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.ml ................................ 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:

```c
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.

<table>
<thead>
<tr>
<th></th>
<th>num</th>
<th>string</th>
<th>bool</th>
<th>int</th>
<th>while</th>
<th>for</th>
<th>not</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>void</td>
<td>if</td>
<td>elif</td>
<td>else</td>
<td>new</td>
<td>return</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>and</td>
<td>or</td>
<td>Queue</td>
<td>Stack</td>
<td>LinkedList</td>
<td>BSTree</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Function</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Assignment</td>
<td>Right to left</td>
</tr>
</tbody>
</table>

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`

<table>
<thead>
<tr>
<th>Operator</th>
<th>Function</th>
<th>Associativity</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>Left to right</td>
<td><code>x + y = 120</code></td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>Left to right</td>
<td><code>x - y = 80</code></td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>Left to right</td>
<td><code>x * y = 2000</code></td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>Left to right</td>
<td><code>x / y = 50</code></td>
</tr>
<tr>
<td>%</td>
<td>Modulo</td>
<td>Left to right</td>
<td><code>x % y = 0</code></td>
</tr>
<tr>
<td>++</td>
<td>Increment</td>
<td>Left to right</td>
<td><code>x++ = 101</code></td>
</tr>
<tr>
<td>--</td>
<td>Decrement</td>
<td>Left to right</td>
<td><code>x-- = 99</code></td>
</tr>
</tbody>
</table>

3.2.3 Comparison Operators

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

<table>
<thead>
<tr>
<th>Op.</th>
<th>Function</th>
<th>Associativity</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Equal to</td>
<td>Left to right</td>
<td><code>(x == y)</code> returns <code>false</code></td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
<td>Left to right</td>
<td><code>(x != y)</code> returns <code>true</code></td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>Left to right</td>
<td><code>(x &gt; y)</code> returns <code>true</code></td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td>Left to right</td>
<td><code>(x &gt;= y)</code> returns <code>true</code></td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>Left to right</td>
<td><code>(x &gt; y)</code> returns <code>false</code></td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
<td>Left to right</td>
<td><code>(x &gt;= y)</code> returns <code>false</code></td>
</tr>
</tbody>
</table>

3.2.4 Logical Operators

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Function</th>
<th>Associativity</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>Logical AND</td>
<td>Left to right</td>
<td><code>(x and y)</code> returns <code>false</code></td>
</tr>
<tr>
<td>or</td>
<td>Logical OR</td>
<td>Left to right</td>
<td><code>(x or y)</code> returns <code>true</code></td>
</tr>
<tr>
<td>not</td>
<td>Logical NOT</td>
<td>Left to right</td>
<td><code>not x</code> returns <code>false</code></td>
</tr>
</tbody>
</table>

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. The list 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.

```
um fivePointOh = 5.0;
int wordCount = 1255;
string missionStatement = "Strux rocks!";
bool isTired = true;
int i; \:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\:\\n```

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:

```c
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:

```c
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:

```c
bool october = false;
if (october == true) {
  print("It's October!");
```
An if-elif-else statement presents the opportunity to introduce more (infinite, in fact) conditional statements.

```python
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 using 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.
Initialize 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.

```cpp
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`.

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

Library Functions  There are several built-in 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`.

```cpp
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.

```cpp
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.

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

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

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

** 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.

```cpp
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:

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

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

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

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

Add
To append items to the tail of the LinkedList, 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.

```cpp
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.

```cpp
numList.delete(3); // (returns true)
```

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

```cpp
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.

```plaintext
int[2] intArray = [5, 2];
string[2] name = ["hello", "world"];
num[0] numArray = [];
```

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

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

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

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

**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()`:

```plaintext
int[7] arr = [10, 100, 30, 90, 40, 50, 70];
arr.quickSort();
arr.show();
```

Using `.showQuickSort()`

```plaintext
int[9] arr = [1, 4, 3, 6, 7, 2, 99, 23, 37];
arr.showQuickSort();
```

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]

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:

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

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

```cpp
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:

```cpp
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.

```cpp
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

```cpp
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

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

will print this to the console:

```
1
```

**string**  The expression

```cpp
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

```cpp
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

```cpp
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 \texttt{isTrue()}\], and the formals include one variable \( x \) of type \texttt{bool}. Note that if this method is defined to be

\begin{verbatim}
num isTrue(bool x) {
    if (x == true) {
        return true;
    }
    return false;
}
\end{verbatim}

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

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

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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Data Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Josh</td>
<td>Language Guru</td>
<td>Tree</td>
</tr>
<tr>
<td>Kofi</td>
<td>System Architect</td>
<td>Linked list, quick sort</td>
</tr>
<tr>
<td>Millie</td>
<td>Tester</td>
<td>Stack, queue</td>
</tr>
<tr>
<td>Sophie</td>
<td>Manager</td>
<td>Array</td>
</tr>
</tbody>
</table>

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 to master, excluding merge commits

---

**milliyang #1**
69 commits 17,839 ++ 1,584 --

---

**sophstac #2**
66 commits 3,898 ++ 3,021 --

---

**Fredrick-Tam #3**
43 commits 2,847 ++ 1,462 --

---

**jcblartlett25 #4**
25 commits 14,613 ++ 12,785 --

---
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:

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

And another example of a unit test:

```c
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.

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

```c
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;
        }
    }
```
6.3 Error Handling

Here is an example of how we terminated gracefully:

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

Returns:

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

Another example of error handling:

```c
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.

```c
int main() {
    Queue::num qn = new Queue::num();
    qn.add(3.6);
    qn.add(4.216);
    qn.add(-5.25);
    qn.add(6.4);
    qn.show();
}
```
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* }
@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"
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*)

declare i8* @top(%struct.Stack*)

declare i32 @stack_size(%struct.Stack*)

declare void @stack_show_int(%struct.Stack*)


```mlir
define i32 @main() {
  entry:
  %qn = alloca %struct.Queue*  
  %init = call %struct.Queue* @initQueue()  
  store %struct.Queue* %init, %struct.Queue** %qn  
  %qn1 = load %struct.Queue*, %struct.Queue** %qn  
  %malloccall = tail call i8* @malloc(i32 ptrtoint (double* getelementptr (double, double* null, i32 1) to i32))  
  %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))
%tmp26 = bitcast i8* %malloccall.25 to i32*
store i32 4, i32* %tmp26
%ptr27 = bitcast i32* %tmp26 to i8*
call void @enqueue(%struct.Queue* %qi24, i8* %ptr27)
%qi28 = load %struct.Queue*, %struct.Queue** %qi
%malloccall.29 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32))
%tmp30 = bitcast i8* %malloccall.29 to i32*
store i32 356, i32* %tmp30
%ptr31 = bitcast i32* %tmp30 to i8*
call void @enqueue(%struct.Queue* %qi28, i8* %ptr31)
%qi32 = load %struct.Queue*, %struct.Queue** %qi
call void @dequeue(%struct.Queue* %qi32)
%qi33 = load %struct.Queue*, %struct.Queue** %qi
%malloccall.34 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32))
%tmp35 = bitcast i8* %malloccall.34 to i32*
store i32 89494, i32* %tmp35
%ptr36 = bitcast i32* %tmp35 to i8*
call void @enqueue(%struct.Queue* %qi33, i8* %ptr36)
%qi37 = load %struct.Queue*, %struct.Queue** %qi
%malloccall.38 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32))
%tmp39 = bitcast i8* %malloccall.38 to i32*
store i32 22, i32* %tmp39
%ptr40 = bitcast i32* %tmp39 to i8*
call void @enqueue(%struct.Queue* %qi37, i8* %ptr40)
%qi41 = load %struct.Queue*, %struct.Queue** %qi
call void @queue_show_int(%struct.Queue* %qi41)
%qs = alloca %struct.Queue*
%init42 = call %struct.Queue* @initQueue()
store %struct.Queue* %init42, %struct.Queue** %qs
%qs43 = load %struct.Queue*, %struct.Queue** %qs
%malloccall.44 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%tmp45 = bitcast i8* %malloccall.44 to i8**
store i8* getelementptr inbounds ([5 x i8], [5 x i8]* @string, i32 0, i32 0), i8** %tmp45
%ptr46 = bitcast i8** %tmp45 to i8*
call void @enqueue(%struct.Queue* %qs43, i8* %ptr46)
%qs47 = load %struct.Queue*, %struct.Queue** %qs
%malloccall.48 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%tmp49 = bitcast i8* %malloccall.48 to i8**
store i8* getelementptr inbounds ([5 x i8], [5 x i8]* @string.1, i32 0, i32 0), i8** %tmp49
%ptr50 = bitcast i8** %tmp49 to i8*
call void @enqueue(%struct.Queue* %qs47, i8* %ptr50)
%qs51 = load %struct.Queue*, %struct.Queue** %qs
%malloccall.52 = tail call i8* @malloc(i32 ptrtoint (i1**
  getelementptr (i1*, i1** null, i32 1) to i32))
%tmp53 = bitcast i8* %malloccall.52 to i8**
store i8* getelementptr inbounds ([7 x i8], [7 x i8]*
  @string.2, i32 0, i32 0), i8** %tmp53
%ptr54 = bitcast i8** %tmp53 to i8*
call void @enqueue(%struct.Queue* %qs51, i8* %ptr54)
%qs55 = load %struct.Queue*, %struct.Queue** %qs
%malloccall.56 = tail call i8* @malloc(i32 ptrtoint (i1**
  getelementptr (i1*, i1** null, i32 1) to i32))
%tmp57 = bitcast i8* %malloccall.56 to i8**
store i8* getelementptr inbounds ([7 x i8], [7 x i8]*
  @string.3, i32 0, i32 0), i8** %tmp57
%ptr58 = bitcast i8** %tmp57 to i8*
call void @enqueue(%struct.Queue* %qs55, i8* %ptr58)
%qsSize = alloca i32
%qs59 = load %struct.Queue*, %struct.Queue** %qs
%isEmpty = call i32 @queue_size(%struct.Queue* %qs59)
store i32 %isEmpty, i32* %qsSize
%qsSize60 = load i32, i32* %qsSize
%printf = call i32 (i8*, ...) @printf(i8* getelementptr
  inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32 0), i32
  %qsSize60)
%qs61 = load %struct.Queue*, %struct.Queue** %qs
call void @queue_show_string(%struct.Queue* %qs61)
%qb = alloca %struct.Queue*
%init62 = call %struct.Queue* @initQueue()
store %struct.Queue* %init62, %struct.Queue** %qb
%qb63 = load %struct.Queue*, %struct.Queue** %qb
%malloccall.64 = tail call i8* @malloc(i32 ptrtoint (i1*
  getelementptr (i1, i1* null, i32 1) to i32))
%tmp65 = bitcast i8* %malloccall.64 to i1*
store i1 true, i1* %tmp65
%ptr66 = bitcast i1* %tmp65 to i8*
call void @enqueue(%struct.Queue* %qb63, i8* %ptr66)
%qb67 = load %struct.Queue*, %struct.Queue** %qb
%malloccall.68 = tail call i8* @malloc(i32 ptrtoint (i1*
  getelementptr (i1, i1* null, i32 1) to i32))
%tmp69 = bitcast i8* %malloccall.68 to i1*
store i1 false, i1* %tmp69
%ptr70 = bitcast i1* %tmp69 to i8*
call void @enqueue(%struct.Queue* %qb67, i8* %ptr70)
%qb71 = load %struct.Queue*, %struct.Queue** %qb
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.

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

Target code for this program:

```cpp
; 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* }

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 %struct.Stack* @initStack()

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

declare void @pop(%struct.Stack*)

declare i8* @top(%struct.Stack*)
declare i32 @stack_size(%struct.Stack*)
declare void @stack_show_int(%struct.Stack*)
declare void @stack_show_float(%struct.Stack*)
declare void @stack_show_string(%struct.Stack*)
declare i32* @cQuickSort(i32*, i32)
declare void @cShowQuickSort(i32*, i32)
declare double* @cQuickfSort(double*, i32)
declare void @cShowfQuickSort(double*, i32)
declare i8** @cQuicksSort(i8**, i32)
declare void @cShowsQuickSort(i8**, i32)
declare %struct.BSTree* @initBSTree()
declare void @addIntToTree(%struct.BSTree*, i8*)
declare void @addNumToTree(%struct.BSTree*, i8*)
declare void @deleteIntFromTree(%struct.BSTree*, i32)
declare void @deleteNumFromTree(%struct.BSTree*, double)
declare i1 @treeContainsInt(%struct.BSTree*, i32)
declare i1 @treeContainsFloat(%struct.BSTree*, double)
declare void @showIntTree(%struct.BSTree*)
declare void @showNumTree(%struct.BSTree*)
declare i32 @printbig(i32)
define i32 @main() {
  entry:
  %tree1 = alloca %struct.BSTree*
  %init = call %struct.BSTree* @initBSTree()
store %struct.BSTree* %init, %struct.BSTree** %tree1
%tree11 = load %struct.BSTree*, %struct.BSTree** %tree1
call void @showIntTree(%struct.BSTree* %tree11)
ret i32 0
}

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 }

45
28 | '4'   { 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 }  
45 | '∀'   { OR }   
46 | '¬'   { 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 }
8.2 parser.mly

