nodable Programming Language Final Report

Karen Shi
Manager
ks3650@columbia.edu

Ajita Bala
Language Guru
ab4420@columbia.edu

Ariel Goldman
System Architect
apg2164@barnard.edu

Naviya Makhija
Tester
nm3076@columbia.edu

April 27, 2021

Contents

1 Introduction **7**

2 nodable Tutorial **7**
- 2.1 Environment Setup . . . 7
- 2.2 Compilation Guide . . . 7
- 2.3 Language Tutorial . . . 7

3 Language Reference Manual **8**
- 3.1 Lexical Conventions . . . 8
  - 3.1.1 Character Set . . . 8
  - 3.1.2 Comments . . . 8
  - 3.1.3 Identifiers . . . 8
  - 3.1.4 Keywords . . . 8
  - 3.1.5 Constants . . . 9
  - 3.1.6 Elementary Operations and Spacing . . . 9
- 3.2 Operators . . . 9
  - 3.2.1 Arithmetic Operators . . . 9
  - 3.2.2 Unary Operators . . . 10
  - 3.2.3 Assignment Operators . . . 10
  - 3.2.4 Relational Operators . . . 10
  - 3.2.5 Logical Operators . . . 10
- 3.3 Type specifiers . . . 11
  - 3.3.1 Type Specifiers . . . 11
- 3.4 Variables . . . 11
  - 3.4.1 Lists . . . 11
  - 3.4.2 Nodes . . . 12
  - 3.4.3 Trees . . . 13
  - 3.4.4 Graphs . . . 14
  - 3.4.5 Functions . . . 14
- 3.5 Statements . . . 15
  - 3.5.1 Expression Statements . . . 16
  - 3.5.2 Iterative Statements . . . 16
  - 3.5.3 Conditional Statements . . . 16
- 3.6 Scope . . . 17

4 Project Plan **17**
- 4.1 Development Process . . . 17
  - 4.1.1 Planning . . . 17
  - 4.1.2 Testing . . . 18
- 4.2 Style Guide . . . 18
- 4.3 Roles and Responsibilities . . . 18
- 4.4 Software Development Environment . . . 18
- 4.5 Project Timeline . . . 19
- 4.6 Project Log . . . 20

5 Architectural Design 40
5.1 Component Diagram . . . 40
5.2 Component Design and Authors . . . 40
5.2.1 Scanner . . . 40
5.2.2 Parser and AST . . . 40
5.2.3 Semantic Checking . . . 40
5.2.4 Code generation . . . 41
5.2.5 C Libraries . . . 41

6 Test Plan 41
6.1 Source Language Programs . . . 41
6.1.1 List Operations . . . 41
6.1.2 Tree Traversals . . . 51
6.1.3 Check Tree Balance . . . 57
6.2 Automation . . . 65
6.2.1 ./testall.sh . . . 65
6.3 Testing Strategy . . . 69

7 Lessons Learned 70
7.1 Naviya . . . 70
7.2 Karen . . . 71
7.3 Ajita . . . 71
7.4 Ariel . . . 71

8 Appendix 71
8.1 Scanner - scanner.mll . . . 71
8.2 Parser - nodableparser.mly . . . 73
8.3 AST - ast.ml . . . 77
8.4 Semantic Checker - semant.ml . . . 79
8.5 Semantically-checked Abstract Syntax Tree - sast.ml . . . 85
8.6 Code generation- codegen.ml . . . 87
8.7 nodable.ml . . . 95
8.8 C Standard Library . . . 96
8.8.1 c-library.h . . . 96
8.8.2 c-library.c . . . 97
8.9 Test Suites . . . 100
8.9.1 test-add1.nd . . . 100
8.9.2 test-add2.nd . . . 100
8.9.3 test-arith2.nd . . . 100
8.9.4 test-arith3.nd . . . 101
8.9.5 test-comment1.nd . . . 101
8.9.6 test-comment2.nd . . . 101
8.9.7 test-complexintuse.nd . . . 102
8.9.8 test-complexstruse.nd . . . 102
8.9.9 test-div1.nd . . . 102
8.9.10 test-div2.nd . . . 103

8.9.11 test-eq1.nd 103
8.9.12 test-fib.nd 103
8.9.13 test-float1.nd 103
8.9.14 test-float2.nd 104
8.9.15 test-float3.nd 104
8.9.16 test-for1.nd 105
8.9.17 test-func1.nd 105
8.9.18 test-func2.nd 105
8.9.19 test-func3.nd 106
8.9.20 test-func4.nd 106
8.9.21 test-func5.nd 106
8.9.22 test-func6.nd 107
8.9.23 test-func7.nd 107
8.9.24 test-func8.nd 107
8.9.25 test-func9.nd 108
8.9.26 test-gcd.nd 108
8.9.27 test-geq1.nd 108
8.9.28 test-global1.nd 108
8.9.29 test-global2.nd 109
8.9.30 test-global3.nd 109
8.9.31 test-global4.nd 110
8.9.32 test-if1.nd 110
8.9.33 test-if2.nd 110
8.9.34 test-if3.nd 111
8.9.35 test-if4.nd 111
8.9.36 test-if5.nd 111
8.9.37 test-if6.nd 111
8.9.38 test-leq1.nd 112
8.9.39 test-list1.nd 112
8.9.40 test-list2.nd 112
8.9.41 test-list3.nd 113
8.9.42 test-list4.nd 113
8.9.43 test-listappend.nd 113
8.9.44 test-listdec.nd 114
8.9.45 test-listdemo.nd 114
8.9.46 test-listempty.nd 116
8.9.47 test-listfunct1.nd 116
8.9.48 test-listnest.nd 117
8.9.49 test-listsize.nd 117
8.9.50 test-listupdate.nd 117
8.9.51 test-local1.nd 118
8.9.52 test-local2.nd 118
8.9.53 test-local3.nd 118
8.9.54 test-mod1.nd 119
8.9.55 test-mult1.nd 119
8.9.56 test-mult2.nd 119

8.9.57 test-neq1.nd 119
8.9.58 test-node1.nd 119
8.9.59 test-node2.nd 120
8.9.60 test-node3.nd 120
8.9.61 test-node4.nd 120
8.9.62 test-node5.nd 120
8.9.63 test-node6.nd 121
8.9.64 test-printbool.nd 121
8.9.65 test-printflt.nd 121
8.9.66 test-printint.nd 121
8.9.67 test-printstr.nd 122
8.9.68 test-recurs1.nd 122
8.9.69 test-sub1.nd 122
8.9.70 test-sub2.nd 122
8.9.71 test-sub3.nd 123
8.9.72 test-tree.nd 123
8.9.73 test-tree2.nd 123
8.9.74 test-tree3.nd 123
8.9.75 test-tree4.nd 124
8.9.76 test-treeheight.nd 124
8.9.77 test-while1.nd 125
8.9.78 test-while2.nd 126
8.10 Fail Tests 126
8.10.1 fail-assign1.nd 126
8.10.2 fail-assign2.nd 126
8.10.3 fail-assign3.nd 127
8.10.4 fail-dead1.nd 127
8.10.5 fail-dead2.nd 127
8.10.6 fail-expr1.nd 128
8.10.7 fail-expr2.nd 128
8.10.8 fail-expr3.nd 128
8.10.9 fail-float1.nd 128
8.10.10 fail-float2.nd 129
8.10.11 fail-for1.nd 129
8.10.12 fail-for2.nd 129
8.10.13 fail-for3.nd 129
8.10.14 fail-for4.nd 130
8.10.15 fail-for5.nd 130
8.10.16 fail-func1.nd 130
8.10.17 fail-func2.nd 130
8.10.18 fail-func3.nd 131
8.10.19 fail-func4.nd 131
8.10.20 fail-func5.nd 131
8.10.21 fail-func6.nd 132
8.10.22 fail-func7.nd 132
8.10.23 fail-func8.nd 132

8.10.24 fail-func9.nd 133
8.10.25 fail-global1.nd 133
8.10.26 fail-global2.nd 133
8.10.27 fail-if1.nd 133
8.10.28 fail-if2.nd 134
8.10.29 fail-if3.nd 134
8.10.30 fail-nomain.nd 134
8.10.31 fail-print.err 134
8.10.32 fail-printbool.nd 134
8.10.33 fail-printstr.nd 135
8.10.34 fail-return1.nd 135
8.10.35 fail-return2.nd 135
8.10.36 fail-while1.nd 135
8.10.37 fail-while2.nd 135

1 Introduction

nodable is an imperative, statically-typed graph programming language designed to help users create, use, manipulate, and search graphs. Graphs are essential for data representation in computer science, but the most popular programming languages require users to implement graphs with low-level data structures. *nodable*'s aim is to simplify the implementation of graphs and their algorithms by featuring built-in data types for nodes to store information about nodes and links to other nodes, allowing for graph functionalities. *nodable*'s syntax contains elements of the Java and C programming languages.

2 nodable Tutorial

2.1 Environment Setup

In order to run nodable, you must first have OCaml, LLVM, and docker installed on your machine. In order to download nodable, you should then go to your terminal and run:

```
git clone https://github.com/kshi18/nodable.git
```

2.2 Compilation Guide

You must follow these steps each time you open a terminal before you run nodable code. First, in your terminal, within the nodable folder, run this (all one command):

```
docker run --rm -it -v `pwd`:/home/nodable -w=/home/nodable
    columbiasedwards/plt
```

Next run:

```
make
```

This will build the compiler and will also automatically run all of our tests.

2.3 Language Tutorial

Let's create your first nodable program. Inside the nodable folder, create a file named welcome.nd and inside the file, copy this code:

```
int main()
{
   prints("Welcome!");
   return 0;
}
```

To run this program, first go into the nodable file in your terminal and follow the steps of the compilation guide above. Next, enter these commands into the terminal:

```
./nodable.native < welcome.nd > welcome.ll
llc -relocation-model=pic welcome.ll > welcome.s
cc -o welcome.exe welcome.s c_library.o -lm
./welcome.exe > welcome.out
```

You should now see a file called ’welcome.out’ in the nodable folder. If you open welcome.out, you will see that your program printed the message: Welcome!

Congratulations! You have just written your first nodable code.

3 Language Reference Manual

3.1 Lexical Conventions

3.1.1 Character Set

Our language’s character set is **a-zA-Z0-9,-;<>*+/=!%$&|{}()[]_**

3.1.2 Comments

Comments are denoted by **//** preceding and following the desired text to be commented.

```
// this is a single line comment //

// this is //
// a multi-line //
// comment //
```

3.1.3 Identifiers

An identifier in our language must begin with a letter, and is optionally followed by letters, numbers, or underscores. An identifier name is thus defined by the following regular expression:

```
[’a’-’z’ ’A’-’Z’][’a’-’z’ ’A’-’Z’ ’0’-’9’ ’\_’]*
```

3.1.4 Keywords

These keywords are reserved words that should not be used as identifiers:

```
int float string boolean true false
```

```
if else while for void return
node list
```

3.1.5 Constants

nodable supports integer, floating point, and boolean constants, otherwise known as literals, inside of expressions. String literals are represented internally as arrays of characters.

3.1.6 Elementary Operations and Spacing

Each expression is delimited by a semicolon and sets of expressions are marked by curly braces {}, with the exception of single-line if-else statements. Whitespace and indentation do not play a role in the flow of the code.

3.2 Operators

In *nodable*, there are five categories of operators: arithmetic, unary, assignment, relational, and logical.

From highest to lowest precedence, where highest order operators bind tighter than lower order ones:

1. Unary
2. Arithmetic
3. Relational
4. Logical
5. Assignment

3.2.1 Arithmetic Operators

nodable provides the five basic arithmetic operators: *addition, subtraction, multiplication, division,* and *modulo.* All arithmetic operators are left-associative binary operators, and multiplication, division, and modulo have higher operator precedence than addition and subtraction.

Operator	**Description**	**Example**
`+`	Addition	`a + b;`
`-`	Subtraction	`a - b;`
`*`	Multiplication	`a * b;`
`/`	Division	`a / b;`
`%`	Modulus	`a % b;`

Each of these operators can be performed on integers and floating point expressions, but must be applied to expressions of the same data type. For the operators % and /, if the right operand is 0, the result is undefined.

For example:

```
int i1 = 1;
int i2 = 2;
float f = 1.5;
// i1 + i2 will compile
// i1 * f will not compile
```

Note that binary operators must be used on expressions of the same type (int with int, float with float); otherwise, the code will not compile.

3.2.2 Unary Operators

Unary operators require only one operand and are grouped from right to left.

Operator	Description	Example
`!`	Negation	`a = !b;`
`-`	Minus	`a = -3;`

3.2.3 Assignment Operators

The assignment operator is a right-associative binary operator. The value to the right of the operator is stored into the variable on the left of the operator.

Operator	Description	Example
`=`	Assignment	`int a = 5;`

3.2.4 Relational Operators

Relational operators are left-associative binary operators used to check the relationship between two operands (literals or variables). The type on each side of these operators should be either an integer or float. All relational operators return a boolean value.

Type	Description	Example
`>`	Greater than	`3 > 1;`
`<`	Less than	`3 < 5;`
$\geq$	Greater than or equal to	`1` $\geq$ `1;`
$\leq$	Less than or equal to	`3` $\leq$ `5;`

3.2.5 Logical Operators

Logical operators are left-associative binary operators used to assert whether multiple values or expressions, or the negation of a value or expression, are true. Logical operators are used with boolean values, and return a boolean value (0 or 1).

Operator	Description	Example
`&&`	logical AND	`a && b;`
`||`	logical OR	`a || b;`

3.3 Type specifiers

Declarations specify the interpretation (value and type) given to a set of identifiers.

3.3.1 Type Specifiers

nodable has the following type specifiers:

typ:
int
float
string
boolean
void
list<typ>
node<typ>

nodable supports the 4 primitive data types and void. The type of the list or node must be specified when the list is declared and can be any one of the four primitive data types (int, float, string, boolean), as well as lists or nodes.

3.4 Variables

Variables can be declared and initialized globally or locally in a function as follows (excluding void as a type specifier):

```
typ ID;
typ ID = expr;
```

3.4.1 Lists

Lists are mutable collections of primitive data types or of lists or nodes. Lists can simply be declared or initialized as empty or with values.

```
list<typ> ID;
list<typ> ID = [];
list<typ> ID = [args_list];
```

Users are able to obtain the size, as well as append or update values to the lists.

Built-in list functions (e.g. for list of nodes of integer type)

```
void append(list<node<int>> t, node<int> d);
// adds an element d to the end of the list t

list<node<int>> update_elem(node<int> d, list<node<int>> t, index n);
// updates an existing element in the list t at the given index n with a
    new element d

int size(list<node<int>> t);
// returns the number of elements in list t
```

3.4.2 Nodes

Nodes are modeled on C structs on the backend. Each node contains a field 'data', and can also have 'left' and 'right' child attributes.

The node struct in the C library:

```
struct Node {
  void *data;
   struct Node* left;
   struct Node* right;
};
```

Nodes can simply be declared with one of the four primitive data types or initialized to a '$' symbol followed by a literal.

```
node<typ> identifier;
node<typ> identifier = $literal;
```

The 'data' field of the node can be accessed using the dot operator.

```
node<int> a = $1;
print(a.data);
// prints 1
```

Users can declare null nodes by assigning the value to 0, which are instantiated in the backend as NULL.

```
node<int> null_node = $0;
```

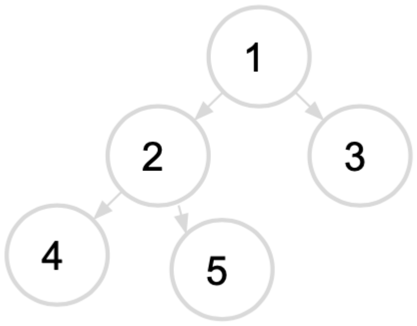
Built-in node functions (e.g. for nodes of integer type)

```
void add_left(node<int> a, node<int> b);
// assigns node b as the left child node of a
```

```
void add_right(node<int> a, node<int> b);
// assigns node b as the right child node of a
node<int> get_left(node<int> a);
// returns the left child node of a
node<int> get_right(node<int> a);
// returns the right child node of a
```

3.4.3 Trees

nodable allows the user to work with binary trees using the node data type's .left and .right attributes and the built-in node functions mentioned above. For example, this binary tree can be created using the code below:

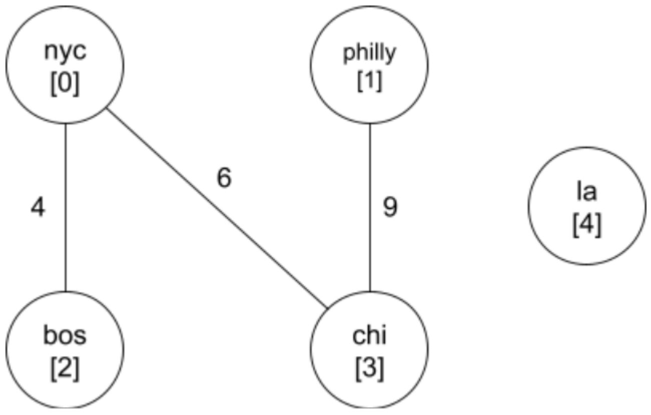


```
    node<int> root;
    node<int> c1;
    node<int> c2;
    node<int> c3;
    node<int> c4;
    node<int> null;

    root = $1;

    c1 = $2;
    add_left(root, c1);

    c2 = $3;
    add_right(root, c2);

    c3 = $4;
    add_left(c1, c3);

    c4 = $5;
    add_right(c1, c4);

    null = $0;
    add_left(c2, null);
    add_right(c2, null);
    add_left(c3, null);
    add_right(c3, null);
    add_left(c4, null);
    add_right(c4, null);
```

3.4.4 Graphs

nodable does not have a graph data type. However, graphs can be represented in our language through lists of nodes and adjacency lists of integers.

The nodes of a graph can be stored in a list where the ID of each node is their index in this list. The edges of the graph can be stored in an adjacency list, as seen in the example of a weighted undirected graph below:



```
node<string> nyc = $"nyc";
node<string> philly = $"philadelphia";
node<string> bos = $"boston";
node<string> chi = $"chicago";
node<string> la = $"los angeles";
list<node<string>> nodelist = [nyc, philly, bos, chi, la];
list<list<list<int>>> adj_list = [[[2, 4], [3,6]], [[3, 9]], [[0, 4]],
    [[0, 6], [1, 9]], []]
```

3.4.5 Functions

nodable declares functions in a similar way to the standard C style, but uses curly braces to determine scope. Every program requires at least the main function and can have additional functions defined.

return-type identifier (*args-list*) { *function definition* }

The args-list is a list of arguments inside of parentheses. This list must include type specifiers and an identifier for each argument. An argument can be of any type, including graph types.

The function definition can include a list of statements and expressions of unlimited length. If the function has a return type, the definition must contain a return statement with a value indicated by the type-specifier. Otherwise, if the function has a void return type, it does not require a return statement. The function signature must specify the data types of the parameter list, as well as the return type.

nodable also requires the usage of semicolons at the end of each statement to reduce confusion and ambiguity.

Here are some written examples of function definitions:

```
int add(int x, int y) {
    return x + y;
}
```

Functions must be written above the main function when they're called:

```
int main(){
    add(2, 7);
}
```

3.5 Statements

Statements in *nodable* are executed in sequence with a series of statements acting as a list of statements. Statements can be expression statements, conditional statements, and iteration statements. All statements are delimited by semicolons.

```
stmt:
    expr;
    typ ID;
    typ = expr;
    return expr_opt;
    {stmt_list}
    if (expr) stmt
    if (expr) stmt else stmt
    while (expr) stmt
    for (expr_opt; expr; expr_opt) stmt
```

3.5.1 Expression Statements

An expression statement is the most basic form of statement. These include assignments, declarations, operations, and function calls.

Here are some examples:

```
int x = 5;
x = x + 1;
int y = add(4, 5) + add(9, 10);
```

3.5.2 Iterative Statements

Iterative statements in *nodable* include while loops and for loops.

While Loop Based on Java syntax, this loop executes the inner statements if the outer boolean expression is satisfied. It terminates once the condition is false.

```
int i = 5;
while (i > 0) {
   print(i);
   i = i - 1;
}
```

For Loop Based on Java syntax, For Loops allow iteration through lists and other data structures/objects so that expressions can be executed on that data. *nodable* has two types of for loops. There is a loop that iterates through the data structure:

```
int v;
int count = 0;
for (v = 0; v < 5; v = v + 1) {
    if (v == 0)
        count = count + 1;
}
```

3.5.3 Conditional Statements

These are conditional statements established based on Java syntax to allow users to selectively execute expression and other statements. *nodable* includes if statements (with no else) and if-else statements.

```
string weather = "";
if(temp >= 90)) {
    weather = "hot";
}
else {
    weather = "cold";
}
```

3.6 Scope

nodable uses curly brackets {} to determine scope (functions, conditional statements, and iterative statements). Any variable declared within a particular code block cannot be referenced outside of it, and any variable declared outside of code blocks are regarded as global variables and can be used anywhere in the program.

