acaa333336a5f888ea1f36833fe53f9a
Author: sophstad <srs2231@columbia.edu>
Date:   Tue Dec 12 07:35:06 2017 +0000

    Array len working; size argument removed for quicksort

commit a49339ee14cc059670233df7f0b8c73a98e21f41
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 12 01:17:28 2017 -0500

    tests for inputting array in queue, linkedlist, and stack

commit dcec61c3e9db3035aec83881cea65dc699b7b508
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 12 00:48:42 2017 -0500

    comment so that it's easier to read

commit 722839818d9eb29bfc015d6899d7e4a88464fb43
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 12 00:45:55 2017 -0500

    all tests for quicksort passes

commit 7866b8899aeff8fde1d649593132df037b0d6133
Merge: f5dc16a aade2d6
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 12 00:20:23 2017 -0500

    rebase w master

commit f5dc16a6e4f0834d704d71d3c8fe77f1a042083d
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 18:25:44 2017 -0500

    changing return type of show functions

commit 81dbb553395bbe789e4a7058965ac5d66b60c856
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 18:11:44 2017 -0500

    trying to change actual params for quicksorts

commit 7b7f3bfd4b8967acaa930326c30cfd220d829002
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 18:05:12 2017 -0500
```

```
    adding tests for quicksorts

commit 5f26ee25dc383b511d9d7990db5757b63afe7007
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 17:38:12 2017 -0500

    trying quicksort show for int

commit a8d27b7d4d8d4bfd568b06ecca6857fdb5eb2e1d
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 17:25:14 2017 -0500

    initial step for quciskort with nums

commit 459fb5883a0448de3e325b353acc3775e30b867a
Merge: 23d041a b451d04
Author: Millie Yang <millie.yang@columbia.edu>
Date:   Tue Dec 12 00:15:38 2017 -0500

    Merge pull request #24 from sophstad/test

    Test

commit b451d04cee4db7869df9db30bc43b9ec9cda4dcb
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 12 00:15:08 2017 -0500

    more tests

commit cdd7381e3d6131cd92ade7396da77e098912fc34
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 12 00:07:13 2017 -0500

    more tests

commit e45b86d8b804b62544389daa9e10e7401b7d29d2
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Dec 11 23:56:01 2017 -0500

    tests and queue size works

commit aade2d60e183cbcfec19169fb1beda522d8ff18a
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 18:25:44 2017 -0500
```

```
    changing return type of show functions

commit e9efc7f42bb91a55102c2ef8416a152412250faf
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 18:11:44 2017 -0500

    trying to change actual params for quicksorts

commit f45f33dfe8a079128cb8d3605e749a40a1e8a383
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 18:05:12 2017 -0500

    adding tests for quicksorts

commit bffecb7f4dbbba44a4bb08375dc62b481db7fce8
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 17:38:12 2017 -0500

    trying quicksort show for int

commit 870a0c3000df3cb0a3b0df200aa9d35573964bfc
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sun Dec 10 17:25:14 2017 -0500

    initial step for quciskort with nums

commit 23d041a42c875ebebc41a52595abd81b45078eca
Merge: de47c1e aff7947
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Sun Dec 10 17:10:58 2017 -0500

    Merge pull request #21 from sophstad/kofi-quicksort

    quicksort

commit de47c1ed6b30ec21707426d63b98d519dcdeb59c
Merge: 2fb4f19 9dacb73
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Sun Dec 10 17:03:53 2017 -0500

    Merge pull request #23 from sophstad/ss/bugfix-increment

    Bugfix: Increment/decrement

commit 9dacb735247cbae1dd7a16df760ee33cbdec8ad1
Author: sophstad <srs2231@columbia.edu>
```

```
Date:   Sun Dec 10 21:48:28 2017 +0000

    Bugfix: fix increment/decrement

commit 2fb4f193efe9997b72bb3a5250e3b28b57742ee6
Merge: df9fb10 5272610
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Sun Dec 10 16:33:53 2017 -0500

    Merge pull request #22 from sophstad/ss/fix-object-syntax

    Improve object creation syntax

commit 5272610011cb0ee6411b38928e3a6f80c769d558
Author: sophstad <srs2231@columbia.edu>
Date:   Sun Dec 10 19:02:35 2017 +0000

    Improve object creation syntax

commit df9fb10fad5066a7acf1da7d7b5160f2b84264c2
Merge: 90adda1 15e18ae
Author: Millie Yang <millie.yang@columbia.edu>
Date:   Sun Dec 10 00:13:53 2017 -0500

    Merge pull request #20 from sophstad/ss/show-queue-stack

    Show Queue & Stack

commit 15e18aeed0a4e30a6d4fb383d0f989e69eb1bd87
Author: sophstad <srs2231@columbia.edu>
Date:   Sun Dec 10 05:09:46 2017 +0000