Josh, Kofi, Millie, Sophie

```verbatim
%%
%{ open Ast %}

%token SEMI LPAREN RPAREN LBRACE RBRACE LBRACK RBRACK COMMA DOUBLECOL
%token PLUS MINUS TIMES DIVIDE INCR DECR MOD ASSIGN NOT
%token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
%token RETURN NULL IF ELIF ELSE NEW FOR WHILE NUM INT BOOL
%token STRING VOID DOT
%token QUEUE LINKEDLIST STACK BSTREE
%token <float> NUM_LITERAL
%token <int> INT_LITERAL
%token <string> STRING_LITERAL
%token <string> ID
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%nonassoc ELIF
%left INCR DECR
%right ASSIGN
%left OR
%left AND DOT
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%right MOD
%right NOT NEG

%start program
%type <Ast.program> program
%
```
program:
    decls EOF { $1 }

decls:
    /* nothing */ { [] }
    | decls fdecl { $2 :: $1 }

fdecl:
    typ ID LPAREN forms_opt RPAREN LBRACE stmt_list RBRACE
    { { typ = $1;
        fname = $2;
        formals = $4;
        body = List.rev $7 } }

forms_opt:
    /* nothing */ { [] }
    | formal_list { List.rev $1 }

formal_list:
    typ ID { [(1,2)] }
    | formal_list COMMA typ ID { (3,4) :: $1 }

primitive:
    NUM { Num }
    | INT { Int }
    | STRING { String }
    | BOOL { Bool }
    | VOID { Void }

ds_type:
    primitive LBRACK INT_LITERAL RBRACK { Arraytype($1, $3) }
    | QUEUE DOUBLECOL primitive { QueueType($3) }
    | LINKEDLIST DOUBLECOL primitive { LinkedListType($3) }
    | STACK DOUBLECOL primitive { StackType($3) }
    | BSTREE DOUBLECOL primitive { BSTreeType($3) }

typ:
    primitive { $1 }
    | ds_type { $1 }

stmt_list:
    /* nothing */ { [] }
    | stmt_list stmt { $2 :: $1 }

stmt:
    expr SEMI { Expr $1 }
| RETURN SEMI { Return Noexpr } |
| RETURN expr SEMI { Return $2 } |
| LBRACE stmt_list RBRACE { Block(List.rev $2) } |
| IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([]) ) } |
| IF LPAREN expr RPAREN stmt ELSE stmt { If($3, $5, $7) } |
| IF LPAREN expr RPAREN stmt else_stmt { If($3, $5, $6) } |
| FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt { For($3, $5, $7, $9) } |
| WHILE LPAREN expr RPAREN stmt { While($3, $5) } |

else_stmt: 
  ELIF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([]) ) } |
  ELIF LPAREN expr RPAREN stmt else_stmt { If($3, $5, $6) } |
  ELIF LPAREN expr RPAREN stmt ELSE stmt { If($3, $5, $7) } |

expr_opt: 
  /* nothing */ { Noexpr } |
  expr { $1 } |

expr: 
  literal { $1 } |
  expr PLUS expr { Binop($1, Add, $3) } |
  expr MINUS expr { Binop($1, Sub, $3) } |
  expr TIMES expr { Binop($1, Mult, $3) } |
  expr DIVIDE expr { Binop($1, Div, $3) } |
  expr MOD expr { Binop($1, Mod, $3) } |
  expr EQ expr { Binop($1, Equal, $3) } |
  expr NEQ expr { Binop($1, Neq, $3) } |
  expr LT expr { Binop($1, Less, $3) } |
  expr LEQ expr { Binop($1, Leq, $3) } |
  expr GT expr { Binop($1, Greater, $3) } |
  expr GEQ expr { Binop($1, Geq, $3) } |
  expr AND expr { Binop($1, And, $3) } |
  expr OR expr { Binop($1, Or, $3) } |
  MINUS expr %prec NEG { Unop(Neg, $2) } |
  NOT expr { Unop(Not, $2) } |
  expr INCR { Postop($1, Incr) } |
  expr DECR { Postop($1, Decr) } |
  typ ID { Assign($1, $2, Noexpr) } |
  typ ID ASSIGN expr { Assign($1, $2, $4) } |
  expr DOT ID LPAREN actuals_opt RPAREN { ObjectCall($1, $3, $5) } |
  ID ASSIGN expr { Reassign($1, $3) } |
  } |
8.3 ast.ml

Josh, Kofi, Millie, Sophie

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

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

type uop = Neg | Not
type typ = Num | Int | String | Bool | Void | AnyType |
" NumberType |
\[ \rightarrow \text{Arraytype of typ } | \text{QueueType of typ } | \]
\[ \rightarrow \text{LinkedListType of typ } | \text{StackType of typ } | \text{BSTreeType of typ } \]

type bind = typ * string

type expr =
\[ \text{NumLit of float } | \text{IntLit of int } | \text{StringLit of string } | \text{BoolLit of bool } | \text{Null } | \text{Id of string } | \text{Binop of expr * op * expr } | \text{Unop of uop * expr } | \text{Postop of expr * op } | \text{Assign of typ * string * expr } | \text{Reassign of string * expr } | \text{FuncCall of string * expr list } | \text{QueueLit of typ * expr list } | \text{BSTreeLit of typ * expr list } | \text{LinkedListLit of typ * expr list } | \text{StackLit of typ * expr list } | \text{ObjectCall of expr * string * expr list } | \text{Noexpr } | \text{ArrayLit of expr list } | \text{ArrayAccess of string * expr } | \text{ArrayElementAssign of string * expr * expr } \]

type stmt =
\[ \text{Block of stmt list } | \text{Expr of expr } | \text{Return of expr } | \text{If of expr * stmt * stmt } | \text{For of expr * expr * expr * stmt } | \text{While of expr * stmt } \]

type func_decl = {
\[ \text{typ : typ ; } | \text{fname : string ; } | \text{formals : bind list ; } | \text{body : stmt list ; } \]
type program = func_decl list

(* Pretty-printing functions *)

let string_of_op = function
  Add -> "+
  Sub -> "-
  Mult -> "*
  Div -> "/
  Mod -> "/\n  Equal -> ",="
  Neq -> ",!="
  Less -> ",<"
  Leq -> ",\<="
  Greater -> ",>"
  Geq -> ",\>="
  And -> "and"
  Or -> "or"
  Incr -> ",++"
  Decr -> ",--"

let string_of_uop = function
  Neg -> "-
  Not -> "not"

let rec string_of_typ = function
  Num -> "num"
  Int -> "int"
  String -> "string"
  Bool -> "bool"
  Void -> "void"
  Arraytype(ty, len) -> string_of_typ ty ^ "]" ^ string_of_int len ^ "]"
  QueueType(ty) -> "Queue " ^ string_of_typ ty
  BSTreeType(ty) -> "BSTree " ^ string_of_typ ty
  LinkedListType(ty) -> "LinkedList " ^ string_of_typ ty
  StackType(ty) -> "Stack " ^ string_of_typ ty
  AnyType -> "AnyType"
  NumberType -> "NumberType"

let rec string_of_expr = function
  StringLit(s) -> s
  NumLit(f) -> string_of_float f
  IntLit(i) -> string_of_int i
  BoolLit(true) -> "true"
| BoolLit(false) -> "false" |
| Null -> "null" |
| Id(s) -> s |
| Binop(e1, o, e2) -> string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2 |
| Unop(o, e) -> string_of_uop o ^ string_of_expr e |
| Postop(e, o) -> string_of_expr e ^ string_of_op o |
| Assign(t, v, e) -> string_of_typ t ^ " " ^ v ^ " = " ^ string_of_expr e |
| Reassign(v, e) -> v ^ "=" ^ string_of_expr e |
| FuncCall(f, el) -> f ^ "(" ^ String.concat "\), " ^ (List.map string_of_expr el) ^ ")" |
| ObjectCall(o, f, e1) -> string_of_expr o ^ "." ^ f ^ "\(" ^ String.concat "\), " ^ (List.map string_of_expr e1) ^ ")" |
| ArrayLit a -> "[" ^ String.concat "\], " ^ (List.map string_of_expr a) ^ "]" |
| ArrayAccess(v, i) -> v ^ "[" ^ string_of_expr i ^ "]" |
| ArrayElementAssign(s, i, e) -> s ^ "[" ^ string_of_expr i ^ "] = " ^ string_of_expr e |
| Noexpr -> "" |
| QueueLit(typ, e1) -> "new " ^ "Queue" ^ "::" ^ string_of_typ typ ^ ";
" ^ (List.map string_of_expr e1) ^ ")" |
| LinkedListLit(typ, e1) -> "new " ^ "LinkedList" ^ "::" ^ string_of_typ typ ^ ";
" ^ (List.map string_of_expr e1) ^ ")" |
| BSTreeLit(typ, e1) -> "new " ^ "BSTree" ^ "::" ^ string_of_typ typ ^ ";
" ^ (List.map string_of_expr e1) ^ ")" |
| StackLit(typ, e1) -> "new " ^ "Stack" ^ "::" ^ string_of_typ typ ^ ";
" ^ (List.map string_of_expr e1) ^ ")" |

let rec string_of_stmt = function |
| Block(stmts) -> |
| "\n" ^ String.concat "\n" ^ (List.map string_of_stmt stmts) ^ ";
" |
| Expr(expr) -> string_of_expr expr ^ ";
" |
| Return(expr) -> "return " ^ string_of_expr expr ^ ";
" |
| If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s |
| If(e, s1, s2) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2 |
| For(e1, e2, e3, s) -> |

53
"for (" ^ string_of_expr e1 ^ " ; " ^ string_of_expr e2 ^ " ; " ^ string_of_expr e3 ^ ") " ^ string_of_stmt s
| While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s

let string_of_vdecl (t, id) = string_of_typ t ^ " " ^ id ^ ";\n"

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

let string_of_program (funcs) =
    String.concat "\n" (List.map string_of_fdecl funcs)

8.4 semant.ml

Josh, Kofi, Millie, Sophie

(* Semantic checking for the MicroC compiler *)
open Ast

module StringMap = Map.Make(String)

(* Semantic checking of a program. Returns void if successful, *
* throws an exception if something is wrong. Checks each *
* function *)

let check (functions) =

  (* Raise an exception if the given list has a duplicate *)
  let report_duplicate exceptf list =
    let rec helper = function
      | n1 :: n2 :: _ when n1 = n2 -> raise (Failure (exceptf ^=
          n1))
      | _ :: t -> helper t
      | [] -> ()
    in helper (List.sort compare list)
  in


(* Raise an exception if a given binding is to a void type *)
let check_not_void exceptf = function
  (Void, n) -> raise (Failure (exceptf n))
| _ -> ()
in

(* Raise an exception of the given rvalue type cannot be
assigned to
the given lvalue type *)
let check_assign lvaluet rvaluet err =
  if lvaluet = rvaluet then rvaluet
  else if lvaluet = Num && rvaluet = AnyType then lvaluet
  else if lvaluet = Int && rvaluet = AnyType then lvaluet
  else if lvaluet = String && rvaluet = AnyType then lvaluet
  else if lvaluet = Bool && rvaluet = AnyType then lvaluet
  else if lvaluet = Num && rvaluet = NumberType then lvaluet
  else if lvaluet = Int && rvaluet = NumberType then lvaluet
  else raise err
in

(**** Checking Functions ****)
if List.mem "print" (List.map (fun fd -> fd.fname) functions)
then raise (Failure ("function print may not be defined"))
  else ();
if List.mem "delete" (List.map (fun fd -> fd.fname) functions)
then raise (Failure ("function delete may not be defined"))
  else ();
report_duplicate (fun n -> "duplicate function " ^ n)
(List.map (fun fd -> fd.fname) functions);

(* Function declaration for a named function *)
let built_in_decls = StringMap.add "print"
  { typ = Void; fname = "print"; formals = [(Num, "x")];
    body = [] }

  (StringMap.add "printb"
  { typ = Void; fname = "printb"; formals = [(Bool, "x")];
    body = [] })

  (StringMap.add "add"
  { typ = Void; fname = "add"; formals = [(AnyType, "x")];
    body = [] })

  (StringMap.add "show"
{ typ = Void; fname = "show"; formals = [];
  body = [] }

(StringMap.add "peek"
{ typ = AnyType; fname = "peek"; formals = [];
  body = [] })

(StringMap.add "remove"
{ typ = Void; fname = "remove"; formals = [];
  body = [] })

(StringMap.add "get"
{ typ = AnyType; fname = "get"; formals = [(Int, "x")];
  body = [] })

(StringMap.add "size"
{ typ = Int; fname = "size"; formals = [];
  body = [] })

(StringMap.add "delete"
{ typ = Void; fname = "delete"; formals = [(NumberType,
\[gap\] "x")];
  body = [] })

(StringMap.add "contains"
{ typ = Bool; fname = "contains"; formals = [(NumberType,
\[gap\] "x")];
  body = [] })

(StringMap.add "fquickSort"
{ typ = Void; fname = "fquickSort"; formals = [];
  body = [] })

(StringMap.add "fshowQuickSort"
{ typ = Void; fname = "fshowQuickSort"; formals = [];
  body = [] })

(StringMap.add "quickSort"
{ typ = Void; fname = "quickSort"; formals = [];
  body = [] })

(StringMap.singleton "showQuickSort"
{ typ = Void; fname = "showQuickSort"; formals = [];
  body = [] })

))))))))))))
let function_decls = List.fold_left (fun m fd -> StringMap.add fd.fname fd m) built_in_decls functions

let function_decl s = try StringMap.find s function_decls with Not_found -> raise (Failure ("unrecognized function " ^ s))

let _ = function_decl "main" in (* Ensure "main" is defined *)

let check_function func =
  List.iter (check_not_void (fun n -> "illegal void formal " ^ n ^ " in " ^ func.fname)) func.formals;
  report_duplicate (fun n -> "duplicate formal " ^ n ^ " in " ^ func.fname) (List.map snd func.formals);

(* Store type and names of variables - formal *)
let variables = ref (List.fold_left (fun m (t, n) -> StringMap.add n t m) StringMap.empty (func.formals)) in

(* Helper: Returns type of identifier *)
let type_of_identifier name = try StringMap.find name (!variables) with Not_found -> raise (Failure ("undeclared identifier " ^ name)) in

(* Helper: Check if variable is already declared *)
let check_var_decl var_name err = if StringMap.mem var_name (!variables) then raise err in

let array_typ = function
  Arraytype(typ, _) -> typ
_ -> raise(Failure("Expecting an array and was not an array"))

let arr_lit_len = function
  ArrayLit(el) -> List.length el
  _ -> -1

let invalid_arr_size s e =
  let literal_len = arr_lit_len e in
  if literal_len == -1
  then false
  else
    let decl_len = (match s with
      Arraytype(_, len) -> len) in
    if decl_len == literal_len then false else true

let get_type = function
  QueueType(typ) -> typ
  | LinkedListType(typ) -> typ
  | StackType(typ) -> typ
  | BSTreeType(typ) -> typ
  | _ -> Void