4 Project Plan

4.1 Development Process

4.1.1 Planning

As we were working on nodable, we relied on group Zoom meetings in order to plan our next steps. At these meetings we would discuss progress made since the last meeting, work on any difficult development issues as a group, and add new tasks to our project tracker spreadsheet. The project tracker spreadsheet listed tasks, the target date to complete them by, who was working on each task, and the task status. We had at least two of these group meetings a week, with one of them being at our standing Monday at 3 pm meeting time, and our other meeting times varied weeekly. Additionally, we would often meet as pairs to work together on a piece of nodable implementation.

Another key part of planning how to build nodable was feedback from TAs and Professor. Unfortunately, teaching staff changes during the semester kept us from having a specific time to meet with a TA each week, we attempted to make sure we were receiving feedback from the teaching staff about our progress and next steps on a weekly basis.

Between meetings we communicated over Facebook Messenger, which allowed us to ask each other questions and give each other updates on nodable easily and efficiently.

4.1.2 Testing

Whenever we implemented a new feature in our language we implemented thorough testing of the feature to make sure its behavior was as expected. We were careful to make sure we always did this testing right after implementing the feature, so that we could catch any errors in our implementations quickly. Our tests included checking the behavior of our primitive datatypes, variable assignment, operators, function declaration, control flow, lists, and nodes. We tested complicated use cases, such as using nodes in tree traversal, in addition to our simple ones to make sure our implementation did not start behaving oddly in these more complicated use cases. We also had tests that were designed to fail, which we wrote to test that the code failed when expected to and provided the user with the appropriate error message to help them fix their code. By the end of the project, we had over 120 test cases.

4.2 Style Guide

In developing nodable, we generally used these style guidelines:

- Keep line lengths to 80 characters or less.
- For indentation, using tabs, 2 spaces, or 4 spaces are permissible, but within a document the choice of how to indent should remain consistent.
- Use underscores rather then camelCase where possible.

4.3 Roles and Responsibilities

We all worked very collaboratively throughout the creation of nodable, often coding in pairs or groups, because we found we were able to code more efficiently and effectively if we worked together. Therefore, it is hard to define certain roles and responsibilities that belonged to each group member, but none-the-less here is an idea of what each person focused on:

- Ariel Goldman: Testing, Documentation, AST
- Ajita Bala: Scanner, Codegen, C libraries
- Naviya Makhija: Semant, Codegen, Testing, C libraries
- Karen Shi: Codegen, Parser, Testing, C libraries

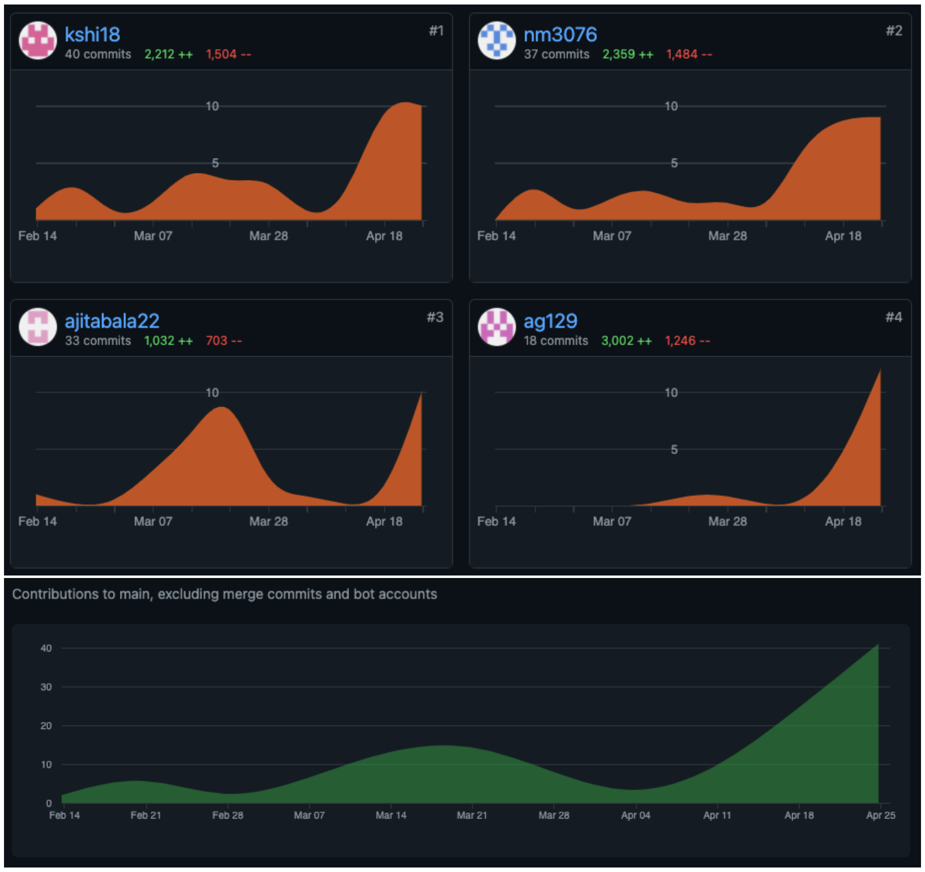
4.4 Software Development Environment

We developed our code in Visual Studio Code. We chose Visual Studio Code because of its Live Share plug in option that allows people to type code on the same document (which is stored on one person's computer) at the same time, similar to a google doc. Using Live Share while on a zoom call allowed for easy virtual collaboration. We also used Github in order to share our project and our updates to it across our computers. The language we developed in was OCaml version 4.11.1.

4.5 Project Timeline

2/3	Language proposal completed
2/24	LRM written
2/24	Parser written
3/24	Hello World milestone: We have our first program running end to end
4/4	Operators, variables, data types, control flow and functions
4/23	Lists are working
4/24	Nodes are working
4/25	We can build trees and graphs using our nodes
4/26	Final Report completed

4.6 Project Log



```
commit 5080d75f25370e5cd0e6adcc72a656d55e6f7230
Merge: d602337 2effc8b
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Mon Apr 26 17:08:32 2021 -0400

    Merge branch 'main' of https://github.com/kshi18/nodable into main

commit d6023378238ceb310bc67ac26d232d56098a5338
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Mon Apr 26 17:08:21 2021 -0400

    graph fixes

commit 2effc8b6c16b1f50ea2d5df401c6812709ab494f
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Apr 26 15:39:26 2021 -0400
```

```
    cleaned up the scanner

commit 7ffb9c54c892f4aafe56d36700fc51605ca53044
Merge: 1bc4b97 4a3abc0
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Apr 26 01:42:38 2021 -0400

    merging

commit 1bc4b972a5cfbc2d5a882cd4206aac85be522f7f
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Apr 26 01:40:45 2021 -0400

    simplified tree traversal test

commit 4a3abc0ca2e56ad702163f2bf576b6e259ec1b92
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 26 13:30:19 2021 +0800

    fixed tree balance function

commit bb911a38096251bb47cbfa524017d8bb8949bb29
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Mon Apr 26 01:01:27 2021 -0400

    check tree balance (not working)

commit 103ab4decd7eadc74aeb4c055a002a1ee6f010aa
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 26 12:57:15 2021 +0800

    variables merged and tests edited

commit ff2bc811677db261a6acef9196d7f872b9d5f2d9
Merge: f47dc82 201ec80
Author: Naviya Makhija <43485094+nm3076@users.noreply.github.com>
Date:   Mon Apr 26 12:52:39 2021 +0800

    Merge pull request #10 from kshi18/variables

    variable changes

commit f47dc824a707627f7e6d991978685aff59e5f980
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 26 12:50:58 2021 +0800

    cleaned up testing branch

commit 0fce1f24ee43020d5fb22e467867c931428bc653
Merge: 0d72596 f95291f
```

```
Author: Naviya Makhija <43485094+nm3076@users.noreply.github.com>
Date:   Mon Apr 26 12:37:53 2021 +0800

    Merge pull request #9 from kshi18/updating_test_suite

    Updating test suite

commit f95291f0f326482d10a78020280365fcc861d839
Merge: df80f18 0d72596
Author: Naviya Makhija <43485094+nm3076@users.noreply.github.com>
Date:   Mon Apr 26 12:37:42 2021 +0800

    Merge branch 'main' into updating_test_suite

commit 0d72596f58ef3bc4f963f785ab1b35fd2a5b0697
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 26 12:33:40 2021 +0800

    cleaned up tests after merge

commit db640556a15b10176aff3283e0a36e3880e599e3
Merge: a82af1e 9d9a601
Author: Naviya Makhija <43485094+nm3076@users.noreply.github.com>
Date:   Mon Apr 26 12:26:24 2021 +0800

    Merge pull request #8 from kshi18/tree_attempt

    Merge Trees

commit 9d9a6011cc5bdb002623d40139821e0a5a2e3921
Merge: dabfe59 a82af1e
Author: Naviya Makhija <43485094+nm3076@users.noreply.github.com>
Date:   Mon Apr 26 12:25:02 2021 +0800

    Merge branch 'main' into tree_attempt

commit df80f18396378710697ca3f74491cb4df444ac5c
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Mon Apr 26 00:22:21 2021 -0400

    testdemo insertion sort

    test demo now features an inserion sort function

commit de5a24ed882592a0a017e27d189d39eb0e501db5
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Mon Apr 26 00:04:28 2021 -0400

    reverse list function
    I added a reverse list function to the listdemo

commit db92e91978cbeeb85832acdf60e6b335970397c3
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 23:47:56 2021 -0400

    list demo now uses nodes

    So far list demo creates the last and tells you its length and
        average value

commit 201ec804412be94d21b6a567d70f38f16379a2b6
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 26 11:22:56 2021 +0800

    parser errors fixed

commit 1d0f0b539620e05d7113a88e01414e86d1f4470e
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 26 10:59:40 2021 +0800

    fixed size function

commit 486eee1c4112d8ee196e589f95b0a66a455cc18f
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 22:23:31 2021 -0400

    Working on updating with int lists

    Working on the demo test, and trying to make updating with lists work

commit dabfe594c19fb8746330544396028c58ef36825e
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Apr 25 20:18:37 2021 -0400

    simplified tree height

commit 1e49c729712f2acf34067f4550a467e47c251c43
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Apr 25 19:51:59 2021 -0400

    tree_height test working

commit 9cac1c9345a1a29bb9165919961b417d72bac220
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 19:40:29 2021 -0400

    Fixed the fail tests

    I have thee fail tests functioning properly: the make file runs them
```

```
        and the errors are as expected

commit 09243310f0fe9c97c8e366b9ab940e748cbb5d64
Author: kshi18 <karenshi0914@gmail.com>
Date:   Sun Apr 25 17:11:57 2021 -0400

    tree traversal works

commit f44e68c15bd9795c18edf05e06120de6b5c730c8
Author: kshi18 <karenshi0914@gmail.com>
Date:   Sun Apr 25 15:42:04 2021 -0400

    tree traversal function

commit 8553c1aa7b886b315c3ed77a00d05dc306983df2
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Apr 25 15:34:23 2021 -0400

    trees get_left and get_right working

commit 0ccf86f67c45b9d4c87b09aeb88f24c5ecdb4f75
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Apr 25 14:24:23 2021 -0400

    tree node .data works + tree tests

commit 7e2a69cc3596ffcc8f9d8a21c6897bac979df34e
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 14:08:17 2021 -0400

    Check the preeformance of strings being declared and being used

commit e3f1c66e3e5aac11654ff869d74808442d75ba15
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 13:40:25 2021 -0400

    Renamed the a test the complexintuse test

commit 6f643f4e635a5c91a8d077fd118cf61d9483ae8d
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 13:35:47 2021 -0400

    fixe more extensions

    fixed .nd.mc and .mc to .nd

commit 1979f1a6d4deedea209e16781a9f85e7c8ed177e
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 13:32:12 2021 -0400
```

```
    Fixed file end error

    Switched .nd.mc to .nd

commit 19dbef3f6e99e1a5c237e002576cca9bcf7f4ff8
Merge: ea93545 2dbab10
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 13:16:16 2021 -0400

    Merged the variable branch into this branch

commit ea935452d70ce9be4b4c1ecf6b465f8ca2aa74ba
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 13:10:47 2021 -0400

    Merged graph_attempt2 in and changed some file extensions

    Merged with graph_attempt2 and switched remaining .mc files to .nd
        files

commit 0f073508c06a170426491e02c471e85faaa550fe
Merge: 9e41e94 2d2466b
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 13:01:24 2021 -0400

    Merged with graph_attempt2

commit 9e41e94f71d5888fc1815c465a65977179f8f6c6
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 12:37:00 2021 -0400

    Fixed the print fail tests

    Fixed the print fail tests to be relavent to our language, not microc

commit 2dbab10249a65670a0f5e59990cc27c4fbe2ebc0
Author: kshi18 <karenshi0914@gmail.com>
Date:   Sun Apr 25 12:29:46 2021 -0400

    slight changes to variables

commit 2a3476f3179c4a408daf7f037f18f2195cc15dc3
Author: kshi18 <karenshi0914@gmail.com>
Date:   Sun Apr 25 12:25:53 2021 -0400

    all tests working now

commit 82ac5b5c56ae5b438bedf69c7ac9a4c43b5b6137
Author: kshi18 <karenshi0914@gmail.com>
Date:   Sun Apr 25 12:07:05 2021 -0400
```

```
    resolved variable issue, a few errors and failed tests to fix

commit 684bbd9c4fb877afaf7728873508eb71e3a81356
Author: nm3076 <nm3076@columbia.edu>
Date:   Sun Apr 25 23:30:41 2021 +0800

    attempt at trees

commit 477640e7df4c12a33b2dc76826e3a60f4d07ff5e
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Sun Apr 25 11:16:26 2021 -0400

    Fixed the .err files

    Fixed the .err test files to match our program output

commit 2d2466b3398a2f3c626ad7b0fa65fc80220aea7a
Author: nm3076 <nm3076@columbia.edu>
Date:   Sun Apr 25 21:17:01 2021 +0800

    append and recursion working

commit 5023c2b085eb5da61dba3bf88658fce1937f8dbd
Author: nm3076 <nm3076@columbia.edu>
Date:   Sun Apr 25 18:51:19 2021 +0800

    update_elem is user callable

commit 65abe928c826b07397d3999f06b9a726827cace4
Author: kshi18 <karenshi0914@gmail.com>
Date:   Sun Apr 25 04:02:32 2021 -0400

    working on update_elem function

commit a82af1e30a8911f9e5c7e9d86643c8a29565738f
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Apr 25 03:56:43 2021 -0400

    add_elem started, not working currently

commit cea518e82005beb44f526b1d01ae983eadda58f7
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Apr 25 02:21:42 2021 -0400

    more list changes

commit 2c57d319818b7e162be0a521562e0e7ad93c8442
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Apr 25 02:15:55 2021 -0400
```

```
    list changes

commit b789489af0aed4036a6b7a0183873ddbfe2ea97b
Merge: 94b89dd 62212e8
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Apr 25 02:04:33 2021 -0400

    Merge branch 'main' of https://github.com/kshi18/nodable into main

commit 94b89ddc2b37b5dc62cf3c959757859a6dd7aebb
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Apr 25 01:59:46 2021 -0400

    list indexing changes

commit 62212e8a9f3db83458e6e7d7d5599b8ec39c9a64
Author: kshi18 <karenshi0914@gmail.com>
Date:   Sat Apr 24 12:31:00 2021 -0400

    changed test file extension to .nd

commit cf018494e83b05800551cd7d6453ba925022fb95
Merge: d07f29e d3d2acd
Author: kshi18 <karenshi0914@gmail.com>
Date:   Sat Apr 24 12:20:38 2021 -0400

    Merge pull request #7 from kshi18/node_attempt

    Node attempt

commit d3d2acda0935f0ea5dc085e9523455e417175dcf
Author: kshi18 <karenshi0914@gmail.com>
Date:   Sat Apr 24 00:52:39 2021 -0400

    updated node tests

commit 5cd2255d9101bffa6ceb2b7f98e40b580939b60a
Author: kshi18 <karenshi0914@gmail.com>
Date:   Fri Apr 23 23:47:54 2021 -0400

    updated the way nodes are parsed

commit 7a928f4656ca7a059b514a4ceb304a1fc9e80026
Author: kshi18 <karenshi0914@gmail.com>
Date:   Fri Apr 23 23:44:16 2021 -0400

    another node test added

commit f4faad8a597bebd1c5b3a018cf82ac0a9b136352
```

```
Author: kshi18 <karenshi0914@gmail.com>
Date:   Fri Apr 23 23:32:16 2021 -0400

    list of nodes test works

commit 67937eac10b15b85041af0d94ed9391a104fa55a
Author: kshi18 <karenshi0914@gmail.com>
Date:   Fri Apr 23 22:30:42 2021 -0400

    DOLLAR for node literals

commit 81aaedb1576a436dc2a3f40c832ae95fd3695b6a
Author: kshi18 <karenshi0914@gmail.com>
Date:   Fri Apr 23 22:26:30 2021 -0400

    get_data for nodes

commit 150966f877bbacca2972be2b1da14760921bb243
Author: kshi18 <karenshi0914@gmail.com>
Date:   Fri Apr 23 19:18:52 2021 -0400

    added a few more basic node tests

commit 305f4cad00253109d66484ef7d06a7656a5aaf00
Author: kshi18 <karenshi0914@gmail.com>
Date:   Fri Apr 23 18:31:04 2021 -0400

    test suite runs fail tests automatically

commit cbb0d406f0f69a85fef85ae98153bb89d000e8fe
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Fri Apr 23 18:03:31 2021 -0400

    Micro C failed tests

    Failed tests from micro c. Still need to figure out how to make them
        run attomatically when we call make.

commit b2bc80cc4d6a62ca0ffe720bd1ad23a9c098a310
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Fri Apr 23 17:56:59 2021 -0400

    Added a test that used lists and functions

commit 15defc616268c6af93371ee5facb45838c99b1b6
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Fri Apr 23 17:13:26 2021 -0400

    I added a second comment tester that checks multiline comments and
        comments following code on the same line
```

```
commit d8dda121f0d347fe68b0ea10a7cee336f38703b9
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Fri Apr 23 17:05:50 2021 -0400

    I added the first comment test

commit 4ac7b76b29d0adc1819a531f787f5498c1f5d047
Author: nm3076 <nm3076@columbia.edu>
Date:   Sat Apr 24 01:17:18 2021 +0800

    node first draft

commit 129948c5cec90c363a0570f307866c0bc4620535
Merge: 41d3a37 d07f29e
Author: nm3076 <nm3076@columbia.edu>
Date:   Fri Apr 23 17:38:54 2021 +0800

    Merge branch 'main' of https://github.com/kshi18/nodable into
        node_attempt

commit 41d3a371ae5c03c73c6045a1686470717e76a509
Author: nm3076 <nm3076@columbia.edu>
Date:   Fri Apr 23 17:37:58 2021 +0800

    attemping nodes

commit d07f29e20b143e812c151a60e790463469a2df00
Merge: 0de0aad 64db95a
Author: Naviya Makhija <43485094+nm3076@users.noreply.github.com>
Date:   Fri Apr 23 15:04:45 2021 +0800

    Merge pull request #6 from kshi18/list_fix

    Nested lists and size()

commit 64db95a320b6c45330c22ebb489691d29df746a0
Author: nm3076 <nm3076@columbia.edu>
Date:   Fri Apr 23 15:03:11 2021 +0800

    cleaned up tests

commit 6206d01da9b9e26f3c320d4e44d0ce5e92092f80
Author: nm3076 <nm3076@columbia.edu>
Date:   Fri Apr 23 14:56:29 2021 +0800

    nested lists and user size func added

commit 0de0aad84acde245862c9b6537d3a5b6359e642b
Merge: 6069737 8c13e60
```

```
Author: Naviya Makhija <43485094+nm3076@users.noreply.github.com>
Date:   Fri Apr 23 12:39:08 2021 +0800

    Merge pull request #5 from kshi18/list_attempt

    Merging lists

commit 8c13e6020a735a8ea2be83ae3dc9d923dbcc9122
Author: nm3076 <nm3076@columbia.edu>
Date:   Thu Apr 22 20:17:16 2021 +0800

    lists working for all data types

commit 3ee640f3edba6a7022472dcf59cd1e858828b736
Author: nm3076 <nm3076@columbia.edu>
Date:   Thu Apr 22 17:57:11 2021 +0800

    fixed list functions

commit 816937cf3dad1be4c97bfea51fc0c49e568daf59
Author: nm3076 <nm3076@columbia.edu>
Date:   Thu Apr 22 10:10:05 2021 +0800

    lists working w backwards indexing

commit 8885d0116f2955f7e31f10e5bcee5cdddf64a36d
Author: nm3076 <nm3076@columbia.edu>
Date:   Wed Apr 21 21:04:39 2021 +0800

    parsing error fixed - fix function linking

commit 30b9ca24e3faaacad4c00106306c094739c458bf
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Apr 19 22:42:14 2021 -0400

    fixed a few more warnings

commit 9aae18e46f4a0d641e036fb101ee82a062fce3a6
Author: nm3076 <nm3076@columbia.edu>
Date:   Tue Apr 20 06:24:44 2021 +0800

    warning for indexing left

commit 1c0decb06000341212b1034ce08fac922756af3c
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Apr 19 17:27:21 2021 -0400

    parsing error in lists

commit adf450733ec0a67e2f3e5bf4667e5dc5e090bc6a
```

```
Author: nm3076 <nm3076@columbia.edu>
Date:   Fri Apr 16 06:35:22 2021 +0800

    list in c from OH

commit b724c3e74339627894a46d225110df7b772d9b84
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Apr 12 03:06:25 2021 -0400

    func3 and float3 tests working

commit c36de2c11dc6491e25faca41695930f6edfe989f
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 12 09:33:22 2021 +0800

    majority of tests fixed - func3 and float3 left

commit 078bc5dbf347313d89c379caed0d8a8acc684de3
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 12 07:54:07 2021 +0800

    if and while complete with functions

commit 6ec07df6eb378c81caa6f1fae0316c3f283cecef
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 12 07:50:09 2021 +0800

    functions implemented

commit 475533217bfa73ce225c4fc1c2ec23b7a8e44702
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 12 06:50:17 2021 +0800

    cursory tests for for while and if done

commit 60ee7df1d75830b13dd98fbce66af0c21743fd41
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 12 06:32:07 2021 +0800

    cursory operator tests finalized

commit 60d107a12dbe950ca96bc302ef2d39e7150fbb42
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Apr 12 05:39:01 2021 +0800

    finalized arithmetic and data type testing

commit 6069737cd3ca5577ccafb37fc62d35c054a56f09
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Mon Apr 5 10:43:46 2021 -0400
```

```
    added check_bool_expr and changed print_bool

commit 09afddea44d0846ced84146e938ed3b81058aa4c
Merge: 80c282c a15a504
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Fri Apr 2 00:37:25 2021 -0400

    Merge pull request #4 from kshi18/working_on_loops

    If statements and loops compile

commit 80c282cb7899a405bdb2bf612445b95e5dcdaf6a
Merge: 9038e9d dfd26c9
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Fri Apr 2 00:34:58 2021 -0400

    Merge pull request #3 from kshi18/variables

    Variables

commit dfd26c9975e5a75746cfbe5fa0e83b84ceb52519
Merge: 5fd09eb 9038e9d
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Fri Apr 2 00:34:47 2021 -0400

    Merge branch 'main' into variables

commit a15a50490b385ad7439183d5ba2f07d9baf6d9f8
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Fri Apr 2 00:34:36 2021 -0400

    If statements and loops compile

commit 9038e9d76acbb35decc2f6ca28966faeac17d142
Merge: 58907c5 a312a4f
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Fri Apr 2 00:29:13 2021 -0400

    Merge pull request #2 from kshi18/ops

    Ops

commit a312a4f5f82b5581d560410e3a9e51aa6176b75d
Merge: 88598b0 58907c5
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Fri Apr 2 00:27:11 2021 -0400

    Merge branch 'main' into ops
```

```
commit 58907c532ec9372f17f1a53ce6c9ab98dc3d9b8e
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Fri Apr 2 00:13:33 2021 -0400

    added float and bool (and tests)

commit 5fd09ebd8233e3a62f60e7d2f2d7269e5ee0836f
Author: nm3076 <nm3076@columbia.edu>
Date:   Thu Apr 1 02:07:50 2021 -0400

    global local vars and assign op

commit 232f114f40f93c819be6336ee8316043a689767d
Author: nm3076 <nm3076@columbia.edu>
Date:   Thu Apr 1 02:07:17 2021 -0400

    DS store in git ignore

commit 88598b002311d8916c4edfd52c8ed8fb5a5070bc
Author: kshi18 <karenshi0914@gmail.com>
Date:   Thu Apr 1 05:56:12 2021 +0000

    modulo works

commit 8b6e62c28a614d4761b95f7b55bcfc3abea89c4c
Author: kshi18 <karenshi0914@gmail.com>
Date:   Thu Apr 1 05:30:16 2021 +0000

    mult and div testing suites working

commit 8d076925d195a0f4b0520b321fc6e04a8e6f891d
Author: kshi18 <karenshi0914@gmail.com>
Date:   Wed Mar 31 17:57:40 2021 +0000

    subtraction test suites added

commit cd37db1297f60744a268952a9209632b0f868334
Author: kshi18 <karenshi0914@gmail.com>
Date:   Wed Mar 31 16:49:11 2021 +0000

    addition tests working

commit 267776d169481d28d2fe72e74d00ec2948484a5b
Merge: d35e344 1b5fa75
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Mar 28 16:52:12 2021 -0400

    Merge pull request #1 from kshi18/hello-world

    Hello world milestone
```

```
commit 1b5fa750d27a62725fa3d5cd6314160e0de3dd2b
Author: kshi18 <karenshi0914@gmail.com>
Date:   Wed Mar 24 13:43:31 2021 -0400

    updated README

commit e7249b2b55de6a4feb80741af68905b2f44477d6
Author: Karen Shi <karenshi0914@gmail.com>
Date:   Wed Mar 24 17:40:16 2021 +0000

    creating tar.gz

commit 5ac54477deccd3bd3595294e40a9ff1923c20dfd
Author: kshi18 <karenshi0914@gmail.com>
Date:   Wed Mar 24 12:37:28 2021 -0400

    updated README

commit 8847e5b5cf74349c800bac5b44109e4565731a2c
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Wed Mar 24 00:41:35 2021 -0400

    updated readme for hello world

commit f433fda1b378cad60732b54b4584479274a9eea3
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Wed Mar 24 00:09:57 2021 -0400

    print string functionality added

commit 4d68706ac3023fd7e59cbaea3f1f4f566f8a2069
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Tue Mar 23 18:16:36 2021 -0400

    print int updated

commit 98239e7303d95897b19c88eeedb61c41c36ff96e
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Tue Mar 23 18:10:37 2021 -0400

    printint works!!!!!!!!!
    Co-authored-by: kshi18 <kshi18@users.noreply.github.com>
    Co-authored-by: ag129 <ag129@users.noreply.github.com>
    Co-authored-by: Naviya Makhija <nm3076@users.noreply.github.com>

commit 0eb9dcbcf73a3032cd6c9b32956085929a796f8b
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Mon Mar 22 21:52:43 2021 -0400
```

```
    Co-authored-by: kshi18 <kshi18@users.noreply.github.com>
    Co-authored-by: ag129 <ag129@users.noreply.github.com>
    Co-authored-by: Naviya Makhija <nm3076@users.noreply.github.com>

commit 096f77e6677264b4d77ffa2f6042799463260f4f
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Mon Mar 22 20:05:16 2021 -0400

    part of codegen.ml

commit 3a1a919b7e5f64a642a50efca8d69a93302a9db3
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Mar 22 17:26:49 2021 -0400

    LLVM module error fixed

commit 6b9a3f65a2c53df117adc747e1f38a79e9324e6f
Author: ag129 <62814033+ag129@users.noreply.github.com>
Date:   Mon Mar 22 16:54:04 2021 -0400

    Starting Codegen

    Working on codegen. We created the codegen.ml file, the
        test-printint.mc file and the test-printint.out file. We also
        modified nodable.ml

commit 829f9052b157e999435d8bd09e6d12674554ca45
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Mar 21 11:32:41 2021 -0400

    added hello world test

commit 19f7a6199408d483feaf345006531d373f4146d4
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Mar 21 11:25:02 2021 -0400

    fixed void and exhaustive pattern matching issues

commit d6ae10b43542e7f70cde0818a7c6b5e632abe022
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Mar 21 10:53:22 2021 -0400

    minor changes to semant.ml

commit fefe279771edb74869662b626c41b8c173a261d0
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Mar 21 02:23:15 2021 -0400

    semant minor fix
```

```
commit ad288169cfd92e1cff0e08c3d0f692611636e65a
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sun Mar 21 02:21:38 2021 -0400

    first draft sast and semant.ml

commit 4110eec0d3d7b624a124bb338adee98aec95b547
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sat Mar 20 10:27:41 2021 -0400

    added docker command to readme

commit 625d212488c1b5227fc04de169d7d2b414b1cca4
Author: kshi18 <karenshi0914@gmail.com>
Date:   Fri Mar 19 19:14:14 2021 -0400

    Makefile compiles without test suite

commit e6da1a51a89a5b607deedcfd0ee2a15ff9a82fb0
Author: nm3076 <nm3076@columbia.edu>
Date:   Fri Mar 19 13:23:19 2021 -0400

    updated makefile and basic testing suite

commit d35e3448a8c530c6c12b0a58e2675888d9d9979d
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Fri Mar 19 10:25:35 2021 -0400

    fixed multiply inconsistency in parser

commit b778d0729017a43889080202599fac5c44218313
Author: kshi18 <karenshi0914@gmail.com>
Date:   Thu Mar 18 17:16:32 2021 -0400

    make command works

commit 2a473e011f300336196b05554c603dbeb308e3d3
Author: kshi18 <karenshi0914@gmail.com>
Date:   Thu Mar 18 17:11:06 2021 -0400

    fixed lit errors in scanner

commit 6f83ccb4a43a31414bd5a9de8b7aa78143658584
Author: kshi18 <karenshi0914@gmail.com>
Date:   Thu Mar 18 17:07:11 2021 -0400

    fixed int_lit error

commit b58ede4555fdfcc0ae093d5005f32c17be25a88e
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
```

```
Date:   Wed Mar 17 22:48:10 2021 -0400

    fixed keyword mismatches btwn scanner and parser

commit 3c1e5aa710c3673e19422a30038c8e8e3b62625e
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Wed Mar 17 22:33:35 2021 -0400

    edited if statements in parser

commit 327db9d5cc8d8fe65e824a61daff1550cef98191
Author: nm3076 <nm3076@columbia.edu>
Date:   Wed Mar 17 02:49:21 2021 -0400

    comment in Makefile abt .mli

commit 9139a8d2ac914666d6f7ed76563aa32383cbf16b
Author: kshi18 <karenshi0914@gmail.com>
Date:   Wed Mar 17 02:47:02 2021 -0400

    debugged parser and ast from make

commit 308286ebf4bc4e36823b5f4961adc712430a8742
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Tue Mar 16 19:15:07 2021 -0400

    added makefile

commit abcdd029e9cab6f9ca0539c9f538eb7c688c2f62
Merge: ace6b80 2f40365
Author: nm3076 <nm3076@columbia.edu>
Date:   Tue Mar 16 14:12:07 2021 -0400

    Merge branch 'main' of https://github.com/kshi18/nodable into main

    Merging test.ml

commit ace6b80e0c6433bbf17f93d58103d14ebfc2adb6
Author: nm3076 <nm3076@columbia.edu>
Date:   Tue Mar 16 14:11:14 2021 -0400

    Co-authored-by: ag129 <ag129@users.noreply.github.com>

commit 2f403653c0f3af49aff69a2f555a320d90a23486
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Tue Mar 16 13:31:02 2021 -0400

    added test1.ml

commit f5a7f8da4bb145c425f79b681bb2aaa6be15e3d3
```

```
Merge: 230f0e9 dfaf8be
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Mar 15 18:16:55 2021 -0400

    Merge branch 'main' of https://github.com/kshi18/nodable into main

    Merging scanner and parser

commit 230f0e94b7ef87952a970e6d9fb5eb0ee6a12148
Author: nm3076 <nm3076@columbia.edu>
Date:   Wed Mar 10 00:48:24 2021 -0500

    fixed errors for compilation draft 1 ast

commit dfaf8be9ac8a22e4c78ce857dff4bd6827b3aa6e
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Tue Mar 9 22:58:18 2021 -0500

    no errors in scanner.mll!

commit 79420da979a1469d1f2998f88f9d0e60e1e16523
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Tue Mar 9 22:53:32 2021 -0500

    fixed more errors in scanner.mll

commit 1da2610c75bfb1e9b8bab92bded99ca191cbcf13
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Tue Mar 9 22:47:54 2021 -0500

    fixed some syntax errors in scanner.mll

commit 758f5381163c4c67bc22ad2873fa850fa0323250
Author: kshi18 <karenshi0914@gmail.com>
Date:   Tue Mar 9 22:39:42 2021 -0500

    first draft of scanner

commit 1978f441ef0ac2f0f52a5e2f68fd31381124b762
Author: nm3076 <nm3076@columbia.edu>
Date:   Tue Mar 9 22:26:18 2021 -0500

    first draft of ast

commit e2e7eee9db56a2f9db3440ab11be5694ec5ef715
Author: nm3076 <nm3076@columbia.edu>
Date:   Wed Feb 24 11:12:09 2021 -0500

    Updated vdecl to include nodes, graphs and trees
```