    Add support and testing for bool

commit aff7947a9843d83493842b7edace072449dfbc2e
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Dec 9 22:35:22 2017 -0500

    quicksort works, uncommented hardcoded path

commit b02ce5c253615f6a0f9a2bc154f645406f9f7972
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Dec 9 22:29:23 2017 -0500

    quick sort test passes
```

```
commit 5e3d0e425abed451bcb5a6506b8fc24317d0cb99
Merge: bf7d327 2fe22be
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Dec 9 22:26:56 2017 -0500

    fixed a bug in master and rebased

commit bf7d327a79a46f04a53db6f91bb7a816ab6d164e
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sat Dec 9 18:07:31 2017 -0500

    compiling quicksort.c

commit a137cd5eeef31f6f58986d45831a66942f593025
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sat Dec 9 17:59:34 2017 -0500

    trying to get quicksort for ints to work

commit 26b6cd41c4c076840ecef5a126eede5806586df6
Author: sophstad <srs2231@columbia.edu>
Date:   Sun Dec 10 00:40:33 2017 +0000

    Implement .show() for stacks

commit b6dfbdcb096f4f5804956bc3c1b802951fc9dd2e
Author: sophstad <srs2231@columbia.edu>
Date:   Mon Dec 4 21:54:39 2017 +0000

    Implement .show() for queues

commit 2fe22be8ac60858ce0bffa39ee4dc803e98cd7ce
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sat Dec 9 18:07:31 2017 -0500

    compiling quicksort.c

commit f0d3777a74ac06e6acb1e087cd8ae5d8fadba11f
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sat Dec 9 17:59:34 2017 -0500

    trying to get quicksort for ints to work

commit 90adda1ee838adeadd01c7332ee8f09f72ad6147
Merge: 84e0a8a 792296a
Author: Sophie Stadler <sophstad@users.noreply.github.com>
```

```
Date:   Sat Dec 9 17:38:35 2017 -0500

    Merge pull request #19 from sophstad/hood_functions

    Putting functions under the hood

commit 792296a1938bdc2d002135dd7dbe7badba2d61df
Author: sophstad <srs2231@columbia.edu>
Date:   Sat Dec 9 18:40:59 2017 +0000

    Combine show function into one

commit 84e0a8a217bbb906d82ffb9303dcf9609932377d
Merge: 69ae1e5 bc45572
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Fri Dec 8 17:01:58 2017 -0500

    Merge pull request #13 from sophstad/kofi-quicksort

    Quicksort implementation for arrays in c

commit bc4557232f1d29605ae96517d3287b81842f1a8a
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Fri Dec 8 16:52:53 2017 -0500

    fix to show pivot at each stage

commit 40784da76e15dd3bb4436aa7dffa2b4be2e0a14e
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Fri Dec 8 16:03:51 2017 -0500

    Update Makefile

commit 41a4fd499774b3de66c05f953545f61c35509f82
Author: Millie Yang <my2440@columbia.edu>
Date:   Fri Dec 8 16:00:14 2017 -0500

    buggy print

commit 4f837aa8e7f87265010bf3b9e23c52d232a3de2a
Author: Millie Yang <my2440@columbia.edu>
Date:   Fri Dec 8 14:50:55 2017 -0500

    clean up

commit 57565f71fbff665436306c7726efab83c46f510d
```

```
Author: Millie Yang <my2440@columbia.edu>
Date:   Fri Dec 8 14:47:11 2017 -0500

    putting functions under the hood

commit 69ae1e55c847e08a1f60cc129cd2ea27992a2cc6
Merge: b5b3657 3de6018
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Fri Dec 8 13:56:44 2017 -0500

    Merge pull request #18 from sophstad/millie-show

    Show for linkedlist

commit 3de6018b1a609c4b813dc6360138ccdd7dfe1ac1
Merge: 2fdbf4c b5b3657
Author: sophstad <srs2231@columbia.edu>
Date:   Fri Dec 8 18:54:25 2017 +0000

    Merge branch 'master' of https://github.com/sophstad/strux into millie-show

commit 2fdbf4cd3ee8103225235cb84ea279d0a622b3f2
Author: Millie Yang <my2440@columbia.edu>
Date:   Fri Dec 8 12:35:33 2017 -0500

    SHOW STRIng works

commit b5b3657a47977479e7c6baa7f3a06e2f60c4ef31
Merge: c77faec c5b837b
Author: Millie Yang <millie.yang@columbia.edu>
Date:   Fri Dec 8 11:22:27 2017 -0500

    Merge pull request #17 from sophstad/millie-queue-list

    Stack, Queue, LinkedList

commit c5b837b17dbd4580250113a6d76e52e98c45c187
Author: sophstad <srs2231@columbia.edu>
Date:   Fri Dec 8 03:43:04 2017 +0000

    Update tarball tests and delete bad output file

commit b39727a341540f2f48b5f210e431b69a992c3488
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 17:20:12 2017 -0500
```