(* Return the type of an expression or throw an exception *)

let rec expr = function
  NumLit _ -> Num
  | IntLit _ -> Int
  | StringLit _ -> String
  | QueueLit (t, _) -> QueueType(t)
  | LinkedListLit (t, _) -> LinkedListType(t)
  | StackLit (t, _) -> StackType(t)
  | BSTreeLit (t, _) -> BSTreeType(t)
  | BoolLit _ -> Bool
  | Id s -> type_of_identifier s
  | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 = expr e2 in
    (match op with
      Add | Sub | Mult | Div | Mod when t1 = Int && t2 = Int
        -> Int
      | Add | Sub | Mult | Div | Mod when t1 = Num && t2 = Num
        -> Num
      | Equal | Neq when t1 = t2
        -> Bool
    )
Less | Leq | Greater | Geq when t1 = Int && t2 = Int
→ Bool
Less | Leq | Greater | Geq when t1 = Num && t2 = Num
→ Bool
And | Or when t1 = Bool && t2 = Bool
→ Bool
_ -> raise (Failure ("illegal binary operator " ^
  string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
  string_of_typ t2 ^ " in " ^ string_of_expr e))
)
Postop (e, op) ->
  let t1 = expr e in
  (match op with
   | Incr when t1 = Int -> Int
   | Decr when t1 = Int -> Int
   | Incr when t1 = Num -> Num
   | Decr when t1 = Num -> Num
   | _ -> raise (Failure("illegal unary operator " ^
                 string_of_uop op ^ " on " ^ string_of_expr e))
  )
Unop(op, e) as ex -> let t = expr e in
  (match op with
   | Neg when t = Num -> Num
   | Neg when t = Int -> Int
   | Not when t = Bool -> Bool
   | _ -> raise (Failure("illegal unary operator " ^
                 string_of_uop op ^ " in " ^ string_of_expr e))
  )
Noexpr -> Void
Assign(typ, var, e) as ex ->
  let lt = (match typ with
            | Arraytype(t, _) -> t
            | _ -> typ
         ) in
  let rt = expr e
  and invalid_arr = invalid_arr_size typ e in
  if invalid_arr then raise (Failure("Invalid length
               declaration"))
  else
    if rt == Void then raise (Failure("must initialize
               variable with a value.
               
   else
    ignore (check_assign lt rt (Failure ("illegal
               assignment " ^ string_of_typ typ ^ " = " ^
               string_of_typ rt ^ " in " ^ string_of_expr e)));
    check_var_decl var (Failure("duplicate declaration of
               variable " ^ var));
    let _ =

(match func.fname with
  | _ -> variables := StringMap.add var typ
      <- (!variables)
  )
  in rt

| Reassign(var, e) as ex ->
  let rt = expr e and lt = type_of_identifier var in
  check_assign lt rt (Failure ("illegal assignment " ~
      <- string_of_typ lt ~
      " = " ~ string_of_typ rt ~ " in "
      <- ~ string_of_expr ex))
| FuncCall(fname, actuals) as call ->
  if fname = "print"
  then (if List.length actuals == 1
         then let arg_type = string_of_typ (expr (List.hd
             <- actuals)) in
             if arg_type = string_of_typ (Num) ||
                arg_type = string_of_typ (Int) ||
                arg_type = string_of_typ (String) ||
                arg_type = string_of_typ (Bool) ||
                arg_type = string_of_typ (AnyType)
                then Void
                else raise (Failure ("illegal actual
                    <- argument found in print " ~
                    <- string_of_typ (expr (List.hd
                        <- actuals)) ~
                    " in " ~ string_of_expr (List.hd
                        <- actuals)))
            else raise (Failure ("expecting 1 argument in " ~
                    <- string_of_expr call)))
  else let fd = function_decl fname in
    if List.length actuals != List.length fd.formals
    then raise (Failure ("expecting " ~ string_of_int
                    (List.length fd.formals) ~ " arguments in " ~
                    <- string_of_expr call))
    else List.iter2 (fun (ft, _) e -> let et = expr e in
                          ignore (check_assign ft et
                          (Failure ("illegal actual argument found " ~
                          <- string_of_typ et ~
                          " expected " ~ string_of_typ ft ~ " in " ~
                          <- string_of_expr e))))
        fd.formals actuals;
        fd.typ
  | ArrayLit(el) -> expr (List.nth el 0)
| ArrayAccess (var, el) as element ->
|   if expr el != Int
|   then raise (Failure "Invalid element access in " ~
|                 ~ string_of_expr element)
|   else array_typ (type_of_identifier var)
| ArrayElementAssign (s, i, e) as ex ->
|   let lt =
|     if expr i != Int
|     then raise (Failure "invalid element access in " ~
|                    ~ string_of_expr ex)
|     else array_typ (type_of_identifier s)
|     in
|     let rt = expr e in
|     check_assign lt rt (Failure "illegal assignment " ~
|                       ~ string_of_typ lt ~ " = " ~ string_of_typ rt ~ " in
|                       ~ " ~ string_of_expr ex));
| ObjectCall(oname, fname, actuals) as objectcall ->
|   let fd = function_decl fname in
|   let returntype = ref (fd.typ) in
|   if List.length actuals != List.length fd.formals then
|     raise (Failure "expecting " ~ string_of_int
|             ~ (List.length fd.formals) ~
|             " arguments in " ~ string_of_expr objectcall))
|   else
|     List.iter2 (fun (ft, _) e -> let et = expr e in
|                   if fname = "add" then
|                     let acttype = expr oname in
|                     let actqtype = get_type acttype in
|                     ignore(check_assign actqtype et (Failure ~
|                     ~ "illegal actual add argument found " ~
|                     ~ string_of_typ et ~
|                     ~ " expected " ~ string_of_typ actqtype ~ " in " ~
|                     ~ ~ string_of_expr e)))
|                   else if fname = "remove" then
|                     let acttype = expr oname in
|                     let actqtype = get_type acttype in
|                     ignore(check_assign actqtype et (Failure ~
|                     ~ "illegal actual remove argument found " ~
|                     ~ string_of_typ et ~
|                     ~ " expected " ~ string_of_typ actqtype ~ " in " ~
|                     ~ ~ string_of_expr e)))
|                   else if fname = "quickSort" then
|                     let acttype = expr oname in
|                     let actatype = array_typ acttype in
|                     ignore(check_assign acttype et (Failure ~
|                     ~ "illegal actual quickSort argument found " ~
|                     ~ string_of_typ et ~
|                     ~ " expected " ~ string_of_typ acttype ~ " in " ~
|                     ~ ~ string_of_expr e)))
|                   else
|                     let et = expr e in
|                     if fname = "add" then
|                       let acttype = expr oname in
|                       let actqtype = get_type acttype in
|                       ignore(check_assign actqtype et (Failure ~
|                       ~ "illegal actual add argument found " ~
|                       ~ string_of_typ et ~
|                       ~ " expected " ~ string_of_typ actqtype ~ " in " ~
|                       ~ ~ string_of_expr e)))
ignore(check_assign actotype et (Failure ~
  "illegal actual size argument found " ~
  string_of_typ et ~
  " expected " ~ string_of_typ actatype ~ " in "
  ~ string_of_expr e)))

else if fname = "delete" then
  let acttype = expr oname in
  match acttype with
  | BSTreeType(actqtype) -> ignore(check_assign
    actqtype et (Failure ("illegal actual
    delete argument found " ~ string_of_typ
    et ~
    " expected " ~ string_of_typ actqtype ~ " type
    ~ " in " ~ string_of_expr e)))
  | LinkedListType(_) -> ignore(check_assign
    Int et (Failure ("illegal actual delete
    argument found " ~ string_of_typ et ~
    " expected int type " ~ " in "
    ~ string_of_expr e)))

else if fname = "contains" then
  let acttype = expr oname in
  match acttype with
  | BSTreeType(actqtype) -> ignore(check_assign
    actqtype et (Failure ("illegal actual
    contains argument found " ~ string_of_typ
    et ~
    " expected " ~ string_of_typ actqtype ~ " type
    ~ " in " ~ string_of_expr e)))
  | _ -> raise (Failure (".contains() is only
    applicable to the the tree data type"))

else ignore (check_assign ft et (Failure
  "illegal actual argument found " ~
  string_of_typ et ~
  " expected " ~ string_of_typ ft ~ " in " ~
  ~ string_of_expr e)))

!returntype

let check_bool_expr e = if expr e != Bool
  then raise (Failure ("expected Boolean expression in " ~
    ~ string_of_expr e))
  else () in
let rec stmt = function
  Block sl -> let rec check_block = function
            [Return _ as s] -> stmt s
            | Return _ :: _ -> raise (Failure "nothing may follow a
           ← return")
            | Block sl :: ss -> check_block (sl @ ss)
            | s :: ss -> stmt s ; check_block ss
            | [] -> ()
           in check_block sl
  | Expr e -> ignore (expr e)
  | Return e -> let t = expr e in if t = func.typ then ()
           ← else
           raise (Failure ("return gives " ^ string_of_typ t ^ 
           ← expected " ^ string_of_typ func.typ ^ 
           ← string_of_expr e))
  | If(p, b1, b2) -> check_bool_expr p; stmt b1; stmt b2
  | For(e1, e2, e3, st) -> ignore (expr e1); check_bool_expr
           ← e2;
           ignore (expr e3); stmt st
  | While(p, s) -> check_bool_expr p; stmt s
in
stmt (Block func.body)
in
List.iter check_function functions

8.5 codegen.ml

Josh, Kofi, Millie, Sophie

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

module L = Llvm
module A = Ast
module StringMap = Map.Make(String)
let context = L.global_context ()
let queuem = L.MemoryBuffer.of_file "queue.bc"
let listm = L.MemoryBuffer.of_file "linkedlist.bc"
let stackm = L.MemoryBuffer.of_file "stack.bc"
let quicksortm = L.MemoryBuffer.of_file "quicksort.bc"
let bstreem = L.MemoryBuffer.of_file "bstree.bc"
let qqm = Llvm_bitreader.parse_bitcode context queuem
let list_qm = Llvm_bitreader.parse_bitcode context listm
let stack_qm = Llvm_bitreader.parse_bitcode context stackm
let bstree_qm = Llvm_bitreader.parse_bitcode context bstreem
let quicksort_qm = Llvm_bitreader.parse_bitcode context quicksortm

let the_module = L.create_module context "Strux"
and f_t = L.double_type context (* float *)
and i8_t = L.i8_type context (* print type *)
and i1_t = L.i1_type context (* bool type *)
and void_t = L.void_type context (* void type *)
and str_t = L.pointer_type (L.i8_type context) (* string *)
and i32_t = L.i32_type context
and queue_t = L.pointer_type (match L.type_by_name qqm
  "struct.Queue") | Some x -> x)
and stack_t = L.pointer_type (match L.type_by_name stack_qm
  "struct.Stack") | Some x -> x)
and linkedlist_t = L.pointer_type (match L.type_by_name list_qm
  "struct.LinkedList") | Some x -> x)
and bstree_t = L.pointer_type (match L.type_by_name bstree_qm
  "struct.BSTree") | Some x -> x)

let rec ltype_of_typ = function (* LLVM type for AST type *)
  | A.Num -> f_t
  | A.Int -> i32_t
  | A.String -> str_t
  | A.Bool -> i1_t
  | A.Void -> void_t
  | A.Arraytype(t, _) -> L.pointer_type (ltype_of_typ t)
  | A.QueueType _ -> queue_t
  | A.LinkedListType _ -> linkedlist_t
  | A.BSTreeType _ -> bstree_t
  | A.StackType _ -> stack_t
  | A.QueueType _ -> queue_t
  | A.LinkedListType _ -> linkedlist_t
| A.BSTreeType _ -> bstree_t |
| A.StackType _ -> stack_t |
| A.AnyType -> str_t |
| A.NumberType -> str_t |
| _ -> raise(Failure("Invalid Data Type")) |

and translate (functions) =

(* -------BUILT IN FUNCTIONS----------- *)

(* print *)
let printf_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t []] in
let printf_func = L.declare_function "printf" printf_t the_module in

(* built-in queue functions *)
let initQueue_t = L.function_type queue_t [| ] in
let initQueue_f = L.declare_function "initQueue" initQueue_t the_module in
let enqueue_t = L.function_type void_t [| queue_t; L.pointer_type i8_t[]] in
let enqueue_f = L.declare_function "enqueue" enqueue_t the_module in
let dequeue_t = L.function_type void_t [| queue_t ] in
let dequeue_f = L.declare_function "dequeue" dequeue_t the_module in
let peek_t = L.function_type (L.pointer_type i8_t) [| queue_t ] in
let peek_f = L.declare_function "peek" peek_t the_module in
let sizeQ_t = L.function_type i32_t [| queue_t ] in
let sizeQ_f = L.declare_function "queue_size" sizeQ_t the_module in
let q_show_t = L.function_type void_t [| queue_t ] in
let q_show_int = L.declare_function "queue_show_int" q_show_t the_module in
let q_show_float = L.declare_function "queue_show_float" q_show_t the_module in
let q_show_string = L.declare_function "queue_show_string" q_show_t the_module in

(*built-in linkedlist functions*)
let initList_t = L.function_type linkedlist_t [| ] in
let initList_f = L.declare_function "initList" initList_t the_module in
let add_t = L.function_type void_t [| linkedlist_t; L.pointer_type i8_t[]] in
let add_f = L.declare_function "add" add_t the_module in  
let delete_t = L.function_type void_t [| linkedlist_t; i32_t → |] in  
let delete_f = L.declare_function "delete" delete_t the_module in  
let get_t = L.function_type (L.pointer_type i8_t) [| linkedlist_t; i32_t |] in  
let get_f = L.declare_function "get" get_t the_module in  
let sizeList_t = L.function_type i32_t [| linkedlist_t |] in  
let sizeList_f = L.declare_function "size" sizeList_t the_module in  
let l_show_t = L.function_type void_t [| linkedlist_t |] in  
let l_show_int = L.declare_function "ll_show_int" l_show_t the_module in  
let l_show_float = L.declare_function "ll_show_float" l_show_t the_module in  
let l_show_string = L.declare_function "ll_show_string" l_show_t the_module in  
(*built-in stack functions*)  
let initStack_t = L.function_type stack_t [| |] in  
let initStack_f = L.declare_function "initStack" initStack_t the_module in  
let push_t = L.function_type void_t [| stack_t; L.pointer_type i8_t |] in  
let push_f = L.declare_function "push" push_t the_module in  
let pop_t = L.function_type void_t [| stack_t |] in  
let pop_f = L.declare_function "pop" pop_t the_module in  
let top_t = L.function_type (L.pointer_type i8_t) [| stack_t |] in  
let top_f = L.declare_function "top" top_t the_module in  
let sizeS_t = L.function_type i32_t [| stack_t |] in  
let sizeS_f = L.declare_function "stack_size" sizeS_t the_module in  
let s_show_t = L.function_type void_t [| stack_t |] in  
let s_show_int = L.declare_function "stack_show_int" s_show_t the_module in  
let s_show_float = L.declare_function "stack_show_float" s_show_t the_module in  
let s_show_string = L.declare_function "stack_show_string" s_show_t the_module in  

