Gantry

A JSON Data Processing Language

Audrey Copeland, Walter Meyer, Taimur Samee, Rizwan Syed

Contents

1 Introduction **4**
- 1.1 Language Whitepaper . . . 4

2 Language Tutorial **13**
- 2.1 Requirements . . . 13
- 2.2 Installation on Ubuntu 16.04 LTS . . . 13
- 2.3 Compiling the Compiler . . . 13
- 2.4 Testing the Compiler . . . 13
- 2.5 Compiling a Program . . . 14
- 2.6 Compiling a Program to LLVM IR . . . 14

3 Language Reference Manual **14**
- 3.1 Lexical Conventions . . . 14
  - 3.1.1 Tokens . . . 14
  - 3.1.2 Comments . . . 14
  - 3.1.3 Identifiers . . . 15
  - 3.1.4 Keywords . . . 15
  - 3.1.5 Operators . . . 15
  - 3.1.6 Constants . . . 15
  - 3.1.7 Character Escape Sequences . . . 16
- 3.2 Expressions . . . 16
  - 3.2.1 Functions . . . 17
  - 3.2.2 Built-In Functions . . . 17
  - 3.2.3 Operator Precedence . . . 23
- 3.3 Statements . . . 23
  - 3.3.1 Expression-Statements . . . 23
  - 3.3.2 Control-Statements . . . 24
  - 3.3.3 Jump Statements . . . 25
  - 3.3.4 Comparison Operators . . . 26
  - 3.3.5 Assignment Expressions . . . 26
  - 3.3.6 Identifiers . . . 26
  - 3.3.7 Arrays . . . 27
  - 3.3.8 Objects . . . 28
- 3.4 Grammar . . . 29

4 Project Plan **32**
- 4.1 Process . . . 32
  - 4.1.1 Planning . . . 32
  - 4.1.2 Specification . . . 33
  - 4.1.3 Development . . . 33
  - 4.1.4 Testing . . . 33
- 4.2 Programming Style Guide . . . 34

4.2.1 General Guidelines . . . 34
4.2.2 OCaml . . . 34
4.2.3 C . . . 34
4.2.4 Gantry . . . 35
4.3 Project Timeline . . . 35
4.3.1 Roles and Responsibilities . . . 35
4.4 Software Development Environment . . . 36
4.5 Project Log . . . 37

5 Architectural Design **94**
5.1 Block Diagram . . . 94
5.1.1 Top-Level . . . 94
5.1.2 Scanner . . . 94
5.1.3 Parser and AST . . . 94
5.1.4 Semantic Checking . . . 95
5.1.5 Code Generation . . . 95

6 Test Plan **97**
6.1 Sample Programs . . . 97
6.1.1 Greatest Common Divisor . . . 97
6.1.2 Docker API calls . . . 100
6.2 Test Automation . . . 104

7 Reflections **105**
7.1 Lessons Learned . . . 105
7.2 Advice for Future Teams . . . 108

8 Appendix **110**
8.1 Code Listing . . . 110
8.1.1 Introduction . . . 110
8.1.2 Core Code . . . 112
8.1.3 Library Functions . . . 154
8.2 Test Suite . . . 179
8.2.1 Test Script . . . 179
8.2.2 Test Cases . . . 184
8.3 Applications . . . 217
8.3.1 blackjack.gty . . . 217
8.3.2 docker.gty . . . 224
8.3.3 docker-run.gty . . . 226
8.3.4 test_string_input.gty . . . 228

1 Introduction

The Gantry Language was designed to support the programmatic manipulation of JSON data by implementing C-like syntax and semantics along with JSON-like[1] data types. Gantry is a weakly-typed language that is statically-typed with the notable exception of Objects, whose elements, while statically-typed, are polymorphic and may change at runtime.

Gantry's distinguishing features are its user-friendly aggregate data types: Objects and Arrays. Objects are mutable key-value data structures with polymorphic keys that can be infinitely nested. Arrays contain a single type, while being static and limited to a single dimension. Gantry also provides String, HTTP, and Object libraries that provide the functionality necessary to interact with JSON-based web APIs.

This document outlines our initial, admittedly idealistic, ideas which ultimately informed our language design in ways there were both justifiably technical and simply the the product of time and knowledge constraints. While we are proud of what we were able to accomplish, we admittedly fell short of our original specification. Interestingly, we found that the implementation challenges that we encountered exposed our technical misunderstandings insofar that we had not been forced to confront many of the harsh realities of implementation that underlie the "magic" of the programming languages that we have learned and used in practice up to this point. So, without further ado and weakly-guised appeals to project-based pedagogy in computer science, here is Gantry!

1.1 Language Whitepaper

This is our original language proposal for reference purposes.

1. Introduction

The Gantry Language is designed to make algorithmic processing of JSON data simpler. Gantry will allow for the programmatic manipulation of JSON data by implementing C-like syntax and semantics along with JSON-like[2] data types and structures.

1.1 *Implementation*

Gantry will be designed to interface with the Docker container runtime via its API, which uses JSON as a data-exchange format.

1.2 *Use Cases*

[1] http://www.ietf.org/rfc/rfc4627.txt
[2] http://www.ietf.org/rfc/rfc4627.txt

Gantry's JSON-centric features will make it easy to develop programs that interact with a multitude of JSON-based APIs. Specifically, Gantry can be used to facilitate the orchestration of Linux Containers using the Docker[3] JSON API. Containers are a form of Operating System Virtualization typically used to deploy applications (Web Servers, Database Servers, etc.) into lightweight, isolated, and immutable objects. For example, a Gantry program will be able to create, destroy, start and stop Containers using the Docker JSON API based on real-time Container properties returned from the Docker API.

2. Lexical Conventions

2.1 Comments

Comments are lines beginning with two forward slashes, or blocks beginning with /* and ending with */

```
// This is a comment
```

```
/*
This is a comment
in block format
*/
```

2.2 Identifiers

An identifier (a variable) is a sequence of alphanumeric characters that may begin with any alpha-numeric character.

2.3 Keywords

func	int	float
string	char	arr
true	false	null
if	elif	else
for	while	object
continue	break	return

2.4 Operators

[3]https://www.docker.com/what-docker

Operator	Syntax	Operands
Arithmetic	a + b, a - b, a * b, a / b	int, float
Assignment	a = b	int, float, bool, char, string
Equal	a == b	int, float, bool, char, string
Not Equal	a != b	int, float, bool, char, string
Comparison	a <= b, a <b , a >= b, a >b	int, float
Logical AND	a && b	bool
Logical OR	a‖b	bool
Logical NOT	!a	bool
Concatenation	a ^ b	string

2.5 Built-In Functions

Gantry includes built-in functions to handle some fundamental operations that are useful for parsing JSON-formatted data.

2.5.1 jsonify()

The jsonify() function will take an object as a parameter and will convert the object into a JSON-formatted string. e.g.:

Listing 1: jsonify()

```
1  string course = "PLT";
2  int students = 125;
3  string location = "NWC";
4  object course_obj;
5  string course_obj.course = course;
6  int course_obj.students = students;
7  string course_obj.location = location;
8  string course_str = jsonify(course_obj);
9  print(course_str);
10 // prints {course="PLT",students=125,location="NWC"}
```

2.5.2 objectify()

The objectify() function will take a string as a parameter and will attempt to produce a representation of that JSON-formatted string as an object with its nested component data types. e.g:

Listing 2: objectify()

```
1  string str = "{course=\"PLT\",students=125,location=\"NWC\"}";
2  object course_obj = objectify(str);
3  string course_name = course_obj.course;
```

```
int course_enrollment = course_obj.students;
string course_location = course_obj.location;
print(course_name);
// prints "PLT"
```

2.5.3 arrify()

The arrify() function will take a string as a parameter and will attempt to produce a representation of that JSON-formatted string as an array. e.g:

Listing 3: arrify()

```
string str = "[{course=\"PLT\",students=125,location=\"NWC\"},
{course=\"CS Theory\",students=200,location=\"NWC\"}]";
arr courses_arr = arrify(str);
object first_course = courses_arr[0];
object second_course = courses_arr[1];
string first_course_name = first_course.course;
string second_course_name = second_course.course;
string output_string = first_course_name ^ " and " ^ second_course_name;
print(output_string);
// prints "PLT and CS Theory"
```

2.5.4 length()

The length() function will take an array as a parameter and will return the number of elements in the array. This function can also take a string as a parameter since we implement a string as an array of characters.

Listing 4: length()

```
arr student_arr = ["Joe", "Bob", "Alan"];
int arr_length = length(student_arr);
print(arr_length);
// prints 3
```

2.5.5 print()

The print() function will take a parameter of any type defined in our language and print its string representation. *Print* requires text to be bound by "" and can be concatenated with strings or integers using the hat symbol.

Listing 5: print()

```
string course_name = "PLT";
print("This is the course name: "^ course_name);
// prints "This is the course name : PLT"
```

2.5.6 to_string()

The to_string() function will take a parameter of any type defined in our language and return it as a string. This method will be called by print.

Listing 6: to_string()

```
int course_enrollment = 3;
string course_enrollment_string = to_string(course_enrollment);
```

2.5.7 http_get()

The http_get() function will take a server and port as a parameter along with a URI to perform an HTTP GET request to.

Listing 7: http_get()

```
/*
  Returns a json object of containers running on
  a particular Docker engine.
*/
string uri = "/v1.19/containers/json"
string cons = http_get("192.168.0.9", 80, uri);
object cons_arr = arrify(cons);
print(cons_arr);
```

2.5.8 http_post()

The http_post() function will take a server, port, URI, and POST data as parameters to form an HTTP POST request.

Listing 8: http_post()

```
/*
  Returns a json object of a newly created container
  running on a particular Docker engine.
*/
```

```
string post_data = "{"Image": "centos", "Cmd": ["echo", "hello world"]}"
string uri = "/v1.19/containers/create"
string con = http_post("192.168.0.9", 80, uri, post_data);
object con_obj = objectify(con);
print(con_obj);
```

2.5.9 sort()

The sort() function will take an array as an argument, and will by default sort it in ascending order. The function will optionally accept sort order (asc or dsc) and a key value to sort on. For example, 'sort_key=Size' would search inside the objects in the array and sort based on the values associated with the 'Size' key. If no 'Size' key exists in one or more of the objects present in the array, the sort function will fail to sort the array.

Listing 9: sort()

```
/*
  Returns a sorted list.
*/
arr my_number_list = [3, 9, 2, 1];
arr my_sorted_number_list = sort(my_number_list, asc);
print(my_sorted_number_list);
// prints '[1, 2, 3, 9]'
```

2.6 Constants

2.6.1 Integers

Integers are a sequence of numeric characters [0-9] in decimal notation. They are 32-bit signed integers in the range of -2,147,483,648 to 2,147,483,647.

2.6.2 Floats

2.6.3 Characters

Characters are standard ASCII characters with associated integer values from 0 to 127.