```
    Implement stack, queue, linkedlist with one size() function

commit 3b81b4791a937afc706c7ca07107aec66d0954f7
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 7 17:43:03 2017 -0500

    fix for displaying full array

commit 8d9b73a8f970576109ac05ae73bd8ceee07eec1e
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 7 16:35:03 2017 -0500

    sort method needs array size as arg

commit c324a750b0802b1e6532ada54832de6d0b0d0b17
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Dec 7 03:33:18 2017 -0500

    looks good now, but have things to discuss with team

commit 2db4f9cc04c45e170ee70b3b92537970d2c8cc76
Author: sophstad <srs2231@columbia.edu>
Date:   Thu Dec 7 02:20:03 2017 +0000

    Fix filenames; works on DigitalOcean

commit c1d357d8375474465b3ea3e592a2d53da431682a
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 20:58:29 2017 -0500

    buggy code that runs

commit 45961ba76da9b31fa934473eac070bf449e1953e
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 20:56:44 2017 -0500

    buggy code

commit e4b9a2c38174592b1d89740b5a4984ce6f049243
Merge: 70708f3 83a6a79
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 20:36:08 2017 -0500

    conflicts w rebase fixed

commit 70708f3d4c5ad29afe8c87a99c3b2598d27a881c
```

```
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 20:33:42 2017 -0500

    rebased w master

commit 0643bcc7a00d841509487ec08a46c7ef184629de
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Wed Dec 6 02:15:09 2017 -0500

    have a basic printing for the tree done, also began implementing the syntax in the AST,

commit bb57177a13808ae9b8575916ac300154e8cede62
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Wed Nov 29 13:11:00 2017 -0500

    some more basic printing

commit b62b4364d001c9d4aae6f2fa981686aaef469800
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Tue Nov 28 13:42:54 2017 -0500

    starting on pretty print

commit 1679135f385916bf4b4105858f9be777fb517e91
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Mon Nov 27 14:43:02 2017 -0500

    Basic BSTree

commit 269fc10378d2b2d805973c21fd3f0eb6290c3bfb
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 15:40:00 2017 -0500

    anytype assignment works

commit 2e716d52107d6b0ace6c0851236b775f327822de
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 15:24:04 2017 -0500

    bug free

commit dbc38ff497ae1e4d447ea6184df98f470d9d3bcc
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 15:00:51 2017 -0500

    printing float works
```

```
commit ac2b5ee385796d74a43cbc1017978ffc3b9c378e
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Wed Dec 6 11:23:19 2017 -0500

    removing vertical bars from the print

commit 83a6a79b6c27cc83d2cde05a141c46640516679b
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Wed Dec 6 02:15:09 2017 -0500

    have a basic printing for the tree done, also began implementing the syntax in the AST,

commit 80214582820b9f94634a3e7abbbb9bcae2267744
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 01:48:36 2017 -0500

    lil progress

commit a97b56a552b5ef28886b6cbb4c191245f81440aa
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Dec 6 01:45:19 2017 -0500

    show works in codegen, but think there is something wrong with implementation in nc

commit 928f188e895a001c040c7e1c16b812ee5bf1e24c
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 5 23:28:22 2017 -0500

    I think sophie knows how to fix this

commit 69cd32e3af41d058087da3f36356eeb00f4fddaa
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 5 23:15:16 2017 -0500

    trying to implement size function that recognizes which functionn to call

commit 8a68d290c82e292c60e8e577f142b5ae78c9764c
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 5 22:25:30 2017 -0500

    initial commit, queue peek and list get works

commit 80a3c26e397b423cccd286623e4626f31424cc40
Merge: 203876b c77faec
Author: Joshua Bartlett <jcbart12595@gmail.com>
```

```
Date:   Tue Dec 5 21:50:52 2017 -0500

    Merge branch 'master' of https://github.com/sophstad/strux into josh/tree

commit c77faec67e9c766c0a1fb61f4925574316192ca8
Merge: 2026606 473d3e8
Author: Millie Yang <millie.yang@columbia.edu>
Date:   Tue Dec 5 20:48:42 2017 -0500

    Merge pull request #16 from sophstad/ss/bugfix-dequeue

    Fix dequeue

commit 2026606617fdfa6d209793f6a5a6130e0ed563fd
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 5 20:40:39 2017 -0500

    oops didnt mean to hardcode path

commit 35db5ae86e896a16b9c0aff024d9ea36874a171a
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 5 20:39:26 2017 -0500

    forces people to run generateModules before running testall

commit 7d2ac0b2f8560c039c2c761fb6c4fcad1ff43f64
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Dec 5 20:23:35 2017 -0500

    removed bug that hides we haven't created .o's

commit 473d3e8dea2f0253b9fd1bd4d0bcbe6d7932bd66
Author: sophstad <srs2231@columbia.edu>
Date:   Mon Dec 4 22:18:40 2017 +0000

    Bugfix: dequeue doesn't accept arguments

commit aa59470643ef6c489eafdabc973708a843524a65
Author: sophstad <srs2231@columbia.edu>
Date:   Mon Dec 4 20:51:06 2017 +0000