(* quicksort array functions *)  
(*takes in int[] to do quick sort *)
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
let int_quickSort_f = L.declare_function "cQuickSort"
  "int_quickSort_t the_module in
let int_show_quickSort_t = L.function_type void_t [|
  L.pointer_type (ltype_of_typ A.Int); i32_t |] in
let int_show_quickSort_f = L.declare_function "cShowQuickSort"
  "int_show_quickSort_t the_module in

(*takes in num[] to do quick sort *)
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
let num_quickSort_f = L.declare_function "cQuickfSort"
  "num_quickSort_t the_module in
let num_show_quickSort_t = L.function_type void_t [|
  L.pointer_type (ltype_of_typ A.Num); i32_t |] in
let num_show_quickSort_f = L.declare_function "cShowfQuickSort" num_show_quickSort_t the_module in

(*takes in string[] to do quick sort *)
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
let string_quickSort_f = L.declare_function "cQuicksSort"
  "string_quickSort_t the_module in
let string_show_quickSort_t = L.function_type void_t [|
  L.pointer_type (ltype_of_typ A.String); i32_t |] in
let string_show_quickSort_f = L.declare_function "cShowsQuickSort" string_show_quickSort_t the_module in

(*built-in bstree functions*)
let initBSTree_t = L.function_type bstree_t [ | | ] in
let initBSTree_t = L.declare_function "initBSTree"
  "initBSTree_t the_module in
let bstreeadd_t = L.function_type void_t [ | bstree_t; |
  L.pointer_type i8_t[] |] in
let bstreeadd_int_f = L.declare_function "addIntToTree"
  "bstreeadd_t the_module in
let bstreeadd_float_f = L.declare_function "addNumToTree"
  "bstreeadd_float_f the_module in
let bstreeadd_int_t = L.function_type void_t [ | bstree_t; |
  i32_t[] |] in
let bstreeadd_int_f = L.declare_function "deleteIntFromTree" bstreeadd_int_t the_module in
let bstreeadd_float_f = L.declare_function "deleteNumFromTree" bstreeadd_float_f the_module in
let bstreeadd_int_t = L.function_type void_t [ | bstree_t; |
  i32_t[] |] in
let bstreeadd_int_f = L.declare_function "deleteIntFromTree" bstreeadd_int_t the_module in

(*built-in bstree functions*)
let bstree_delete_float_t = L.function_type void_t [ bstree_t; f_t] in

let bstree_delete_float_f = L.declare_function "deleteNumFromTree" bstree_delete_float_t the_module in

let bstreecontains_int_t = L.function_type i1_t [ bstree_t; i32_t] in

let bstreecontains_int_f = L.declare_function "treeContainsInt" bstreecontains_int_t the_module in

let bstreecontains_float_t = L.function_type i1_t [ bstree_t; f_t] in

let bstreecontains_float_f = L.declare_function "treeContainsFloat" bstreecontains_float_t the_module in

let bstree_show_t = L.function_type void_t [ bstree_t ] in

let bstree_show_int = L.declare_function "showIntTree" bstree_show_t the_module in

let bstree_show_float = L.declare_function "showNumTree" bstree_show_t the_module in

(*print big *)

let printbig_t = L.function_type i32_t [ i32_t ] in

let printbig_func = L.declare_function "printbig" printbig_t the_module in

(* Define each function (arguments and return type) so we can call it *)

let function_decls =

let function_decl m func_decl =
  let name = func_decl.A.fname
  and formal_types =
    Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) func_decl.A.formals)
  in let ftype = L.function_type (ltype_of_typ func_decl.A.typ) formal_types in
  StringMap.add name (L.define_function name ftype the_module, func_decl) m in

  List.fold_left function_decl StringMap.empty functions in

(* Format str for printf *)

let string_format_str b = L.build_global_stringptr "%s\n" "fmt" b

and int_format_str b = L.build_global_stringptr "%d\n" "fmt" b

and float_format_str b = L.build_global_stringptr "%f\n" "fmt" b in

let format_str x_type builder =
let b = builder in
  match x_type with
  | A.Int -> int_format_str b
  | A.Num -> float_format_str b
  | A.String -> string_format_str b
  | A.Bool -> int_format_str b
  | _ -> raise (Failure ("Invalid printf type"))

let int_format_zeroth b = L.build_global_stringptr "[\%d"
  ↪ "fmt" b
and num_format_zeroth b = L.build_global_stringptr "[\%f"
  ↪ "fmt" b
and string_format_zeroth b = L.build_global_stringptr "[\%s"
  ↪ "fmt" b in

let int_format_arr b = L.build_global_stringptr ", \%d"
  ↪ "fmt" b
and num_format_arr b = L.build_global_stringptr ", \%f"
  ↪ "fmt" b
and string_format_arr b = L.build_global_stringptr ", \%s"
  ↪ "fmt" b in

let int_format_last b = L.build_global_stringptr ", \%d]\n"
  ↪ "fmt" b
and num_format_last b = L.build_global_stringptr ", \%f]\n"
  ↪ "fmt" b
and string_format_last b = L.build_global_stringptr ", \%s]\n"
  ↪ "fmt" b in

let format_arr_print x_type index length builder =
  let b = builder in
  if index == 0 then
    match x_type with
    | A.Int -> int_format_zeroth b
    | A.Num -> num_format_zeroth b
    | A.String -> string_format_zeroth b
    | A.Bool -> int_format_zeroth b
    | _ -> raise (Failure ("Invalid array print type"))
  else if index == length - 1 then
    match x_type with
    | A.Int -> int_format_last b
    | A.Num -> num_format_last b
    | A.String -> string_format_last b
    | A.Bool -> int_format_last b
    | _ -> raise (Failure ("Invalid array print type"))
else
  match x_type with
  | A.Int -> int_format_arr b
  | A.Num -> num_format_arr b
  | A.String -> string_format_arr b
  | A.Bool -> int_format_arr b
  | _ -> raise (Failure ("Invalid array print type"))
in

let build_function_body func_decl =
  let (the_function, _) = StringMap.find func_decl.A.fname
  in
  let llbuilder = L.builder_at_end context (L.entry_block the_function) in

  (* Construct the function's "locals": formal arguments and
   * locally declared variables. Allocate each on the stack,
   * initialize their value, if appropriate, and remember their values in the
   * "locals" map *)

  (* function to add local variables to a map *)
  let local_vars =
    let add_formal m (t,n) p = L.set_value_name n p;
    let local = L.build_alloca (ltype_of_typ t) n llbuilder
    in
    ignore (L.build_store p local llbuilder);
    StringMap.add n (t,local) m in

  (* add formals to the map *)
  ref (List.fold_left2 add_formal StringMap.empty
      func_decl.A.formals
      (Array.to_list (L.params the_function))) in

  (* Return the value or the type for a variable or formal
   * argument *)
  (* All the tables have the structure (type, llvalue) *)

  let lookup n : L.llvalue =
    try (snd (StringMap.find n !local_vars))
    with Not_found -> (raise (Failure ("Variable not found!")))
    in

  let name_to_type n : A.typ =
try (fst (StringMap.find n !local_vars))
with Not_found -> (raise (Failure ("Variable not found")))

(* Array creation, initialization, access *)

let create_array t len builder =
    let ltype = ltype_of_typ t in
    let size_t = L.build_intcast (L.size_of ltype) i32_t "tmp" builder in
    let total_size = L.build_mul size_t len "tmp" builder in
    let total_size = L.build_add total_size (L.const_int i32_t 1) "tmp" builder in
    let arr_malloc = L.build_array_malloc ltype total_size "tmp" builder in
    let arr = L.build_pointercast arr_malloc (L.pointer_type ltype) "tmp" builder in
    arr

let initialize_array t el builder =
    let len = L.const_int i32_t (List.length el) in
    let arr = create_array t len builder in
    let _ =
        let assign_value i =
            let index = L.const_int i32_t i in
            let index = L.build_add index (L.const_int i32_t 1) "tmp" builder in
            let _val = L.build_gep arr [| index |] "tmp" builder in
            L.build_store (List.nth el i) _val builder
        in
        for i = 0 to (List.length el)-1 do
            ignore (assign_value i)
        done
    in
    arr

let access_array arr index assign builder =
    let index = L.build_add index (L.const_int i32_t 1) "tmp" builder in
    let arr = L.build_load arr "tmp" builder in
    let _val = L.build_gep arr [| index |] "tmp" builder in
    if assign then _val else L.build_load _val "tmp" builder in
let get_array_len = function
  A.Id name -> (match (name_to_type name) with
     A.Arraytype(_, len) -> len
     | _ -> raise (Failure "Can't get the length of this object"))
in

let is_array = function
  A.Id name -> (match (name_to_type name) with
     A.Arraytype(_, _) -> true
     | _ -> false)
in

let rec get_array_index e = match e with
  A.IntLit x -> x
  | A.Binop (e1, op, e2) -> (match op with
    A.Add -> (get_array_index e1) + (get_array_index e2)
    | A.Sub -> (get_array_index e1) - (get_array_index e2)
    | A.Mult -> (get_array_index e1) * (get_array_index e2)
    | A.Div -> (get_array_index e1) / (get_array_index e2)
    | _ -> 0 (* If index is a variable we can't check, so default to 0 *)
  )
in

let get_type = function
  A.Id name -> (match (name_to_type name) with
     A.QueueType(typ) -> typ
     | A.LinkedListType(typ) -> typ
     | A.BSTreeType(typ) -> typ
     | A.StackType(typ) -> typ
     | A.Arraytype(typ, _) -> typ
     | _ as typ -> typ)
in

let rec gen_type = function
  A.IntLit _ -> A.Int
  | A.NumLit _ -> A.Num
  | A.StringLit _ -> A.String
  | A.BoolLit _ -> A.Bool
| A.ArrayLit el -> A.Arraytype (gen_type (List.nth el 0), \n   \rightarrow List.length el) |
| A.ArrayElementAssign (_, _, el) -> gen_type el |
| A.Id name -> (match (name_to_type name) with |
   | A.Arraytype(t, _) -> t |
   | _ as ty -> ty) |
| A.Unop(_,e) -> gen_type e |
| A.Binop(e1,_,_) -> gen_type e1 |
| A.Postop(e, _) -> gen_type e |
| A.Assign(_,var,_) -> gen_type (A.Id(var)) |
| A.Reassign(var,_) -> gen_type (A.Id(var)) |
| A.FuncCall(var,_) -> let fdecl = |
   | List.find (fun x -> x.A.fname = var) |
   | \rightarrow functions in |
   | fdecl.A.typ |
| A.ArrayAccess(id,_) -> gen_type (A.Id(id)) |
| A.ObjectCall(_, "size", _) -> A.Int |
| A.ObjectCall(obj, "peek", _) -> get_type obj |
| A.ObjectCall(obj, "get", _) -> get_type obj |
| A.ObjectCall(obj, "contains", _) -> A.Bool |
| A.Noexpr -> A.Void |

let call_size_ptr = function |
| A.Id name -> (match (name_to_type name) with |
   | A.QueueType _ -> sizeQ_f |
   | A.LinkedListType _ -> sizeList_f |
   | A.StackType _ -> sizeS_f |
   | _ -> raise (Failure ("Invalid data structure type - size \n   \rightarrow function"))) |

let call_add_ptr ds_type = function |
| A.Id name -> (match (name_to_type name) with |
   | A.QueueType _ -> enqueue_f |
   | A.LinkedListType _ -> add_f |
   | A.StackType _ -> push_f |
   | A.BSTreeType _ -> (match ds_type with |
   | A.Int -> bstreeadd_int_f |
   | A.Num -> bstreeadd_float_f |
   | _ -> raise (Failure ("Invalid data structure type - add \n   \rightarrow function"))) |

let call_pop_ptr = function |
| A.Id name -> (match (name_to_type name) with |
A.QueueType _ -> dequeue_f
| A.StackType _ -> pop_f
| _ -> raise (Failure ("Invalid data structure type - delete function"))

let call_show_ptr data_type = function
  A.Id name -> (match (name_to_type name) with
    A.QueueType _ -> (match data_type with
      A.Int -> q_show_int
    | A.Num -> q_show_float
    | A.Bool -> q_show_int
    | A.String -> q_show_string)
    | A.StackType _ -> (match data_type with
      A.Int -> s_show_int
    | A.Num -> s_show_float
    | A.Bool -> s_show_int
    | A.String -> s_show_string)
    | A.LinkedListType _ -> (match data_type with
      A.Int -> l_show_int
    | A.Num -> l_show_float
    | A.Bool -> l_show_int
    | A.String -> l_show_string)
    | A.BSTreeType _ -> (match data_type with
      A.Int -> bstree_show_int
    | A.Num -> bstree_show_float)
    | _ -> raise (Failure ("Invalid data structure type - show function")))
  in

let call_delete_ptr ds_type = function
  A.Id name -> (match (name_to_type name) with
    A.LinkedListType _ -> delete_f
    | A.BSTreeType _ -> (match ds_type with
      A.Int -> bstree delete_int_f
    | A.Num -> bstree delete_float_f)
    | _ -> raise (Failure ("Invalid data structure type - delete function")))
  in

let call_contains_ptr ds_type = function
  A.Id name -> (match (name_to_type name) with
    A.BSTreeType _ -> (match ds_type with
      A.Int -> bstree contains_int_f
    | A.Num -> bstree contains_float_f)
let call_quicksort_ptr data_type = function
  A.Id name -> (match (name_to_type name) with
    A.Arraytype(_, _) -> (match data_type with
      A.Int -> int_quickSort_f
    | A.Num -> num_quickSort_f
    | A.String -> string_quickSort_f)
    | _ -> raise (Failure ("Cannot perform quicksort on this
data_type")))
  in