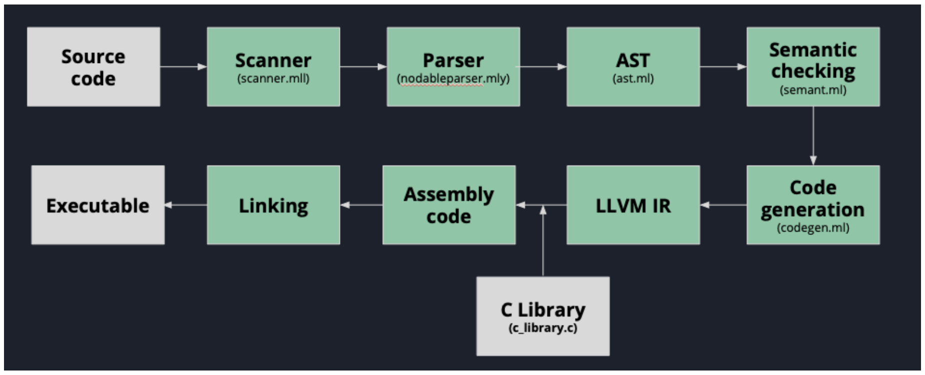
```
commit e83cbb49de8bede00805d2033d1a3a5bcdaa2345
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Feb 22 19:35:15 2021 -0500

    Parser compiles

commit 00e69275473f167321b813e0cc40ca0665d96dfc
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Feb 22 19:13:49 2021 -0500

    down to 11 shift/reduce errors

commit 0fb33f5f32ca823ec3bbc2bc1bc7869a1ca0424f
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Feb 22 18:56:18 2021 -0500

    now 34 errors

commit ef0c2b9ce9e41f2fc60a13fb9eb2085534fa4643
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Feb 22 18:42:46 2021 -0500

    down to 45 shift/reduce errors

commit 4914a5d719ec9f416f94db377ce4718919956118
Author: kshi18 <karenshi0914@gmail.com>
Date:   Mon Feb 22 12:42:36 2021 -0500

    fixed stmt actions, need to fix shift/reduce conflicts

commit 62d030bac219e8f78d1216a04b960a92cb933543
Author: nm3076 <nm3076@columbia.edu>
Date:   Mon Feb 22 01:58:27 2021 -0500

    Working on parser errors - need to fix stmt actions

commit 8a3f4751157703f894409c2f6248449689bb8699
Author: nm3076 <nm3076@columbia.edu>
Date:   Sun Feb 21 23:37:26 2021 -0500

    First draft of parser

commit c0bb8c2860019c793cf8b5b9267d2d0f331df103
Author: Ajita Bala <47126432+ajitabala22@users.noreply.github.com>
Date:   Sat Feb 20 14:10:56 2021 -0500

    just opened scanner, parser, and ast files

commit 6c6ad30c39699db69d27b6a2c2e171d0eebbc00d
Author: kshi18 <karenshi0914@gmail.com>
```

```
Date:   Mon Feb 15 21:07:18 2021 -0500

    Initial commit
```

---

5 Architectural Design

5.1 Component Diagram



5.2 Component Design and Authors

5.2.1 Scanner

Authors: Ajita and Karen

Our scanner (scanner.ml) reads in text from the source program and returns a stream of tokens.

5.2.2 Parser and AST

Authors: Ariel, Naviya, Ajita, Karen

The parser (nodableparser.mly) reads in tokens from the scanner and produces an abstract syntax tree (ast.ml) using rules from nodable's grammar. The parser checks list and node types.

5.2.3 Semantic Checking

Authors: Ariel, Naviya, Ajita, Karen

The semantic checker (semant.ml) goes through each node of the abstract syntax tree and type checks - makes sure that each operator has the appropriate operands. It uses information from the symbol table to implement C-style variable scope and ensure the source program is semantically consistent. The type-checked expressions are added to a semantically-checked abstract syntax tree (sast.ml).

5.2.4 Code generation

Authors: Ariel, Naviya, Ajita, Karen

The code generator (codegen.ml) steps through each node of the SAST and returns an LLVM module. The code generator constructs expressions bottom-up and constructs basic blocks for control-flow statements. It creates a map of global variables, allocates formal arguments and local variables on the stack, declares external functions for lists and nodes, defines each function, then fills in the body of each function. Lists were allocated on the heap by calling external C functions.

5.2.5 C Libraries

Authors: Ariel, Naviya, Ajita, Karen

Nodable used C structs to internally represent lists and nodes. Lists were implemented as a linked list.