    Bugfix: redirect next pointer when enqueueing

commit 4b19933dc7a9827c25ec748f757b76f91b9e8e4a
Author: sophstad <srs2231@columbia.edu>
Date:   Mon Dec 4 04:14:56 2017 +0000
```

```
    Improve test coverage

commit bc7189377469dac121cb0f0049897fdc3d06909c
Merge: 7bbc874 fa9db5d
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Thu Nov 30 20:34:21 2017 -0500

    Merge pull request #15 from sophstad/ss/and-or

    Implement and & or

commit fa9db5df0310a76133e1b7f61ec446350234e8c1
Author: sophstad <srs2231@columbia.edu>
Date:   Fri Dec 1 00:38:24 2017 +0000

    Implement and/or

commit dfd3ec9c22a65f61248108a9ac32d29051e3f675
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Nov 30 16:12:23 2017 -0500

    made quicksort step info more clear

commit a76e94e81273c2cbe8f209cd4a7870c0bb162647
Merge: 3e3cf61 7bbc874
Author: sophstad <srs2231@columbia.edu>
Date:   Thu Nov 30 19:24:24 2017 +0000

    Merge branch 'master' into kofi-quicksort

commit 7bbc8743c1971be622fa3c8a000d6d43671b05aa
Merge: 45322a8 a7eba42
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Thu Nov 30 14:20:31 2017 -0500

    Merge pull request #14 from sophstad/millie-queue

    System compatibility fix

commit a7eba420ff40fa1ab302a9587200d5a88524cbd8
Author: Millie Yang <my2440@columbia.edu>
Date:   Wed Nov 29 21:18:37 2017 -0500

    Build passing on Ubuntu 16.04
```

```
commit 3e3cf610a0762693d1419e5c6999ae5fca117e83
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Nov 30 03:46:14 2017 -0500

    commenting out main function

commit 4868c76a5cd378acb5fa27b6807895a998f50014
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Thu Nov 30 03:44:59 2017 -0500

    have quicksort implementation for int and num arrays

commit 203876b7347c5ae926d16fa2f73fd39902c5c7da
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Wed Nov 29 13:11:00 2017 -0500

    some more basic printing

commit 45322a8161522aa4ac68179c4b231df90e162104
Merge: af1514b c2bf04a
Author: Millie Yang <millie.yang@columbia.edu>
Date:   Tue Nov 28 21:47:19 2017 -0500

    Merge pull request #12 from sophstad/millie-queue

    Implemented linkedlist and queue data structures

commit c2bf04a19109372d2312deecd31f92f4e0bb705f
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 21:36:32 2017 -0500

    made changes according to soph's comment

commit 7a47745be82294d108e042ae75d6108e363b1003
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 20:32:28 2017 -0500

    more failure checks

commit 9ee9330d52413c645ec05fc854a6d757754b3cbb
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 20:26:32 2017 -0500

    adding failure tests

commit 9f13e00f64ad079183e302147f79efee7b049167
```

```
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 20:09:33 2017 -0500

    linkedlist remove works

commit 0eb7de81b3e60f365bf0b819cb6f7bf3937029c9
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 19:24:03 2017 -0500

    add for linkedlist works

commit 31958f8656627d3408968d66c3cb20593f0e7b83
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 18:55:00 2017 -0500

    added linkedlist initialization

commit 173c4f0c8b8499eee6e040d63019ba5058902dfb
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 18:15:54 2017 -0500

    renamed for cleaner code

commit 8d2c59647b9b3b34ce46096303b9fc732df3f88f
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 18:09:19 2017 -0500

    size works

commit 3daf892f379eeadac4ca68a74fae0b9707ea7d3f
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Tue Nov 28 13:42:54 2017 -0500

    starting on pretty print

commit 9cbea0829fe79bee6c95cca6df80d0e6cb9bddca
Merge: dcc208b a3f329f
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 12:50:14 2017 -0500

    buggy peek but trying to get it to work

commit dcc208b385ad43ba04edc1a0bbec8ab021d7e61c
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 12:48:29 2017 -0500
```

```
    buggy peek method, waiting to modify print

commit c298eb64f7083705dd4e886a7db5ff947be79293
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 12:19:01 2017 -0500

    comparing differences to master to avoid conflicts

commit b9b38e0e9f3110d62449b2ef139e6b6c4a92aa0f
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 11:36:08 2017 -0500

    dequeue works

commit e4767e70e353f75118136685880e20fdaa9324d8
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 02:33:09 2017 -0500

    enqueue works

commit cba425d8b1f0d162374b23edffea8a332da84453
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 01:53:07 2017 -0500

    queue works

commit a3f329f63da6d6ba817b41fd8d53c07a3d8d9d4b
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 12:19:01 2017 -0500

    comparing differences to master to avoid conflicts

commit af1514b3b20ce62b1de9c67d84e634f2bc352a06
Merge: c331101 0963508
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Tue Nov 28 12:16:02 2017 -0500