let call_show_quicksort_ptr data_type = function
  A.Id name -> (match (name_to_type name) with
    A.Arraytype(_, _) -> (match data_type with
      A.Int -> int_show_quickSort_f
    | A.Num -> num_show_quickSort_f
    | A.String -> string_show_quickSort_f)
    | _ -> raise (Failure ("Cannot perform quicksort on this
data_type")))
  in

let call_peek_ptr = function
  A.Id name -> (match (name_to_type name) with
    A.QueueType _ -> peek_f
    | A.StackType _ -> top_f
    | _ -> raise (Failure ("Invalid data structure type - peek
function")))
  in

let init_bstree_add typ = match typ with
  | A.Int -> bstreeadd_int_f
  | A.Num -> bstreeadd_float_f
  | _ -> raise (Failure ("Invalid tree constructor"))
  in

let rec expr_generator llbuilder = function
  A.NumLit(n) -> L.const_float f_t n
  | A.IntLit(i) -> L.const_int i32_t i
  | A.BoolLit(b) -> L.const_int i1_t (if b then 1 else 0)
  | A.StringLit(s) -> L.build_global_stringptr s "string" llbuilder
  | A.Id s -> L.build_load (lookup s) s llbuilder
  | A.QueueLit (typ, act) ->
let d_ltyp = ltype_of_typ typ in
let queue_ptr = L.build_call initQueue_f [ | ] "init"
← llbuilder in
let add_element elem =
let d_ptr = match typ with
| A.QueueType _ -> expr_generator llbuilder elem
| _ ->
let element = expr_generator llbuilder elem in
let d_ptr = L.build_malloc d_ltyp "tmp" llbuilder in
ignore (L.build_store element d_ptr llbuilder);
← d_ptr in
let void_d_ptr = L.build_bitcast d_ptr (L.pointer_type i8_t) "ptr" llbuilder in
ignore (L.build_call enqueue_f [ | queue_ptr; void_d_ptr | ] "" llbuilder)
in ignore (List.map add_element act);
queue_ptr
| A.LinkedListLit (typ, act) ->
let d_ltyp = ltype_of_typ typ in
let list_ptr = L.build_call initList_f [ | ] "init"
← llbuilder in
let add_element elem =
let d_ptr = match typ with
| A.LinkedListType _ -> expr_generator llbuilder elem
| _ ->
let element = expr_generator llbuilder elem in
let d_ptr = L.build_malloc d_ltyp "tmp" llbuilder in
ignore (L.build_store element d_ptr llbuilder);
← d_ptr in
let void_d_ptr = L.build_bitcast d_ptr (L.pointer_type i8_t) "ptr" llbuilder in
ignore (L.build_call add_f [ | list_ptr; void_d_ptr | ] "" llbuilder)
in ignore (List.map add_element act);
list_ptr
| A.StackLit (typ, act) ->
let d_ltyp = ltype_of_typ typ in
let stack_ptr = L.build_call initStack_f [ | ] "init"
← llbuilder in
let add_element elem =
let d_ptr = match typ with
| A.StackType _ -> expr_generator llbuilder elem
| _ ->
let element = expr_generator llbuilder elem in
let d_ptr = L.build_malloc d_ltyp "tmp" llbuilder in
← d_ptr in
478  | ignore (L.build_store element d_ptr llbuilder);
479  |   ↓  d_ptr in
480 | let void_d_ptr = L.build_bitcast d_ptr (L.pointer_type
481 |   → i8_t) "ptr" llbuilder in
482 | ignore (L.build_call push_f [| stack_ptr; void_d_ptr
483 |   → |] "" llbuilder)
484 | in ignore (List.map add_element act);
485 | stack_ptr
486 | | A.BSTreeLit (typ, act) ->
487 | let d_ltyp = ltype_of_typ typ in
488 | let bstree_ptr = L.build_call initBSTree_f [| |] "init"
489 |   → llbuilder in
490 | let obj_method = init_bstree_add typ in
491 | let add_element elem =
492 |   let d_ptr = match typ with
493 |     | A.BSTreeType _ → expr_generator llbuilder elem
494 |     | _ →
495 |     | let element = expr_generator llbuilder elem in
496 |     | let d_ptr = L.build_malloc d_ltyp "tmp" llbuilder in
497 |     | ignore (L.build_store element d_ptr llbuilder);
498 |     |   ↓  d_ptr in
499 |     | let void_d_ptr = L.build_bitcast d_ptr (L.pointer_type
500 |       → i8_t) "ptr" llbuilder in
501 |     | ignore (L.build_call obj_method [| bstree_ptr;
502 |       → void_d_ptr |] "" llbuilder)
503 |     | in ignore (List.map add_element act);
504 | bstree_ptr
505 | | A.Binop (e1, op, e2) ->
506 | let e1' = expr_generator llbuilder e1 in
507 | let e2' = expr_generator llbuilder e2
508 | and num_ops = (match op with
509 |   | A.Add   -> L.build_fadd
510 |   | A.Sub   -> L.build_fsub
511 |   | A.Mult  -> L.build_fmul
512 |   | A.Div   -> L.build_fdiv
513 |   | A.Mod   -> L.build_frem
514 |   | A.And   -> L.build_and
515 |   | A.Or    -> L.build_or
516 |   | A.Equal -> L.build_fcmp L.Fcmp.Oeq
517 |   | A.Neq   -> L.build_fcmp L.Fcmp.One
518 |   | A.Less  -> L.build_fcmp L.Fcmp.Ult
519 |   | A.Leq   -> L.build_fcmp L.Fcmp.Ole
520 |   | A.Greater -> L.build_fcmp L.Fcmp.Ogt
521 |   | A.Geq   -> L.build_fcmp L.Fcmp.Oge
522 |   | _ → L.build_fcmp L.Fcmp.Oeq
523 | )
and int_ops = (match op with
  | A.Add -> L.build_add
  | A.Sub -> L.build_sub
  | A.Mult -> L.build_mul
  | A.Div -> L.build_sdiv
  | A.Mod -> L.build_srem
  | A.And -> L.build_and
  | A.Or -> L.build_or
  | A.Equal -> L.build_icmp L.Icmp.Eq
  | A.Neq -> L.build_icmp L.Icmp.Ne
  | A.Less -> L.build_icmp L.Icmp.Slt
  | A.Leq -> L.build_icmp L.Icmp.Sle
  | A.Greater -> L.build_icmp L.Icmp.Sgt
  | A.Geq -> L.build_icmp L.Icmp.Sge
  | _ -> L.build_icmp L.Icmp.Eq)

in

if ((L.type_of e1 = f_t) && (L.type_of e2 = f_t)) then
  num_ops e1 e2 "tmp" llbuilder
else int_ops e1 e2 "tmp" llbuilder
| A.Unop (op, e) ->
  let e' = expr_generator llbuilder e in
  let int_unops op e' = (match op with
    | A.Neg -> L.build_neg e' "tmp" llbuilder
    | A.Not -> L.build_not e' "tmp" llbuilder
  )
and num_unops op e' = (match op with
  | A.Neg -> L.build_fneg e' "tmp" llbuilder )

in
if ((L.type_of e' = f_t))
then num_unops op e'
else int_unops op e'
| A.Postop (e, op) ->
  let e' = expr_generator llbuilder e in
  let llval = (match e with
    | A.Id(s) -> s
    | _ -> raise (Failure("This input type cannot be
      incremented/decremented"))
  )
and op_typ = (match op with
    | A.Incr -> A.Add
    | A.Decr -> A.Sub
  )
and num_typ = if ((L.type_of e' = f_t))
then A.NumLit(1.0) 
else A.IntLit(1) in

expr_generator llbuilder (A.Reassign(llval, A.Binop(e, op_typ, num_typ)))

| A.Assign (t, s, e) ->
  let _ = (local_vars := StringMap.add s (t,  
  !lbuildaloca (ltype_of_typ t)) s llbuilder)  
  let e' = expr_generator llbuilder e and llval = lookup  
  in  
  s in
  if (len < index_int) || (index_int < 0)
  then raise (Failure ("Array index out of bounds"))
  else access_array llval index false llbuilder

| A.Reassign (s, e) ->
  let e' = expr_generator llbuilder e and llval = lookup  
  in
  s in
  if (len < index_int) || (index_int < 0)
  then raise (Failure ("Array index out of bounds"))
  else access_array llval index false llbuilder

| A.ArrayLit el -> let t = gen_type (List.nth el 0) in
  initialize_array t (List.map (expr_generator llbuilder)) llbuilder

| A.ArrayAccess (s, i) ->
  let index = expr_generator llbuilder i  
  and llval = lookup s in
  if (len < index_int) || (index_int < 0)
  then raise (Failure ("Array index out of bounds"))
  else access_array llval index false llbuilder

| A.ArrayElementAssign (s, i, e) ->
  let e' = expr_generator llbuilder e in
  let index = expr_generator llbuilder i in
  let llval = lookup s in
  let var = access_array llval index true llbuilder in
  ignore (L.build_store e' var llbuilder); e'

| A.FuncCall("print", [e]) ->
  let e' = expr_generator llbuilder e in
  let e_type = gen_type e in
  L.build_call printf_func [| (format_str e_type  
  !llbuilder); e' |] "printf" llbuilder

| A.FuncCall ("printbig", [e]) ->
  L.build_call printbig_func [| (expr_generator  
  !llbuilder e) |] "printbig" llbuilder

| A.FuncCall (f, act) ->
let (fdef, func_decl) = StringMap.find f
  function_decls in
let actuals = List.rev (List.map (expr_generator llbuilder) (List.rev act)) in
let result =
  (match func_decl.A.typ with
    A.Void -> ""
    | _ -> f ^ "_result") in
L.build_call fdef (Array.of_list actuals) result
  llbuilder

| A.ObjectCall (obj, "remove", []) ->
  let obj_val = expr_generator llbuilder obj in
  let obj_method = call_pop_ptr obj in
  ignore (L.build_call obj_method [| obj_val |] "" llbuilder); obj_val
| A.ObjectCall (obj, "peek", []) ->
  let obj_val = expr_generator llbuilder obj in
  let obj_type = get_type obj in
  let obj_method = call_peek_ptr obj in
  let val_ptr = L.build_call obj_method [| obj_val |]
    "val_ptr" llbuilder in
  let dtyp = ltype_of_typ obj_type in
  let ptr = L.build_bitcast val_ptr (L.pointer_type dtyp)
    "d_ptr" llbuilder in
  (L.build_load ptr "d_ptr" llbuilder)
| A.ObjectCall (obj, "show", []) ->
  let obj_val = expr_generator llbuilder obj in
  let obj_type = get_type obj in
  let print_array arr builder =
    let len = get_array_len obj in
    let llval = lookup (match arr with
      A.Id(s) -> s
      | _ -> raise (Failure("variable not found"))) in
    for index = 0 to len - 1 do
      let item = access_array llval (L.const_int i32_t
        index) false builder in
      (L.build_call printf_func [| (format_arr_print
        obj_type index len llbuilder); item |] "printf" llbuilder)
    done
  in
  if is_array obj then print_array obj llbuilder
else (let obj_method = call_show_ptr obj_type obj in
    ignore (L.build_call obj_method [| obj_val |] "")
    lbuilder));

obj_val

| A.ObjectCall (obj, "size", []) ->
  let e = expr_generator lbuilder obj in
  let obj_size = call_size_ptr obj in
  let size_ptr = L.build_call obj_size [| e |] "isEmpty"
    lbuilder in
  size_ptr

| A.ObjectCall (obj, "add", [e]) ->
  let obj_val = expr_generator lbuilder obj in
  let obj_type = get_type obj in
  let e_val = expr_generator lbuilder e in
  let d_ltyp = L.type_of e_val in
  let d_ptr = L.build_malloc d_ltyp "tmp" lbuilder in
  ignore (L.build_store e_val d_ptr lbuilder);
  let obj_method = call_add_ptr obj_type obj in
  let void_e_ptr = L.build_bitcast d_ptr (L.pointer_type i8_t) "ptr" lbuilder in
  ignore (L.build_call obj_method [| obj_val; void_e_ptr |]
    lbuilder); obj_val

| A.ObjectCall (obj, "delete", [e]) ->
  let obj_val = expr_generator lbuilder obj in
  let obj_type = get_type obj in
  let e_val = expr_generator lbuilder e in
  let obj_method = call_delete_ptr obj_type obj in
  ignore (L.build_call obj_method [| obj_val; e_val |] ""
    lbuilder);

obj_val

| A.ObjectCall (obj, "contains", [e]) ->
  let obj_val = expr_generator lbuilder obj in
  let obj_type = get_type obj in
  let e_val = expr_generator lbuilder e in
  let obj_method = call_contains_ptr obj_type obj in
  let result = L.build_call obj_method [| obj_val; e_val |]
    "res" lbuilder in
  result

| A.ObjectCall (l, "get", [e]) ->
  let l_ptr = expr_generator lbuilder l in
  let e_val = expr_generator lbuilder e in
  let l_type = get_type l in
  let val_ptr = L.build_call get_f [| l_ptr; e_val |]
    "val_ptr" lbuilder in
  let l_dtyp = ltype_of_typ l_type in
let d_ptr = L.build_bitcast val_ptr (L.pointer_type ~ l_dtyp) "d_ptr" llbuilder in
(L.build_load d_ptr "d_ptr" llbuilder)
| A.ObjectCall(a, "quickSort", []) ->
  let a_val = expr_generator llbuilder a in
  let len = L.const_int i32_t (get_array_len a) in
  let obj_method = call_quicksort_ptr (get_type a) a in
  ignore (L.build_call obj_method [| a_val; len |] ""
    ~ llbuilder); a_val
| A.ObjectCall(a, "showQuickSort", []) ->
  let a_val = expr_generator llbuilder a in
  let len = L.const_int i32_t (get_array_len a) in
  let obj_method = call_show_quicksort_ptr (get_type a) a
    ~ in
  ignore (L.build_call obj_method [| a_val; len |] ""
    ~ llbuilder); a_val in

(* Invoke "f llbuilder" if the current block doesn't
already have a terminal (e.g., a branch). *)
let add_terminal llbuilder f =
  match L.block_terminator (L.insertion_block llbuilder) with
  Some _ -> ()
| None -> ignore (f llbuilder) in