6 Test Plan

6.1 Source Language Programs

6.1.1 List Operations

```
//  *** tests/test-listdemo.nd ***  //
int average(list <node<int>> a)
{
    int sum;
    int i;
    sum = 0;
    for(i=0; i < size(a); i=i+1)
    {
        sum = sum + a[i].data;
    }
    return sum/(size(a));
}

void printElements(list <node<int>> a)
{
    int i;
    for(i=0; i < size(a); i=i+1)
    {
        print(a[i].data);
    }
}

void reverse(list <node<int>> a)
{
    int i;
    node<int> temp;
    for(i=0; i<(size(a)/2); i = i+1)
    {
        temp = a[i];
        a = update_elem(a[size(a)-i-1], a, i);
        a = update_elem(temp, a, size(a)-i-1);
    }
}

void selectionSort(list <node<int>> a)
{
    int i;
    node<int> min;
    int minIndex;
    node<int> temp;
    int j;
    for(i=0; i < size(a)-1; i=i+1)
    {
        min = a[i];
        minIndex = i;
        for(j = i+1; j<size(a); j=j+1)
        {
            if(min.data > a[j].data)
            {
                min = a[j];
                minIndex=j;
            }
        }

        temp = a[i];
        a = update_elem(min, a, i);
        a = update_elem(temp, a, minIndex);

    }
}


int main()
{
    prints("Let's make the list of nodes with values 2,2,4,1,7");
    list<node<int> > t;
    node<int> a;
    node<int> b;
    node<int> c;
    node<int> d;
    node<int> e;
    a = $2;
    b = $2;
    c = $4;
    d = $1;
    e = $7;

    t = [a, b, c, d, e];
    prints("Now lets print out each element in order:");
    printElements(t);

    prints("Now lets see how long it is:");
    print(size(t));

    prints("Now lets get its average value and print it (rounded down to
        an integer):");
    print(average(t));

    prints("Now let's reverse the list");
    reverse(t);
    prints("Now let's print out each element in their new order:");
    printElements(t);

    prints("Now lets do selection sort to order the list");
    selectionSort(t);

    prints("Now lets print out every element of our newly ordered
        list:");
    printElements(t);

    return 0;
}

//   *** listdemo.ll ***   //
; ModuleID = 'nodable'
source_filename = "nodable"

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@str = private unnamed_addr constant [51 x i8] c"Let's make the list of
    nodes with values 2,2,4,1,7\00"
@str.3 = private unnamed_addr constant [42 x i8] c"Now lets print out
    each element in order:\00"
@str.4 = private unnamed_addr constant [29 x i8] c"Now lets see how long
    it is:\00"
@str.5 = private unnamed_addr constant [74 x i8] c"Now lets get its
    average value and print it (rounded down to an integer):\00"
@str.6 = private unnamed_addr constant [27 x i8] c"Now let's reverse the
    list\00"
@str.7 = private unnamed_addr constant [53 x i8] c"Now let's print out
    each element in their new order:\00"
@str.8 = private unnamed_addr constant [45 x i8] c"Now lets do selection
    sort to order the list\00"
```

```
@str.9 = private unnamed_addr constant [60 x i8] c"Now lets print out
    every element of our newly ordered list:\00"
@fmt.10 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.11 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.12 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.13 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.14 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.15 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.16 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.17 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.18 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.19 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.20 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.21 = private unnamed_addr constant [4 x i8] c"%s\0A\00"

declare i8* @init_node(i8*, ...)

declare i8* @get_data(i8*, ...)

declare i8* @get_left(i8*, ...)

declare i8* @get_right(i8*, ...)

declare i8* @add_left(i8*, ...)

declare i8* @add_right(i8*, ...)

declare i8* @list_init(...)

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

declare i8* @add_elem(i8*, i8*, ...)

declare i8* @append(i8*, i8*, ...)

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

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

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

define i32 @main() {
entry:
  %e = alloca i8*
  %d = alloca i8*
  %c = alloca i8*
  %b = alloca i8*
  %a = alloca i8*
  %t = alloca i8*
```

```
%printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x
    i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr inbounds
    ([51 x i8], [51 x i8]* @str, i32 0, i32 0))
%malloccall = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc = bitcast i8* %malloccall to i32*
store i32 2, i32* %data_malloc
%data_bitcast = bitcast i32* %data_malloc to i8*
%init_node = call i8* (i8*, ...) @init_node(i8* %data_bitcast)
store i8* %init_node, i8** %a
%malloccall1 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc2 = bitcast i8* %malloccall1 to i32*
store i32 2, i32* %data_malloc2
%data_bitcast3 = bitcast i32* %data_malloc2 to i8*
%init_node4 = call i8* (i8*, ...) @init_node(i8* %data_bitcast3)
store i8* %init_node4, i8** %b
%malloccall5 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc6 = bitcast i8* %malloccall5 to i32*
store i32 4, i32* %data_malloc6
%data_bitcast7 = bitcast i32* %data_malloc6 to i8*
%init_node8 = call i8* (i8*, ...) @init_node(i8* %data_bitcast7)
store i8* %init_node8, i8** %c
%malloccall9 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc10 = bitcast i8* %malloccall9 to i32*
store i32 1, i32* %data_malloc10
%data_bitcast11 = bitcast i32* %data_malloc10 to i8*
%init_node12 = call i8* (i8*, ...) @init_node(i8* %data_bitcast11)
store i8* %init_node12, i8** %d
%malloccall13 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc14 = bitcast i8* %malloccall13 to i32*
store i32 7, i32* %data_malloc14
%data_bitcast15 = bitcast i32* %data_malloc14 to i8*
%init_node16 = call i8* (i8*, ...) @init_node(i8* %data_bitcast15)
store i8* %init_node16, i8** %e
%list_init = call i8* (...) @list_init()
%e17 = load i8*, i8** %e
%0 = call i8* (i8*, i8*, ...) @add_elem(i8* %list_init, i8* %e17)
%d18 = load i8*, i8** %d
%1 = call i8* (i8*, i8*, ...) @add_elem(i8* %list_init, i8* %d18)
%c19 = load i8*, i8** %c
%2 = call i8* (i8*, i8*, ...) @add_elem(i8* %list_init, i8* %c19)
%b20 = load i8*, i8** %b
%3 = call i8* (i8*, i8*, ...) @add_elem(i8* %list_init, i8* %b20)
%a21 = load i8*, i8** %a
%4 = call i8* (i8*, i8*, ...) @add_elem(i8* %list_init, i8* %a21)
store i8* %list_init, i8** %t
```

```
  %printf22 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr
      inbounds ([42 x i8], [42 x i8]* @str.3, i32 0, i32 0))
  %t23 = load i8*, i8** %t
  call void @printElements(i8* %t23)
  %printf24 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr
      inbounds ([29 x i8], [29 x i8]* @str.4, i32 0, i32 0))
  %t25 = load i8*, i8** %t
  %size = call i32 (i8*, ...) @size(i8* %t25)
  %printf26 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %size)
  %printf27 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr
      inbounds ([74 x i8], [74 x i8]* @str.5, i32 0, i32 0))
  %t28 = load i8*, i8** %t
  %average_result = call i32 @average(i8* %t28)
  %printf29 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %average_result)
  %printf30 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr
      inbounds ([27 x i8], [27 x i8]* @str.6, i32 0, i32 0))
  %t31 = load i8*, i8** %t
  call void @reverse(i8* %t31)
  %printf32 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr
      inbounds ([53 x i8], [53 x i8]* @str.7, i32 0, i32 0))
  %t33 = load i8*, i8** %t
  call void @printElements(i8* %t33)
  %printf34 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr
      inbounds ([45 x i8], [45 x i8]* @str.8, i32 0, i32 0))
  %t35 = load i8*, i8** %t
  call void @selectionSort(i8* %t35)
  %printf36 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr
      inbounds ([60 x i8], [60 x i8]* @str.9, i32 0, i32 0))
  %t37 = load i8*, i8** %t
  call void @printElements(i8* %t37)
  ret i32 0
}

define void @selectionSort(i8* %a) {
entry:
  %a1 = alloca i8*
  store i8* %a, i8** %a1
  %j = alloca i32
  %temp = alloca i8*
  %minIndex = alloca i32
  %min = alloca i8*
  %i = alloca i32
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %merge27, %entry
  %i40 = load i32, i32* %i
  %a41 = load i8*, i8** %a1
  %size42 = call i32 (i8*, ...) @size(i8* %a41)
  %tmp43 = sub i32 %size42, 1
  %tmp44 = icmp slt i32 %i40, %tmp43
  br i1 %tmp44, label %while_body, label %merge45

while_body:                                       ; preds = %while
  %a2 = load i8*, i8** %a1
  %i3 = load i32, i32* %i
  %get_elem = call i8* (i32, i8*, ...) @get_elem(i32 %i3, i8* %a2)
  store i8* %get_elem, i8** %min
  %i4 = load i32, i32* %i
  store i32 %i4, i32* %minIndex
  %i5 = load i32, i32* %i
  %tmp = add i32 %i5, 1
  store i32 %tmp, i32* %j
  br label %while6

while6:                                           ; preds = %merge,
    %while_body
  %j24 = load i32, i32* %j
  %a25 = load i8*, i8** %a1
  %size = call i32 (i8*, ...) @size(i8* %a25)
  %tmp26 = icmp slt i32 %j24, %size
  br i1 %tmp26, label %while_body7, label %merge27

while_body7:                                      ; preds = %while6
  %min8 = load i8*, i8** %min
  %data = call i8* (i8*, ...) @get_data(i8* %min8)
  %data9 = bitcast i8* %data to i32*
  %data10 = load i32, i32* %data9
  %a11 = load i8*, i8** %a1
  %j12 = load i32, i32* %j
  %get_elem13 = call i8* (i32, i8*, ...) @get_elem(i32 %j12, i8* %a11)
  %data14 = call i8* (i8*, ...) @get_data(i8* %get_elem13)
  %data15 = bitcast i8* %data14 to i32*
  %data16 = load i32, i32* %data15
  %tmp17 = icmp sgt i32 %data10, %data16
  br i1 %tmp17, label %then, label %else

merge:                                            ; preds = %else, %then
  %j22 = load i32, i32* %j
  %tmp23 = add i32 %j22, 1
  store i32 %tmp23, i32* %j
  br label %while6

then:                                             ; preds = %while_body7
  %a18 = load i8*, i8** %a1
  %j19 = load i32, i32* %j
  %get_elem20 = call i8* (i32, i8*, ...) @get_elem(i32 %j19, i8* %a18)
  store i8* %get_elem20, i8** %min
  %j21 = load i32, i32* %j
  store i32 %j21, i32* %minIndex
  br label %merge

else:                                             ; preds = %while_body7
  br label %merge

merge27:                                          ; preds = %while6
  %a28 = load i8*, i8** %a1
  %i29 = load i32, i32* %i
  %get_elem30 = call i8* (i32, i8*, ...) @get_elem(i32 %i29, i8* %a28)
  store i8* %get_elem30, i8** %temp
  %i31 = load i32, i32* %i
  %a32 = load i8*, i8** %a1
  %min33 = load i8*, i8** %min
  %update_elem = call i8* (i8*, i8*, i32, ...) @update_elem(i8* %min33,
      i8* %a32, i32 %i31)
  store i8* %update_elem, i8** %a1
  %minIndex34 = load i32, i32* %minIndex
  %a35 = load i8*, i8** %a1
  %temp36 = load i8*, i8** %temp
  %update_elem37 = call i8* (i8*, i8*, i32, ...) @update_elem(i8*
      %temp36, i8* %a35, i32 %minIndex34)
  store i8* %update_elem37, i8** %a1
  %i38 = load i32, i32* %i
  %tmp39 = add i32 %i38, 1
  store i32 %tmp39, i32* %i
  br label %while

merge45:                                          ; preds = %while
  ret void
}

define void @reverse(i8* %a) {
entry:
  %a1 = alloca i8*
  store i8* %a, i8** %a1
  %temp = alloca i8*
  %i = alloca i32
  store i32 0, i32* %i
  br label %while
```

```
while:                                            ; preds = %while_body,
    %entry
  %i21 = load i32, i32* %i
  %a22 = load i8*, i8** %a1
  %size23 = call i32 (i8*, ...) @size(i8* %a22)
  %tmp24 = sdiv i32 %size23, 2
  %tmp25 = icmp slt i32 %i21, %tmp24
  br i1 %tmp25, label %while_body, label %merge

while_body:                                       ; preds = %while
  %a2 = load i8*, i8** %a1
  %i3 = load i32, i32* %i
  %get_elem = call i8* (i32, i8*, ...) @get_elem(i32 %i3, i8* %a2)
  store i8* %get_elem, i8** %temp
  %i4 = load i32, i32* %i
  %a5 = load i8*, i8** %a1
  %a6 = load i8*, i8** %a1
  %a7 = load i8*, i8** %a1
  %size = call i32 (i8*, ...) @size(i8* %a7)
  %i8 = load i32, i32* %i
  %tmp = sub i32 %size, %i8
  %tmp9 = sub i32 %tmp, 1
  %get_elem10 = call i8* (i32, i8*, ...) @get_elem(i32 %tmp9, i8* %a6)
  %update_elem = call i8* (i8*, i8*, i32, ...) @update_elem(i8*
      %get_elem10, i8* %a5, i32 %i4)
  store i8* %update_elem, i8** %a1
  %a11 = load i8*, i8** %a1
  %size12 = call i32 (i8*, ...) @size(i8* %a11)
  %i13 = load i32, i32* %i
  %tmp14 = sub i32 %size12, %i13
  %tmp15 = sub i32 %tmp14, 1
  %a16 = load i8*, i8** %a1
  %temp17 = load i8*, i8** %temp
  %update_elem18 = call i8* (i8*, i8*, i32, ...) @update_elem(i8*
      %temp17, i8* %a16, i32 %tmp15)
  store i8* %update_elem18, i8** %a1
  %i19 = load i32, i32* %i
  %tmp20 = add i32 %i19, 1
  store i32 %tmp20, i32* %i
  br label %while

merge:                                            ; preds = %while
  ret void
}

define void @printElements(i8* %a) {
entry:
  %a1 = alloca i8*
  store i8* %a, i8** %a1
  %i = alloca i32
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %while_body,
    %entry
  %i7 = load i32, i32* %i
  %a8 = load i8*, i8** %a1
  %size = call i32 (i8*, ...) @size(i8* %a8)
  %tmp9 = icmp slt i32 %i7, %size
  br i1 %tmp9, label %while_body, label %merge

while_body:                                       ; preds = %while
  %a2 = load i8*, i8** %a1
  %i3 = load i32, i32* %i
  %get_elem = call i8* (i32, i8*, ...) @get_elem(i32 %i3, i8* %a2)
  %data = call i8* (i8*, ...) @get_data(i8* %get_elem)
  %data4 = bitcast i8* %data to i32*
  %data5 = load i32, i32* %data4
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x
      i8], [4 x i8]* @fmt.16, i32 0, i32 0), i32 %data5)
  %i6 = load i32, i32* %i
  %tmp = add i32 %i6, 1
  store i32 %tmp, i32* %i
  br label %while

merge:                                            ; preds = %while
  ret void
}

define i32 @average(i8* %a) {
entry:
  %a1 = alloca i8*
  store i8* %a, i8** %a1
  %i = alloca i32
  %sum = alloca i32
  store i32 0, i32* %sum
  store i32 0, i32* %i
  br label %while

while:                                            ; preds = %while_body,
    %entry
  %i9 = load i32, i32* %i
  %a10 = load i8*, i8** %a1
  %size = call i32 (i8*, ...) @size(i8* %a10)
  %tmp11 = icmp slt i32 %i9, %size
  br i1 %tmp11, label %while_body, label %merge

while_body:                                       ; preds = %while
  %sum2 = load i32, i32* %sum
  %a3 = load i8*, i8** %a1
  %i4 = load i32, i32* %i
  %get_elem = call i8* (i32, i8*, ...) @get_elem(i32 %i4, i8* %a3)
  %data = call i8* (i8*, ...) @get_data(i8* %get_elem)
  %data5 = bitcast i8* %data to i32*
  %data6 = load i32, i32* %data5
  %tmp = add i32 %sum2, %data6
  store i32 %tmp, i32* %sum
  %i7 = load i32, i32* %i
  %tmp8 = add i32 %i7, 1
  store i32 %tmp8, i32* %i
  br label %while

merge:                                            ; preds = %while
  %sum12 = load i32, i32* %sum
  %a13 = load i8*, i8** %a1
  %size14 = call i32 (i8*, ...) @size(i8* %a13)
  %tmp15 = sdiv i32 %sum12, %size14
  ret i32 %tmp15
}

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

6.1.2 Tree Traversals

```
//   *** tests/test-treetraversal.nd *** //
void printPreOrder (node<int> n) {
    if (n.data == 0) return;

    print(n.data);
    printPreOrder(get_left(n));
    printPreOrder(get_right(n));
}

void printPostOrder (node<int> n) {
    if (n.data == 0) return;

    printPostOrder(get_left(n));
    printPostOrder(get_right(n));
    print(n.data);
}

void printInOrder (node<int> n) {
    if (n.data == 0) return;

    printInOrder(get_left(n));
    print(n.data);
    printInOrder(get_right(n));
}
```

```
int main (){
   node<int> root;
   node<int> c1;
   node<int> c2;
   node<int> c3;
   node<int> c4;
   node<int> null;

   root = $1;

   c1 = $2;
   add_left(root, c1);

   c2 = $3;
   add_right(root, c2);

   c3 = $4;
   add_left(c1, c3);

   c4 = $5;
   add_right(c1, c4);

   null = $0;

   //leaves//
   add_left(c2, null);
   add_right(c2, null);
   add_left(c3, null);
   add_right(c3, null);
   add_left(c4, null);
   add_right(c4, null);

   prints("Preorder traversal of tree:");
   printPreOrder(root);

   prints("Postorder traversal of tree:");
   printPostOrder(root);

   prints("Inorder traversal of tree:");
   printInOrder(root);

   return 0;
}


//   *** treetraversal.ll ***  //
; ModuleID = 'nodable'
source_filename = "nodable"
```

```
@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@str = private unnamed_addr constant [28 x i8] c"Preorder traversal of
    tree:\00"
@str.3 = private unnamed_addr constant [29 x i8] c"Postorder traversal
    of tree:\00"
@str.4 = private unnamed_addr constant [27 x i8] c"Inorder traversal of
    tree:\00"
@fmt.5 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.6 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.7 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.8 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.9 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.10 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.11 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.12 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.13 = private unnamed_addr constant [4 x i8] c"%s\0A\00"

declare i8* @init_node(i8*, ...)

declare i8* @get_data(i8*, ...)

declare i8* @get_left(i8*, ...)

declare i8* @get_right(i8*, ...)

declare i8* @add_left(i8*, ...)

declare i8* @add_right(i8*, ...)

declare i8* @list_init(...)

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

declare i8* @add_elem(i8*, i8*, ...)

declare i8* @append(i8*, i8*, ...)

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

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

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

define i32 @main() {
entry:
  %null = alloca i8*
  %c4 = alloca i8*
  %c3 = alloca i8*
```

```
%c2 = alloca i8*
%c1 = alloca i8*
%root = alloca i8*
%malloccall = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc = bitcast i8* %malloccall to i32*
store i32 1, i32* %data_malloc
%data_bitcast = bitcast i32* %data_malloc to i8*
%init_node = call i8* (i8*, ...) @init_node(i8* %data_bitcast)
store i8* %init_node, i8** %root
%malloccall1 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc2 = bitcast i8* %malloccall1 to i32*
store i32 2, i32* %data_malloc2
%data_bitcast3 = bitcast i32* %data_malloc2 to i8*
%init_node4 = call i8* (i8*, ...) @init_node(i8* %data_bitcast3)
store i8* %init_node4, i8** %c1
%c15 = load i8*, i8** %c1
%root6 = load i8*, i8** %root
%add_left = call i8* (i8*, ...) @add_left(i8* %root6, i8* %c15)
%malloccall7 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc8 = bitcast i8* %malloccall7 to i32*
store i32 3, i32* %data_malloc8
%data_bitcast9 = bitcast i32* %data_malloc8 to i8*
%init_node10 = call i8* (i8*, ...) @init_node(i8* %data_bitcast9)
store i8* %init_node10, i8** %c2
%c211 = load i8*, i8** %c2
%root12 = load i8*, i8** %root
%add_right = call i8* (i8*, ...) @add_right(i8* %root12, i8* %c211)
%malloccall13 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc14 = bitcast i8* %malloccall13 to i32*
store i32 4, i32* %data_malloc14
%data_bitcast15 = bitcast i32* %data_malloc14 to i8*
%init_node16 = call i8* (i8*, ...) @init_node(i8* %data_bitcast15)
store i8* %init_node16, i8** %c3
%c317 = load i8*, i8** %c3
%c118 = load i8*, i8** %c1
%add_left19 = call i8* (i8*, ...) @add_left(i8* %c118, i8* %c317)
%malloccall20 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc21 = bitcast i8* %malloccall20 to i32*
store i32 5, i32* %data_malloc21
%data_bitcast22 = bitcast i32* %data_malloc21 to i8*
%init_node23 = call i8* (i8*, ...) @init_node(i8* %data_bitcast22)
store i8* %init_node23, i8** %c4
%c424 = load i8*, i8** %c4
%c125 = load i8*, i8** %c1
%add_right26 = call i8* (i8*, ...) @add_right(i8* %c125, i8* %c424)
```