    Merge pull request #11 from sophstad/ss/remove-global-assign

    Bugfix: don't treat variables in main() as global

commit 0963508185fe33e8d24b8f88eaf5a11d2266f0aa
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Mon Nov 27 23:24:48 2017 -0500

    Bugfix: don't treat variables in main() as global
```

```
commit 0ac8e1072131f13dd383157376e786a160c40a0b
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 11:36:08 2017 -0500

    dequeue works

commit c33110136002c79e4502c63474560c0fa4821ab2
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Tue Nov 28 11:17:28 2017 -0500

    Add .o files to gitignore

commit 0ac985638da39a95a73be6b84ae82a06510c4b5a
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 02:33:09 2017 -0500

    enqueue works

commit dc59ebd49bdaebd538e236af4ca7caec67215436
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 28 01:53:07 2017 -0500

    queue works

commit 11e1ae8392323bd5733f71c7d10becb5ebebc4f5
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 23:39:56 2017 -0500

    oops didnt need two tests that did the same thing

commit 3cd1bed9c25ef2df42da47d5982accb16e422fe9
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 23:38:42 2017 -0500

    fixed return 1 test

commit 7e3d5c8372ea0fc1f6cb27685d0bbba5c1fc589e
Merge: ce0fa03 55396e3
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 23:33:26 2017 -0500

    Merge branch 'master' of github.com:sophstad/strux

commit ce0fa03a70b00fb4762a21558df2a0aa67a908df
Author: Millie Yang <my2440@columbia.edu>
```

```
Date:   Mon Nov 27 23:32:57 2017 -0500

    fixed test

commit 55396e341c9ecb757b222e9b8af02239ce5950aa
Merge: c16e36c 23f0fc7
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Mon Nov 27 22:41:16 2017 -0500

    Merge pull request #10 from sophstad/linkedlist-kofi

    added linkedlist c file

commit 23f0fc79fadcedc7357fc7f9bb9a296f10a5384a
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Mon Nov 27 22:40:09 2017 -0500

    added linkedlist c file

commit c16e36cfc38d14c79001325d73164cdc1b5fa847
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 18:20:22 2017 -0500

    fixed test result

commit 56e770b1bb264cbb291bc05cfd2e37142e5f9e09
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 18:16:18 2017 -0500

    removed duplicate files

commit d1fad34fe0279003394bdb2b313da0f1b1f16ddf
Merge: 4d118eb 6f38054
Author: Millie Yang <millie.yang@columbia.edu>
Date:   Mon Nov 27 18:14:29 2017 -0500

    Merge pull request #9 from sophstad/millie-queue-stack

    Integrating c library

commit 6f38054711305d36c6fcde5154e3d2801e987aec
Merge: 4771c19 4d118eb
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 18:13:58 2017 -0500

    merging
```

```
commit 4771c19ad217d6bc8ab3891ad36e04cd10318e97
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 17:48:21 2017 -0500

    all tests pass

commit 1a022fd7608e5baf74a2774a1473110ba3c870b4
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 17:38:13 2017 -0500

    integrated queue n stack

commit ea3590febde8a5a15a57f30580463c147f08ba29
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 17:33:08 2017 -0500

    print big works

commit a2634ba09939474eea0681b008a8722162cdba99
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 16:33:24 2017 -0500

    changed ml file

commit 0b0d59deb57dc599b4cebee0eaa07246c190c3cd
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 16:30:49 2017 -0500

    trying to revert back to original code to chcek for errors

commit 91c785e203ef64157ec06f0f2a2b39801dc3bb57
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 16:18:19 2017 -0500

    printbig integration

commit 5791d5214da1e5ffa35c405242ca3b939ff32c20
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 27 15:18:00 2017 -0500

    modified, but still not workign

commit c87c308c40d02085d3d70e541415d28c644a339d
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Mon Nov 27 14:43:02 2017 -0500
```

```
    Basic BSTree

commit 4d118ebbbd93ee6acb9fe8c7aa4de9e09fb1440f
Merge: 801fab9 b470130
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Sat Nov 25 21:54:10 2017 -0500

    Merge pull request #7 from sophstad/ss/arrays

    Implement Arrays

commit b470130a454e2420aa5fd4958cd834fd7df61a28
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 14:34:14 2017 -0500

    More cleaning up

commit 9ab86424e85f162263e52d8a763faa95216fbe5a
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 14:03:50 2017 -0500

    Add testing for string arrays

commit da832e9df111eac278ae5f34b18a75e14155bbee
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 14:00:56 2017 -0500

    Fix up spacing

commit 52cfaced6a7dcf3fae5cb6e411d4b6c8a5b42527
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 13:53:18 2017 -0500

    Conclude merge resolution

commit e432a608c0b7b6fc11b4b742fc5ae79d729e94b5
Merge: ec89d63 df9068b
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 13:49:08 2017 -0500