(* Statement generator *)
let rec stmt_generator llbuilder = function
| A.Block stmtlist -> List.fold_left stmt_generator
  llbuilder stmtlist
| A.Return e -> ignore (match func_decl.A.typ with
  A.Void -> L.build_ret_void llbuilder
  | _ -> L.build_ret
    ~ (expr_generator llbuilder e)
    ~ llbuilder); llbuilder
| A.Expr se -> ignore (expr_generator llbuilder se);
  llbuilder
| A.If (predicate, s1, s2) -> generate_if predicate s1 s2
  llbuilder
| A.While (predicate, body) -> generate_while predicate
  body llbuilder
| A.For (e1, e2, e3, s) -> stmt_generator llbuilder (A.Block [A.Expr e1 ; A.While (e2, A.Block [s ; A.Expr ~ e3]) ] )

and generate_if predicate s1 s2 llbuilder = 
  let bool_val = expr_generator llbuilder predicate in 
  let merge_bb = L.append_block context "merge" the_function in 

  let then_bb = L.append_block context "then" the_function in 
  add_terminal (stmt_generator (L.builder_at_end context then_bb) s1) 
  (L.build_br merge_bb); 

  let else_bb = L.append_block context "else" the_function in 
  add_terminal (stmt_generator (L.builder_at_end context else_bb) s2) 
  (L.build_br merge_bb); 

  ignore (L.build_cond_br bool_val then_bb else_bb llbuilder); 
  L.builder_at_end context merge_bb

and generate_while predicate body llbuilder = 
  let pred_bb = L.append_block context "while" the_function in 
  ignore (L.build_br pred_bb llbuilder); 

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

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

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

  (* Build the code for each statement in the function *) 
  let llbuilder = stmt_generator llbuilder (A.Block 
  → func_decl.A.body) in
add_terminal llbuilder (match func_decl.A.typ with
  | A.String -> L.build_ret (L.build_global_stringptr ""
    ↪
    →
  )
  | A.Void -> L.build_ret_void
  | A.Num -> L.build_ret (L.const_float f_t 0.)
  | A.Bool -> L.build_ret (L.const_int i1_t 0)
  | _ -> L.build_ret_void)

in
List.iter build_function_body functions;

8.6 strux.ml

Josh, Kofi, Millie, Sophie

module StringMap = Map.Make(String)

type action = Ast | LLVM_IR | Compile

let _ =
let action = ref Compile in
let set_action a () = action := a in
let speclist = [
  ("-a", Arg.Unit (set_action Ast), "Print the AST");
  ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated 
    LLVM IR");
  ("-c", Arg.Unit (set_action Compile),
    "Check and print the generated LLVM IR (default)");
] in
let usage_msg = "usage: ./strux.native [-a|-l|-c] 
	[file.strux]" in
let channel = ref stdin in
Arg.parse speclist (fun filename -> channel := open_in
  ⇔ 
  filename) usage_msg;
let lexbuf = Lexing.from_channel !channel in
let ast = Parser.program Scanner.token lexbuf in
Semant.check ast;
match !action with
  | Ast -> print_string (Ast.string_of_program ast)
8.7 bstree.h

Josh

```c
#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);
```
8.8 bstree.c

Josh

```c
#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)
```
{ /* Node is empty, place the new node here */
    if (node == NULL)
        return createNode(data);

    if (*((int *) data) < *((int *) node->data))
        node->left = addIntToTreeHelper(node->left, data);
    else if (*((int *) data) > *((int *) node->data))
        node->right = addIntToTreeHelper(node->right, data);

    return node;
}

void addIntToTree(struct BSTree *tree, void *data)
{
    if (tree->root == NULL) {
        tree->root = createNode(data);
    } else {
        addIntToTreeHelper(tree->root, data);
    }
}

/* Adds the given data to the tree */
struct BSTreeNode *addNumToTreeHelper(struct BSTreeNode *node, void *data)
{
    /* Node is empty, place the new node here */
    if (node == NULL)
        return createNode(data);

    if (*((double *) data) < *((double *) node->data))
        node->left = addNumToTreeHelper(node->left, data);
    else if (*((double *) data) > *((double *) node->data))
        node->right = addNumToTreeHelper(node->right, data);

    return node;
}

void addNumToTree(struct BSTree *tree, void *data)
{
    if (tree->root == NULL) {
        tree->root = createNode(data);
    } else {
        addNumToTreeHelper(tree->root, data);
    }
}
struct BSTreeNode *getMin(struct BSTreeNode *node)
{
    /* Invalid node given */
    if (node == NULL)
        return NULL;
    if (node->left)
        return getMin(node->left);
    else
        return node;
}

struct BSTreeNode *deleteIntFromTreeHelper(struct BSTreeNode *node, int data)
{
    struct BSTreeNode *temp;
    /* element not found */
    if (node == NULL)
        return NULL;
    if (data < *(int *) node->data)
        node->left = deleteIntFromTreeHelper(node->left, data);
    else if (data > *(int *) node->data)
        node->right = deleteIntFromTreeHelper(node->right, data);
    /* element found */
    else{
        /* two children present, handle accordingly */
        if (node->left && node->right) {
            temp = getMin(node->right);
            node->data = temp->data;
            node->right = deleteIntFromTreeHelper(node->right, *(int *) temp->data);
        }
        /* zero or one child present, handle accordingly */
        else {
            temp = node;
        }
        /* other cases */
    }
}
if (node->left == NULL)
    node = node->right;
else if (node->right == NULL)
    node = node->left;

free(temp);
}

return node;

void deleteIntFromTree(struct BSTree *tree, int data)
{
    if (tree->root->left == NULL && tree->root->right == NULL) {
        if ( *(int *) tree->root->data == data) {
            struct BSTreeNode *temp = tree->root;
            tree->root = NULL;
            free(temp);
        }
        return ;
    }
    deleteIntFromTreeHelper(tree->root, data);
}

struct BSTreeNode *deleteNumFromTreeHelper(struct BSTreeNode *node, double data)
{
    struct BSTreeNode *temp;
    /* element not found */
    if (node == NULL)
        return NULL;
    if ( data < *(double *) node->data)
        node->left = deleteNumFromTreeHelper(node->left, data);
    else if ( data > *(double *) node->data)
        node->right = deleteNumFromTreeHelper(node->right, data);
    /* element found */
else{
    /* two children present, handle accordingly */
    if (node->left && node->right) {
        temp = getMin(node->right);
        node->data = temp->data;
        node->right = deleteNumFromTreeHelper(node->right, *(int *) temp->data);
    }

    /* zero or one child present, handle accordingly */
    else {
        temp = node;
        if (node->left == NULL)
            node = node->right;
        else if (node->right == NULL)
            node = node->left;
        free(temp);
    }
}
return node;

void deleteNumFromTree(struct BSTree *tree, double data)
{
    if (tree->root->left == NULL && tree->root->right == NULL) {
        if ( *(double *) tree->root->data == data) {
            struct BSTreeNode *temp = tree->root;
            tree->root = NULL;
            free(temp);
        }
        return ;
    }
    deleteNumFromTreeHelper(tree->root, data);
}
```c
int treeContainsIntHelper(struct BSTreeNode *node, int data)
{
    /* Not found */
    if (node == NULL)
        return 0;

    /* Keep searching */
    if (data < *(int *) node->data)
        return treeContainsIntHelper(node->left, data);
    else if (data > *(int *) node->data)
        return treeContainsIntHelper(node->right, data);

    /* Element found */
    else
        return 1;
}

int treeContainsNumHelper(struct BSTreeNode *node, double data)
{
    /* Not found */
    if (node == NULL)
        return 0;

    /* Keep searching */
    if (data < *(double *) node->data)
        return treeContainsNumHelper(node->left, data);
    else if (data > *(double *) node->data)
        return treeContainsNumHelper(node->right, data);

    /* Element found */
    else
        return 1;
}

int treeContainsNum(struct BSTree *tree, double data)
{
    return treeContainsNumHelper(tree->root, data);
}

int treeContainsInt(struct BSTree *tree, int data)
{
    return treeContainsIntHelper(tree->root, data);
}
```
int treeHeight(struct BSTreeNode* node) {
    if (node == NULL) return 0;
    else {
        /* get the height of both children */
        int leftHeight = treeHeight(node->left);
        int rightHeight = treeHeight(node->right);

        /* take the larger of the two */
        if (leftHeight > rightHeight)
            return (leftHeight + 1);
        else
            return (rightHeight + 1);
    }
}

int printLeftChild(struct BSTreeNode *tree, int offset, int depth, char s[50][255], int typ) {
    char b[20];
    if (!tree) return 0;

    if (typ == FLOATING) {
        sprintf(b, "%.2lf", *(double *)tree->data);
    } else {
        sprintf(b, "%04d", *(int *)tree->data);
    }

    int left = printLeftChild(tree->left, offset, depth + 1, s, typ);
    int right = printRightChild(tree->right, offset + left + DATA_WIDTH, depth + 1, s, typ);

    for (int i = 0; i < DATA_WIDTH; i++)
        s[2 * depth][offset + left + i] = b[i];

    if (depth) {
        for (int i = 0; i < DATA_WIDTH + right; i++)
            s[2 * depth - 1][offset + left + DATA_WIDTH/2 + i] = '-';
    }
```c
s[2 * depth - 1][offset + left + DATA_WIDTH/2] = '*';
s[2 * depth - 1][offset + left + DATA_WIDTH + right +
    DATA_WIDTH/2] = '*';
}

return left + DATA_WIDTH + right;
}
int printRightChild(struct BSTreeNode *tree, int offset, int
    depth, char s[50][255], int typ)
{
    char b[20];
    if (!tree) return 0;
    if (typ == FLOATING)
        sprintf(b, "(%.2lf)", *(double *)tree->data);
    else
        sprintf(b, "(%04d)", *(int *)tree->data);
    int left = printLeftChild(tree->left, offset, depth + 1, s,
        typ);
    int right = printRightChild(tree->right, offset + left +
        DATA_WIDTH, depth + 1, s, typ);
    for (int i = 0; i < DATA_WIDTH; i++)
        s[2 * depth][offset + left + i] = b[i];
    if (depth) {
        for (int i = 0; i < left + DATA_WIDTH; i++)
            s[2 * depth - 1][offset - DATA_WIDTH/2 + i] = '-';
        s[2 * depth - 1][offset + left + DATA_WIDTH/2] = '*';
        s[2 * depth - 1][offset - DATA_WIDTH/2 - 1] = '*';
    }
    return left + DATA_WIDTH + right;
}
void showTree(struct BSTreeNode *root, int typ)
{
    int maxDepth = treeHeight(root);
    char s[50][255];
```
for (int i = 0; i < 50; i++)
    sprintf(s[i], "%80s", " ");

printLeftChild(root, 0, 0, s, typ);

int until;

if (maxDepth * 2 - 1 > 50)
    until = 50;
else
    until = maxDepth * 2 - 1;

for (int i = 0; i < maxDepth*2-1; i++)
    printf("%s
", s[i]);
}
}

void showIntTree(struct BSTree *tree)
{
    if (tree->root) {
        showTree(tree -> root, INTEGER);
        printf("\n%sn", "");
    } else {
        printf("%s
", "Tree is empty!");
    }
}

void showNumTree(struct BSTree *tree)
{
    if (tree->root) {
        showTree(tree -> root, FLOATING);
        printf("\n%sn", "");
    } else {
        printf("%s
", "Tree is empty!");
    }
}

8.9  linkedlist.c

Kofi, Sophie
#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 ListNode* list = (struct ListNode*) malloc(sizeof(struct ListNode));
    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++;
    }
}
```c
int isEmpty(struct LinkedList *list) {
    return (list->head == NULL);
}

void delete(struct LinkedList* list, int index) {
    if (isEmpty(list)) {
        return;
    }

    if (index == 0) {
        struct ListNode* temp = list->head;
        list->head = list->head->next;
        list->size--;
        free(temp);
        return;
    }

    int pointer = 0;
    struct ListNode* temp = list->head;
    while(pointer < index-1) {
        temp = temp->next;
        pointer++;
    }
    temp->next = temp->next->next;
    temp = temp->next;
    free(temp);
    list->size--;
}

void* get(struct LinkedList* list, int index) {
    struct ListNode* node = list->head;
    while (index > 0) {
        node = node->next;
        index--;
    }
    return node->data;
}

struct ListNode* access(struct LinkedList* list, int index) {
    struct ListNode* temp = list->head;
    while (index > 0) {
        temp = temp->next;
        index--;
    }
    return temp;
}
```c
int size(struct LinkedList *list) {
    return (list->size);
}

void printLlBorder(struct LinkedList *list, int typ) {
    int i;
    for (i = 0; i < list->size; i++) {
        int len = 0;
        char tmp[256];
        if (typ == INTEGER) {
            len = sprintf(tmp, " %d ", *(int *) get(list, i));
        } else if (typ == FLOATING) {
            len = sprintf(tmp, " %f ", *(double *) get(list, i));
        } else if (typ == STRING) {
            len = sprintf(tmp, " %s ", *(char **) get(list, i));
        }
        int j;
        printf("+");
        for (j = 0; j < len; j++) {
            printf("-");
        }
        printf("+ ");
        printf("+------+
");
    }
}

void printIndexes(struct LinkedList *list, int typ) {
    int i;
    for (i = 0; i < list->size; i++) {
        int len = 0;
        char tmp[256];
        if (typ == INTEGER) {
            len = sprintf(tmp, "| %d |->", *(int *) get(list, i));
        } else if (typ == FLOATING) {
            len = sprintf(tmp, "| %f |->", *(double *) get(list, i));
        } else if (typ == STRING) {
            len = sprintf(tmp, "| %s |->", *(char **) get(list, i));
        }
        printf("%-*d", len, i);
        printf("+-----+\n");
    }
}
```

void ll_simple_show(struct LinkedList *list, int typ) {
    int i;
    for (i = 0; i < list->size; i++) {
        printf("[ ");
        if (typ == INTEGER) {
            printf("%d", *(int *) get(list, i));
        } else if (typ == FLOATING) {
            printf("%f", *(double *) get(list, i));
        } else if (typ == STRING) {
            printf("%s", *(char **) get(list, i));
        }
        printf(" ]");
        if (access(list, i) -> next != NULL) {
            printf(" -> ");
        }
    }
    printf(" -> [ NULL ]\n");
}

void ll_show(struct LinkedList *list, int typ) {
    if (list->size == 0) {
        printf("LinkedList is empty!");
        return;
    }

    if (list->size > 10) {
        ll_simple_show(list, typ);
        return;
    }

    printLlBorder(list, typ);

    int i;
    for (i = 0; i < list->size; i++) {
        printf(" | %d |->", *(int *) get(list, i));
    } else if (typ == FLOATING) {
        printf(" | %f |->", *(double *) get(list, i));
    } else if (typ == STRING) {
        printf(" | %s |->", *(char **) get(list, i));
    }
8.10  queue.c

Millie, Sophie