2.6.3.1 Character Constants

Character constants have special meaning and specific functionality. In Gantry, the following character constants are defined:

- '\n' will be used for line break.

- '\' will be used as an escape character.

2.6.4 Strings

Strings are immutable null-terminated arrays of characters.

2.7 Control Flow

Gantry supports basic control flow features, including if-else, for loops, and while loops.

2.7.1 If-Else

Listing 10: If-Else

```
int number = 10;
if(number > 5) {
    print("Number is greater than 5");
} else {
    print("Number is less than 5");
}

// prints "Number is greater than 5"
```

2.7.2 Loops

2.7.2.1 For Loop

Listing 11: For Loop

```
int number = 10;
for(int example = 1; example < 6; example++) {
    number = number + 1;
}
print(number);

// prints 15
```

2.7.2.2 While Loop

Listing 12: While Loop

```
int example = 1;
int number = 10;
while(example < 6) {
    number = number + 1;
    example = example + 1;
```

```
}
print(number);

// prints 15
```

2.8 Composite Types

2.8.1 Arrays

Arrays are an ordered list of values of a single type. The elements of an array can be indexed using bracket notation.

Listing 13: Arrays

```
arr nums = [101, 346, 1032, 2, 25];
int i = nums[0];
print(i); // prints 101

arr strs = ["hello", "world"];
string str = strs[1];
print(str); // prints "world"
```

2.8.2 Objects

Objects are mutable collections of key value pairs. Values of an object can be accessed and modified using dot notation.

Listing 14: Sample Code

```
object class = {
    name: "PLT",
    students: "125"
};

print(class.name); // prints PLT

class.students = 100; // change # of students
print(class.students); // prints 100
```

3. Sample Code

3.1 Function to query the Docker API for running Containers

Listing 15: Sample Code

```
object host = {
    ip: "192.168.1.3",
    port: "443",
    ver: "v1.19"
};

arr func listCont(object host) {
    string uri = "/" ^ host.ver ^ "/containers/json";
    arr cons = http_get(host.ip, host.port, uri);
    return cons;
}
```

3.2 Connect to the Docker Engine and Get Container Information

The following example program establishes a connection with a Docker engine on a running host at http://192.168.1.3/. First, the program gets the number of containers running on the engine and sorts them based on particular container object keys. Then it prints the container names in sorted orders. If there are no containers running on the engine it reports that instead.

Listing 16: Sample Code

```
object myhost = {
    ip: "192.168.1.3",
    port: "443",
    ver: "v1.19"
};

arr container_list = listCont(myhost);
if (length(container_list) > 0) {

    // sort container list by given key and order
    arr sorted_list = sort(container_list, asc, sort_key="size");
    arr sorted_list2 = sort(container_list, dsc, sort_key="name");

    // prints names of containers from smallest to largest size
    for(int i = 0 ; i < length(sorted_list); i++){
        print(sorted_list[i].name);
    }
}
else {
    print("There are no containers running on this engine.");
}
```

2 Language Tutorial

2.1 Requirements

OCaml 4.02 (or higher)
LLVM 3.8
m4 1.4.17 (or higher)
gcc 5.4 (or higher)
make 4.1 (or higher)
libcurl4
pkg-config (Linux-specific requirement)

2.2 Installation on Ubuntu 16.04 LTS

```
apt-get install ocaml opam m4 pkg-config llvm-3.8 libcurl4-gnutls-dev
opam init --auto-setup
opam install -y ocamlfind llvm.3.8
```

Versions other than the aforementioned may work, but have not been tested. Additionally, you should be able to get Gantry compiled on the Operating System of your choice provided you have the prerequisites installed – namely, *ocaml, opam, m4, llvm, and libcurl.*

2.3 Compiling the Compiler

To compile Gantry (compilers need compiling too), navigate to the the Gantry working directory and run:

```
make
```

This will produce the compiler binary 'gantry.native' with which you can compile your own programs, as detailed in the instructions to follow.

2.4 Testing the Compiler

To test the compiler you can run:

```
./test_all.sh
```

This will compile and execute all of the tests provided in the *tests/* directory. If these test work, you can reasonably assume that the compiler is functioning on your system and begin compiling your own Gantry (.gty) programs.

2.5 Compiling a Program

Gantry provides a wrapper script that will compile your source (.gty) program and perform the appropriate linking with the libraries that Gantry uses. For example, to compile a program called *program.gty* you can do the following:

```
1 int main () {
2   print_s("Hello World");
3 }
```

```
./gantry_comp.sh program.gty
```

This will produce a native binary called *program* that you can execute.

2.6 Compiling a Program to LLVM IR

If you would like to produce the LLVM IR output from your program, you can run the following to send it to *stdout*:

```
./gantry.native < program.gty
```

3 Language Reference Manual

The *Gantry Language Reference Manual* details Gantry's syntactical and semantic specifications along with any relevant implementation specific details.

3.1 Lexical Conventions

3.1.1 Tokens

Gantry has five types of tokens: identifiers, keywords, operators, constants, and separators.

3.1.2 Comments

Comments are lines beginning with two forward slashes, or blocks beginning with /* and ending with */ . Note that comments are not tokens.

```
// This is a comment
```

```
/*
This is a comment
in block format
*/
```

3.1.3 Identifiers

An identifier, or variable name, is a sequence of alphanumeric characters or underscores that must begin with a letter or number. Identifiers are case-sensitive and may not be a Gantry keyword.

3.1.4 Keywords

int	float	string	bool
array	object	true	false
if	else	while	for

3.1.5 Operators

The following table details the operators in Gantry along with their associated operands.

Operator	Syntax	Operands
Arithmetic	a + b, a - b, a * b, a / b	int, float
Assignment	a = b	int, float, bool, string
Equal	a == b	int, float, bool, string
Not Equal	a != b	int, float, bool, string
Comparison	a <= b, a <b , a >= b, a >b	int, float
Logical AND	a && b	bool
Logical OR	a‖b	bool
Logical NOT	!a	bool
Concatenation	a^b	string

See section *3.3.3* for the order of operations.

3.1.6 Constants

There are four kinds of constants in Gantry: *int*, *float*, *string*, and *bool*.

int

An int is a sequence of numeric characters [0-9] in decimal notation. They are 32-bit signed integers in the range of -2,147,483,648 to 2,147,483,647. An int must contain at least one digit.

float

Floats are real numbers with integer and decimal parts separated by a decimal point. They are 64-bit signed values in the range -3.4×10^{-38} to 3.4×10^{38}, with precision of up to 6 decimal places.

bool

Booleans are 8-bit primitives with values *true* and *false*.

string

A string is an immutable sequence of zero or more ASCII characters or character escape sequences.

3.1.7 Character Escape Sequences

Character escape sequences allow for the use of certain ASCII characters in strings that overlap with language tokens as well as certain non-printable or spacing characters. The backslash character '\' signifies the beginning of a character escape sequence. The following character escape sequences are supported:

- \n yields a newline
- \r yields a carriage return
- \t yields a tab
- \b yields a backspace
- \\ yields a backslash
- \f yields a form feed
- \" yields a double-quote

3.2 Expressions

An expression in Gantry represents a value. Expressions consist of one or more operands and zero or more operators, where one, and only one, operator must exist between two operands. The following are examples of expressions:

```
42
2 + 2
3 - 1
-5.0
3 / 2
```

Expressions can also be calls to functions, array subscripts, etc. Consider the following examples:

```
foo(3)
bar[2]
```

3.2.1 Functions

A function in Gantry must be declared in the following format:

<type> <identifier> (optional-typed, comma-separated list of parameters) { statements }

A function must be declared with and return the same single type. A function may also include a list of typed and comma-separated parameters that will be lexically scoped into the body of the function.

Listing 17: Function Declaration

```
int repMsg(int times, string message) {
    for (int i = 0; i <= times; i++) {
        print_s(message);
    }
    return 0;
}
```

3.2.2 Built-In Functions

Gantry includes built-in functions that are useful for interacting with JSON-formatted data and HTTP JSON APIs.

Print Functions

There are five print functions: print_b(), print_d(), print_i(), print_k, and print_s(), which take as parameters, booleans, floats, integers, objects, and strings, respectively. The following is an example using the print_s() function:

Listing 18: print_s()

```
int main(){
    string course_name = "PLT";
    print_s("This is the course name: "^ course_name);

    return 0;
}
// prints "This is the course name : PLT"
```

string_length()

The string_length() function takes a string as a parameter and returns the number of characters in said string.

Listing 19: string_length()

```
int main(){
    string s = "hello";
    int s_len = string_length(s);
    print_i(s_len);

    return 0;
}
// prints 5
```

slice()

The slice() function takes a string and two integer indices as parameters. It returns a substring or "slice" of the original string starting from the first index up to and excluding the second index.

Listing 20: slice()

```
int main(){
    string s = "foobar";
    string sliced = slice(s, 2, 5);
    print_s(sliced);
    return 0;
}
/*
Outputs: oba
*/
```

get_string()

The get_string() function takes a request as a parameter which will be output to the user and returns the user input.

Listing 21: get_string()

```
int main(){
       string y = "Please enter a string : ";
       string s = get_string(y);
       return 0;
}
/*
Doesn't print anything, s will hold user input until user hits return.
*/
```

stoint()

The stoint() function takes a string of an integer as a parameter and returns an integer.

Listing 22: stoint()

```
int main(){
        string x = "3";
        int y = stoint(x);
        print_i(y);
}

// 3
```

obj_stringify()

The obj_stringify() function takes an Object as a parameter and returns it as a JSON-encoded string.

Listing 23: obj_stringify()

```
int main () {
  object a = {|
    string test_key : "Hello",
    int x : 123,
    object b : {| string hello : "World!" |}
    |};

  string s = obj_stringify(a);
  print_s(s);

  return 0;
}
// Outputs: { "b" : { "hello" : "World!" },"x" : 123 , "test_key" : "Hello" }
```

obj_addkey()

The obj_addkey() function takes an Object, a key name, and an integer (representing the data type to be stored), and data (of any type) as parameters and adds a key to an existing object (that must have been at least declared at compile time). The function returns what amounts to an object reference (i.e. you are not spawning a new distinct object) with the new key added.

The following table indicates which integer values need to be used with the *obj_addkey()* function.

Integer	Data Type
3	int
4	float
5	object
6	string
7	bool
8	int array
9	float array
10	string array
11	bool array

Listing 24: obj_addkey()

```
int main () {
  string array arr = ["foo", "barr", "zoo"];

  object a = {|
    string test_key : "Hello",
    int x : 123,
    object b : {| string hello : "World!" |}
    |};

  object o = obj_addkey(a, "key1", 10, arr);
  string s = obj_stringify(a);
  print_s(s);

  return 0;
}
// Outputs: { "key1" : [ "foo" , "barr" , "zoo" ] , "b" : { "hello" : "World!"
    },"x" : 123 , "test_key" : "Hello" }
```

arr_stringify()

The arr_stringify() function takes an Array as a parameter and returns it as a JSON-encoded string.