```
  %malloccall27 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
  %data_malloc28 = bitcast i8* %malloccall27 to i32*
  store i32 0, i32* %data_malloc28
  %data_bitcast29 = bitcast i32* %data_malloc28 to i8*
  %init_node30 = call i8* (i8*, ...) @init_node(i8* %data_bitcast29)
  store i8* %init_node30, i8** %null
  %null31 = load i8*, i8** %null
  %c232 = load i8*, i8** %c2
  %add_left33 = call i8* (i8*, ...) @add_left(i8* %c232, i8* %null31)
  %null34 = load i8*, i8** %null
  %c235 = load i8*, i8** %c2
  %add_right36 = call i8* (i8*, ...) @add_right(i8* %c235, i8* %null34)
  %null37 = load i8*, i8** %null
  %c338 = load i8*, i8** %c3
  %add_left39 = call i8* (i8*, ...) @add_left(i8* %c338, i8* %null37)
  %null40 = load i8*, i8** %null
  %c341 = load i8*, i8** %c3
  %add_right42 = call i8* (i8*, ...) @add_right(i8* %c341, i8* %null40)
  %null43 = load i8*, i8** %null
  %c444 = load i8*, i8** %c4
  %add_left45 = call i8* (i8*, ...) @add_left(i8* %c444, i8* %null43)
  %null46 = load i8*, i8** %null
  %c447 = load i8*, i8** %c4
  %add_right48 = call i8* (i8*, ...) @add_right(i8* %c447, i8* %null46)
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x
      i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr inbounds
      ([28 x i8], [28 x i8]* @str, i32 0, i32 0))
  %root49 = load i8*, i8** %root
  call void @printPreOrder(i8* %root49)
  %printf50 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr
      inbounds ([29 x i8], [29 x i8]* @str.3, i32 0, i32 0))
  %root51 = load i8*, i8** %root
  call void @printPostOrder(i8* %root51)
  %printf52 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr
      inbounds ([27 x i8], [27 x i8]* @str.4, i32 0, i32 0))
  %root53 = load i8*, i8** %root
  call void @printInOrder(i8* %root53)
  ret i32 0
}

define void @printInOrder(i8* %n) {
entry:
  %n1 = alloca i8*
  store i8* %n, i8** %n1
  %n2 = load i8*, i8** %n1
  %data = call i8* (i8*, ...) @get_data(i8* %n2)
  %data3 = bitcast i8* %data to i32*
  %data4 = load i32, i32* %data3
  %tmp = icmp eq i32 %data4, 0
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %n5 = load i8*, i8** %n1
  %get_left = call i8* (i8*, ...) @get_left(i8* %n5)
  call void @printInOrder(i8* %get_left)
  %n6 = load i8*, i8** %n1
  %data7 = call i8* (i8*, ...) @get_data(i8* %n6)
  %data8 = bitcast i8* %data7 to i32*
  %data9 = load i32, i32* %data8
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x
      i8], [4 x i8]* @fmt.5, i32 0, i32 0), i32 %data9)
  %n10 = load i8*, i8** %n1
  %get_right = call i8* (i8*, ...) @get_right(i8* %n10)
  call void @printInOrder(i8* %get_right)
  ret void

then:                                             ; preds = %entry
  ret void

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

define void @printPostOrder(i8* %n) {
entry:
  %n1 = alloca i8*
  store i8* %n, i8** %n1
  %n2 = load i8*, i8** %n1
  %data = call i8* (i8*, ...) @get_data(i8* %n2)
  %data3 = bitcast i8* %data to i32*
  %data4 = load i32, i32* %data3
  %tmp = icmp eq i32 %data4, 0
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %n5 = load i8*, i8** %n1
  %get_left = call i8* (i8*, ...) @get_left(i8* %n5)
  call void @printPostOrder(i8* %get_left)
  %n6 = load i8*, i8** %n1
  %get_right = call i8* (i8*, ...) @get_right(i8* %n6)
  call void @printPostOrder(i8* %get_right)
  %n7 = load i8*, i8** %n1
  %data8 = call i8* (i8*, ...) @get_data(i8* %n7)
  %data9 = bitcast i8* %data8 to i32*
  %data10 = load i32, i32* %data9
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x
      i8], [4 x i8]* @fmt.8, i32 0, i32 0), i32 %data10)
  ret void

then:                                             ; preds = %entry
  ret void

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

define void @printPreOrder(i8* %n) {
entry:
  %n1 = alloca i8*
  store i8* %n, i8** %n1
  %n2 = load i8*, i8** %n1
  %data = call i8* (i8*, ...) @get_data(i8* %n2)
  %data3 = bitcast i8* %data to i32*
  %data4 = load i32, i32* %data3
  %tmp = icmp eq i32 %data4, 0
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %n5 = load i8*, i8** %n1
  %data6 = call i8* (i8*, ...) @get_data(i8* %n5)
  %data7 = bitcast i8* %data6 to i32*
  %data8 = load i32, i32* %data7
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x
      i8], [4 x i8]* @fmt.11, i32 0, i32 0), i32 %data8)
  %n9 = load i8*, i8** %n1
  %get_left = call i8* (i8*, ...) @get_left(i8* %n9)
  call void @printPreOrder(i8* %get_left)
  %n10 = load i8*, i8** %n1
  %get_right = call i8* (i8*, ...) @get_right(i8* %n10)
  call void @printPreOrder(i8* %get_right)
  ret void

then:                                             ; preds = %entry
  ret void

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

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

6.1.3 Check Tree Balance

```
//   *** tests/test-checktreebalance.nd *** //
int abs(int value) {
    int ret;
    if (value <= 0) {
        ret = value * -1;
    }
    else {
        ret = value;
    }
    return ret;
}
int tree_height(node<int> root) {
    int left_height;
    int right_height;
    int max;

    if (root.data == 0) {
        max = 0;
        return max;
    }
    else {
        left_height = tree_height(get_left(root));
        right_height = tree_height(get_right(root));
        if (left_height < right_height) {
            max = right_height + 1;
        }
        else {
            max = left_height + 1;
        }

    }
    return max;
}

boolean check_balance(node<int> root) {
    int left_height;
    int right_height;
    int difference;
    boolean result;
    if (root.data == 0) {
        result = true;
        return result;
    }
    left_height = tree_height(get_left(root));
    right_height = tree_height(get_right(root));
    difference = left_height - right_height;
    if (abs(difference) <= 1 && check_balance(get_left(root)) &&
         check_balance(get_right(root))) {
        result = true;
    }
    else {
        result = false;
    }
    return result;
}

int main() {
    boolean result;
    node<int> root;
    node<int> c1;
    node<int> c2;
    node<int> c3;
    node<int> c4;
    node<int> null;

    root = $1;

    c1 = $2;
    add_left(root, c1);

    c2 = $3;
    add_right(root, c2);

    c3 = $4;
    add_left(c1, c3);

    c4 = $5;
    add_right(c1, c4);

    null = $0;
    add_left(c2, null);
    add_right(c2, null);
    add_left(c3, null);
    add_right(c3, null);
    add_left(c4, null);
    add_right(c4, null);

    result = check_balance(root);
    printb(result);
    return 0;
}


//  *** treebalance.ll ***  //
; ModuleID = 'nodable'
source_filename = "nodable"

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@str = private unnamed_addr constant [46 x i8] c"The result of the check
    balance function is: \00"
```

```
@fmt.3 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.4 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.5 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.6 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.7 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.8 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt.9 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.10 = private unnamed_addr constant [4 x i8] c"%g\0A\00"
@fmt.11 = private unnamed_addr constant [4 x i8] c"%s\0A\00"

declare i8* @init_node(i8*, ...)

declare i8* @get_data(i8*, ...)

declare i8* @get_left(i8*, ...)

declare i8* @get_right(i8*, ...)

declare i8* @add_left(i8*, ...)

declare i8* @add_right(i8*, ...)

declare i8* @list_init(...)

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

declare i8* @add_elem(i8*, i8*, ...)

declare i8* @append(i8*, i8*, ...)

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

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

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

define i32 @main() {
entry:
  %null = alloca i8*
  %c4 = alloca i8*
  %c3 = alloca i8*
  %c2 = alloca i8*
  %c1 = alloca i8*
  %root = alloca i8*
  %result = alloca i1
  %malloccall = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
      (i32, i32* null, i32 1) to i32))
  %data_malloc = bitcast i8* %malloccall to i32*
  store i32 1, i32* %data_malloc
  %data_bitcast = bitcast i32* %data_malloc to i8*
```

```
%init_node = call i8* (i8*, ...) @init_node(i8* %data_bitcast)
store i8* %init_node, i8** %root
%malloccall1 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc2 = bitcast i8* %malloccall1 to i32*
store i32 2, i32* %data_malloc2
%data_bitcast3 = bitcast i32* %data_malloc2 to i8*
%init_node4 = call i8* (i8*, ...) @init_node(i8* %data_bitcast3)
store i8* %init_node4, i8** %c1
%c15 = load i8*, i8** %c1
%root6 = load i8*, i8** %root
%add_left = call i8* (i8*, ...) @add_left(i8* %root6, i8* %c15)
%malloccall7 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc8 = bitcast i8* %malloccall7 to i32*
store i32 3, i32* %data_malloc8
%data_bitcast9 = bitcast i32* %data_malloc8 to i8*
%init_node10 = call i8* (i8*, ...) @init_node(i8* %data_bitcast9)
store i8* %init_node10, i8** %c2
%c211 = load i8*, i8** %c2
%root12 = load i8*, i8** %root
%add_right = call i8* (i8*, ...) @add_right(i8* %root12, i8* %c211)
%malloccall13 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc14 = bitcast i8* %malloccall13 to i32*
store i32 4, i32* %data_malloc14
%data_bitcast15 = bitcast i32* %data_malloc14 to i8*
%init_node16 = call i8* (i8*, ...) @init_node(i8* %data_bitcast15)
store i8* %init_node16, i8** %c3
%c317 = load i8*, i8** %c3
%c118 = load i8*, i8** %c1
%add_left19 = call i8* (i8*, ...) @add_left(i8* %c118, i8* %c317)
%malloccall20 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc21 = bitcast i8* %malloccall20 to i32*
store i32 5, i32* %data_malloc21
%data_bitcast22 = bitcast i32* %data_malloc21 to i8*
%init_node23 = call i8* (i8*, ...) @init_node(i8* %data_bitcast22)
store i8* %init_node23, i8** %c4
%c424 = load i8*, i8** %c4
%c125 = load i8*, i8** %c1
%add_right26 = call i8* (i8*, ...) @add_right(i8* %c125, i8* %c424)
%malloccall27 = tail call i8* @malloc(i32 ptrtoint (i32* getelementptr
    (i32, i32* null, i32 1) to i32))
%data_malloc28 = bitcast i8* %malloccall27 to i32*
store i32 0, i32* %data_malloc28
%data_bitcast29 = bitcast i32* %data_malloc28 to i8*
%init_node30 = call i8* (i8*, ...) @init_node(i8* %data_bitcast29)
store i8* %init_node30, i8** %null
%null31 = load i8*, i8** %null
```

```
  %c232 = load i8*, i8** %c2
  %add_left33 = call i8* (i8*, ...) @add_left(i8* %c232, i8* %null31)
  %null34 = load i8*, i8** %null
  %c235 = load i8*, i8** %c2
  %add_right36 = call i8* (i8*, ...) @add_right(i8* %c235, i8* %null34)
  %null37 = load i8*, i8** %null
  %c338 = load i8*, i8** %c3
  %add_left39 = call i8* (i8*, ...) @add_left(i8* %c338, i8* %null37)
  %null40 = load i8*, i8** %null
  %c341 = load i8*, i8** %c3
  %add_right42 = call i8* (i8*, ...) @add_right(i8* %c341, i8* %null40)
  %null43 = load i8*, i8** %null
  %c444 = load i8*, i8** %c4
  %add_left45 = call i8* (i8*, ...) @add_left(i8* %c444, i8* %null43)
  %null46 = load i8*, i8** %null
  %c447 = load i8*, i8** %c4
  %add_right48 = call i8* (i8*, ...) @add_right(i8* %c447, i8* %null46)
  %root49 = load i8*, i8** %root
  %check_balance_result = call i1 @check_balance(i8* %root49)
  store i1 %check_balance_result, i1* %result
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x
      i8], [4 x i8]* @fmt.2, i32 0, i32 0), i8* getelementptr inbounds
      ([46 x i8], [46 x i8]* @str, i32 0, i32 0))
  %result50 = load i1, i1* %result
  %printf51 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4
      x i8], [4 x i8]* @fmt, i32 0, i32 0), i1 %result50)
  ret i32 0
}

define i1 @check_balance(i8* %root) {
entry:
  %root1 = alloca i8*
  store i8* %root, i8** %root1
  %result = alloca i1
  %difference = alloca i32
  %right_height = alloca i32
  %left_height = alloca i32
  %root2 = load i8*, i8** %root1
  %data = call i8* (i8*, ...) @get_data(i8* %root2)
  %data3 = bitcast i8* %data to i32*
  %data4 = load i32, i32* %data3
  %tmp = icmp eq i32 %data4, 0
  br i1 %tmp, label %then, label %else

merge:                                          ; preds = %else
  %root6 = load i8*, i8** %root1
  %get_left = call i8* (i8*, ...) @get_left(i8* %root6)
  %tree_height_result = call i32 @tree_height(i8* %get_left)
  store i32 %tree_height_result, i32* %left_height
  %root7 = load i8*, i8** %root1
  %get_right = call i8* (i8*, ...) @get_right(i8* %root7)
  %tree_height_result8 = call i32 @tree_height(i8* %get_right)
  store i32 %tree_height_result8, i32* %right_height
  %left_height9 = load i32, i32* %left_height
  %right_height10 = load i32, i32* %right_height
  %tmp11 = sub i32 %left_height9, %right_height10
  store i32 %tmp11, i32* %difference
  %difference12 = load i32, i32* %difference
  %abs_result = call i32 @abs(i32 %difference12)
  %tmp13 = icmp sle i32 %abs_result, 1
  %root14 = load i8*, i8** %root1
  %get_left15 = call i8* (i8*, ...) @get_left(i8* %root14)
  %check_balance_result = call i1 @check_balance(i8* %get_left15)
  %tmp16 = and i1 %tmp13, %check_balance_result
  %root17 = load i8*, i8** %root1
  %get_right18 = call i8* (i8*, ...) @get_right(i8* %root17)
  %check_balance_result19 = call i1 @check_balance(i8* %get_right18)
  %tmp20 = and i1 %tmp16, %check_balance_result19
  br i1 %tmp20, label %then22, label %else23

then:                                             ; preds = %entry
  store i1 true, i1* %result
  %result5 = load i1, i1* %result
  ret i1 %result5

else:                                             ; preds = %entry
  br label %merge

merge21:                                          ; preds = %else23, %then22
  %result24 = load i1, i1* %result
  ret i1 %result24

then22:                                           ; preds = %merge
  store i1 true, i1* %result
  br label %merge21

else23:                                           ; preds = %merge
  store i1 false, i1* %result
  br label %merge21
}

define i32 @tree_height(i8* %root) {
entry:
  %root1 = alloca i8*
  store i8* %root, i8** %root1
  %max = alloca i32
  %right_height = alloca i32
  %left_height = alloca i32
  %root2 = load i8*, i8** %root1
  %data = call i8* (i8*, ...) @get_data(i8* %root2)
  %data3 = bitcast i8* %data to i32*
  %data4 = load i32, i32* %data3
  %tmp = icmp eq i32 %data4, 0
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %merge12
  %max19 = load i32, i32* %max
  ret i32 %max19

then:                                             ; preds = %entry
  store i32 0, i32* %max
  %max5 = load i32, i32* %max
  ret i32 %max5

else:                                             ; preds = %entry
  %root6 = load i8*, i8** %root1
  %get_left = call i8* (i8*, ...) @get_left(i8* %root6)
  %tree_height_result = call i32 @tree_height(i8* %get_left)
  store i32 %tree_height_result, i32* %left_height
  %root7 = load i8*, i8** %root1
  %get_right = call i8* (i8*, ...) @get_right(i8* %root7)
  %tree_height_result8 = call i32 @tree_height(i8* %get_right)
  store i32 %tree_height_result8, i32* %right_height
  %left_height9 = load i32, i32* %left_height
  %right_height10 = load i32, i32* %right_height
  %tmp11 = icmp slt i32 %left_height9, %right_height10
  br i1 %tmp11, label %then13, label %else16

merge12:                                          ; preds = %else16, %then13
  br label %merge

then13:                                           ; preds = %else
  %right_height14 = load i32, i32* %right_height
  %tmp15 = add i32 %right_height14, 1
  store i32 %tmp15, i32* %max
  br label %merge12

else16:                                           ; preds = %else
  %left_height17 = load i32, i32* %left_height
  %tmp18 = add i32 %left_height17, 1
  store i32 %tmp18, i32* %max
  br label %merge12
}

define i32 @abs(i32 %value) {
entry:
  %value1 = alloca i32
  store i32 %value, i32* %value1
  %ret = alloca i32
  %value2 = load i32, i32* %value1
  %tmp = icmp sle i32 %value2, 0
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else, %then
  %ret6 = load i32, i32* %ret
  ret i32 %ret6

then:                                             ; preds = %entry
  %value3 = load i32, i32* %value1
  %tmp4 = mul i32 %value3, -1
  store i32 %tmp4, i32* %ret
  br label %merge

else:                                             ; preds = %entry
  %value5 = load i32, i32* %value1
  store i32 %value5, i32* %ret
  br label %merge
}

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

6.2 Automation

All of our test files were stored in the /tests/ subdirectory. Test cases are given the format test-*.nd and fail cases have a format of fail*.nd. The expected output of each test program is stored in the respective *.out file.

The testing process was automated based on the testall.sh script from MicroC. This iterates through the /tests/ subdirectory, runs each file and compares the output to the corresponding *.out file. If the test passes, it prints 'OK', otherwise it prints the error. All the testing results are also stored in 'testall.log'. This script exists in the Makefile and is automatically called during 'make'.

6.2.1 ./testall.sh

```
#!/bin/sh

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

# Path to the LLVM interpreter
LLI="lli"
#LLI="/usr/local/opt/llvm/bin/lli"

# Path to the LLVM compiler
```

```
LLC="llc"

# Path to the C compiler
CC="cc"

# Path to the nodable compiler. Usually "./nodable.native"
# Try "_build/nodable.native" if ocamlbuild was unable to create a
    symbolic link.
NODABLE="./nodable.native"
#NODABLE="_build/nodable.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] [.nd 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/.nd//'`
    reffile=`echo $1 | sed 's/.nd$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

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

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.ll ${basename}.s
        ${basename}.exe ${basename}.out" &&
    Run "$NODABLE" "$1" ">" "${basename}.ll" &&
    Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">"
        "${basename}.s" &&
    Run "$CC" "-o" "${basename}.exe" "${basename}.s" "c_library.o" "-lm"
        &&
    Run "./${basename}.exe" > "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

    # Report the status and clean up the generated files

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

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

    echo -n "$basename..."

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

    generatedfiles=""

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

    # Report the status and clean up the generated files

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

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

```
shift `expr $OPTIND - 1`

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

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

if [ ! -f c_library.o ]
then
    echo "Could not find c_library.o"
    echo "Try \"make c_library.o\""
    exit 1
fi

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

for file in $files
do
    case $file in
   *test-*)
       Check $file 2>> $globallog
       ;;
   *fail-*)
       CheckFail $file 2>> $globallog
       ;;
   *)
       echo "unknown file type $file"
       globalerror=1
       ;;
    esac
done

exit $globalerror
```

6.3 Testing Strategy

All of our test cases are shown in Appendix 8.9 (Test Cases) and 8.10 (Fail Cases). We tested at a feature level first and tested each new feature as it was implemented to ensure it was robust (e.g many tests for operators, arithmetic, etc.). Since our test cases were primarily built in parallel with the features that we implemented, they were chosen to implement very specific aspects of each feature to ensure that they did not collapse in edge cases and that our edge cases have supporting error messages. As we developed more features, the test cases became more complex as they employed more features in tandem to ensure that they worked together.

In general, each member of the team worked on testing their own features as they implemented them. Most new features were implemented and tested on new branches before merging. After merging, new test cases were written to deal with cross-feature testing. While Naviya focused on managing the testing process and debugging test cases and Ariel worked on fail tests, most testing was done on an as-needed basis and implemented by everyone.

7 Lessons Learned

7.1 Naviya

This project taught me a lot about planning and coordinating with others. I think it took us a lot of time to understand each person's strengths and get our footing as a group, especially with a lot of the logistic changes that were happening. However, once we started doing that we were much more effective. I realized that with a project of this magnitude it isn't as important to get together and speak for a long time, but rather to make decisions as a group in a quick and efficient way.

I was also particularly surprised by the difference by in our project ambition and our achievements at each stage in the process. I hadn't realized how much goes on in the back end in order to help languages come to life. I learned a lot about the compiler architecture, especially when working on the C connecting library and understanding how memory management works. In the beginning our conversations would spin out into overly ambitious ideas, but by the end we had a much more concrete understanding of what was feasible and how it worked and could implement our features much more quickly. The process was extremely frustrating at times, but it was so great to see how it all came together in the end!

My biggest advice would be to ensure that you spend time in the beginning creating a robust environment for running your files and tests. It really simplifies the process later. Also working towards smaller goals and coding incrementally was really helpful for us! But most of all, it's important for everyone on the team to work together and is super efficient if you play off each other and one person starts to implement, another debugs and then you go back and forth!

7.2 Karen

I learned a lot about functional programming, compiler architecture, and the importance of robust testing and incremental coding. Keeping the architectural design component diagram in mind was instrumental in helping me see how the pieces fit together and generally see the larger picture, especially as we started working on the more complicated SAST, code generation, and linking of C libraries. Rather than solely focusing on specific components of the compiler, we attempted to implement features from end to end. From doing this, I was able to understand how certain features were implemented step by step from start to finish and definitely underestimated how much happens behind the scenes when we code.

Working on such a large-scale team project, I also learned a lot about collaboration and communication. It was important for us to be realistic about what we could accomplish, which turned out to be much less than what we imagined in our proposal, and set short-term, attainable tasks to move towards the goal. I would advise future teams to set these shorter-term goals towards and make sure everyone is aligned. Having teammates to bounce ideas off of and to tag-team on the implementation, debugging, and testing is invaluable!