```c
#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;
    return q;
}
```
```c
q->front = 0;
q->rear = 0;
return q;
}

int queue_size(struct Queue *queue) {
    return queue->size;
}

void enqueue(struct Queue *q, void *data) {
    struct Node *node = (struct Node*)malloc(sizeof(struct
→ Node));
    node->data = data;
    node->next = NULL;
    if (q->size == 0) {
        q->front = q->rear = node;
        q->size++;
        return;
    }
    q->rear->next = node;
    q->rear = node;
    q->size++;
}

void dequeue(struct Queue *q) {
    if (q->size == 0) {
        return;
    }
    struct Node *node = q->front;
    if (q->size == 1) {
        q->front = NULL;
        q->rear = NULL;
        q->size--;
    }
    else {
        q->front = q->front->next;
        q->size--;
    }
    free(node);
}

void *peek(struct Queue *q) {
    if(q->size == 0) {
        return NULL;
    }
    return q->front->data;
```
int printBorder(struct Queue *q, int typ) {
    int totalChars = 0;
    int size = q->size;

    int i;

    totalChars += printf("%s", "+");
    struct Node *curr = q->front;
    for (i = 0; i < size; i++) {
        int len = 0;
        char tmp[1000];
        if (typ == INTEGER) {
            len = sprintf(tmp, " %d ", *(int *) curr->data);
        } else if (typ == FLOATING) {
            len = sprintf(tmp, " %f ", *(double *) curr->data);
        } else if (typ == STRING) {
            len = sprintf(tmp, " %s ", *(char **) curr->data);
        }

        int j;
        for (j = 0; j < len; j++) {
            totalChars += printf("-");
        }
        totalChars += printf("+");
        curr = curr->next;
    }
    printf("\n");
    return totalChars;
}

void printHeadTail(int len) {
    int padding = len - strlen("Tail");
    printf("%-*s%s\n", padding, "Head", "Tail");
}

void queue_simple_show(struct Queue *q, int typ) {
    int i;
    struct Node *curr = q->front;
    printf("Head->");
    printf("\n");
    for (i = 0; i < q->size; i++) {

printf("[ ");
    if (typ == INTEGER) {
        printf("%d", *(int *) curr->data);
    } else if (typ == FLOATING) {
        printf("%f", *(double *) curr->data);
    } else if (typ == STRING) {
        printf("%s", *(char **) curr->data);
    }
    curr = curr->next;
    printf(" ]");
}
printf(" <-Tail
");
}

void queue_show(struct Queue *q, int typ) {
    if (q->size == 0) {
        printf("Queue is empty!");
        return;
    }
    if (q->size > 10) {
        queue_simple_show(q, typ);
        return;
    }

    // Print top border
    printBorder(q, typ);
    int i;
    int size = q->size;
    struct Node *curr = q->front;
    printf("%s", "|");
    for (i = 0; i < size; i++) {
        if (typ == INTEGER) {
            printf(" %d |", *(int *) curr->data);
        } else if (typ == FLOATING) {
            printf(" %f |", *(double *) curr->data);
        } else if (typ == STRING) {
            printf(" %s |", *(char **) curr->data);
        }
        curr = curr->next;
    }
    printf("\n");
    int totalChars = printBorder(q, typ);
    printHeadTail(totalChars);
```c
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

```c
#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 // allocate value of x to z, set x to y and y to z
26 z = *x;
27  *x = *y;
28  *y = z;
29  return ;
30 }
31
32
33 void sswap(char **x, char **y)
34 {
35  char *temp = *x;
36  *x = *y;
37  *y = temp;
38 }
39
// prints out int list
40 void display(int a[], int length) {
41  int i;
42  printf("[ ");
43  for(i = 1;i<=length;i++) {
44    printf("%d ",a[i]);
45  }
46  printf("]\n");
47 }
48
// prints out float list
49 void fdisplay(double a[], int length) {
50  int i;
51  printf("[ ");
52  for(i = 1;i<=length;i++) {
53    printf("%f ",a[i]);
54  }
55  printf("]\n");
56 }
57
// prints out string list
58 void sdisplay(char* a[], int length) {
59  int i;
60  printf("[ ");
61  for(i = 1;i<=length;i++) {
62    printf("%s ",a[i]);
63  }
64 }
// Function that uses median of three partitioning
int median_of_3(int a[], int left, int right)
{
    // get middle of array
    int middle = (left + right)/2;

    // rearrange arrange now with pivot known
    if (a[middle] < a[left]) {
        swap(&a[left], &a[middle]);
    }
    if (a[right] < a[left]) {
        swap(&a[left], &a[right]);
    }
    if (a[right] < a[middle]) {
        swap(&a[middle], &a[right]);
    }

    swap(&a[middle], &a[right - 1]);
    return a[right - 1];
}

double fmedian_of_3(double a[], int left, int right)
{
    // get middle of array
    int middle = (left + right)/2;

    // rearrange arrange now with pivot known
    if (a[middle] < a[left]) {
        fswap(&a[left], &a[middle]);
    }
    if (a[right] < a[left]) {
        fswap(&a[left], &a[right]);
    }
    if (a[right] < a[middle]) {
        fswap(&a[middle], &a[right]);
    }

    fswap(&a[middle], &a[right - 1]);
    return a[right - 1];
}
char *smedian_of_3(char a[], int left, int right)
{
    // get middle of array
    int middle = (left + right)/2;

    // rearrange arrange now with pivot known
    if (strcmp(a[middle], a[left]) < 0) {
        sswap(&a[left],&a[middle]);
    }
    if (strcmp(a[right], a[left]) < 0) {
        sswap(&a[left],&a[right]);
    }
    if (strcmp(a[right], a[middle]) < 0) {
        sswap(&a[middle],&a[right]);
    }
    sswap(&a[middle], &a[right - 1]);
    return a[right - 1];
}

void quickSort(int a[], int left, int right)
{
    if (left < right) {
        int pivot = median_of_3(a,left,right);
        if (left == right - 1) return;
        int i = left;
        int j = right - 1;
        for ( ; ;) {
            while(a[++i]<pivot) {}
            while(pivot<a[--j]) {}
            if ( i < j ) {
                swap(&a[i],&a[j]);
            } else {
                break ;
            }
            swap(&a[i],& a[right -1]);
            quickSort(a, left, i-1);
            quickSort(a, i+1, right);
        }
        return ;
    }
}

void fquickSort(double a[], int left, int right)
{
if (left < right) {
    double pivot = fmedian_of_3(a, left, right);

    if (left == right - 1) return;
    int i = left;
    int j = right - 1;
    for ( ; ;) {
        while(a[++i]<pivot) {}
        while(pivot<a[--j]) {}
            if ( i < j) {
                fswap(&a[i], &a[j]);
            } else {
                break ;
            }
        fswap(&a[i], &a[right -1]);
        fquickSort(a, left, i-1);
        fquickSort(a, i+1, right);
    }

    return ;
}

void squickSort(char *a[], int left, int right)
{
    if (left < right) {
        char *pivot = smedian_of_3(a, left, right);
        //printf("%s\n", pivot);

        if (left == right - 1) return;
        int i = left;
        int j = right - 1;
        for ( ; ;) {
            while(strcmp(a[++i],pivot)<0) {}
            while(strcmp(pivot,a[--j])<0) {}
                if ( i < j) {
                    sswap(&a[i], &a[j]);
                } else {
                    break ;
                }
            sswap(&a[i], &a[right -1]);
            squickSort(a, left, i-1);
            squickSort(a, i+1, right);
        }
    }
// ======== show quicksorts =============

int partition(int a[], int left, int right, int length)
{
    int x = a[right];
    int i = left-1;
    for(int j=left; j<right; j++)
    {
        if(a[j] <= x)
        {
            i++;
            swap(&a[i], &a[j]);
            if (a[i] != a[j] && i != j) {
                printf("numbers swapped: %d,%d\n", a[j], a[i]);
                printf("array after swap: ");
                display(a, length);
            }
        }
    }
    swap(&a[i+1], &a[right]);
    if (a[i+1] != a[right] && ((i+1) != right)) {
        printf("numbers swapped: %d,%d\n", a[i+1], a[right]);
        printf("array after swap: ");
        display(a, length);
    }
    return i+1;
}

int fpartition(double a[], int left, int right, int length)
{
    double x = a[right];
    int i = left-1;
    for(int j=left; j<right; j++)
    {
        if(a[j] <= x)
        {
            i++;
            fswap(&a[i], &a[j]);
            if (a[i] != a[j] && i != j) {
                printf("numbers swapped: %f,%f\n", a[j], a[i]);
                printf("array after swap: ");
                fdisplay(a, length);
            }
        }
    }
}
```c
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);
            }
        }
    }
    ss.swap(&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("At this step:\n");
    printf("current array: ");
```

display(a, length);

// rearrange arrange now with pivot known
if (a[middle] < a[left]) {
    swap(&a[left],&a[middle]);
    printf("pivot swapped: %d,%d\n", a[middle],a[left]);
    printf("array after swap: ");
    display(a, length);
}
if (a[right] < a[left]) {
    swap(&a[left],&a[right]);
    printf("pivot swapped: %d,%d\n", a[right],a[left]);
    printf("array after swap: ");
    display(a, length);
}
if (a[right] < a[middle]) {
    swap(&a[middle],&a[right]);
    printf("pivot swapped: %d,%d\n", a[right],a[middle]);
    printf("array after swap: ");
    display(a, length);
}

swap(&a[middle], &a[right - 1]);
if (a[middle] != a[right-1] && (middle != (right-1))) {
    printf("numbers swapped: %d,%d\n", a[middle],a[right-1]);
    printf("array after swap: ");
    display(a, length);
}
printf("pivot is %d\n", a[right-1]);
return a[right - 1];

double show_fmedian_of_3(double a[], int left, int right, int length) {
    // get middle of array
    int middle = (left + right)/2;

    printf("At this step:\n");
    printf("current array: ");
    fdisplay(a, length);

    // rearrange arrange now with pivot known
    if (a[middle] < a[left]) {
        fswap(&a[left],&a[middle]);
printf("pivot swapped: %f,%f\n", a[middle],a[left]);
printf("array after swap: ");
fdisplay(a, length);
}

if (a[right] < a[left]) {
    fswap(&a[left],&a[right]);
    printf("pivot swapped: %f,%f\n", a[right],a[left]);
    printf("array after swap: ");
    fdisplay(a, length);
}

if (a[right] < a[middle]) {
    fswap(&a[middle],&a[right]);
    printf("pivot swapped: %f,%f\n", a[right],a[middle]);
    printf("array after swap: ");
    fdisplay(a, length);
}

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

char *show_smedian_of_3(char *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: ");
    sdisplay(a, length);

    // rearrange arrange now with pivot known
    if (strcmp(a[middle], a[left]) < 0) {
        sswap(&a[left],&a[middle]);
        printf("pivot swapped: %s,%s\n", a[middle],a[left]);
        printf("array after swap: ");
        sdisplay(a, length);
    }
    if (strcmp(a[right], a[left]) < 0) {
        sswap(&a[right],&a[left]);
        printf("pivot swapped: %s,%s\n", a[right],a[left]);
        printf("array after swap: ");
        sdisplay(a, length);
    }
}
sdisplay(a, length);}

if (strcmp(a[right], a[middle]) < 0) {
    sswap(&a[middle], &a[right]);
    printf("pivot swapped: %s,%s\n", a[right], a[middle]);
    printf("array after swap: ");
    sdisplay(a, length);
}

sswap(&a[middle], &a[right - 1]);
if (strcmp(a[middle], a[right-1]) != 0 && (middle !=
    (right-1))) {
    printf("strings swapped: %s,%s\n", a[middle], a[right-1]);
    printf("array after swap: ");
    sdisplay(a, length);
}
printf("pivot is %s\n", a[right-1]);
return a[right-1];

void showQuickSort(int a[], int left, int right, int length)
{
    if(left < right)
    {
        show_median_of_3(a, left, right, length);
        int q = partition(a, left, right, length);
        showQuickSort(a, left, q-1, length);
        showQuickSort(a, q+1, right, length);
    }
}

void showfQuickSort(double a[], int left, int right, int length)
{
    if(left < right)
    {
        show_fmedian_of_3(a, left, right, length);
        int q = fpartition(a, left, right, length);
        showQuickSort(a, left, q-1, length);
        showQuickSort(a, q+1, right, length);
    }
}
void showsQuickSort(char *a[], int left, int right, int length)
{
    if(left < right)
    {
        show_smedian_of_3(a, left, right, length);

        int q = spartition(a, left, right, length);
        showsQuickSort(a, left, q-1, length);
        showsQuickSort(a, q+1, right, length);
    }
}

void cShowQuickSort(int a[], int length) {
    int right = length;
    length = length;
    showQuickSort(a, 1, right, length);
    printf("%s\n", "QuickSort complete! Final Result: ");
    display(a, length);
}

void cShowfQuickSort(double a[], int length) {
    int right = length;
    length = length;
    showfQuickSort(a, 1, right, length);
    printf("%s\n", "QuickSort complete! Final Result: ");
    fdisplay(a, length);
}

void cShowsQuickSort(char *a[], int length) {
    int right = length;
    length = length;
    showsQuickSort(a, 1, right, length);
    printf("%s\n", "QuickSort complete! Final Result: ");
    sdisplay(a, length);
}

void cQuickSort(int a[], int length) {
    length = length;
    quickSort(a, 1, length);
}

113
```c
void cQuickfSort(double a[], int length) {
    length = length;
    fquickSort(a, 1, length);
}

void cQuicksSort(char *a[], int length) {
    length = length;
    sqquickSort(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, 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[] = {"abz", "cdf", "eadfaer", "baweaw", "dwaw"};
    cQuicksSort(a, 4);
    sswap(&a[0], &a[1]);
    sdisplay(a, 5);