    Merge branch 'ss/arrays' of https://github.com/sophstad/strux into ss/arrays

commit ec89d63e61cbca95bd13a219df3f805aa2fc6130
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 13:44:14 2017 -0500
```

```
    Arrays working

commit 537a6cc736fac611f2791aa17530c8d224b5f32c
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 01:07:31 2017 -0500

    WIP

commit a89d6272abef3097809dc14af2a296a9613cb95f
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Wed Nov 15 18:15:41 2017 -0500

    Allow arrays to be created; test included

    Todo
    - allow index assignment, e.g. a[1] = 5;
    - Fix integer operations

commit 456f74e05d54735e6082f548252aece13347203f
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Wed Nov 15 17:16:27 2017 -0500

    should work with int

commit f1b3144a12e2c5869994c1658e49e9e05e73e7c2
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Tue Nov 14 10:57:53 2017 -0500

    Arrays minus codegen - uses num

commit 801fab9122da30a565a44567b6f32889e66ee22a
Merge: ab000fa b5a9c9b
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Fri Nov 24 10:59:20 2017 -0500

    Merge pull request #6 from sophstad/ss/assign

    Single line variable assignment

commit b5a9c9b752468d9bafbe95c02fbb8034fe15feec
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 10:53:19 2017 -0500

    Error handling for variable declaration without assignment
```

```
commit 37b40f495171e0fec3d1fc75cef3996e5652392c
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 01:42:32 2017 -0500

    Improve test to verify that variable reassignment works

commit 31fae7c464844d75bb669c3b1d6046f047243a27
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Fri Nov 24 01:38:31 2017 -0500

    Add single line variable assignment and fix tests

commit 332e802eef2412d7aa69e74b445458353bb49230
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 20 21:04:17 2017 -0500

    updated with some tests passing and others failing

commit d04a306c010613c243d6386ff51fb6865f6aeaa6
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 20 19:39:37 2017 -0500

    fixed sth, or maybe not?

commit e94dfac9023eb67c2eaec4493c8b325f8a1a5d31
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 20 18:44:50 2017 -0500

    queue added

commit 862c27f6af1235af351512162da7103094d20fd1
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 20 18:30:37 2017 -0500

    added type queue

commit 4bc6c2bd9b6fc16e547c891159739bef4e3b41f6
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 20 17:53:00 2017 -0500

    stack and q data structures

commit ab000fab68af2179651cda0fce862c17979bcb33
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 20 16:01:43 2017 -0500
```

```
    added lines to clean up some stuff

commit 7df85cddb09deac55a494efa9f7573bf591b56d6
Merge: ed1db9f 8a064f2
Author: Millie Yang <millie.yang@columbia.edu>
Date:   Sat Nov 18 17:33:59 2017 -0500

    Merge pull request #5 from sophstad/implement_int

    Implement Integer

commit 8a064f284d587b3cf55b45596605ad91d66b2d6f
Merge: d3fbb24 8d0c9b0
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sat Nov 18 17:22:07 2017 -0500

    Merge branch 'implement_int' of https://github.com/sophstad/strux into implement_int

commit d3fbb24f445047be5eafd72381dd42fb8bb5f706
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sat Nov 18 17:21:50 2017 -0500

    cleaning up semant for int/num ops

commit 8d0c9b0d4d5864f22611472f82cbd23262330f46
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Sat Nov 18 12:29:50 2017 -0500

    Fix several tests

commit 43f5775e132d75222eeb5c137358f37cf86a6336
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Sat Nov 18 03:52:03 2017 -0500

    made int type with associated ops

commit ed1db9f47cc9247fd1d61ca51e0a6bd83bc581a7
Author: Millie Yang <my2440@columbia.edu>
Date:   Fri Nov 17 16:11:23 2017 -0500

    renamed tests

commit 1174e1ef2be6396859bccf5781226349a8250f07
Author: Millie Yang <my2440@columbia.edu>
Date:   Thu Nov 16 17:57:46 2017 -0500
```

```
    modify test

commit 8015d62babc9b05e2ae0ca5f21926c8d692119ca
Author: Millie Yang <my2440@columbia.edu>
Date:   Thu Nov 16 17:39:00 2017 -0500

    fixed tests

commit 9fe6347617a1a3706971743d51435c5e652c0a54
Author: Millie Yang <my2440@columbia.edu>
Date:   Thu Nov 16 17:05:39 2017 -0500

    no main test

commit b18d383bc4ff65bdaa901a6c7e45b059fe7df526
Author: Millie Yang <my2440@columbia.edu>
Date:   Thu Nov 16 17:00:16 2017 -0500

    string assignment

commit df9068bb59097216cc069ffed19b568915dc141d
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Wed Nov 15 18:15:41 2017 -0500