Listing 25: arr_stringify()

```
int main () {
  int array x = [1, 2, 3];
  string s = arr_stringify(x);
  print_s(s);
  return 0;
}
// Outputs: [ 1 , 2 , 3 ]
```

arr_length()

The arr_length() function takes an Array as a parameter and returns its length as an integer.

Listing 26: arr_length()

```
int main () {
  string array a = ["Hello", "Arrays!"];

  for(int i = 0; i < arr_length(a); i++) {
    print_s(a[i]);
  }

}
/* Outputs:
Hello
Arrays!
*/
```

stringcmp()

The stringcmp() function takes two strings as parameters and returns -1 if the first string is less than the second, 1 if the first string is greater than the second, and 0 if they are equal. This sorting is lexicographic.

Listing 27: stringcmp()

```
int main(){
    string a = "a";
    string animal = "animal";
    string b = "boy";
    string a_cap = "A";
    string b_cap = "B";

    comp = stringcmp(a, a_cap);
    print_s("a compared to A , boy compared to A , A compared to B");
    print_i(comp);
    comp = stringcmp(b, a_cap);
    print_i(comp);
    comp = stringcmp(a_cap, b_cap);
    print_i(comp);
    return 0;
}

/*
```

```
a compared to A , boy compared to A , A compared to B
1
1
-1
*/
```

httpget()

The httpget() function takes in a URL as a parameter, sends a GET request to the target URL and and returns the server response in a string. Port numbers for requests can be specified within the URL. Note that Gantry makes use of libcurl for HTTP functions, and requires that libcurl4 be installed on the machine.

Listing 28: http_get()

```
/*
  Returns a json string corresponding to the containers running on
  a particular Docker engine.
*/
int main() {
    string cons = httpget("http://192.168.0.9:32000/v1.19/conainers/json");
    print_s cons;
    return 0;
}
```

httppost()

The httppost() function takes a URL and a string containing the POST request body, sends a POST request to the target URL and returns the server response in a string. Port numbers for requests can be specified within the URL. Use of obj_stringify() for complex JSON data is encouraged, but for simple cases a correctly-formatted string may be more convenient.

Listing 29: http_post()

```
/*
  Returns a json object of a newly created container
  running on a particular Docker engine.
*/
int main() {

    object post = {|
        string image : "centos",
        string cmd : "[echo, hello world]"
    |};

    string url = "http://192.168.0.9:80/v1.19/containers/create";
    string body = obj_stringify(post);
    string con = http_post(url, body);
```

```
15     return 0;
16 }
```

3.2.3 Operator Precedence

The following table lists the operator precedence. Note that lower numeric values correspond to higher priority.

Precedence	Operand	Description	Associativity
1	[ ]	Brackets (array access)	Left-to-right
2	^	String Concatenation	Left-to-right
3	- !	Unary minus Logical negation	Right-to-left
4	* /	Multiplication Division	Left-to-right
5	+ -	Addition Subtraction	Left-to-Right
6	< <= > >=	Relational less-than Relational less-than or equal to Relational greater-than Relational greater-than or equal to	Left-to-right
7	== !=	Relational Equality Relational Inequality	Left-to-right
8	++ --	Postfix increment Postfix decrement	Left-to-right
9	&&	Logical AND	Left-to-right
10	||	Logical OR	Left-to-right
11	= :	Assignment Colon	Right-to-left

3.3 Statements

A statement in Gantry performs an action such as evaluation or control-flow. A statement may also contain expressions.

3.3.1 Expression-Statements

Although statements differ from expressions in that an expression represents a value and a statement performs an action, we can combine these two concepts syntactically by adding a succeeding semicolon to any expression. This produces an expression-statement wherein the value represented by the expression in this context is evaluated *only* because it is also a

statement.

Listing 30: Expression-Statements

```
42;
2 + 2;
3 - 1;
foo();
bar();
```

3.3.2 Control-Statements

The conditional expressions that dictate conditional statement evaluation or control-flow must be booleans or evaluate to booleans.

Listing 31: If-Statement

```
int main(){
    int x = 1;

    if(x == 0){
        // test fails
        print_s("If/else failed");
    } else {
        // test passes
        print_s("If/else works!");
    }
return 0;
}

// prints "If/else works!"
```

Listing 32: While-Loop

```
int main(){
    int x = 10;
    while (x < 12) {
        print_i(x);
        x = x+1;
    }
return 0;
}
// prints 10 11
```

Listing 33: For-Loop

```
int main(){
    int x = 1;

    for (x = 2; x > 0; x--){
        print_i(x);
    }

    for(int x = 0; x < 3; x++){
        print_i(x);
    }

    print_i(x);
    return 0;
}
// prints 2 1 0 1 2 3
```

3.3.3 Jump Statements

Jump statements cause unconditional jumps to other parts of the code, allowing for the transfer of control to other parts of the program.

return

Return statements end the current function and return control to the caller. Any number of return statements are allowed in a function, but each *return* must only return a single value that matches the return type of the function it is within. Note that a function of return type *null* will not support statements that *return* a value.

Listing 34: return

```
bool isHeader(string s) {
    if(s) {
        return true;
    } else {
        return false;
    }
}
```

3.3.4 Comparison Operators

Equality Operators

There are two equality operators which can be used to evaluate the content of two operands: == and ! =. Such operands must be of the same type, where valid types are *int*, *float*, *bool*, and *string*. The equality evaluation will return a *boolean* value of either *true* or *false*.

Relational Operators

There are four relational operators which can be used to compare two operands.: $<$, $>$, $\leq$, and $\geq$. Such operands must be of the same type, where valid types are *int* and *float*. The relational evaluation will return a boolean value of either *true* or *false*.

Logical Operators

There are three logical operators: && (AND), || (OR), and ! (NOT), where AND and OR evaluate two operands, and NOT evaluates a single operand. All operands must be of type *bool*. The logical evaluation will return a boolean value of either *true* or *false*.

3.3.5 Assignment Expressions

An assignment expression assigns a value to an identifier. An assignment expression must include a type and a value to which the identifier will be initialized. Valid types are *bool, int, float, string, array*, and *object*. Note that *objects* and *arrays* are aggregate types.

3.3.6 Identifiers

Global variables or identifiers must be declared in the following format *outside* of a function block and before any function declarations:

<type> <identifier>;

Listing 35: Global Identifier Declarations

```
1 int g;
2 int function () {
3     g = 42;
4 }
5 // initializes a global identifier g with a value of 42
```

Once the global variable has been declared it can be initialized within a function scope and mutated by any function in the program.

Local variables or identifiers must be declared and initialized in the following format:

<type> <identifier> = value of *type*;

Listing 36: Local Identifier Declarations

```
int function () {
  int y = 42;
}
// initializes an integer named y with a value of 42
```

See section *3.3.5* for valid types.

3.3.7 Arrays

Arrays are an aggregate data type that contain a single type (int, float, string, bool, or object) and are static (their size cannot change). Arrays are mutable and passed by reference between functions. Only single-dimensional arrays are supported.

Arrays must be declared and initialized in the following format:

<type> array <identifier> = [comma-separated values of *type*]

Listing 37: Array Declarations

```
int array exampleArray2 = [1,10,100];
// initializes an array of integers with three elements
```

Subscripts may be used to access or modify individual elements of an array. A subscript may consist of any expression that evaluates to an integer, as long as the integer is within the bounds of the array. Note that array indices start at 0 and not 1.

Listing 38: Array Subscripting

```
int array exampleArray2 = [1,10,100];
int val2 = exampleArray2[1];
// val2 is 10
exampleArray2[1] = 20;
// exampleArray2 = [1, 20, 100]
```

See section *3.3.5* for valid types.

Note: Arrays are *passed-by-reference* between functions.

3.3.8 Objects

Objects are mutable key-value data structures with polymorphic keys that can be infinitely nested.

Objects must be declared and initialized in the following format:

object <identifier> = {| <type> key : value |}

Object Declarations and Object Dot Notation

Listing 39: Object Declarations and Object Dot Notation

```
int x = 1;
object v = {|
               int i: 42,
               float f: 3.14159,
               string s: "hello world",
               bool b: true,
               int array i_a = [ "PLT", "2017" ]
               object nest = {| int y : 64 |}
           |}

int j = v.i;
// j is 42
v.s = "hello mars";
// key s in object v is "hello mars"
```

As indicated in the previous example, Object dot notation can be used to access or modify the value of a key that is a member of an object. Dot notation can also be chained if there are nested objects. *Note:* Using combined array access and object dot notation is not currently supported. You must assign an array from an object to an array before accessing the array in question.

See section *3.3.5* for valid types.

Note: Objects are *passed-by-reference* between functions.

Object Runtime and Errors

Note that the dynamic and polymorphic nature of Objects (keys can be added at runtime and can change their type when replaced) may result in runtime errors during certain assignment operations.

Listing 40: Object Runtime Errors

```
object v = {|
               int i: 1,
               int j: x,
               string k: "hello world"
           |}
int j = v.k;
// This will compile, but it will cause a runtime error
```

The previous example will compile, but result in a runtime error. While this is a contrived example, getting dynamic data from an API may result in unintended assignments like the aforementioned wherein a key has been added or replaced in an unanticipated way. Specifically, the Gantry Object runtime will attempt to make the assignment and fail because the data type "requested" from the key in question is mismatched with the primitive.

Listing 41: Object Runtime Errors

```
object v = {|
               int i: 1,
               int j: x,
               string k: "hello world"
           |}
int j = 2;
v.k = j;
int i = v.k;
print_i(i);
// This will compile and run
```

The aforementioned polymorphic operations are valid and will not result in a runtime error. That is, the Gantry Object runtime will perform *type enforcement* such that object keys can always be changed. However, the reverse operation, when primitives are assigned from keys may result in a runtime error because of dynamic-type enforcement at runtime.