    // }
```
#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) {

stack->top = NULL;
stack->size--;
} else {
    stack->top = stack->top->next;
    stack->size--;
}
free(node);
}

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

void printLine(int size) {
    printf("+");
    int i;
    for (i = 0; i < size; i++) {
        printf("-");
    }
    printf("+");
}

void stack_show(struct Stack *stack, int typ) {
    if (stack->size == 0) {
        printf("Stack is empty!");
        return;
    }
    int INTEGER_MAX = 13;
    int FLOATING_MAX = 20;
    int STRING_MAX = 48;
    if (typ == INTEGER) {
        printLine(INTEGER_MAX);
    } else if (typ == FLOATING) {
        printLine(FLOATING_MAX);
    } else if (typ == STRING) {
        printLine(STRING_MAX);
    }
    printf(" <- Top\n");
    int i;
    int size = stack->size;
struct Node *curr = stack->top;
for (i = 0; i < size; i++) {
    if (typ == INTEGER) {
        printf("| %-d |\n", INTEGER_MAX - 2, *(int *)
             curr->data);
        printLine(INTEGER_MAX);
    } else if (typ == FLOATING) {
        printf("| %-f |\n", FLOATING_MAX - 2, *(double
             curr->data);
        printLine(FLOATING_MAX);
    } else if (typ == STRING) {
        printf("| %-s |\n", STRING_MAX - 2, *(char **)
             curr->data);
        printLine(STRING_MAX);
    }
    printf("\n");
    curr = curr->next;
}
}

void stack_show_int(struct Stack* stack)
{
    stack_show(stack, INTEGER);
}

void stack_show_float(struct Stack* stack)
{
    stack_show(stack, FLOATING);
}

void stack_show_string(struct Stack* stack)
{
    stack_show(stack, STRING);
}

8.13  utils.h

Sophie

#ifndef type_consts
#define type_consts
#define INTEGER 0
#define FLOATING 1
#define STRING 2

117
8.14  testall.sh

Josh, Kofi, Millie, Sophie

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

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

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

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

Check() {
    error=0
    basename=`echo $1 | sed \s/\.*\///`

s/.strux//
refff="echo $1 | sed 's/.strux$//'"
basedir="echo $1 | sed 's/\([-\/[\-\/]\)]+$'//."

echo -n "$basename..."
echo 1>&2
echo "##### Testing $basename" 1>&2
generatedfiles=""

# generatedfiles="$generatedfiles $basename.ll
" "$generatedfiles="$generatedfiles $basename.ll $basename.s
" "$generatedfiles="$generatedfiles $basename.exe $basename.out" &&
Run "$STRUX" "$1" "$basename.ll" &&
Run "$LLC" "$basename.ll" "$basename.s" &&
Run "$CC" "$basename.exe" "$basename.s" "queue.bc"
" "$bstree.bc" "linkedlist.bc" "stack.bc" "quicksort.bc" &&
Run "./$basename.exe" "$basename.out" &&
Compare $basename.out $refff.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/\([-\/[\-\/]\)]+$'//."
  refff="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}.err
→ ${basename}.diff" &&
RunFail "$STRUX" "<$ 1 "2>" "${basename}.err ">>"
→ $globallog &&
Compare ${basename}.err ${reffile}.err ${basename}.diff

# Report the status and clean up the generated files

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

while getopts kdpsh c; do
  case $c in
  k) # Keep intermediate files
     keep=1
     ;;
  h) # Help
     Usage
     ;;
  esac
done

shift `expr $OPTIND - 1`

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

which "$LLI" >> $globallog || LLIFail
if [ ! -f linkedlist.bc ]
  then
    echo "Could not find linkedlist.bc"
    echo "Try "/linkStrux.sh"
    exit 1
fi

if [ ! -f queue.bc ]
  then
    echo "Could not find queue.bc"
    echo "Try "/linkStrux.sh"
    exit 1
fi

if [ ! -f bstree.bc ]
  then
    echo "Could not find bstree.bc"
    echo "Try "/linkStrux.sh"
    exit 1
fi

if [ ! -f stack.bc ]
  then
    echo "Could not find stack.bc"
    echo "Try "/linkStrux.sh"
    exit 1
fi

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

for file in $files
do
case $file in
  *test-*)
    Check $file 2>> $globallog
    ;;
  *fail-*)
    CheckFail $file 2>> $globallog
    ;;
  *)
    echo "unknown file type $file"
  esac
done
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* }

@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()
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*)
declare i8* @top(%struct.Stack*)
declare i32 @stack_size(%struct.Stack*)
declare void @stack_show_int(%struct.Stack*)
declare void @stack_show_float(%struct.Stack*)
declare void @stack_show_string(%struct.Stack*)
declare i32* @cQuickSort(i32*, i32)
declare void @cShowQuickSort(i32*, i32)
declare double* @cQuickfSort(double*, i32)
declare void @cShowfQuickSort(double*, i32)
declare i8** @cQuicksSort(i8**, i32)
declare void @cShowsQuickSort(i8**, i32)
declare %struct.BSTree* @initBSTree()
declare void @addIntToTree(%struct.BSTree*, i8*)
declare void @addNumToTree(%struct.BSTree*, i8*)
declare void @deleteIntFromTree(%struct.BSTree*, i32)
declare void @deleteNumFromTree(%struct.BSTree*, double)
declare i1 @treeContainsInt(%struct.BSTree*, i32)
declare i1 @treeContainsFloat(%struct.BSTree*, double)
declare void @showIntTree(%struct.BSTree*)
declare void @showNumTree(%struct.BSTree*)
declare i32 @printbig(i32)

define i32 @gcd(i32 %n1, i32 %n2) {
  entry:
  %n11 = alloca i32
  store i32 %n1, i32* %n11
  %n22 = alloca i32
  store i32 %n2, i32* %n22
  %gcd = alloca i32
  store i32 1, i32* %gcd
  %i = alloca i32
  store i32 1, i32* %i
  br label %while
  while:
  %merge, %entry
  %i15 = load i32, i32* %i
  %n116 = load i32, i32* %n1
  %tmp17 = icmp sle i32 %i15, %n116
  %i18 = load i32, i32* %i
  %n219 = load i32, i32* %n2
  %tmp20 = icmp sle i32 %i18, %n219
  %tmp21 = and i1 %tmp17, %tmp20
  br i1 %tmp21, label %while_body, label %merge22
  while_body:
  %merge, %while
  %n13 = load i32, i32* %n1
  %i4 = load i32, i32* %i
  %tmp = srem i32 %n13, %i4
  %tmp5 = icmp eq i32 %tmp, 0
  %n26 = load i32, i32* %n2
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 a414f6774ff94e3afa17fddb32161f24f658be2
Author: Joshua Bartlett <jcbart12595@gmail.com>
Date: Wed Dec 20 15:41:03 2017 -0500

final touches to bstree

commit 4d1391bf8ec03795e7d9adf6ff11e4a4db5b788
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 3f9c220dca8c545e8a8fec65e73deea46a64dac
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
few changes here and there

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

adding tests

commit ae81717a8397afeb86eb6c5167d8f92ef87f430d
Author: sophstad <srs2231@columbia.edu>
Date:   Tue Dec 19 05:37:49 2017 +0000

Remove strux.sh file

commit e9ea07be8d9fd13a1c32aa20c5235b3c833cf095b
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

Merge: 7452b1d5a5310f
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
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

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 94f63b5b81b156a7d14cbeae99f9a2dd1e16eb97f
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 83c7243a5ace064da6a47cc7e33d978ff21b5
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 b8b4e499e8251bdcd4576694a9e2395a737ae63
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 84581d881569a787c1925cb9ec1e45162fb3a98af
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 2704ce1222ec817faec61159e50c9ae98b5e69b61
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 8ec06e7a994bdfa2a3c83bcd85a3217d1aadcd66a
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 a4581912d7741c599990a18b964cf914d1d2acad
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 c09d362d3f83cd7c7496e8408c1b5271d08479c
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 bef318e0a76e6b962241b3b8ebe33d39ee7c5bb0
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 508e14754bcdbb16e77dbe4d24003cdead7f3e66d4
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 3912cb9ab6a903082ad160ceaeaa013051c1a48d0
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 760515ac9071d7efee1ae7c9b32cc450770af3e75
Author: sophstad <srs2231@columbia.edu>
Date: Tue Dec 12 21:57:01 2017 +0000

Show array works
Array len working; size argument removed for quicksort

tests for inputting array in queue, linkedlist, and stack

comen so that it's easier to read

all tests for quicksort passes

rebase w master

changing return type of show functions

trying to change actual params for quicksorts
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 a8d27b7d4d8d4bfdbd68b06ec6a6857f9eb5eb2e1d
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 b451d04ceee4db7869df9db30bc43b9ec9cda4dcb
Author: Millie Yang <my2440@columbia.edu>
Date: Tue Dec 12 00:15:08 2017 -0500

more tests

commit cdd7381e3d6131cd92ad7396da77e098912fc34
Author: Millie Yang <my2440@columbia.edu>
Date: Tue Dec 12 00:07:13 2017 -0500

more tests

commit e45b86d8b804b62544389da9e10e7401b7d29d2
Author: Millie Yang <my2440@columbia.edu>
Date: Mon Dec 11 23:56:01 2017 -0500

tests and queue size works

commit aade2d60e183cbe2e19169f9b1beda522d8ff18a
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 bffecb7f4dabb4a4b08375dc62b481db7fce8
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 quicksort with nums

commit 23d041a42c875ebebc41a52595abd81b45078eca
Merge: de47c1e affer7947
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 9dacb735247cbae1dd7a16df760ee33cdec8ad1
Author: sophstad <srs2231@columbia.edu>
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 df9fb10fad5066a7acf1d7d7b5160f2b84264c2
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 15e18aeeed04a4e30a6d4fb383d0f989e69eb1bd87
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

quicksort test passes
fixed a bug in master and rebased

compiling quicksort.c

trying to get quicksort for ints to work

Implement .show() for stacks

Implement .show() for queues

compiling quicksort.c

trying to get quicksort for ints to work
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 84e0a8a217bb906d82ff9303cf960932377d
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 <sphst@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 57565f71fbbf665436306c7726efab83c46f510d
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 3de6018b1a609c4b813dc6360138ccdd7df1a1c
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 8d9b73a8f970576109ac05ae73bd8ceee07ee1e
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 2db4f9cc04c45e170ee70b392537970d2c8cc76
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
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 dbc38ff497ae1e4d447e6a6184df98f470d9d3b0b
Author: Millie Yang <my2440@columbia.edu>
Date: Wed Dec 6 15:00:51 2017 -0500

printing float works
removing vertical bars from the print

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

lil progress

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

I think sophie knows how to fix this

trying to implement size function that recognizes which function to call

initial commit, queue peek and list get works
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 2026606617fd6a6d209793f6a5a6130e0ed563fd
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 aa59470643e6f6c489eafdb9c73708a843524a65
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 bc7189377469dac121cb0f0049897f0dc3d06909c
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 9ee9330d52413c645ec05f854a6d757754b3cbb
Author: Millie Yang <my2440@columbia.edu>
Date: Tue Nov 28 20:26:32 2017 -0500

adding failure tests

commit 9f13e00f64ad079183e302147f79efee7b049167
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
dehque works

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

commit cba425d8b1f0d162374b23edffe8a332da84453
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 c16e36cfc38d14c79001325d73164cdec1b5fa847
Author: Millie Yang <my2440@columbia.edu>
Date: Mon Nov 27 18:20:22 2017 -0500

fixed test result

commit 56e770b1bb264cbb291bc05cf2d2e37142e5f9e09
Author: Millie Yang <my2440@columbia.edu>
Date: Mon Nov 27 18:16:18 2017 -0500

removed duplicate files

commit difad34fe0279003394bdb2b313da0f1b1f16ddf
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

154
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 check for errors

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

printbig integration

commit 5791d5214dale5ffa35c405242ca3b939ff32c20
Author: Millie Yang <my2440@columbia.edu>
Date: Mon Nov 27 15:18:00 2017 -0500

modified, but still not working

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 9ab86424e85f162263e52d8a763fa95216fbe5a
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 52cfaced6a7dce3fae5c6e411d4b6c8a5b42527
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 a89d6272abe0f3097809dc14af2a296a9613cb95f
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 b5a9c9b752468d9ba6fe95c02fbb8034fe1feec
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 e94dfac9023eb67c2eaecc4493c8b325f8a1a5d31
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 7df85cdddb09deac55a494efa9f7573bf591b56d6
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

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 8015d62bab9b05e2ae0ca5f21926c8d692119ca
Author: Millie Yang <my2440@columbia.edu>
Date: Thu Nov 16 17:39:00 2017 -0500

fixed tests

commit 9fe6347617a1a370697174d51435c5e652c0a54
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 a6056734a575d5603b5aaac8ef781e4b2ef5494a0
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 666a0815aa6f2f56badd3ac4946760b90973fb7c8
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, didn't 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 def05b4e1856d88fd3ed8f73ca352eaa3520bab
Author: Fredrick-Tam <tamf295@yahoo.com>
Date: Tue Nov 14 03:48:44 2017 -0500

have increment and decrement working

commit 7c4b5919069d669d669dccc291bbae13fac5622cf7169
Merge: a9adec1 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 a9adec1cb31bddd18454755c957a843af4529beea3
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 70022c11c3b5b292084f4dcd97624930c621e8
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 4f51a883e412da2da4a18d93cd752b91bec8465
Author: Millie Yang <my2440@columbia.edu>
Date: Mon Nov 13 19:41:04 2017 -0500

printed bool and got return working - one bug

commit ad508626bc787be27fc0d908d78355f1ef6d05d8
Author: Millie Yang <my2440@columbia.edu>
Date: Sat Nov 11 19:02:15 2017 -0500

162
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

163
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 6354e3cae63e53bd1f3ae3aeb960d17b2646157ea68
Author: Millie Yang <my2440@columbia.edu>
Date: Fri Nov 10 11:34:40 2017 -0500

adding necessary tests

commit 567a9d854aa2eeeb319d2d72d55f3c0097ee4394
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/sophstard/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 errors 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 d0f64a760f08b11cc93c34a60368ffabdb7a9ddc5b
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/sophstard/strux

commit 5b4c6553631f825d611e2ae3caee3e3e6819a41
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/sophstard/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 24b4b29bd60610bc405fccc8a3f9c6f1f6861cb1c
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
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