7.3 Ajita

Although the whole process was very stressful, it was incredibly rewarding - I learned about working in a group on a large-scale, long-term project, about compiler architecture, functional programming, memory management, language design, and so much more. I think the coolest thing about this project was understanding and learning the ”magic” behind many of the tools I've worked with over the past few years and never thought much about.

7.4 Ariel

I learned a lot about how to work on a large coding project as a group, as most of my computer science courses up until now have been individual assignments. I learned how to organize a git repo so that people are able to figure out what is going on in each branch and how to check that code changes don't break other parts of your code before you merge. I learned how important it is to stay on the same page with your teammates so that the code you are all developing works well together. I also learned that one great benefit of working on a coding project as a team is that you have people to help you debug!

8 Appendix

8.1 Scanner - scanner.mll

```
(*Authors:
Ajita Bala ab4420
Karen Shi ks3650*)

(* Ocamllex Scanner for nodable *)

{ open Nodableparser }

let digit = ['0' - '9']
let letter = ['a' - 'z' 'A' - 'Z']
let integers = digit+
let dec = (integers '.' digit*)
let name = letter ['a' - 'z' 'A' - 'Z' '0'-'9' '_']*

rule token =
parse
[' ' '\n' '\r' '\t'] {token lexbuf}
| '+' { PLUS }
| '-' { MINUS }
| '*' { TIMES }
| '/' { DIVIDE }
| '%' { MODULO }
| '=' { ASSIGN }
| ',' { COMMA }
| ';' { SEMI }
| '(' { LPAREN }
| ')' { RPAREN }
| '{' { LBRACE }
| '}' { RBRACE }
| '[' { LSQUARE }
| ']' { RSQUARE }
| '.' { DOT }
| '$' { DOLLAR }


| "==" { EQ }
| "!=" { NEQ }
| '!' { NOT }
| '<' { LT }
| '>' { GT }
| ">=" { GEQ }
| "<=" { LEQ }
| "true" { BOOL_LIT(true) }
| "false" { BOOL_LIT(false) }

| "if" { IF }
| "else" { ELSE }
| "&&" { AND }
| "||" { OR }
| "for" { FOR }
| "while" { WHILE }
| "return" { RETURN }


| "void" { VOID }
| "int" { INT }
| "float" { FLOAT }
| "boolean" { BOOL }
| "string" { STRING }
| "node" { NODE }
| "list" { LIST }
| "true"  { BOOL_LIT(true) }
| "false" { BOOL_LIT(false) }

| name as lxm { ID(lxm) }
| integers as lxm { INT_LIT(int_of_string lxm) }
| digit+ '.' digit* as lxm { FLT_LIT(lxm) }
| '\"' ([^'\"']* as lxm) '\"' {STR_LIT(lxm)}

| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

| "//" {comment lexbuf}

and comment = parse
"//" {token lexbuf}
| _ {comment lexbuf} (* Ignore other characters *)
```

8.2 Parser - nodableparser.mly

```
/* Ocamlyacc parser for nodable */
(*Authors:
Ajita Bala
Naviya Makhija
Ariel Goldman
Karen Shi
*)

%{ open Ast

let scope = ref 1

let parse_error err =
   print_endline err;
   flush stdout

%}
```

```
%token SEMI COMMA
%token LPAREN RPAREN LBRACE RBRACE LSQUARE RSQUARE ASSIGN DOT DOLLAR

%token PLUS MINUS TIMES DIVIDE MODULO
%token NOT OR AND LT GT LEQ GEQ EQ NEQ
%token NEW

%token INT FLOAT BOOL STRING NODE LIST VOID
%token <int> INT_LIT
%token <string> FLT_LIT STR_LIT ID
%token <bool> BOOL_LIT

%token RETURN FOR IF ELIF ELSE WHILE
%token EOF

%start program
%type <Ast.program> program

%nonassoc NOELSE
%nonassoc ELSE
%nonassoc LPAREN LSQUARE LBRACE
%nonassoc RPAREN RSQUARE RBRACE

%left COMMA
%right ASSIGN
%left OR
%left AND
%left DOT
%left EQ NEQ
%left LEQ GEQ LT GT
%left PLUS MINUS
%left TIMES DIVIDE MODULO
%right NOT NEG

%%

program:
  decls EOF { $1 }

decls:
   /* nothing */      { ([], [])             }
   | decls vdecl      { (($2 :: fst $1), snd $1) }
   | decls fdecl      { (fst $1, ($2 :: snd $1)) }
   | decls glb_vdecl  { (($2 :: fst $1), snd $1) }


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

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

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

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

literal:
   ID                          { Id($1) }
   | INT_LIT                      { Lit_Int($1) }
   | FLT_LIT                      { Lit_Flt($1) }
   | STR_LIT                      { Lit_Str($1) }
   | BOOL_LIT                     { Lit_Bool($1) }
   | LPAREN expr RPAREN              { $2 }
   | ID LPAREN args_opt RPAREN       { Call($1, $3) }
   | literal LSQUARE expr RSQUARE     { List_Access($1, $3) }
   | literal DOT ID               { Attr($1, $3) }

expr:
   literal                           { $1 }
   | node_expr                         { $1 }
   | expr PLUS expr               { Binop ($1, Add, $3) }
   | expr MINUS expr              { Binop ($1, Sub, $3) }
   | expr DIVIDE expr             { Binop ($1, Div, $3) }
   | expr TIMES expr              { Binop ($1, Mult, $3) }
   | expr MODULO expr             { Binop ($1, Mod, $3) }
   | expr LT expr                 { Binop ($1, Less, $3) }
   | expr GT expr                 { Binop ($1, Greater, $3) }
   | expr LEQ expr                   { Binop ($1, Leq, $3) }
   | expr GEQ expr                   { Binop ($1, Geq, $3) }
   | expr EQ expr                 { Binop ($1, Eq, $3) }
   | expr NEQ expr                   { Binop ($1, Neq, $3) }
   | expr AND expr                   { Binop ($1, And, $3) }
   | expr OR expr                 { Binop ($1, Or, $3) }
   | MINUS expr %prec NOT      { Unop(Neg, $2)     }
   | NOT expr                     { Unop(Not, $2)     }
   | ID ASSIGN expr               { Assign($1, $3)    }
   | LSQUARE args_opt RSQUARE { Lit_List($2) }
```

```
node_expr:
   DOLLAR literal     { Lit_Node($2) }

args_opt:
  /* nothing */ { [] }
  | args_list { List.rev $1 }

args_list:
    expr                { [$1] }
  | args_list COMMA expr  { $3 :: $1 }

typ:
     INT                 { Int }
   | FLOAT               { Float }
   | BOOL                { Bool }
   | STRING           { String }
   | VOID                { Void }
   | NODE LT typ GT { Node($3) }
   | LIST LT typ GT  { List($3) }

vdecl:
   typ ID SEMI     { ($1, $2) }

glb_vdecl:
   vdecl SEMI{$1}

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

stmt:
   expr SEMI
             { Expr($1) }
   | typ ID SEMI
          { Declare($1, $2, Noexpr) }
   | typ ID ASSIGN expr SEMI
             { Declare($1, $2, Assign($2, $4)) }
   | RETURN expr_opt 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) }
   | WHILE LPAREN expr RPAREN stmt                                    {
       While($3, $5) }
   | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt  { For($3,
       $5, $7, $9) }
```

8.3 AST - ast.ml

```
(* Abstract Syntax Tree and functions for printing it *)
(*Authors:
Ajita Bala
Naviya Makhija
Ariel Goldman
Karen Shi
*)

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

type uop = Not | Neg

type typ = Int | Bool | Float | String | Node of typ | List of typ |
    Void | Any

type bind = typ * string

type expr =
    Id of string
  | Lit_Int of int
  | Lit_Flt of string
  | Lit_Str of string
  | Lit_Bool of bool
  | Lit_List of expr list
  | Lit_Node of expr
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of string * expr
  | Call of string * expr list
  | List_Access of expr * expr
  | Attr of expr * string
  | Noexpr


type stmt =
    Block of stmt list
  | Expr of expr
  | Return of expr
  | If of expr * stmt * stmt
  | For of expr * expr * expr * stmt
  | While of expr * stmt
  | Declare of typ * string * expr
```

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

type program = bind list * func_decl list

(*Pretty Print for Debugging purposes - taken from microc ast.ml and
    refactored*)

let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Eq -> "=="
  | Neq -> "!="
  | Mod -> "%"
  | Less -> "<"
  | Leq -> "<="
  | Greater -> ">"
  | Geq -> ">="
  | And -> "&&"
  | Or -> "||"

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

let rec string_of_expr = function
    Lit_Int(l) -> string_of_int l
  | Lit_Flt(l) -> l
  | Lit_Str(s) -> s
  | Lit_Bool(true) -> "true"
  | Lit_Bool(false) -> "false"
  | Lit_List(l) -> "[" ^ String.concat "," (List.map string_of_expr l) ^
     "]"
  | Lit_Node(e) -> "'" ^ string_of_expr e ^ "'"
  | Attr(e, a) -> string_of_expr e ^ "." ^ a
  | List_Access(l, e) -> string_of_expr l ^ "[" ^ string_of_expr e ^ "]"
  (*| Lit_Char(c) -> String.make 1 c*)
  | Id(s) -> s
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Assign(v, e) -> v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Noexpr -> ""

  let rec string_of_stmt = function
  Block(stmts) ->
    "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
| Expr(expr) -> string_of_expr expr ^ ";\n";
| Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
| If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^
    string_of_stmt s
| If(e, s1, s2) -> "if (" ^ string_of_expr e ^ ")\n" ^
    string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
| For(e1, e2, e3, s) ->
    "for (" ^ string_of_expr e1 ^ " ; " ^ string_of_expr e2 ^ " ; " ^
    string_of_expr e3 ^ ") " ^ string_of_stmt s
| While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s
(*| Declare(t, id, a) -> string_of_typ t ^ " " ^ (match a with Noexpr ->
    id | _ -> string_of_expr a) ^ ";\n"*)

let rec string_of_typ = function
    Any -> "*"
  |  Int -> "int"
  | Bool -> "bool"
  | Float -> "float"
  | String -> "string"
  | Void -> "void"
  | Graph -> "Graph"
  | Node(t) -> "node <" ^ string_of_typ t ^ ">"
  | List(l) -> "list" ^ "<" ^ string_of_typ l ^ ">"


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{\n" ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

let string_of_program (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs)
```

8.4 Semantic Checker - semant.ml

```
(*Authors:
Ajita Bala
Naviya Makhija
```

```
Ariel Goldman
Karen Shi
*)
(* Semantic checking for the nodable compiler *)

(* Tasks:
Takes an abstract syntax tree (Ast) and returns a semantically checked
    AST (Sast) with type checking and syntax checking.
Goal - bind each token to a semantically checked expression and raise a
    failure if incorrect type
*)

open Ast
open Sast

module StringMap = Map.Make(String)

let check (globals, functions) =

  (* Verify a list of bindings has no void types or duplicate names *)
  let check_binds (kind : string) (binds : bind list) =
    List.iter (function
   (Void, b) -> raise (Failure ("illegal void " ^ kind ^ " " ^ b))
      | _ -> ()) binds;
    let rec dups = function
        [] -> ()
      |  ((_,n1) :: (_,n2) :: _) when n1 = n2 ->
     raise (Failure ("duplicate " ^ kind ^ " " ^ n1))
      | _ :: t -> dups t
    in dups (List.sort (fun (_,a) (_,b) -> compare a b) binds)
  in

  check_binds "global" globals;

  (* Collect function declarations for built-in functions: no bodies *)
  let built_in_decls =
    let add_bind map (name, params, ret) = StringMap.add name {
      typ = ret;
      fname = name;
      formals = params;
      body = [] } map
  in List.fold_left add_bind StringMap.empty [("print", [(Int, "x")],
      Void);
                                              ("prints", [(String, "x")],
                                                  Void);
                                              ("printf", [(Float, "x")],
                                                  Void);
                                              ("printb", [(Bool, "x")],
                                                  Void);
```

```
                                           ("size", [(List(Node Int),
                                               "x")], Int);
                                           ("append", [(List(Node Int),
                                               "x"); (Node Int, "y")],
                                               Void);
                                           ("update_elem", [(Node Int,
                                               "x"); (List(Node Int),
                                               "y"); (Int, "z")],
                                               List(Node Int));
                                           ("add_left", [(Node Int, "x");
                                               (Node Int, "y")], Void);
                                           ("add_right", [(Node Int,
                                               "x"); (Node Int, "y")],
                                               Void);
                                           ("get_left", [(Node Int,
                                               "x")], Node Int);
                                           ("get_right", [(Node Int,
                                               "x")], Node Int)
                                           ]
in

  (* Add function name to symbol table *)
  let add_func map fd =
    let built_in_err = "function " ^ fd.fname ^ " may not be defined"
    and dup_err = "duplicate function " ^ fd.fname
    and make_err er = raise (Failure er)
    and n = fd.fname (* Name of the function *)
    in match fd with (* No duplicate functions or redefinitions of
         built-ins *)
         _ when StringMap.mem n built_in_decls -> make_err built_in_err
       | _ when StringMap.mem n map -> make_err dup_err
       | _ -> StringMap.add n fd map
  in

let function_decls = List.fold_left add_func built_in_decls functions
in

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

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

let check_function func =
  (* Make sure no formals or locals are void or duplicates *)
  check_binds "formal" func.formals;
  (*check_binds "local" func.locals; *)

    (* Raise an exception if the given rvalue type cannot be assigned to
      the given lvalue type *)
    let rec check_assign lvaluet rvaluet err =
      match (lvaluet, rvaluet) with
        (List a, List Any) -> List a
        | (Node _, Node a) -> Node a
       | (l, r) -> if l = r then l else raise (Failure err)
    in

    let rec declarations locals = function
        [] -> locals
      | hd::tl -> declarations (match hd with
          Declare (t, id, a) -> (t, id) :: locals
        | _ -> locals) tl
    in

    (* Build local symbol table of variables for this function *)
    let symbols = List.fold_left (fun m (ty, name) -> StringMap.add name
        ty m)
      StringMap.empty (globals @ func.formals @ (declarations []
          func.body))
    in

  let type_of_identifier s =
    try StringMap.find s symbols
    with Not_found -> raise (Failure ("undeclared identifier " ^ s))
  in

  let expr_list lst expr =
    let rec helper typ tlist = function
        [] -> (typ, tlist)
      | hd :: _ when (match (typ, fst (expr hd)) with
          (Node a, Node b) -> a <> b
          | (a, b) -> a <> b) ->
        raise (Failure ("Type inconsistency with list "))
      | hd :: tl -> helper typ (expr hd :: tlist) tl
    in
  let typ = match (List.length lst) with
      0 -> Any
    | _ -> (fst (expr (List.hd lst))) in
  helper typ [] lst
in

  let rec expr = function
    Lit_Int l -> (Int, SLiteral l)
    | Lit_Flt l -> (Float, SFliteral l)
    | Lit_Bool l -> (Bool, SBoolLit l)
    | Lit_Str s -> (String, SStrLit s)
    | Lit_List l -> let (t, l) = expr_list l expr in (List t, SListLit l)
    | Lit_Node n -> let (t, d) = expr n in (Node t, SNodeLit (t, d))
    | List_Access (l, e) ->
      let (tl, _) as l' = expr l in
      let (te, _) as e' = expr e in
    if te != Int then raise (Failure ("list index must be an integer"))
    else (match tl with
        List x -> (x, SListAccess (l', e'))
      | _ -> raise (Failure ("not iterable")))
    | Noexpr    -> (Void, SNoexpr)
    | Id s -> (type_of_identifier s, SId s)
    | Attr (e, p) ->
      let (et, _) as e' = expr e in
      let pt = (match (et, p) with
      (Node t, "data") -> t
      | (Node t, "left") -> t
      | (Node t, "right") -> t
      | (_, _) -> raise (Failure ("no such property"))) in
      (pt, SAttr (e', p))
    | Assign(var, e) as ex ->
    let lt = type_of_identifier var
    and (rt, e') = expr e in
    let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
      string_of_typ rt ^ " in " ^ string_of_expr ex
    in (check_assign lt rt err, SAssign(var, (rt, e')))
    | Unop(op, e) as ex ->
    let (t, e') = expr e in
    let ty = match op with
      Neg when t = Int || t = Float -> t
    | Not when t = Bool -> Bool
    | _ -> raise (Failure ("illegal unary operator " ^
                         string_of_uop op ^ string_of_typ t ^
                         " in " ^ string_of_expr ex))
    in (ty, SUnop(op, (t, e')))
      | Binop(e1, op, e2) as e ->
          let (t1, e1') = expr e1
          and (t2, e2') = expr e2 in
          (* All binary operators require operands of the same type *)
          let same = t1 = t2 in
          (* Determine expression type based on operator and operand
              types *)
          let ty = match op with
           Add | Sub | Mult | Div | Mod when same && t1 = Int -> Int
          | Add | Sub | Mult | Div | Mod when same && t1 = Float -> Float
          | Eq | Neq          when same             -> Bool
          | Less | Leq | Greater | Geq
                    when same && (t1 = Int || t1 = Float) -> Bool
          | And | Or when same && t1 = Bool -> Bool
          | _ -> raise (
         Failure ("illegal binary operator " ^
                     string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
                     string_of_typ t2 ^ " in " ^ string_of_expr e))
          in (ty, SBinop((t1, e1'), op, (t2, e2')))
      | Call(fname, args) as call ->
      let fd = find_func fname in
      let param_length = List.length fd.formals in
      if List.length args != param_length then
        raise (Failure ("expecting " ^ string_of_int param_length ^
                      " arguments in " ^ string_of_expr call))
      else let check_call (ft, _) e =
        let (et, e') = expr e in
        let err = "illegal argument found " ^ string_of_typ et ^
          " expected " ^ string_of_typ ft ^ " in " ^ string_of_expr e
        in (check_assign ft et err, e')
      in
      let args' = List.map2 check_call fd.formals args
      in (fd.typ, SCall(fname, args'))
    in

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

    (* Return a semantically-checked statement i.e. containing sexprs *)
    let rec check_stmt = function
    Expr e -> SExpr (expr e)
    | If(p, b1, b2) -> SIf(check_bool_expr p, check_stmt b1, check_stmt
        b2)
    | For(e1, e2, e3, st) ->
      SFor(expr e1, check_bool_expr e2, expr e3, check_stmt st)
    | While(p, s) -> SWhile(check_bool_expr p, check_stmt s)
    | Declare (typ, id, asn) ->
      let a = expr asn in SDeclare(typ, id, a)
    | Return e -> let (t, e') = expr e in
      if t = func.typ then SReturn (t, e')
        else raise
        (Failure ("return gives " ^ string_of_typ t ^ " expected " ^
            string_of_typ func.typ ^ " in " ^ string_of_expr e))

    (* A block is correct if each statement is correct and nothing
          follows any Return statement. Nested blocks are flattened. *)
      | Block sl ->
        let rec check_stmt_list = function
        [Return _ as s] -> [check_stmt s]
        | Return _ :: _ -> raise (Failure "nothing may follow return")
        | Block sl :: ss -> check_stmt_list (sl @ ss)
        | s :: ss -> check_stmt s :: check_stmt_list ss
        | [] -> []
      in SBlock(check_stmt_list sl)

      in
```

```
{ styp = func.typ;
sfname = func.fname;
sformals = func.formals;
sbody = match check_stmt (Block func.body) with
SBlock(sl) -> sl
| _ -> raise(Failure("internal error: block didn't become a
    block?"))
}
in (globals, List.map check_function functions)
```

8.5 Semantically-checked Abstract Syntax Tree - sast.ml

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

(*Authors:
Ajita Bala
Naviya Makhija
Ariel Goldman
Karen Shi
*)

open Ast

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

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

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

type sprogram = bind list * sfunc_decl list

(* Pretty-printing functions *)

let rec string_of_sexpr (t, e) =
  "(" ^ string_of_typ t ^ " : " ^ (match e with
    SLiteral(l) -> string_of_int l
  | SFliteral(l) -> l
  | SBoolLit(true) -> "true"
  | SBoolLit(false) -> "false"
  | SStrLit(l) -> l
  | SListLit(e) -> "[" ^ String.concat "," (List.map string_of_sexpr e)
      ^ "]"
  | SNodeLit(e) -> "'" ^ string_of_sexpr e ^ "'"
  | SId(s) -> s
  | SBinop(e1, o, e2) ->
  string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
  | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
  | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e
  | SCall(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
  | SListAccess(l, e) -> string_of_sexpr l ^ "[" ^ string_of_sexpr e ^
      "]"
  | SAttr(e, a) -> string_of_sexpr e ^ "." ^ a
  | SNoexpr -> ""
    ) ^ ")"

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


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

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

8.6 Code generation- codegen.ml