3.4 Grammar

Terminals are in *italics*.

program:
    declaration-list ***eof***

declaration-list:
    declaration-list global-declaration$_{opt}$
    declaration-list function-declaration$_{opt}$

global-declaration:
    type-specifier *identifier*;

function-declaration:
    type-specifier *identifier* (function-parameter-list$_{opt}$) { statement-list }

function-parameter-list:
    type-specifier *identifier*
    function-parameter-list, type-specifier *identifier*

type-specifier:
    int
    float
    object
    string
    bool
    null
    int-array
    float-array
    object-array
    string-array
    bool-array

statement-list:
    statement-list$_{opt}$ statement

statement:
    for-statement
    if-statement
    while-statement
    jump-statement
    expression-statement
    { statement-list }

for-statement:
    for (expression ; expression ; expression) statement

if-statement:
    if (expression) statement
    if (expression) statement else statement

while-statement:
    while (expression) statement

jump-statement:

return ;
return expression ;

expression-statement:
expression;

expression-list:
expression
expression-list, expression

expression:
access-expression
array-expression
object-expression
arithmetic-expression
comparison-expression
logical-expression
string-concat-expression
assignment-expression
function-expression
identifier
constant

access-expression:
expression [expression]

array-expression:
[expression-list$_{opt}$]

object-expression:
{| expression-list$_{opt}$ |}

arithmetic-expression:
expression + expression
expression − expression
expression ∗ expression
expression / expression
expression ++
expression −−
− expression

comparison-expression:
expression < expression

expression > expression
expression ≤ *expression*
expression ≥ *expression*
expression==*expression*
expression!=*expression*

logical-expression:
expression && expression
expression || expression
!expression

string-concat-expression:
expression ^ expression

assignment-expression:
expression = expression
type-specifier *identifier* = expression
type-specifier *identifier* : expression

function-expression:
identifier (expression-list$_{opt}$)

constant:
true
false
literal

literal:
int-literal
float-literal
string-literal

4 Project Plan

4.1 Process

4.1.1 Planning

During our first meeting in early September, we established that we would meet twice each week for at least two hours at a time. In early October, we added a third weekly meeting of roughly thirty minutes with our TA, Kai-Zhan, to make sure we were making progress and to bounce ideas off of him. The TAs are an excellent resource, and they can be instrumental

in steering you away from impending doom before you get there. We also worked as a team after this meeting, so we worked together for 8-10 hours per week in addition to the work we did individually.

4.1.2 Specification

The team originally met and brainstormed project ideas. We wanted a language that had practical applications and specific use cases, and we also wanted to set reasonable expectations for what we would deliver.

Our team agreed on a language that would interface with the Docker API, which would require sending http-get and http-post requests to the API. Since these get and post requests would contain and obtain strings of JSON data, we decided that it would be useful for our language to have JSON-like data types. Our initial pseudocode was written in a Python-like language, which does not involve explicitly declaring types. However, we thought it would be interesting to take in these JSON data types and represent them in our language as statically-typed, JSON-like types. Looking at the Docker API and the JSON documentation helped inform the decisions about which types would need to be implemented. We settled on strings, numbers (integers and floats), objects, arrays, and boolean values. Our original language was going to be Docker-specific (for example, with built-in container "objects"). However, we realized that it would be ideal to deliver the functionality of a more general-purpose language that could communicate with any JSON API.

4.1.3 Development

As mentioned in Section 4.4, we set up an Ubuntu 16.04 LTS VM on Google Compute Engine early on in the semester. This served as a homogeneous development environment that meant one person could configure the development environment and provision user accounts for each team member, saving both time and potential headaches related to versioning issues, dependencies, etc. Each person using an identical development environment is a critical best-practice.

Generally, we all coded separately by SSH'ing to the Cloud VM and using VIM. We would then utilize branches in Git for specific features that another member could then pull and work on in tandem if someone needed another set of eyes on a particularly challenging problem.

4.1.4 Testing

We wrote the first iteration of our test suite in October covering checks for more than 40 cases. In retrospect, many of our test cases put the cart before the horse, so to speak, in that we spent time writing tests before features existed. This was a bit naive on our part

despite our intent being to parallelize as much as possible. Ultimately, the best approach was to have each person write their own tests as they implemented a particular feature as a discrete unit, and subsequently integrate that feature with the other features of the language to get better code/test coverage. Professor Edwards stresses that unit testing is not particularly useful for a compiler. We found this to be true, but would suggest that when you are implementing a feature and debugging LLVM IR, it is best and sometimes imperative to start with a simple unit test of the feature you are implementing to reduce the verbosity of your IR output. After you are confident it is working as a unit and the LLVM IR and behavior appear correct in isolation, the test should be expanded for code coverage and richer integration testing before merging into the repository. For example, we had both Arrays and Objects working, but testing revealed Arrays being mishandled when assigned inside of Object literal expressions only after a fully-integrated test.

4.2 Programming Style Guide

4.2.1 General Guidelines

- Each file in our project (with the exception of tests) should have a header that lists the module, the Author, and the Contributors.
- Indentation should be consistent on a per language, per file, basis.
- Code merged into the master branch should compile without warnings.
- Block comments go above the sections of code they reference.
- Block comments should be independent to the line of code they precede.

4.2.2 OCaml

- Indentation for .ml files should use two spaces. Indentation should take place wherever it logically makes sense.
- Indentation for .mly files should be four spaces.
- Nested let expression should be indented.
- No incomplete pattern-matching.
- Code should be well-commented.

4.2.3 C

- A block comment should appear before each function.
- Start curly braces on the same line as the function declaration.

4.2.4 Gantry

- Code blocks should be commented using block commenting.
- Start curly braces on the same line as the function declaration.
- Tests should end with the test output in a comment (see Test Plan).

4.3 Project Timeline

Date	Milestone
September 8	Introductory meeting
September 25	Submitted proposal
October 6	Wrote context-free grammar
October 18	Submitted LRM
October 20	Wrote first test suite
October 22	Scanner, Parser, AST complete
October 25	Started code generation
November 3	Ran "Hello World"
November 8	Finished test suite
November 26	Semantic Checking complete
December 15	Code complete
December 19	Presentation

4.3.1 Roles and Responsibilities

Roles and responsibilities were fluid in that many of us worked on items outside of our immediate scope as defined by the roles.

Audrey Copeland: Tester

Audrey worked on the Test Suite, Code Generation, Semantic checking, the String Library and Stringify functions in C, documentation, and the Grammar.

Walter Meyer: System Architect

Walter wrote the Scanner, Parser and AST, and worked on the Grammar, Code Generation, Testing Suite Automation, the Object Runtime Library in C, and documentation.

Rizwan Syed: Manager

Rizwan was the Project Manager and worked on the Grammar, Testing, and Documentation.

Taimur Samee: Language Guru

Taimur was the Language Guru and worked on the Grammar, Code Generation, and wrote the HTTP Library in C in addition to other C library functions.

4.4 Software Development Environment

We used a shared Ubuntu 16.04 LTS virtual machine hosted on Google Compute Engine[4] (using free education credits from Google). We chose to do this early on in order to make sure everyone was using the same versions of all of the prerequisite software to simplify and speed-up development time. We used Git for source code version control and used a private GitHub[5] repository (using free education credits from GitHub). We also used Travis CI[6] for continuous integration testing, where our entire source tree and test suite would be run by Travis CI in a clean Ubuntu 14.04 VM upon all pushes and pull requests to our GitHub repository in order to ensure failing builds did not go unnoticed. We used VIM as our primary text editor.

[4]https://cloud.google.com/compute/
[5]https://github.com/
[6]https://travis-ci.com/

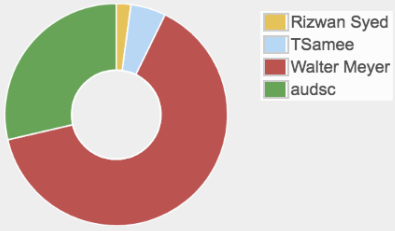
4.5 Project Log

Git Statistics

Statistical information for the repository 'plt' was gathered on 2017/12/13.
The output has been generated by gitinspector 0.5.0dev. The statistical analysis tool for git repositories.

The following historical commit information, by author, was found.

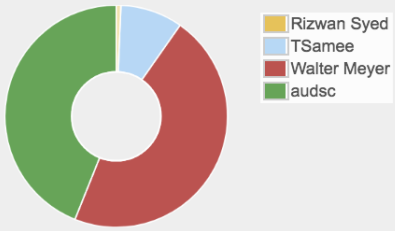
Author ∨	Commits	Insertions	Deletions	% of changes
audsc	96	2477	868	28.63
Rizwan Syed	8	164	83	2.11
TSamee	26	414	183	5.11
Walter Meyer	91	4305	3191	64.15



Below are the number of rows from each author that have survived and are still intact in the current revision.

Author ∨	Rows	Stability	Age	% in comments
audsc	1238	50.0	3.5	8.56
Rizwan Syed	19	11.6	6.8	0.00
TSamee	255	61.6	3.2	7.06
Walter Meyer	1305	30.3	4.1	15.71

Hide minor authors (1) ∧



The following history timeline has been gathered from the repository.

Author	2017W39	2017W40	2017W41	2017W42	2017W43	2017W44	2017W45	2017W46
Rizwan Syed					.			
TSamee					.	.		.
Walter Meyer								
audsc			.					
Modified Rows:	58	2097	2442	250	1176	888	549	1333

Author	2017W47	2017W48	2017W49	2017W50
TSamee				
Walter Meyer				
audsc				
Modified Rows:	930	882	709	371