    Allow arrays to be created; test included

    Todo
    - allow index assignment, e.g. a[1] = 5;
    - Fix integer operations

commit a6056734a575d5603b5aac8ef781e4b2ef5494a0
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Wed Nov 15 17:16:27 2017 -0500

    should work with int

commit 4e2cde0ccac4d29fa01aa464c0602431fd2f16a3
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 14 23:43:09 2017 -0500

    removed duplicate test and clean up

commit 666a0815aa6f2f56badd3ac494676090973fb7c8
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Tue Nov 14 10:57:53 2017 -0500
```

```
    Arrays minus codegen - uses num

commit 57fa27ca2fc78651d41f27d12e762a038cc51e0c
Author: Millie Yang <my2440@columbia.edu>
Date:   Tue Nov 14 14:06:41 2017 -0500

    cleaned up and renamed, didnt do anything

commit de450eb36bd84ec6bea5f917389c46467b1d1813
Merge: 7c4b591 def05b4
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Tue Nov 14 03:51:34 2017 -0500

    Merge pull request #3 from sophstad/kofi-plt

    have increment and decrement working

commit def05b4e185688fdb36d8fb73ca352eaa3520bab
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Tue Nov 14 03:48:44 2017 -0500

    have increment and decremnt working

commit 7c4b5919069d669dcc291bbae13fac5622cf7169
Merge: a9ade1c 28322f1
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Tue Nov 14 02:29:58 2017 -0500

    Merge pull request #1 from sophstad/millie

    printed bool and got return for all primitive types working

commit a9ade1cb31bdd18454755c957a843af4529beea3
Merge: ad50862 70022c1
Author: Fredrick Kofi Tam <Fredrick-Tam@github.com>
Date:   Tue Nov 14 02:26:01 2017 -0500

    Merge pull request #2 from sophstad/kofi-plt

    Fixing minor stuff pertaining to if/elif/else, modulus and float negation

commit 70022c11c3b5b292084f4dcfdd97624930c621e8
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Tue Nov 14 00:22:53 2017 -0500

    fixing neg op for nums
```

```
commit b6b4d56b055f90ba972a4ded46b40b24bcb2d5bd
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Mon Nov 13 23:24:22 2017 -0500

    fixed mod and enabled elif clause

commit 28322f167fcde6af7aaa3a80948cc1f1a03cab4c
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 13 20:35:44 2017 -0500

    added tests

commit dc20e8c2b3c5c465fe93fc03b0b716ca7838a4d8
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 13 20:21:40 2017 -0500

    removed unnecessary files

commit 8ebcbeef6e347e138e81409c1fc5c2c974af2aac
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 13 20:06:35 2017 -0500

    added tests

commit 25c4533168599b89097e21277da6c27f93d8aa41
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 13 20:04:00 2017 -0500

    fixed return type

commit 0325c919283440b7ba852e7d1dd39398853e186e
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 13 19:55:38 2017 -0500

    changed fatal string bug

commit 4f511a883e412da2da4a18d93cd752b91bec8465
Author: Millie Yang <my2440@columbia.edu>
Date:   Mon Nov 13 19:41:04 2017 -0500

    printed bool and got return working - one bug

commit ad508626bc787be27fc0d908d78355f1ef6d0568
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Nov 11 19:02:15 2017 -0500
```

```
    added mod in semant, added tests

commit 650af2c3e2b6960b418852605ff0cc5f620c5014
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Nov 11 18:48:33 2017 -0500

    return type tests

commit b9337102c3107bd3bd48a9f4d904c4b2eca4e00b
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Nov 11 18:46:15 2017 -0500

    added mod test

commit 36924fa3827e7b43cdf2f5d8b3f310ea39297d21
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Nov 11 18:43:35 2017 -0500

    assignment test modifications

commit 184761f78fa112041ad69963f77ca3c461fd4a4e
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Nov 11 18:42:22 2017 -0500

    tests

commit 74c0a5cd76e042ed9021ad7c9cd1f75133ad0ae4
Author: Millie Yang <my2440@columbia.edu>
Date:   Sat Nov 11 18:33:18 2017 -0500

    tests

commit a116af29827841fce0207838eedff425bf4053dd
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Sat Nov 11 18:21:27 2017 -0500

    updated tests/

commit b63f06034c13db3002f398b5b18661ca15ca5559
Merge: f3fd612 8f4e48d
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Sat Nov 11 14:24:28 2017 -0500

    Merge branch 'master' of https://github.com/sophstad/strux
```

```
commit f3fd6121afa5350035cd755518d54acbb391cc17
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Sat Nov 11 14:24:23 2017 -0500

    update comment to say strux

commit 8f4e48d45bd08afad4711f0ff2bc415c1b967f94
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Sat Nov 11 13:15:50 2017 -0500

    Use relative path for ll; update gitignore

commit 54d46d455175fc70f9807422ab721f5c0f313158
Author: Millie Yang <my2440@columbia.edu>
Date:   Fri Nov 10 12:20:44 2017 -0500

    added print and if tests

commit 6354e3cae63e53bd1f3aeb960d17b2646157ea68
Author: Millie Yang <my2440@columbia.edu>
Date:   Fri Nov 10 11:34:40 2017 -0500

    adding necessary tests

commit 567a9d854aa2eebb319d2d72d55f3c0097ee4394
Merge: 56ab4d7 0a04e94
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Fri Nov 10 11:09:49 2017 -0500