```
(*Authors:
Ajita Bala
Naviya Makhija
Ariel Goldman
Karen Shi
*)
(* Code generation: translate takes a semantically checked AST and
produces LLVM IR *)
module L = Llvm
module A = Ast
open Sast

module StringMap = Map.Make(String)

let translate (globals, functions) =
  let context = L.global_context() in
  let the_module = L.create_module context "nodable" in

  let i32_t = L.i32_type context
  and i8_t = L.i8_type context
  and i1_t      = L.i1_type    context (* bool *)
  and float_t   = L.double_type context
  and str_t = L.pointer_type (L.i8_type context)
  and obj_ptr_t = L.pointer_type (L.i8_type context)
  and void_ptr_t = L.pointer_type (L.i8_type context)
  and void_t = L.void_type context in

  (* Return the LLVM type for a nodable type *)
  let ltype_of_typ = function
    A.Int -> i32_t
  (*| A.Bool -> i1_t*)
  | A.Void -> void_t
  | A.Bool -> i1_t
  | A.Float -> float_t
  | A.String -> str_t
  | A.List(_) -> obj_ptr_t
  | A.Node(_) -> obj_ptr_t
  | _ -> raise (Failure "not implemented")
  in

  (* Create a map of global variables after creating each *)
  let global_vars : L.llvalue StringMap.t =
    let global_var m (t, n) =
      let init = match t with
          A.Float -> L.const_float (ltype_of_typ t) 0.0
        | _ -> L.const_int (ltype_of_typ t) 0
      in StringMap.add n (L.define_global n init the_module) m in
    List.fold_left global_var StringMap.empty globals in

    (* string functions*)

     (* NODE FUNCTIONS *)

    let init_t : L.lltype =
      L.var_arg_function_type obj_ptr_t [| void_ptr_t |] in

  let init_node : L.llvalue =
      L.declare_function "init_node" init_t the_module in

  let get_t : L.lltype =
    L.var_arg_function_type void_ptr_t [|obj_ptr_t|] in
  let get_data : L.llvalue =
      L.declare_function "get_data" get_t the_module in


  let get_left_t : L.lltype =
    L.var_arg_function_type void_ptr_t [|obj_ptr_t|] in
  let get_left : L.llvalue =
      L.declare_function "get_left" get_left_t the_module in

  let get_right_t : L.lltype =
      L.var_arg_function_type obj_ptr_t [|obj_ptr_t|] in
  let get_right : L.llvalue =
      L.declare_function "get_right" get_right_t the_module in


let add_left_t : L.lltype =
  L.var_arg_function_type obj_ptr_t [|obj_ptr_t|] in
let add_left : L.llvalue =
    L.declare_function "add_left" add_left_t the_module in
```

```
let add_right_t : L.lltype =
    L.var_arg_function_type obj_ptr_t [|obj_ptr_t|] in
let add_right : L.llvalue =
    L.declare_function "add_right" add_right_t the_module in

    (* list functions *)

  let list_init_t : L.lltype =
      L.var_arg_function_type obj_ptr_t [||] in
  let list_init : L.llvalue =
      L.declare_function "list_init" list_init_t the_module in

  let get_elem_t : L.lltype =
    L.var_arg_function_type void_ptr_t [|i32_t ; obj_ptr_t|] in
  let get_elem : L.llvalue =
    L.declare_function "get_elem" get_elem_t the_module in

  let add_elem_t : L.lltype =
      L.var_arg_function_type void_ptr_t [|obj_ptr_t; void_ptr_t |] in
  let add_elem : L.llvalue =
      L.declare_function "add_elem" add_elem_t the_module in

  let append : L.lltype =
    L.var_arg_function_type void_ptr_t [|obj_ptr_t; void_ptr_t |] in
  let append : L.llvalue =
      L.declare_function "append" append the_module in

  let update_elem_t : L.lltype =
    L.var_arg_function_type obj_ptr_t [|void_ptr_t ; obj_ptr_t ; i32_t|]
        in
  let update_elem : L.llvalue =
      L.declare_function "update_elem" update_elem_t the_module in

  let size_t : L.lltype =
    L.var_arg_function_type i32_t [|obj_ptr_t|] in
  let size : L.llvalue =
      L.declare_function "size" size_t the_module in

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

  let function_decls: (L.llvalue * sfunc_decl) StringMap.t =
    let function_decl m fdecl =
      let name = fdecl.sfname
      and formal_types =
    Array.of_list (List.map(fun(t, _) -> ltype_of_typ t) fdecl.sformals)
      in let ftype =
        L.function_type(ltype_of_typ fdecl.styp) formal_types in
      StringMap.add name (L.define_function name ftype the_module,
          fdecl) m in
    List.fold_left function_decl StringMap.empty functions in

    let build_function_body fdecl =
      let (the_function, _) =
        StringMap.find fdecl.sfname function_decls in
      let builder =
        L.builder_at_end context (L.entry_block the_function) in
      let int_format_str =
        L.build_global_stringptr "%d\n" "fmt" builder
      and float_format_str = L.build_global_stringptr "%g\n" "fmt"
          builder
      and str_format_str = L.build_global_stringptr "%s\n" "fmt" builder
          in

   (* Construct the function's "locals": formal arguments and locally
      declared variables. Allocate each on the stack, initialize their
      value, if appropriate, and remember their values in the "locals"
           map *)
      let local_vars =
        let add_formal m (t, n) p =
          L.set_value_name n p;
    let local = L.build_alloca (ltype_of_typ t) n builder in
          ignore (L.build_store p local builder);
    StringMap.add n local m

      (* Allocate space for any locally declared variables and add the
        * resulting registers to our map *)
       and add_local m (t, n) =
       let local_var = L.build_alloca (ltype_of_typ t) n builder
       in StringMap.add n local_var m
           in

           let formals = List.fold_left2 add_formal StringMap.empty
               fdecl.sformals
              (Array.to_list (L.params the_function)) in
           let rec declare locals = function
              [] -> locals
            | hd::tl -> declare (match hd with
                SDeclare (t, id, _) -> (t, id) :: locals
               | _ -> locals) tl
           in

          let declarations = declare [] fdecl.sbody in
           List.fold_left add_local formals declarations
          in

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

   (* Construct code for an expression; return its value *)
   let rec expr builder ((_, e):sexpr) = match e with
       SLiteral i -> L.const_int i32_t i
     | SBoolLit b -> L.const_int i1_t (if b then 1 else 0)
     | SFliteral l -> L.const_float_of_string float_t l
     | SStrLit l -> L.build_global_stringptr l "str" builder
     | SNoexpr -> L.const_int i32_t 0
     | SId s      -> L.build_load (lookup s) s builder
     | SAssign (s, e) -> let e' = expr builder e in
       ignore(L.build_store e' (lookup s) builder); e'
     | SListAccess (l, e) ->
       let l' = expr builder l in
       let e' = expr builder e in
       (match fst l with
          A.List t ->
           let d_ptr = L.build_call get_elem [|e'; l'|] "get_elem"
                builder in
           match t with
               A.List _ | A.Node _ -> d_ptr
             | _ ->
               let dt_ptr = L.pointer_type (ltype_of_typ t) in
               let d_ptr = L.build_bitcast d_ptr dt_ptr "data" builder
                    in
               L.build_load d_ptr "data" builder)

     | SAttr (o, p) ->
       let tobj = fst o in
       let o' = expr builder o in
       (match (tobj, p) with
       | (A.Node t, "data") ->
           let dt_ptr = L.pointer_type (ltype_of_typ t) in
           let d_ptr = L.build_call get_data [|o'|] "data" builder in
           let d_ptr = L.build_bitcast d_ptr dt_ptr "data" builder in
           L.build_load d_ptr "data" builder
       | (A.Node t, "left") ->
         L.build_call get_left [|o'|] "left" builder
       | (A.Node t, "right") ->
           L.build_call get_right [|o'|] "right" builder
       | (_, _) -> raise (Failure "no such property"))

       | SNodeLit (t, v) ->
       let data_value = expr builder (t, v) in
       let data = L.build_malloc (ltype_of_typ t) "data_malloc"
            builder in
         ignore ( L.build_store data_value data builder);
```

```
    let data = L.build_bitcast data void_ptr_t "data_bitcast"
        builder in
    let node = L.build_call init_node [|data|] "init_node" builder
        in node

  | SListLit li ->
    let rec create_list mylist = (function
    [] -> mylist
    |h :: tail ->
    let (anyt,_) = h in
    let d_ptr = (match anyt with
      A.List _ | A.Node _ -> expr builder h
      | _  -> let elem = L.build_malloc (ltype_of_typ anyt) "data"
          builder in
        let elem_value = expr builder h in
        ignore (L.build_store elem_value elem builder); elem) in
    let elem = L.build_bitcast d_ptr void_ptr_t "data" builder in
    L.build_call add_elem [|mylist; elem|] "" builder;
     create_list mylist tail) in
    let mylist = L.build_call list_init [||] "list_init" builder in
        create_list mylist li
  | SBinop ((A.Float,_ ) as e1, op, e2) ->
    let e1' = expr builder e1
    and e2' = expr builder e2 in
    (match op with
        A.Add    -> L.build_fadd
      | A.Sub    -> L.build_fsub
      | A.Mult   -> L.build_fmul
      | A.Div    -> L.build_fdiv
      | A.Mod    -> L.build_frem
      | A.Eq     -> L.build_fcmp L.Fcmp.Oeq
      | A.Neq    -> L.build_fcmp L.Fcmp.One
      | A.Less   -> L.build_fcmp L.Fcmp.Olt
      | A.Leq    -> L.build_fcmp L.Fcmp.Ole
      | A.Greater -> L.build_fcmp L.Fcmp.Ogt
      | A.Geq    -> L.build_fcmp L.Fcmp.Oge
      | A.And | A.Or ->
        raise (Failure "internal error: semant should have rejected
            and/or on float")
    ) e1' e2' "tmp" builder
| SBinop (e1, op, e2) ->
let e1' = expr builder e1
and e2' = expr builder e2 in
(match op with
  A.Add    -> L.build_add
| A.Sub    -> L.build_sub
| A.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.Eq     -> 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
  ) e1' e2' "tmp" builder
   | SUnop(op, ((t, _) as e)) ->
      let e' = expr builder e in
  (match op with
    A.Neg when t = A.Float -> L.build_fneg
  | A.Neg                  -> L.build_neg
       | A.Not                 -> L.build_not) e' "tmp" builder
     | SCall ("print", [e])
     | SCall ("printb", [e]) -> L.build_call printf_func [|
        int_format_str ; (expr builder e) |]"printf" builder
     | SCall ("printf", [e]) -> L.build_call printf_func [|
        float_format_str ; (expr builder e) |] "printf" builder
   (*| SCall ("printf", [e]) -> L.build_call printf_func [|
       int_format_str ; (expr builder e) |]"printf" builder *)
     | SCall ("prints", [e]) -> L.build_call printf_func [|
        str_format_str ; (expr builder e) |]"printf" builder
     | SCall ("size", [e]) -> L.build_call size [|expr builder e|]
        "size" builder
     | SCall ("append", [e; f]) -> L.build_call append [|expr
        builder e; expr builder f|] "append" builder
     | SCall ("update_elem", [e;f;g]) -> L.build_call update_elem
        [|expr builder e; expr builder f; expr builder g|]
        "update_elem" builder
     | SCall ("add_left", [e; f]) -> L.build_call add_left [|expr
        builder e; expr builder f|] "add_left" builder
     | SCall ("add_right", [e; f]) -> L.build_call add_right [|expr
        builder e; expr builder f|] "add_right" builder
     | SCall ("get_left", [e]) -> L.build_call get_left [|expr
        builder e|] "get_left" builder
     | SCall ("get_right", [e]) -> L.build_call get_right [|expr
        builder e|] "get_right" builder
     | SCall (f, args) ->
     let (fdef, fdecl) = StringMap.find f function_decls in
let llargs = List.rev (List.map (expr builder) (List.rev args)) in
let result = (match fdecl.styp with
                   A.Void -> ""
                 | _ -> f ^ "_result") in
     L.build_call fdef (Array.of_list llargs) result builder
 in

 (* LLVM insists each basic block end with exactly one "terminator"
   instruction that transfers control. This function runs "instr
       builder"
    if the current block does not already have a terminator. Used,
    e.g., to handle the "fall off the end of the function" case. *)

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


let rec stmt builder = function
SBlock sl -> List.fold_left stmt builder sl
| SExpr e -> ignore(expr builder e); builder
| SDeclare (_, _, a) -> ignore(expr builder a); builder
| SReturn e -> ignore(match fdecl.styp with
                         (* Special "return nothing" instr *)
                         A.Void -> L.build_ret_void builder
                         (* Build return statement *)
                        | _ -> L.build_ret (expr builder e) builder );
                  builder
| SIf (predicate, then_stmt, else_stmt) ->
     let bool_val = expr builder predicate in
let merge_bb = L.append_block context "merge" the_function in
     let build_br_merge = L.build_br merge_bb in (* partial function
         *)

let then_bb = L.append_block context "then" the_function in
add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
  build_br_merge;

let else_bb = L.append_block context "else" the_function in
add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
  build_br_merge;

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

  | SWhile (predicate, body) ->
 let pred_bb = L.append_block context "while" the_function in
 ignore(L.build_br pred_bb builder);

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

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

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

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


    let builder = stmt builder (SBlock fdecl.sbody)

      in add_terminal builder (match fdecl.styp with
         A.Void -> L.build_ret_void
         | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))


in List.iter build_function_body functions;
the_module
(* EOF *)
```

8.7 nodable.ml

```
(* Top-level of the nodable compiler: scan & parse the input,
   check the resulting AST and generate an SAST from it, generate LLVM
        IR,
   and dump the module *)

   type action = Ast | Sast | LLVM_IR | Compile

   let () =
     let action = ref Compile in
     let set_action a () = action := a in
     let speclist = [
       ("-a", Arg.Unit (set_action Ast), "Print the AST");
       ("-s", Arg.Unit (set_action Sast), "Print the SAST");
       ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM
           IR");
       ("-c", Arg.Unit (set_action Compile),
         "Check and print the generated LLVM IR (default)");
     ] in
     let usage_msg = "usage: ./nodable.native [-a|-s|-l|-c] [file.nd]" 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 = Nodableparser.program Scanner.token lexbuf in
     match !action with
       Ast -> print_string (Ast.string_of_program ast)
     | _ -> let sast = Semant.check ast in
```

```
      match !action with
        Ast     -> ()
      | Sast    -> print_string (Sast.string_of_sprogram sast)
      | LLVM_IR -> print_string (Llvm.string_of_llmodule
          (Codegen.translate sast))
      | Compile -> let m = Codegen.translate sast in
    Llvm_analysis.assert_valid_module m;
    print_string (Llvm.string_of_llmodule m)
```

8.8 C Standard Library

8.8.1 c-library.h

```
#ifndef _LIST_H_
#define _LIST_H_

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

/* Members of the List */
struct ListObj {
    void *data;
    struct ListObj *next;
};

/* The list itself */
struct List {
    struct ListObj *head;
};

/* The Node struct */
struct Node {
   void *data;
    struct Node* left;
    struct Node* right;

};

/* Function to Initialize list */
struct List *list_init();

/* Function to add elements */
void add_elem(struct List *list, void *data);

/* Adds to the back of the list */
void append (struct List *list, void *data);
```

```
/* Function to index and retrieve list elements */
struct ListObj *get_elem(int n, struct List *list);

/* Function to update list element at given index with given value */
struct List *update_elem(void* data, struct List *list, int index);

static inline int isEmptyList(struct List *list)
{
    return (list->head == NULL);
}

int size(struct List *list);

/* NODE FUNCTIONS START HERE */

struct Node *init_node (void *input);

void *get_data(struct Node *node);

struct Node *get_right (struct Node *node);

struct Node *get_left (struct Node *node);


void add_left (struct Node* node, struct Node* node2);

void add_right (struct Node* node, struct Node* node2);
```

8.8.2 c-library.c

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

#include "c_library.h"

struct List *list_init() {
   struct List *list = (struct List *)malloc(sizeof(struct List));
   list->head = NULL;
   return list;
}

/*adds elements to the front of the list? */
void add_elem(struct List *list, void *data) {
    struct ListObj *node = (struct ListObj *)malloc(sizeof(struct
        ListObj));
    if (node == NULL)
      return;
    node->data = data;
    node->next = list->head;
    list->head = node;
}

void append (struct List *list, void *data) {
       // make the new node that will go to the end of list
    struct ListObj *node = (struct ListObj *)malloc(sizeof(struct
         ListObj));
    node->data = data;
    node->next = NULL;

    // if the list is empty, this node is the head
    if (list->head == NULL) {
   list->head = node;
    }

    // find the last node
    struct ListObj *end = list->head;
    while (end->next != NULL)
   end = end->next;

    // 'end' is the last node at this point
    end->next = node;
}

/* Gets an element by an inputted index */
struct ListObj *get_elem(int n, struct List *list) {
    if( n >= size(list) ) {
        fprintf(stderr, "Index out of bounds.");
      exit(-1);
    }

   struct ListObj *cur = list->head;
   int count = 0;

   while (cur != NULL) {
      if (count == n)
         return (cur->data);

      count++;
      cur = cur->next;
   }
}

/* Updates an element at an inputted index with an inputted value */
struct List *update_elem(void* data, struct List *list, int index) {
   if(isEmptyList(list)) {
      return list;
   }
   struct ListObj *node = list->head;

   ///if index is wrong, return original list
   if( index >= size(list) ) {
                fprintf(stderr, "List Index out of bounds: given %d for
                     list of size %d\n", index, size(list));
                exit(-1);
        }

   if(index == 0){
      node->data = data;
      return list;
   }

   for( int i = 0 ; i!=index ; i++ ) node = node->next;

   node->data = data;

   return list;
}

/* Returns the size of the list */
int size(struct List *list) {

   if (isEmptyList(list)) {
      return 0;
   }
   int count = 0;
   struct ListObj *cur = list->head;

   while(cur != NULL) {
      count++;
      cur = cur->next;
   }
   return count;
}

/* Initializes a node */
struct Node *init_node (void *input) {
   struct Node *node = (struct Node *)malloc(sizeof(struct Node));
   node->data = input;
   node->left = NULL;
   node->right = NULL;
   return node;
}

void *get_data(struct Node *node) {
    return node->data;
}
```

```
struct Node *get_left (struct Node *node) {
   return node->left;
}

struct Node *get_right (struct Node *node) {
   return node->right;
}



void add_left (struct Node* node, struct Node* node2) {
   node->left = node2;
}

void add_right (struct Node* node, struct Node* node2) {
   node->right = node2;
}
```

8.9 Test Suites

8.9.1 test-add1.nd

```
int main()
{
  print(17 + 25);
  return 0;
}
```

8.9.2 test-add2.nd

```
int main()
{
  print(-17 + 25);
  return 0;
}
```

8.9.3 test-arith2.nd

```
int main()
{
  print(1 + 2 * 3 + 4);
  return 0;
}
```

8.9.4 test-arith3.nd

```
int foo(int a)
{
  return a;
}

int main()
{
  int a;
  a = 42;
  a = a + 5;
  print(a);
  return 0;
}
```

8.9.5 test-comment1.nd

```
int main()
{
  //lets test our comments!//
  //yay comments!//
  print(17 + 25);
  return 0;
}
```

8.9.6 test-comment2.nd

```
//my first comment
is this comment//

int foo(int a)
{
  //another comment I have
  is this comment//
  return a; //hi again//
}

int main() //another comment
I have to say//
{
  return 0; //my last comment is
  that nodable is the best//
}
```

8.9.7 test-complexintuse.nd

```
int main(){
    int a;
    a = 3;
    int b;
    if ( a == 3) b = 2;

    print(a);
    print(b);

    int c;
    c = 10;
    print(c);

    return 0;
}
```

8.9.8 test-complexstruse.nd

```
int main(){
    string a;
    a = "hi";
    string b;
    b = "bye";

    prints(a);
    prints(b);

    string c;
    c = "cool";
    prints(c);

    return 0;
}
```

8.9.9 test-div1.nd

```
int main()
{
  print(6 / 2);
  return 0;
}
```

8.9.10 test-div2.nd

```
int main()
{
  print(6 / 5);
  return 0;
}
```

8.9.11 test-eq1.nd

```
int main()
{
  printb(1 == 2);
  return 0;
}
```

8.9.12 test-fib.nd

```
int fib(int x)
{
  if (x < 2) return 1;
  return fib(x-1) + fib(x-2);
}

int main()
{
  print(fib(0));
  print(fib(1));
  print(fib(2));
  print(fib(3));
  print(fib(4));
  print(fib(5));
  return 0;
}
```