Git Log

```
commit 21c693aced5633b2f79d9b0ee95a0ec256bee66b
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Dec 18 15:29:26 2017 -0500

    Added codegen while bug fix

commit 38626b79611bc1634b49ae15fc0f647d2112eff4
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Dec 18 15:28:57 2017 -0500

    Fixed while bug caused by lack of builder context in object lookup

commit 11912ce4c8410da9923c5309613a2e54ab731002
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Dec 16 21:06:27 2017 -0500

    fixed test bug

commit 06f1938d321f3eda9f30cf94244dd77a5df5fe71
Merge: 9d9bd86 b02fc3e
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Dec 16 17:59:40 2017 -0500

    Merge branch 'master' of https://github.com/waltermeyer/plt

commit 9d9bd866971e21ea15ccc0bd2d08eb28adde19aa
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Dec 16 17:59:08 2017 -0500

    Fixed comp warning and fail test

commit b02fc3e304ec8844064244e76c649ef86e4223a5
Merge: 31267c6 840c7fa
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Dec 16 17:57:35 2017 -0500

    Merge branch 'master' of github.com:waltermeyer/plt

commit 31267c61d20e1ab962f32ee7470d32225482c8fe
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Dec 16 17:57:05 2017 -0500

    Get deck id by searching for the key

commit 840c7fa2bd0908fce1b2aedd7a3fdfa6400f40a3
Merge: f671aa6 2109892
```

```
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Dec 16 17:11:48 2017 -0500

    Merge branch 'master' of https://github.com/waltermeyer/plt

commit f671aa6834c734e79ccc2684a10a034bb784fe4a
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Dec 16 17:10:19 2017 -0500

    Fix to semantic for object key value type checking

commit 2109892d265e199e8667f0c4ecb7da386d2d8ea9
Merge: 02e9772 b4308dd
Author: audsc <audsc@users.noreply.github.com>
Date: Sat Dec 16 15:27:50 2017 -0500

    Merge pull request #104 from waltermeyer/f_test

    Add fail tests for custom checks in semantic

commit b4308dd77bc96464e0f895fdf867ed48341fa4da
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Dec 16 15:08:23 2017 -0500

    Add fail tests for custom checks in semantic

commit 02e9772b6ab4c3516f57f1adf7352cf1d451c3f1
Merge: f756803 cf946d6
Author: audsc <audsc@users.noreply.github.com>
Date: Sat Dec 16 11:50:49 2017 -0500

    Merge pull request #103 from waltermeyer/blackjack_update

    Blackjack update

commit cf946d6332acad1b169ea3ee30e2d95ebe8e92b7
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Dec 16 11:39:36 2017 -0500

    Blackjack game with objects, aces as 1 or 11, special blackjack hand.

commit f756803e6d894cd6af9510fa73ee4d1e2dfe4dbf
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Dec 15 13:20:51 2017 -0500

    Updated Docker demo apps.

commit e6438b7b2587024999d6ec6297b7d3b50d5f6bb6
```

```
Merge: b40c53f 696980b
Author: audsc <audsc@users.noreply.github.com>
Date: Fri Dec 15 11:55:49 2017 -0500

    Merge pull request #102 from waltermeyer/fix_obj_stringify

    Fix printing of nested objects

commit 696980ba0bee98242dcdfdf24bac3be9ba343f87
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Dec 15 11:33:28 2017 -0500

    Fix printing of nested objects

commit 0b2217acd96412cd5682c6dae19a1f2725fd4e0d
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Dec 15 10:25:17 2017 -0500

    Blackjack updated w object assignment and access in control flow errors. Also
        added check to semant and cast to eq, neq.

commit 0717e5a4aa8b29fddebf8d4649547bcb3ff2455d
Merge: bc08bb8 b40c53f
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Dec 15 09:10:37 2017 -0500

    Merge branch 'master' of github.com:waltermeyer/plt

commit bc08bb8e183ea9a5f46e2deed0d58c90a3c8077a
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Dec 15 09:10:35 2017 -0500

    updating blackjack

commit b40c53f193da6015a1cac11dcc9cb3e857f6c8d6
Merge: 8bf27cd 9c6a48d
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Dec 14 23:48:34 2017 -0500

    Merge branch 'master' of https://github.com/waltermeyer/plt

commit 8bf27cd76897f83c5761e1f6a439d0dcb2780869
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Dec 14 23:47:00 2017 -0500

    Fixes to while loops and added missing i1 conversions. Also updated demo.

commit 9c6a48d811918b9ba0397e4106bd8ae31032128d
```

```
Merge: c3015b5 258682d
Author: audsc <audsc@users.noreply.github.com>
Date: Thu Dec 14 20:22:57 2017 -0500

    Merge pull request #101 from waltermeyer/fixbool

    fix bool and add tests

commit 258682d44db101ce92af68335995cf9fa09dd2fd
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Dec 14 19:25:28 2017 -0500

    fix bool and add tests

commit c3015b5ad613c1cb07280c61f8049bce3b5860f0
Merge: c5884fd 3bc5870
Author: audsc <audsc@users.noreply.github.com>
Date: Thu Dec 14 18:16:54 2017 -0500

    Merge pull request #100 from waltermeyer/blackjack

    Blackjack

commit c5884fd296e3cf49b9554733e44dbcbd839945a4
Merge: 3612d4b 0ea2113
Author: audsc <audsc@users.noreply.github.com>
Date: Thu Dec 14 12:33:38 2017 -0500

    Merge pull request #98 from waltermeyer/redeclare_semant

    Check that objects aren't declared twice because of issues with memor

commit 3bc58704a352973be2d1773147c3c995365ac2ae
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Dec 14 12:23:23 2017 -0500

    Simple black jack demo game. May have too many print statements

commit 0ea211333a525a09528665bff6443e91088480ae
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Dec 14 08:52:54 2017 -0500

    Check that objects aren't declared twice because of issues with memory when
        object is redeclared and on RHS. Other types can be declared multiple
        times.

    Fail test not test
```

```
    delete test

commit c216d48db1f63107f4b704d4843781b22807ae74
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Dec 13 19:48:18 2017 -0500

    add input function to make black jack app

commit 3612d4b9cf9615372214c8ff0df8cb59921b14ef
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Dec 13 13:23:55 2017 -0500

    get rid of string_eq

commit 2f8253421c615213ee8b8eed87f78dd69e40dbdc
Merge: 3739f01 75c9603
Author: audsc <audsc@users.noreply.github.com>
Date: Wed Dec 13 13:21:22 2017 -0500

    Merge pull request #97 from waltermeyer/obj_array_stringify

    Add obj array stringify and get rid of irrelevant TODOs in semant and

commit 3739f01626e39a9efa6ba0ede1cecdc7415d7071
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Dec 13 12:22:02 2017 -0500

    Removed string_concat from semant, not a user-facing function

commit 3704414232cf7355f0d7463e1f83c43741a262eb
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 21:14:51 2017 -0500

    Adding Docker demo application that can be added to and fixed bug in
        compilation script

commit 0ebbeee49e5f171e6a4cc5a90930cf67ff3af757
Merge: 71dca87 9157cdc
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 14:35:17 2017 -0500

    Merge branch 'master' of https://github.com/waltermeyer/plt

commit 71dca870a82e65dc9b4e575f7ae2188bc139e1dd
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 14:34:49 2017 -0500

    Module attributions and repo cleanup.
```

```
commit 9157cdcd1c92d77e3c32d447160126ac73d36f86
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Dec 12 14:26:10 2017 -0500

    implement gcd in gantry

commit d67cf2c0dc53fd86865eb52e1f7a264dcdd37562
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 14:02:54 2017 -0500

    Cleaned out old/outdated tests

commit b38b560141212969f39e145926daf84efb0d4b8c
Merge: fc0409b 4714397
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 13:57:34 2017 -0500

    Merge pull request #96 from waltermeyer/stringify_array

    Stringify array

commit 47143975b2cb2e9acd663fab341afe4ed4b05f1d (origin/stringify_array)
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Dec 12 13:45:19 2017 -0500

    Test that arrays in objects of all types work at compile time

commit ac59475fabbf8eab664e36d62b8a92fa07f810b2
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 12:47:54 2017 -0500

    Updated test_arr_stringify.gty

commit 5da2de5b3fc0525fdd4bbbe3ef695ae59020986b
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 12:26:31 2017 -0500

    Changes to codegen and reverting back gantrylib_obj.c to attempt fixing arrays
        in objects

commit fc0409b04b2973d76bdcaaec56d22140dadd478e
Merge: b05c894 c67cacd
Author: TSamee <taimur.samee@gmail.com>
Date: Tue Dec 12 11:36:16 2017 -0500

    Merge pull request #95 from waltermeyer/logical_ops
```

```
    Logical ops

commit 6ad02acc6bf390be4334a77a6a3bb55240b6ad23
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 11:26:07 2017 -0500

    Changed struct packing in LLVM, may have introduced regression in C lib that
        needs addressing as a result

commit ec9417ed24682094757076f24c80d9d03b81f96a
Merge: dcaf720 35c49a7
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 10:55:01 2017 -0500

    Merge branch 'master' into stringify_array

commit dcaf720daeaf39ea2dc2457a85ce71a915c71dc9
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 12 10:54:59 2017 -0500

    Changes to obj_getkey for arrays in objects

commit b05c894dabd41d5d6a523cf112d9408339882298
Merge: 35c49a7 7da2d28
Author: audsc <audsc@users.noreply.github.com>
Date: Sun Dec 10 16:26:18 2017 -0500

    Merge pull request #94 from waltermeyer/stringify_array

    Stringify array. This only works with arrays that are added to objects at
        runtime.

commit c67cacd319acf512393fcbfe53300bf43801ca49 (origin/logical_ops)
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Dec 10 06:36:33 2017 -0500

    Removed result casting for logical op build calls when e1 = i8, since
    LLVM gives us i8 already; updated mixed op test with a conditional that
    should fail if precedence is right; removed old AND test

commit 56106256f79b639fd901153af98a1621d8ce680f
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Dec 10 06:21:58 2017 -0500

    Reverted semant to old config for logical ops; modified codegen to cast from
        i8 to i1
    and back correctly. Not sure if we need so many casts though
```

```
commit 4fdfd1b10c9ce82f172aa833c9ae883a528e4cc0
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Dec 10 05:11:12 2017 -0500

    Updated codegen; logical ops still not working

commit 7da2d28cc8c5928bac0d8a09cbdb346da938e46b
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Dec 9 17:39:13 2017 -0500

    Adding array to object using obj_addkey at runtime works for all array types

commit ec4459728efd73c49411f175820ff9da04bd355b
Author: TSamee <taimur.samee@gmail.com>
Date: Fri Dec 8 20:08:09 2017 -0500

    Trying to change Binop to case i1 up to i8 for Booleans

commit b60e238f9b758a5d7463d1cae0d46a6db02126bc
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Dec 7 20:15:06 2017 -0500

    changed for loops in gantrylib_obj.c to be < C99 compatiblt

commit bae0629914adde06db707ea6abdbad5df648abd6
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Dec 7 20:01:24 2017 -0500

    Changed array initial index in codegen, printing arrays works now.

commit ee26b9c3eca982629412d52fe7366b3db7a19cbf
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Dec 7 19:31:49 2017 -0500

    Fixed array index bug

commit 4d73e64ab08da3abc2dd7df4e3cca1b5fb58d35f
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Dec 7 17:55:22 2017 -0500

    array of strings can be put as string but arrays in objects don't work rn

commit e892465604e7d31a9445f3c7f225569c7c89b3fe
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Dec 7 09:30:07 2017 -0500