    Merge branch 'master' of https://github.com/sophstad/strux

commit 0a04e94fa1c8b7f20216d5f61332a5a8ce34d34f
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Wed Nov 8 23:35:42 2017 -0500

    print is working

commit a18f0a4608a02f91d4433c4de38beb25a4682443
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Wed Nov 8 21:05:41 2017 -0500

    BISH IT WORKS

commit 56ab4d720a2042cea4055711a61812480565986b
Merge: d0f64a7 1679e84
Author: Joshua Bartlett <jcbart12595@gmail.com>
```

```
Date:   Wed Nov 8 09:49:08 2017 -0500

    Merge branch 'master' of https://github.com/sophstad/strux

commit 1679e841e18703591795402e2afdd586004b0aca
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Wed Nov 8 04:02:07 2017 -0500

    fixing reduce conflicts

commit 1853124f5896162b4187c57e92b5e571ed559dd2
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Wed Nov 8 03:09:19 2017 -0500

    no erroes when makefile called

commit 4c1ad9456494fb1be9412f9ae6922998b259acc9
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Wed Nov 8 02:59:33 2017 -0500

    major revisions to get parts working, still not there yet

commit d0f64a760f08b1cc93c34a60368ffabd7a9ddc5b
Merge: 848616c 5b4c655
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Tue Nov 7 20:21:27 2017 -0500

    Merge branch 'master' of https://github.com/sophstad/strux

commit 5b4c6553631f825d611e2ae3caecee3e36819a41
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Tue Nov 7 19:57:19 2017 -0500

    Update makefile and gitignore

commit b7288cae18de66d1c7791b198312c96beed0bcae
Merge: d905c61 0b7937e
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Tue Nov 7 19:28:34 2017 -0500

    Merge branch 'master' of https://github.com/sophstad/strux

commit d905c611ca8d1f5ce1d1f0bcec944d3c26b60ed7
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Tue Nov 7 19:27:24 2017 -0500
```

```
    Add makefile and entry point

commit 0b7937edf8eb188853e403962c73b56081ccb625
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Tue Nov 7 17:16:01 2017 -0500

    seperated literals into string and num, started work on codegen and print function

commit a087e28259bb5338b848961a7872e41d85220ca0
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Tue Nov 7 02:34:56 2017 -0500

    starting up codegen file and creating array type in ast

commit 47caf7d575d95e5624dbff29c9d223ded46b7a56
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Mon Nov 6 11:42:59 2017 -0500

    Minor additions

commit 3f56392203a4dfd634ac339497869285f5acc37e
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Thu Nov 2 00:50:26 2017 -0400

    Remove ast syntax error

commit e7f2d4d06359c0fb17833833b83bfbbb2fef93ad
Author: Fredrick-Tam <tamf295@yahoo.com>
Date:   Tue Oct 31 23:55:06 2017 -0400

    added semantic checker file

commit 24b4b29bd60610bc405fcc8a3f9c6f1f6861cb1c
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Tue Oct 31 21:05:02 2017 -0400

    Add ast and parser

commit 8e71c7ad01051f535c81d32ccc90c35c81a5b788
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Thu Oct 26 20:36:15 2017 -0400

    Update scanner, add parser

commit 848616cf4f9d97902212bccfb5d8345fa632d367
Merge: d9a9682 d924749
```

```
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Thu Oct 26 18:14:44 2017 -0400

    Merge branch 'master' of https://github.com/sophstad/strux

commit d92474937eb5203565263a847c4c1312cf771d01
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Thu Oct 19 18:22:07 2017 -0400

    Add double colon for type

commit b3de2dadd87dd99d07601b2b11ef0e316b003e13
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Thu Oct 19 18:19:31 2017 -0400

    Add modulo

commit ba06be934be5592fb68a95dacf72ae7785403a8b
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Thu Oct 19 18:18:31 2017 -0400

    Add brackets

commit 301cdae400f438008770e36a5fa8994a63cd5b3e
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Thu Oct 19 18:15:44 2017 -0400

    Add initial scanner implementation

commit d9a968267995841cf3d5557a85817b55ed73f95d
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date:   Thu Oct 19 16:13:16 2017 -0400

    Adding Language Reference Manual for Strux

commit 68c213192cbeb3bd7b65251444d1839198b31279
Author: Sophie Stadler <srs2231@columbia.edu>
Date:   Mon Sep 25 15:27:32 2017 -0400

    Add proposal pdf

commit b9c288181391ac90e0a8312122c9ab3f78bbb546
Author: Sophie Stadler <sophstad@users.noreply.github.com>
Date:   Sun Sep 17 16:07:25 2017 -0400

    Initial commit
```