8.9.13 test-float1.nd

```
int main()
{
  float a;
  a = 3.14159267;
  printf(a);
  return 0;
}
```

8.9.14 test-float2.nd

```
int main()
{
  float a;
  float b;
  float c;
  a = 3.14159267;
  b = -2.71828;
  c = a + b;
  printf(c);
  return 0;
}
```

8.9.15 test-float3.nd

```
void testfloat(float a, float b)
{
  printf(a + b);
  printf(a - b);
  printf(a * b);
  printf(a / b);
  printb(a == b);
  printb(a == a);
  printb(a != b);
  printb(a != a);
  printb(a > b);
  printb(a >= b);
  printb(a < b);
  printb(a <= b);
}

int main()
{
  float c;
  float d;

  c = 42.0;
  d = 3.14159;

  testfloat(c, d);

  testfloat(d, d);
  return 0;
}
```

8.9.16 test-for1.nd

```
int main()
{
  int i;
  for (i = 0 ; i < 5 ; i = i + 1) {
    print(i);
  }
  print(42);
  return 0;
}
```

8.9.17 test-func1.nd

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

int main()
{
  int a;
  a = add(39, 3);
  print(a);
  return 0;
}
```

8.9.18 test-func2.nd

```
int fun(int x, int y)
{
  return 0;
}

int main()
{
  int i;
  i = 1;

  fun(i = 2, i = i+1);
  print(i);
  return 0;
}
```

8.9.19 test-func3.nd

```
void printem(int a, int b, int c, int d)
{
  print(a);
  print(b);
  print(c);
  print(d);
}

int main()
{
  printem(42,17,192,8);
  return 0;
}
```

8.9.20 test-func4.nd

```
int add(int a, int b)
{
  int c;
  c = a + b;
  return c;
}

int main()
{
  int d;
  d = add(52, 10);
  print(d);
  return 0;
}
```

8.9.21 test-func5.nd

```
int foo(int a)
{
  return a;
}
```

```
int main()
{
  return 0;
}
```

8.9.22 test-func6.nd

```
void foo() {}

int bar(int a, boolean b, int c) { return a + c; }

int main()
{
  print(bar(17, false, 25));
  return 0;
}
```

8.9.23 test-func7.nd

```
int a;

void foo(int c)
{
  a = c + 42;
}

int main()
{
  foo(73);
  print(a);
  return 0;
}
```

8.9.24 test-func8.nd

```
void foo(int a)
{
  print(a + 3);
}

int main()
{
  foo(40);
  return 0;
}
```

8.9.25 test-func9.nd

```
void foo(int a)
{
  print(a + 3);
}

int main()
{
  foo(40);
  return 0;
}
```

8.9.26 test-gcd.nd

```
int gcd(int a, int b) {
  while (a != b) {
    if (a > b) a = a - b;
    else b = b - a;
  }
  return a;
}

int main()
{
  print(gcd(2,14));
  print(gcd(3,15));
  print(gcd(99,121));
  return 0;
}
```

8.9.27 test-geq1.nd

```
int main()
{
    printb(1 >= 2);
    return 0;
}
```

8.9.28 test-global1.nd

```
int a;
int b;

void printa()
{
  print(a);
}

void printbb()
{
  print(b);
}

void incab()
{
  a = a + 1;
  b = b + 1;
}

int main()
{
  a = 42;
  b = 21;
  printa();
  printbb();
  incab();
  printa();
  printbb();
  return 0;
}
```

8.9.29 test-global2.nd

```
boolean i;

int main()
{
  int i;
  i = 42;
  print(i + i);
  return 0;
}
```

8.9.30 test-global3.nd

```
int a;
int b;

int main()
{
  a = 42;
  b= 21;
  print(a);
  print(b);
  return 0;
}
```

8.9.31 test-global4.nd

```
int i;
int j;

int main()
{
  i = 42;
  j = 10;
  print(i + j);
  return 0;
}
```

8.9.32 test-if1.nd

```
int main()
{
  if (true) print(42);
  print(17);
  return 0;
}
```

8.9.33 test-if2.nd

```
int main()
{
  if (true) print(42); else print(8);
  print(17);
  return 0;
}
```

8.9.34 test-if3.nd

```
int main()
{
  if (false) print(42);
  print(17);
  return 0;
}
```

8.9.35 test-if4.nd

```
int main()
{
  if (false) print(42); else print(8);
  print(17);
  return 0;
}
```

8.9.36 test-if5.nd

```
int cond(boolean b)
{
  int x;
  int y;
  if (b) {
    x = 42;
    y = 14;
  }
  else {
    x = 17;
    y = 8;
  }
  return y;
}

int main()
{
 print(cond(true));
 print(cond(false));
 return 0;
}
```

8.9.37 test-if6.nd

```
int cond(boolean b)
{
  int x;
  x = 10;
  if (b)
    if (x == 10)
      x = 42;
  else
    x = 17;
  return x;
}

int main()
{
 print(cond(true));
 print(cond(false));
 return 0;
}
```

8.9.38 test-leq1.nd

```
int main()
{
    printb(1 <= 2);
    return 0;
}
```

8.9.39 test-list1.nd

```
int main()
{
    list <int> a;
    a = [1,2,3];
    print(a[0]);
    print(a[1]);
    print(a[2]);
    return 0;
}
```

8.9.40 test-list2.nd

```
int main()
{
    list <string> a;
    a = ["hi","test"];
    prints(a[0]);
    return 0;
}
```

8.9.41 test-list3.nd

```
int main()
{
    list <float> a;
    a = [3.14159267, 2.7398111];
    printf(a[0]);
    return 0;
}
```

8.9.42 test-list4.nd

```
int main()
{
    list <boolean> a;
    a = [true, false];
    printb(a[1]);
    return 0;
}
```

8.9.43 test-listappend.nd

```
int main(){
    list<node<int> > t;
    node<int> a;
    node<int> b;
    node<int> c;
    node<int> d;
    a = $3;
    b = $9;
    c = $18;
    d = $27;

    t = [a, b, c];
    append(t, d);

    print(t[3].data);
}
```

8.9.44 test-listdec.nd

```
int main()
{
    list <float> a;
    a = [3.14159267, 2.7398111];

    // list <int> b; //

    printf(a[0]);
    return 0;
}
```

8.9.45 test-listdemo.nd

```
int average(list <node<int>> a)
{
    int sum;
    int i;
    sum = 0;
    for(i=0; i < size(a); i=i+1)
    {
        sum = sum + a[i].data;
    }
    return sum/(size(a));
}

void printElements(list <node<int>> a)
{
    int i;
    for(i=0; i < size(a); i=i+1)
    {
        print(a[i].data);
    }
}

void reverse(list <node<int>> a)
{
    int i;
    node<int> temp;
    for(i=0; i<(size(a)/2); i = i+1)
    {
        temp = a[i];
        a = update_elem(a[size(a)-i-1], a, i);
        a = update_elem(temp, a, size(a)-i-1);
    }
}
```

```
void selectionSort(list <node<int>> a)
{
    int i;
    node<int> min;
    int minIndex;
    node<int> temp;
    int j;
    for(i=0; i < size(a)-1; i=i+1)
    {
        min = a[i];
        minIndex = i;
        for(j = i+1; j<size(a); j=j+1)
        {
            if(min.data > a[j].data)
            {
                min = a[j];
                minIndex=j;
            }
        }

        temp = a[i];
        a = update_elem(min, a, i);
        a = update_elem(temp, a, minIndex);

    }
}


int main()
{
    prints("Let's make the list of nodes with values 2,2,4,1,7");
    list<node<int> > t;
    node<int> a;
    node<int> b;
    node<int> c;
    node<int> d;
    node<int> e;
    a = $2;
    b = $2;
    c = $4;
    d = $1;
    e = $7;

    t = [a, b, c, d, e];
    prints("Now lets print out each element in order:");
    printElements(t);

    prints("Now lets see how long it is:");
    print(size(t));
    prints("Now lets get its average value and print it (rounded down to
        an integer):");
    print(average(t));

    prints("Now let's reverse the list");
    reverse(t);
    prints("Now let's print out each element in their new order:");
    printElements(t);

    prints("Now lets do selection sort to order the list");
    selectionSort(t);

    prints("Now lets print out every element of our newly ordered
        list:");
    printElements(t);

    return 0;
}
```

8.9.46 test-listempty.nd

```
int main()
{
    list <float> a;
    a = [];
    return 0;
}
```

8.9.47 test-listfunct1.nd

```
//this shows that a in the main function will not be effected by foo//

void foo(list <string> a)
{
  a=["bye","test"];
}

int main()
{
  list <string> a;
  a = ["hi","test"];
  prints(a[0]);
  foo(a);
  prints(a[0]);
  return 0;
}
```

8.9.48 test-listnest.nd

```
int main()
{
    list <list<string>> a;
    a = [["hi", "i really"], ["hope", "that", "this", "works"]];
    prints(a[1][0]);
}
```

8.9.49 test-listsize.nd

```
int main()
{

    list<node<int> > t;
    node<int> a;
    node<int> b;
    node<int> c;
    node<int> d;
    a = $3;
    b = $9;
    c = $18;
    d = $27;

    t = [a, b, c, d];

    print(size(t));
    return 0;
}
```

8.9.50 test-listupdate.nd

```
int main(){
    list<node<int> > t;
    node<int> a;
    node<int> b;
    node<int> c;
    node<int> d;
    a = $3;
    b = $9;
    c = $18;
    d = $27;
    t = [a, b, c];
    t = update_elem(d, t, 2);

    print(t[2].data);
}
```

8.9.51 test-local1.nd

```
int main()
{
  int i;

  i = 42;
  print(i + i);
  return 0;
}
```

8.9.52 test-local2.nd

```
void foo(boolean i)
{
  int i;

  i = 42;
  print(i + i);
}

int main()
{
  foo(true);
  return 0;
}
```

8.9.53 test-local3.nd

```
int foo(int a, boolean b)
{
  int c;
  boolean d;

  c = a;

  return c + 10;
}

int main() {
 print(foo(37, false));
 return 0;
}
```

8.9.54 test-mod1.nd

```
int main()
{
  print(6 % 2);
  return 0;
}
```

8.9.55 test-mult1.nd

```
int main()
{
  print(3 * 2);
  return 0;
}
```

8.9.56 test-mult2.nd

```
int main()
{
  print(-3 * -2);
  return 0;
}
```

8.9.57 test-neq1.nd

```
int main()
{
  printb(1 != 2);
  return 0;
}
```

8.9.58 test-node1.nd

```
int main() {
    node<string> t = $"test";
    prints(t.data);
    return 0;
}
```

8.9.59 test-node2.nd

```
int main() {
    node<int> t;
    t = $2;
    print(t.data);
    return 0;
}
```

8.9.60 test-node3.nd

```
int main() {
    node<float> t;
    t = $9.6;
    printf(t.data);
    return 0;
}
```

8.9.61 test-node4.nd

```
int main() {
    node<boolean> t;
    t = $false;
    printb(t.data);
    return 0;
}
```

8.9.62 test-node5.nd

```
int main() {
    list <node<int>> t;
    t = [$2, $4, $8];
    print(t[0].data);
    return 0;
}
```

8.9.63 test-node6.nd

```
int main() {
    node<int> t1;
    node<int> t2;
    node<int> t3;
    list<node<int>> t;
    int i;

    t1 = $2;
    t2 = $4;
    t3 = $8;
    t = [t1, t2, t3];

    for (i = 0; i < 3; i = i + 1){
        print(t[i].data);
    }

    print(t1.data);
    print(t2.data);
    print(t3.data);

    return 0;
}
```

8.9.64 test-printbool.nd

```
int main()
{
    printb(true);
    return 0;
}
```

8.9.65 test-printflt.nd

```
int main()
{
    printf(9.2);
    return 0;
}
```

8.9.66 test-printint.nd

```
int main()
```

```
{
    print(4);
    return 0;
}
```

8.9.67 test-printstr.nd

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

8.9.68 test-recurs1.nd

```
int foo (int n) {
   if (n <= 1)
      return n;
   else
      return foo(n - 1) + foo(n - 2);
}

int main () {
    print(foo(3));
}
```

8.9.69 test-sub1.nd

```
int main()
{
  print(25 - 17);
  return 0;
}
```

8.9.70 test-sub2.nd

```
int main()
{
  print(17 - 25);
  return 0;
}
```

8.9.71 test-sub3.nd

```
int main()
{
  print(-25 - 17);
  return 0;
}
```

8.9.72 test-tree.nd

```
int main () {
    node<int> a;
    a = $1;
    node<int> b = $2;
    add_left(a, b);
    print(get_left(a).data);
    return 0;
}
```

8.9.73 test-tree2.nd

```
int main () {
    node<int> a;
    node<int> b;
    node<int> c;
    int result;
    a = $1;
    b = $2;
    c = $3;
    add_right(a, b);
    add_left(a, c);
    print(get_right(a).data);
    print(get_left(a).data);
    return 0;
}
```

8.9.74 test-tree3.nd

```
int main () {
    node<int> a;
    node<int> b;
    node<int> c;
    node<int> d;
    a = $1;
    b = $2;
    c = $3;
    d = $4;
    add_right(a, b);
    add_left(b, c);
    add_right(c, d);
    print(get_right(a).data);
    print(get_left(b).data);
    print(get_right(c).data);
}
```

8.9.75 test-tree4.nd

```
int main () {
    node<int> a;
    node<int> b;
    node<int> c;
    a = $1;
    b = $2;
    c = $300;
    add_left(a, b);
    add_right(b, c);
    print(c.data);
}
```

8.9.76 test-treeheight.nd

```
int tree_height(node<int> root) {
    int left_height;
    int right_height;
    int max;

    if (root.data == 0) {
        max = 0;
    }
    else {
        left_height = tree_height(get_left(root));
        right_height = tree_height(get_right(root));
        if (left_height < right_height) {
            max = right_height + 1;
        }
        else max = left_height + 1;

    }
    return max;
}
```

```
int main() {
   int result;
   node<int> root;
   node<int> c1;
   node<int> c2;
   node<int> c3;
   node<int> c4;
   node<int> null;

   root = $1;

   c1 = $2;
   add_left(root, c1);

   c2 = $3;
   add_right(root, c2);

   c3 = $4;
   add_left(c1, c3);

   c4 = $5;
   add_right(c1, c4);

   null = $0;
   add_left(c2, null);
   add_right(c2, null);
   add_left(c3, null);
   add_right(c3, null);
   add_left(c4, null);
   add_right(c4, null);

   result = tree_height(root);
   print(result);
   return 0;
}
```

8.9.77 test-while1.nd

```
int main()
{
  int i;
  i = 5;
  while (i > 0) {
    print(i);
    i = i - 1;
  }
  print(42);
  return 0;
}
```

8.9.78 test-while2.nd

```
int foo(int a)
{
  int j;
  j = 0;
  while (a > 0) {
    j = j + 2;
    a = a - 1;
  }
  return j;
}

int main()
{
  print(foo(7));
  return 0;
}
```

8.10 Fail Tests

8.10.1 fail-assign1.nd

```
int main()
{
  int i;
  string s;
  i = 4;
  i=7;
  s = "hi";
  i = "bye";
  prints("done");
  return 0;
}
```

8.10.2 fail-assign2.nd

```
int main()
{
  int i;
  string s;
  s=4;
  prints("done");
  return 0;
}
```

8.10.3 fail-assign3.nd

```
void myvoid()
{
  return;
}

int main()
{
  int i;

  i = myvoid(); // Fail: assigning a void to an integer //
  return 0;
}
```

8.10.4 fail-dead1.nd

```
int main()
{
  print(17 + 25);
  return 0;
  print(17+25);
}
```

8.10.5 fail-dead2.nd

```
int main()
{
  int i;

  {
    i = 15;
    return i;
  }
  i = 32; // Error: code after a return //
}
```

8.10.6 fail-expr1.nd

```
int main()
{
  i = 17;
  return 0;
}
```

8.10.7 fail-expr2.nd

```
int main()
{
  int i;
  i = 17 + "hi";
  return 0;
}
```

8.10.8 fail-expr3.nd

```
int a;
float b;

void foo(int c, float d)
{
  int d;
  float e;
  b + a; // Error: float + int //
}

int main()
{
  return 0;
}
```

8.10.9 fail-float1.nd

```
int main()
{
  -3.5 && 1;
  return 0;
}
```

8.10.10 fail-float2.nd

```
int main()
{
  -3.5 && 2.5;
  return 0;
}
```

8.10.11 fail-for1.nd

```
int main()
{
  int i;

  for (i = 0 ; true; i = i + 1) {
    print("hi");
  }


  return 0;
}
```

8.10.12 fail-for2.nd

```
int main()
{
  int i;

  for (i = 0; j < 10 ; i = i + 1) {}

  return 0;
}
```

8.10.13 fail-for3.nd

```
int main()
{
  int i;

  for (i = 0; i ; i = i + 1) {}

  return 0;
}
```

8.10.14 fail-for4.nd

```
int main()
{
  int i;

  for (i = 0; i < 10 ; i = j + 1) {} // j undefined //

  return 0;
}
```

8.10.15 fail-for5.nd

```
int main()
{
  int i;

  for (i = 0; i < 10 ; i = i + 1) {
    foo(); // Error: no function foo //
  }

  return 0;
}
```

8.10.16 fail-func1.nd

```
int foo() {}

int bar() {}

int baz() {}

void bar() {} // Error: duplicate function bar //

int main()
{
  return 0;
}
```

8.10.17 fail-func2.nd

```
int foo(int a, string b, int c) { }
```

```
void bar(int a, string b, int a) {} // Error: duplicate formal a in bar
    //

int main()
{
  return 0;
}
```

8.10.18 fail-func3.nd

```
int foo(int a, string b, int c) { }

void bar(int a, void b, int c) {} // Error: illegal void formal b //

int main()
{
  return 0;
}
```

8.10.19 fail-func4.nd

```
int foo() {}

void bar() {}

int print() {} // Should not be able to define print //

void baz() {}

int main()
{
  return 0;
}
```

8.10.20 fail-func5.nd

```
int foo() {}

int bar() {
  int a;
  void b; // Error: illegal void local b //

  return 0;
}
```

```
int main()
{
  return 0;
}
```

8.10.21 fail-func6.nd

```
void foo(int a, string b)
{
}

int main()
{
  foo(42, "hi");
  foo(42); // Wrong number of arguments //
}
```

8.10.22 fail-func7.nd

```
void foo(int a, string b)
{
}

int main()
{
  foo(42, "hi");
  foo(42, "hi", "bye"); // Wrong number of arguments //
}
```

8.10.23 fail-func8.nd

```
void foo(int a, string b)
{
}

void bar()
{
}

int main()
{
  foo(42, "hi");
  foo(42, bar()); // int and void, not int and string //
}
```

8.10.24 fail-func9.nd

```
void foo(int a, string b)
{
}

int main()
{
  foo(42, "hi");
  foo(42, 42); // Fail: int, not string //
}
```

8.10.25 fail-global1.nd

```
int c;
void a; // global variables should not be void //


int main()
{
  return 0;
}
```

8.10.26 fail-global2.nd

```
int b;
int a;
int b; // Duplicate global variable //

int main()
{
  return 0;
}
```

8.10.27 fail-if1.nd

```
int main()
{
  if (true) {}
  if (false) {} else {}
  if (42) {} // Error: not a valid predicate //
}
```

8.10.28 fail-if2.nd

```
int main()
{
  if (true) {
    foo; // Error: undeclared variable //
  }
}
```

8.10.29 fail-if3.nd

```
int main()
{
  if (true) {
    42;
  } else {
    bar; // Error: undeclared variable //
  }
}
```

8.10.30 fail-nomain.nd

```
//nothing//
```

8.10.31 fail-print.err

```
// Should be illegal to redefine //
void print() {}
```

8.10.32 fail-printbool.nd

```
// Should be illegal to redefine //
void printb() {}
```

8.10.33 fail-printstr.nd

```
// Should be illegal to redefine //
void prints() {}
```

8.10.34 fail-return1.nd

```
int main()
{
  return true; // Should return int //
}
```

8.10.35 fail-return2.nd

```
void foo()
{
  if (true) return 42; // Should return void //
  else return;
}

int main()
{
  return 42;
}
```

8.10.36 fail-while1.nd

```
int main()
{
  int i;

  while (true) {
    i = i + 1;
  }

  while (42) { // Should be boolean //
    i = i + 1;
  }

}
```

8.10.37 fail-while2.nd

```
int main()
{
  int i;

  while (true) {
    i = i + 1;
  }

  while (true) {
    foo(); // foo undefined //
  }

}
```