    int, float, bool works. string is not currently working
```

```
commit 5a90c49df322f668abdeb4a0db484c9d7d4e51ab
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Dec 7 07:46:06 2017 -0500

    Printing array for string and floats works. Need to get rid of extra length
        argument, add printing for string and bool , and add array printing to the
         obj_stringify function

commit 35c49a72f1817267be23cbf73d6eed17b032d21b
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Dec 6 20:48:16 2017 -0500

    Renamed printb.c to gantrylib_printb.c

commit 2d5a1d03794dd7330ff41751b0abe63400376673
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Dec 6 20:44:16 2017 -0500

    Added README for compiler and a compiler script that handles linking, etc. for
         the user.

commit 95108de0868581da5487cf84a46a1f10ba92af6a
Merge: a5f4973 20b7fa3
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 5 23:14:59 2017 -0500

    Merge pull request #93 from waltermeyer/arr_length

    Arr length

commit 20b7fa3afaab01e5b36e373e429c0f1ae54d22e6 (origin/arr_length)
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Dec 5 23:09:06 2017 -0500

    Added ability to assign all array literals to/from object keys and add
        arr_length function

commit d2e35f88e1b28bf6ca337ce861c382bf0edcf17f
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Dec 5 18:50:43 2017 -0500

    add arrs to struct

commit a5f4973ef72c00ffa8342c8841b6ea529cd9df44
Merge: 18da4a8 bd5bb5c
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Dec 4 19:41:29 2017 -0500
```

```
    Merge pull request #87 from waltermeyer/arrays2

    Arrays as structs with length field. More needs to be done

commit bd5bb5c66222b6f650aa9278f05321e7be9f38dc (origin/arrays2)
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Dec 4 19:12:41 2017 -0500

    Added array type field to array struct

commit b43338181a9ba6d16eb132019d34ae88f90f7e41
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Dec 4 18:51:45 2017 -0500

    Added object array 2 test

commit 18da4a87a93fb832173a70be5f1d4acbf7314692
Merge: faf38c2 f788506
Author: audsc <audsc@users.noreply.github.com>
Date: Mon Dec 4 18:51:27 2017 -0500

    Merge pull request #88 from waltermeyer/obj_add_key

    Obj add key

commit f788506d1c7c43c21879e5b32f78c91db1a9f0a0 (origin/obj_add_key)
Merge: 8bf0809 faf38c2
Author: audsc <audsc@users.noreply.github.com>
Date: Mon Dec 4 18:39:04 2017 -0500

    Merge branch 'master' into obj_add_key

commit 8bf08090fc71d4c89062599228c03eaa09fe07f8
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Dec 4 18:36:31 2017 -0500

    Fixed the test and made it better

commit 2288f029fc1bd8e0ac0c2df70df6a9879c7c752f
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Dec 4 18:27:48 2017 -0500

    Fixed adding string and float values.

commit 9c2a53e3d962f32bd73cd41b6f9f221469a8b988
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Dec 4 14:03:22 2017 -0500
```

```
    Arrays as structs with length field. More needs to be done

commit 84e2ccebb1d88b1fd69c1fdb9f480551fde88189
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Dec 3 18:43:57 2017 -0500

    Adding an integer or boolean value works. Float is weird, string is weird.

commit 8631a7c754e775adaf1644d9d8e4559bec8f64c7
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Dec 3 18:23:57 2017 -0500

    Function check in semant is fixed so it doesn't care what type of value is
        passed in. However, strings don't work when passed in as value right now,
        prints garbage

commit 5a1850be2a28d491adba44660b70195ad59fbf1f
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Dec 3 17:44:13 2017 -0500

    Adding a key value pair works if the value is an integer. Need to figure out
        how to add multiple value types to function formals in Semant

commit 19ec219b2ca85392bccaadcf9c185bd0e212aae7
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Dec 3 17:15:32 2017 -0500

    Function obj_addkey is set up in gantrylib_obj.c, however there is an error in
         the casting in codegen for that function. Additionally, not sure how to
        indicate void * parameter time in semantic

commit faf38c2653195f8b3bc501fd901db9e06c9fdd47
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Dec 3 16:15:11 2017 -0500

    Fixed typo error in codegen

commit 2e03e25b62ad8e75fea4b0355ebce348a21f99db
Merge: efb4371 2b2f6f9
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Dec 3 16:11:51 2017 -0500

    Merge branch 'master' of https://github.com/waltermeyer/plt

commit efb4371fdc4f45f201bb238491790b5a9542a7ab
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Dec 3 16:11:41 2017 -0500
```

```
    semant.ml compilation fixes

commit 2b2f6f9abaef90aa5c899ef082885d6c66e53b4a
Merge: b4e2d3e 81ac452
Author: audsc <audsc@users.noreply.github.com>
Date: Fri Dec 1 19:20:26 2017 -0500

    Merge pull request #84 from waltermeyer/more_fail_tests

    Add some more tests

commit cddce6230c1a2cad9dc34f8296a1ce8c5f58aac7
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Dec 1 19:01:34 2017 -0500

    Fixed all compilation warnings and removed continue and break.

commit b4e2d3e4e35f13e84a19fbda97220a2e42a46252
Merge: 9a31f5a ce4dfbf
Author: audsc <audsc@users.noreply.github.com>
Date: Fri Dec 1 19:01:01 2017 -0500

    Merge pull request #85 from waltermeyer/stringify

    Stringify

commit 81ac452c605a83d6de563ae9048699f231d49c17 (origin/more_fail_tests)
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Dec 1 18:28:16 2017 -0500

    Add some more tests

commit ce4dfbfa08106aeb0c5b6dac50eda50519979e50
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Dec 1 18:47:35 2017 -0500

    Remove escape characters in json object to string, use in docker app, and fix
        test to reflect change

commit 71ec355f37025fe960063071cec5a3ccbca4a8cf
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Nov 30 13:50:36 2017 -0500

    Fixed output keys and spacing

commit 660923b3e8d60f6ccb009da2aed5e981b1742723
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Nov 30 13:34:14 2017 -0500
```

```
    Works to stringify multiple objects. Values can be any value types in our lang
        except array

commit ea6401d1f12ee322342720b2877216797e78e600
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Nov 30 12:42:09 2017 -0500

    Object prints correctly, however if more objects are stringified, the old
        string buffer of the old object will be used and not cleared so need to
        initialize as void and make sure buff doesn't grow past length

commit 7bdf737e14391c954e53d429583be781dc65083c
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Nov 30 12:21:35 2017 -0500

    This prints object successfully and removes the comma and paren, but memory
        errors

commit 976cabd750154eee29c8ca89ff73e11faa14ca5a
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Nov 29 19:59:14 2017 -0500

    Correctly outputs object passed in in test_object_stringify. however, there
        are commas at the end of every object and at the end of every value which
        needs to be fixed. Additionally, more testing is needed

commit 9a31f5ab67e1ddfd9c6debbcedc0a2502b296fc1
Merge: 95ce9b8 38a409d
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Nov 29 19:53:53 2017 -0500

    Merge branch 'master' of https://github.com/waltermeyer/plt

commit 95ce9b860e041600cb36afc32f00174c9732df4b
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Nov 29 19:53:37 2017 -0500

    Object keys can be passed to functions and have their types derived based on
        the functions formals. Also, beginning arrays in objects.

commit f71e6ae02064611e0a0c4ec393c130a89e0de101
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Nov 29 19:03:05 2017 -0500

    Correctly traverses object in main however, have not done other tests. Also,
        does not pretty print, not sure if we want it to. Also, handles null key
        by printing junk.
```

```
commit 24d6b5ada281a7dcf30643a9c5c109ba1cfb9601
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Nov 29 16:02:29 2017 -0500

    Failing to grow the buffer

commit 38a409da3b288fe60991f198a317ab1fe9edce07
Merge: f83fc95 0ef77b2
Author: TSamee <taimur.samee@gmail.com>
Date: Tue Nov 28 22:08:55 2017 -0500

    Merge pull request #82 from waltermeyer/float_operations

    Float operations

commit 0ef77b2330bc79b0b2f143c0dd2c093e0b185fbc
Author: TSamee <taimur.samee@gmail.com>
Date: Tue Nov 28 20:15:44 2017 -0500

    Added tests for float equality, comparison and arithmetic

commit 629bda365f66867b6a4028e11e7d11184a2811d9
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Nov 28 20:08:33 2017 -0500

    doesn't work

commit 32c7660401a279fceb7eac99236be2ca9cbbdb08
Author: TSamee <taimur.samee@gmail.com>
Date: Tue Nov 28 19:50:06 2017 -0500

    Added floating-point operators to Binop in codegen and semant,
    test for adding floats

commit f83fc954e5e0602c6f05e1ffb0f9ce982e76927b
Merge: f876b6b 2f22ee2
Author: TSamee <taimur.samee@gmail.com>
Date: Tue Nov 28 12:32:15 2017 -0500

    Merge pull request #77 from waltermeyer/sleep_wrapper

    Added nap(), a wrapper function for sleep() in C

commit f876b6b6d6b23301c0c03fd637a6f9ff09aa7729
Merge: 7e9c1ec d25b3f7
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Nov 28 12:31:27 2017 -0500
```

```
    Merge branch 'master' of https://github.com/waltermeyer/plt

commit 7e9c1ec6f823346141e4559e934f7183d951810f
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Nov 28 12:30:44 2017 -0500

    Primitives can be declared from object keys

commit d25b3f74c9ec7bceeb8cb0e53a5b943eb18fb8fc
Merge: d1daf43 3724cf4
Author: audsc <audsc@users.noreply.github.com>
Date: Mon Nov 27 21:23:07 2017 -0500

    Merge pull request #78 from waltermeyer/boolean_fix

    Boolean fix

commit 3724cf49b2e183c739561192bd0569a0ccb36812 (origin/boolean_fix)
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 27 20:33:03 2017 -0500

    boolean in if and while will work, cast to an i1

commit 2f22ee2232373be9c835836fc491fb51d0d5ea82
Author: TSamee <taimur.samee@gmail.com>
Date: Mon Nov 27 20:24:30 2017 -0500

    Added nap(), a wrapper function for sleep() in C

commit 523594c7b1bdfcd4c1a58c7a23047e260388bd99
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 27 19:46:45 2017 -0500

    [skip ci] casting to i1 type in the not operator works, but casting back to i8
        in order to do not and then a comparison does not work.

commit d1daf43fc58a2576bd2b0c663b237557c6339403
Merge: 0664c09 e621ede
Author: TSamee <taimur.samee@gmail.com>
Date: Mon Nov 27 19:40:06 2017 -0500

    Merge pull request #76 from waltermeyer/print_bool

    Added printing function for booleans in C

commit 0664c09a69e00df6c674f6eeee3bb5f00d5c8e19
Merge: 2111321 3c2bb60
```

```
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Nov 27 19:37:00 2017 -0500

    Merge branch 'master' of https://github.com/waltermeyer/plt

commit 211132101e11d1c5ced3b668b3d954e44e3b4732
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Nov 27 19:36:28 2017 -0500

    Arrays of our primitive types working, but not with objects that is TODO.

commit 3c2bb60fb6c5b3219901a30e80463ebf051ff60a
Merge: e2518ac a8426fe
Author: audsc <audsc@users.noreply.github.com>
Date: Mon Nov 27 19:33:36 2017 -0500

    Merge pull request #74 from waltermeyer/fail_tests

    Fail tests

commit e621ede6b7f04a5a6647c0328c0f12c98ef2cbb8
Author: TSamee <taimur.samee@gmail.com>
Date: Mon Nov 27 19:31:02 2017 -0500

    Wrote printing function for booleans in C, modified semant and Makefile

commit e2518ac50778a8a64194b0f584bc6cb3614bb93e
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 27 19:18:22 2017 -0500

    testing basic math functions with positive and negative integers

commit a8426fe35fac335962d8b342b3f349a157f2f9af (origin/fail_tests)
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Nov 26 09:16:52 2017 -0500

    Adding fail tests. Need to figure out how to link files in fail test... Having
        trouble incorperating this into Edwards script

commit f02017e232d3927c9a7d59211c0838f7b3d493a0
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Nov 25 18:40:56 2017 -0500

    test all doesn't work the way we need it to wrt built in functions

commit 44bf2816bc6128dff8dd2a0dbfa765d5bbb80f94
Merge: 4b75aae 03de626
Author: Walter Meyer <wgmeyer@gmail.com>
```

```
Date: Sat Nov 25 18:32:18 2017 -0500

    Arrays WIP. Semantic checking for Array literal implemented as is codegen.
        Needs testing with ArrAcc expressions.

commit 03de6268e997b08bcd39742811af3c040d6c43ae (origin/array)
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Nov 25 18:21:08 2017 -0500

    Array implementation WIP

commit 4b75aae2f7814ea50f20088edc859e1052b834b2
Merge: d18ec0f 3488790
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Nov 25 11:02:23 2017 -0500

    Polymorphic object keys along with new tests

    Object lib

commit 3488790df50f5efb5c08b9228765968a3a82f169 (origin/object_lib)
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Nov 25 10:48:32 2017 -0500

    More Object changes. Primitives can be assigned from object keys. Nested
        objects can be assigned to existing object keys.

commit 1e8fffaa69b483f4feb0ec383ad957baf93bb7dc
Merge: 9166524 d18ec0f
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Nov 25 08:39:43 2017 -0500

    Merge branch 'master' into object_lib

commit 91665245c83d649d964390ac71f3f057583f4a0c
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 24 16:02:25 2017 -0500

    Objects2 test for object polymorphism

commit 21ec9f64d06e7d74bde251f2abe8f96a5ac18c9f
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 24 16:02:00 2017 -0500

    Objects partially polymporhic in that keys can be assigned from primitives.
        Still need to add code to pull primitives from object when on RHS.

commit 1320a0e27d0374803b640faaacb04bca5cf7bd6f
```

```
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 24 13:33:38 2017 -0500

    WIP polymorphic object keys

commit d18ec0f9c72b277578de3981c8dd2be264155be6
Merge: d8f9b6d ac58f43
Author: audsc <audsc@users.noreply.github.com>
Date: Thu Nov 23 10:43:27 2017 -0500

    Merge pull request #72 from waltermeyer/fix_semant

    Fixes to semant

commit ac58f43fc46aeb38c22e72b15fc36de591b85ddb
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Nov 23 10:32:16 2017 -0500

    Declaring anything with a . should fail.

commit f34f54ae920b330461d4d4c1743e2bcda8cb78c8
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Nov 23 10:02:28 2017 -0500

    Doesn't work, multiple commits that don't work for reference.

commit 56f9095647c1bbe7185a7f0fbabca2b5ccdddc4e
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Nov 23 03:09:52 2017 -0500

    Add check to ID so that if it contains a '.' then the type returned is object,
         to avoid calling 'type_of_identifier' which has no way of knowing what
        the type is if the object has been passed to a new function because the
        object keys are not stored in the symbol table

commit f43086df741e304d2529b0a5a89dfc7374e2a0b0
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Nov 23 02:53:28 2017 -0500

    Check whether variables contain a . and don't check them in function calls if
        they do. This doesn't fix the failure on o.x in the object test

commit db43bb7b8b7c8894f92b9c1682bbfae6898161ea
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Nov 23 02:22:15 2017 -0500

    this doesn't work
```

```
commit eedae82a02b26179f6fccaa2e68742bc85bcbcdf
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Nov 22 17:12:46 2017 -0500

    More hacking with object runtime

commit 8d6cf3ea5f23e7a843fa4328cd72c7bcdbc46170
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Nov 22 15:23:12 2017 -0500

    WIP object runtime library. Do not use yet.

commit 10dca5d862b6bf303e17ad57ad1840f329b78fa3
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Nov 22 14:25:19 2017 -0500

    Clear symbol table at top of each function

commit 9b8acecf2b5d0af0e469dfbb2823dd308287f9ac
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Nov 21 17:24:34 2017 -0500

    Going in circles.

commit 02ee9caa511e53ca48f94fa814362aa702b01763
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Nov 22 13:39:55 2017 -0500

    Add check to make sure function parameters match specified function parameters
         based on microc. However this causes a problem with objects. The object
        keys are not in the symbol table in the new function scope. Will work on
        this more.

commit d8f9b6de3875f2ecddcb6b2a4efbce30422bbcdb
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Nov 21 23:33:30 2017 -0500

    Update codegen.ml

commit b10f940423cf92b851e3aa08592da009268c261f
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Nov 21 17:02:56 2017 -0500

    This doesn't work yet, will have to continue fiddling to get booleans to work

commit fcfb5e7a429c06b50f8f4f154d32a8e41853ac36
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Nov 21 15:44:10 2017 -0500
```

```
    Fixed httpost in semant.ml return type

commit 83d32ca36bb8e53ad0feccb935ef4db7a5a6a151
Merge: f0dfa86 394c98f
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Nov 21 14:57:00 2017 -0500

    Merge branch 'master' of https://github.com/waltermeyer/plt

commit f0dfa86e2660e810930e72ea78b6b600537ec46e
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Nov 21 14:56:36 2017 -0500

    Changed httplib to use application/json headers. Added docker.gty program.
        Modified object test

commit 394c98fb944ffa8682db8a3d9a7ae6d697dc0d67
Merge: 7ce3b5d d1e9578
Author: audsc <audsc@users.noreply.github.com>
Date: Tue Nov 21 11:51:45 2017 -0500

    Merge pull request #70 from waltermeyer/str_comp_ops

    Str comp ops

commit d1e95782c059e9d4f071bf5636615c8fb65dab8b
Merge: 8fdf46e 7ce3b5d
Author: audsc <audsc@users.noreply.github.com>
Date: Tue Nov 21 11:36:52 2017 -0500

    Merge branch 'master' into str_comp_ops

commit 7ce3b5d8182f12d9fe25068dec00d676873bb1f7
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Nov 21 11:32:19 2017 -0500

    Add parenthesis

commit 8fdf46e3432dcb81b893567780bee80bfa177a92
Merge: 538c104 c3059c4
Author: audsc <audsc@users.noreply.github.com>
Date: Tue Nov 21 11:30:41 2017 -0500

    Merge branch 'master' into str_comp_ops

commit c3059c4649aaf4ad1790432b4a92a10cb5e487fb
Merge: 5634f78 7d76fb7
```

```
Author: audsc <audsc@users.noreply.github.com>
Date: Tue Nov 21 11:27:46 2017 -0500

    Merge pull request #62 from waltermeyer/string_length

    Add string length function

commit 538c10416a133b4ea18ff7c0d9266a6932830f35
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Nov 21 10:16:15 2017 -0500

    Add stringeq and have Neq call streq in codegen. Neq was working before too,
       not sure why

commit efd795e7ff362091d1c3451b8169b096b8928abe
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Nov 20 20:36:42 2017 -0500

    Added str comparison code, still WIP

commit 5946e0788fd17a0162561fad99f4834b6db98264
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 20 19:46:35 2017 -0500

    String Compare works but need to add String not equal and clean up

commit 5634f7803ba0ee5ee93e7c7b62cc18b6c87e8346
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Nov 20 17:49:24 2017 -0500

    Fixed to object printing and test

commit e2f27e30f9bb24178dde555fd8d2c264b509b0d8
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Nov 20 15:56:13 2017 -0500

    Resolved conflicts, added print_k to semant.ml

commit 2e7aad7ef510bd9e1a2a05413155d168a0d6a32e
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Nov 20 15:46:13 2017 -0500

    Objects part2 (#64)

    * Adding WIP object runtime stuff. [skip ci]

    * More progress ob Object runtime [skip ci]
```

```
    * Implemented runtime function for Objects print_k() obj_findkey().

    * Minor changes, removing incorrect tests and added bool test. TODO would be
        to create print_b(), but bool printing works with print_i() right now.

    * Update Makefile

commit 7d76fb7e18c29c4d93262d7e0f4d044dc4951377
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 20 10:09:48 2017 -0500

    Add string length function

commit 83010d967c30256c94d3c3adb322fc5d6e628372
Merge: 4e5c427 3c356da
Author: audsc <audsc@users.noreply.github.com>
Date: Sun Nov 19 19:09:24 2017 -0500

    Merge pull request #59 from waltermeyer/string_comp

    String comp + move string functions to gantrylib

commit 3c356da03ce903f25ee138c80586372a14688ed4
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Nov 19 18:54:13 2017 -0500

    Move string comp and string slice to gantrylib_string

commit 8d12182b93eab53b352fbfbf06c153a6a47d034d
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Nov 19 18:43:55 2017 -0500

    Add stringcmp function

commit 4e5c427d3dde442a8a0bf435e3fc489c7977d183
Merge: 21d5e4a 34edcb2
Author: audsc <audsc@users.noreply.github.com>
Date: Sun Nov 19 17:54:37 2017 -0500

    Merge pull request #55 from waltermeyer/remove_elif

    Remove elif

commit 34edcb23edf779b27e7bcbf8baf189d9ed1737e1
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Nov 19 17:53:49 2017 -0500

    Remove elif
```

```
commit 21d5e4a87cfa7ddc6e0e4456d94bcc194ccb80c7
Merge: ac06a14 c8c7879
Author: audsc <audsc@users.noreply.github.com>
Date: Sun Nov 19 17:30:37 2017 -0500

    Merge pull request #54 from waltermeyer/semantic_checking

    Semantic checking

commit c8c7879c4cdb3cea9593a02edad0cde574fc1f14
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Nov 19 17:05:27 2017 -0500

    Add string concat

commit b01fe94c07ad94485dc86590a066cc63403366c8
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Nov 18 15:19:37 2017 -0500

    Fix function parameters in builtins, tests pass. Need to add scoping for
        locals. Not sure whether this involves adding a node in the sast or adding
         a scope variable in semant

commit 1ad90f374967b4912e2a5f2b73db2a3d551b826c
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Nov 18 14:44:13 2017 -0500

    Add all of the expression nodes present in ast, tests pass except for string
        slice

commit 9814beaf531a9e3bfba982783d5fb21b8dd32ce3
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Nov 18 13:17:12 2017 -0500

    add locals under Assign Declaration

commit b49f22742285d541cbef9ac397a38629d597c508
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Nov 18 12:43:52 2017 -0500

    Add globals and function parameters to symbol table

commit 5cff5b9ec3f571e44827558cb529ae280a3091bb
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 17 20:23:09 2017 -0500
```

```
    Add assignment declaration, try to add to hash table from globals list but not
        sure how

commit 9b62c5db88173c9b816220023f62900a1cc7d7b6
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 17 19:08:48 2017 -0500

    Fix assign

commit c5ce9219308b4f8b270909310ff1f76a2ebb8482
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 17 17:54:37 2017 -0500

    Re-add semant and sast from old semant branch

commit ac06a14998bc8263c5bf8801d00ee476fef13f95
Merge: 34922fe 5b79449
Author: audsc <audsc@users.noreply.github.com>
Date: Sun Nov 19 17:12:06 2017 -0500

    Merge pull request #53 from waltermeyer/fix_concat

    Taimur and walt string concat fix

commit 5b794490061f27a109f7ff65951723d85b96ebc0
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 17 20:37:25 2017 -0500

    Taimur and walt string concat fix

commit 34922feab3219e4ffb98883af04dec570f2fa141
Author: TSamee <taimur.samee@gmail.com>
Date: Fri Nov 17 18:36:41 2017 -0500

    Separated tests for GET and POST, corrected no. of arguments to POST in
        codegen

commit bd33d33dd559203229cd55c0e1db39fbff901919
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 17 17:51:40 2017 -0500

    Delete outdated semant and sast

commit 9fdd5fc49c7913232be1ba3f7234cd91d73cfe94
Merge: 4cc9774 4ee06ef
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Nov 15 21:56:04 2017 -0500
```

```
    Merge branch 'master' of https://github.com/waltermeyer/plt

commit 4cc97741a1f0fdea77e3073cedfa161a30b74f56
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Nov 15 21:55:34 2017 -0500

    Added while test, changed bools to i8_t for compatibility with C.

commit 4ee06ef75a0ddf7c5e9b8cc649ff8316fd6b54be
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Nov 15 18:21:52 2017 -0500

    Increment and Decrement Implemented (i.e. For loops working) (#51)

    * Inc/Dec working, problem with dec

    * For loops working and Increment Decrement operators working.

commit 88c5420132a498ec49e2b5d2f82b62af51592472
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Nov 15 01:41:05 2017 -0500

    Objects Implemented in Codegen (#50)

    * Some C files to help figure out how to write objects in llvm

    * Working on C version of our objects
    Nothing useful right now need to check on VM

    * token objects don't update next prev pointers correctly, either because I'm
        doing it wrong, or because of the way alloca works. I do not have internet
         so will check on this later

    * Manually build C JSON object, not done

    * Print function segfaults

    * json_obj.c

    * Update json struct to reflect doubly linked list

    * Codegen object based on Audrey's design in C. Code generation will not work
        as is for this, may need to rethink AST for KeyVal and ObjExp nodes and
        perhaps combine them in some way. WIP.

    * Working on object codegen code

    * Testing object struct definition
```

```
    * Object declaration working with LLVM struct, but key value population is
      still left to implement.

    * WIP objects -- not working right now

    * Objects may be working, won't be able to tell/test until object access is
      done to validate data

    * Objects still WIP, need to verify indices in code gen are correct to value
      printing in object access.

    * Single-level objects working. More work to be done for nested ones.

    * Objects appear to be working correctly.

commit 9efd7a23bd1851232356c9ec242c6027ddc67e4c
Merge: 0abae13 8ef43c4
Author: audsc <audsc@users.noreply.github.com>
Date: Mon Nov 13 19:53:51 2017 -0500

    Merge pull request #49 from waltermeyer/string_slice

    String slice

commit 8ef43c4b1ba39c2518ab4fd60a3b3e6c03004747
Merge: a4f9b56 ebf40c6
Author: TSamee <taimur.samee@gmail.com>
Date: Mon Nov 13 19:51:10 2017 -0500

    Merge branch 'string_slice' of https://github.com/waltermeyer/plt into
      string_slice

commit a4f9b56ae79db17d2ab7b5e63f48e1a9ef753099
Author: TSamee <taimur.samee@gmail.com>
Date: Mon Nov 13 19:50:48 2017 -0500

    Slice works!!

commit ebf40c62656e15e2f0b83a2542427e31bd97ae52
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 13 19:44:23 2017 -0500

    get rid of printf in slice

commit bc8bd4269da352713c3da820bc20e9d41a34a4dc
Merge: c330f94 3233fc0
Author: TSamee <taimur.samee@gmail.com>
```

```
Date: Mon Nov 13 19:42:05 2017 -0500

    Merge branch 'string_slice' of https://github.com/waltermeyer/plt into
        string_slice

commit c330f94b2d0c222a23b7cc0341ca8d2c37110d7e
Author: TSamee <taimur.samee@gmail.com>
Date: Mon Nov 13 19:41:25 2017 -0500

    Added arguments to slice in codegen; fixed slice test

commit 3233fc0e8043a362de6713292207e686bc13b056
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 13 19:39:47 2017 -0500

    switch order of slice args to match proposal

commit 1905c627b9799d0f591afffbbc159151f40564b9
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 13 18:33:21 2017 -0500

    fix output for string slice test

commit bca5fe6ec4522e539903623ec70fed85173d5b84
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 13 18:12:06 2017 -0500

    String splice function gets used however it gets used incorrectly, the test
        returns null when it should return obar

commit 921412239c6d7d3f9419d1fab1f8021542cdccfa
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 13 17:56:42 2017 -0500

    String slice function works on its own

commit 1c4a97cf39ca9a53fcf0b528e3d61e71e97f1de3
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Nov 13 17:15:03 2017 -0500

    Add string slice C function, doesn't work yet

commit 0abae13107942c165930233e2c90aad21d8ebb1f
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Nov 12 23:35:57 2017 -0500

    Update travis to llvm 3.8
```

```
commit d22f3b8c8ab227bab80130a950d8f800b8496627
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Nov 12 23:09:33 2017 -0500

    testing suite fixes. Moved unimplimented tests to todo, we don't want fails
       for these now. There is a regression in test_for. Changes to Makefile and
       test_all.sh. Deleted tests/generate_ref.sh and just rolled into test_all.
       sh. Modified CI to run test suite. Known working tests are not in tests.
       Nothing should be added here unless it works. All Fail tests moved todo
       because those will not work until SAST is working.

commit 2cb4e64060caee3138055fd921519cb29ac7247a
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Nov 12 21:30:43 2017 -0500

    Resolving conflicts from merge

commit 345f60a8748634e6f3774cec5c75a80471b14592
Merge: f31caf5 20301e8
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Nov 12 21:21:57 2017 -0500

    Merge branch 'master' of https://github.com/waltermeyer/plt

commit f31caf567484205938b96c4a9813b0ce3b697b36
Merge: a3514ab 83cabea
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Nov 12 21:12:51 2017 -0500

    Merge branch 'HTTPfuncs'

commit 83cabea016a9c36f4756b942e696d33215452702
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Nov 12 20:40:10 2017 -0500

    Added libcurl linking to test_all.sh, changed args for httppost() in codegen

commit adbf22f7dfbe717795b7c5b580fb5ff950f6ab26
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Nov 12 18:29:53 2017 -0500

    Edited Makefile to use recipe that links in the Curl library

commit 20301e80a5d4cd9719a379890826128712b5d5ef
Merge: 606faf7 8c46365
Author: audsc <audsc@users.noreply.github.com>
Date: Sun Nov 12 17:52:14 2017 -0500
```

```
    Merge pull request #47 from waltermeyer/string_concat

    String concat

commit 8c4636507b8791b08fdc8629e1f5a3b9b0423226
Merge: 317f400 606faf7
Author: audsc <audsc@users.noreply.github.com>
Date: Sun Nov 12 17:48:49 2017 -0500

    Merge branch 'master' into string_concat

commit 317f400f40c7e4c20231620a5f3079825dceb334
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Nov 12 17:47:10 2017 -0500

    string_concat test now works

commit aa9a5d288c0a30a1555ca64c360be4440be80681
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Nov 12 17:14:28 2017 -0500

    Edited Makefile and stringconcat

commit 606faf7eea37a7984ffb0de2e22202209fe8a1d6
Merge: e1c8c38 f579bcd
Author: TSamee <taimur.samee@gmail.com>
Date: Fri Nov 10 21:45:13 2017 -0500

    Merge pull request #46 from waltermeyer/HTTPfuncs

    HTTPfuncs

commit f579bcd9f7df0fc5e7d601948b90efcb1679171d
Author: TSamee <taimur.samee@gmail.com>
Date: Fri Nov 10 20:08:21 2017 -0500

    Deleted old HTTP GET

commit 00a0cb3a559a4f69c7899e989c93400f8e9695d6
Author: TSamee <taimur.samee@gmail.com>
Date: Fri Nov 10 20:07:19 2017 -0500

    Modified Makefile with new HTTP library name

commit b910581c8ef90abb058e4c3e7d719c417c333a39
Author: TSamee <taimur.samee@gmail.com>
Date: Fri Nov 10 20:06:39 2017 -0500
```

```
    Added HTTP POST, renamed file to gantrylib_http

commit 85af52a5fbea906a534b00034c0d12c987962531
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 10 20:04:35 2017 -0500

    Fix concat test, now getting this error message:

    Stored value type does not match pointer operand type!
      store i32 %string_concat, i8** %c
     i8*LLVM ERROR: Broken module found, compilation aborted!

commit 969a023fb027b8ebe0e76c1a7c3136f8508cc25b
Merge: 3343d25 421cb7c
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 10 19:56:51 2017 -0500

    Merge branch 'string_concat' of github.com:waltermeyer/plt into string_concat

commit 3343d258411ce240bd61c7e438d5b4d1cf1ff2e4
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 10 19:49:32 2017 -0500

    add concat test

commit 7a2627353dfa54470946dab1edba95114ccb5a6d
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 10 19:36:03 2017 -0500

    add string concat to binop

commit 39a97323dc445a1192463ebda091d49d103ed156
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 10 19:07:48 2017 -0500

    String concat C function, need to figure out how to integrate with codegen as
        a binop

commit e1c8c382b87b3ad4c1617b07bf786cf9345e4a50
Merge: 49025f6 ba89f7a
Author: audsc <audsc@users.noreply.github.com>
Date: Fri Nov 10 19:51:27 2017 -0500

    Merge pull request #45 from waltermeyer/test_suite

    Test suite

commit 421cb7cf48e7989cef87ed6e04f272468ab9ca27
```

```
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 10 19:49:32 2017 -0500

    add concat test

commit 766fab62aa3d6ebff5d3575bbc0966e9239d8ad7
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 10 19:36:03 2017 -0500

    add string concat to binop

commit 20551e3cf8cffb0be5b690f71494da13f986f445
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Nov 10 19:07:48 2017 -0500

    String concat C function, need to figure out how to integrate with codegen as
        a binop

commit 2d23f2f17fc26ea507d07537e01fc9edf073469a
Author: TSamee <taimur.samee@gmail.com>
Date: Fri Nov 10 16:16:48 2017 -0500

    Added comments, made sure string is null-terminated

commit eb1f39dd9f0c3a2381de56c310e0843497b73148
Author: TSamee <taimur.samee@gmail.com>
Date: Fri Nov 10 15:58:27 2017 -0500

    HTTP get written + tested; possibly remove debugger flags in Makefile later

commit ba89f7a1a6e0874151da3cd6a89ba21eace18a10 (origin/test_suite)
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Nov 8 19:33:45 2017 -0500

    move outdated/unimplemented tests to todo directory, fix output for and while

commit 953a3920fa69b73d2174a34468efb04734a05d6f
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Nov 8 18:00:34 2017 -0500

    List of tests that are currently passing, each time a feature is added this
        will be updated

commit 75891265a7bc9939cade632661fe11d4b9e5b591
Author: TSamee <taimur.samee@gmail.com>
Date: Wed Nov 8 17:55:03 2017 -0500
```

```
    Packaged httpget into its own function; still need to write get request output
        to a string

commit 49025f64c21979c8454b5fcad3e0679eba8fd947
Merge: a3514ab 46e2aa2
Author: audsc <audsc@users.noreply.github.com>
Date: Wed Nov 8 17:45:46 2017 -0500

    Merge pull request #44 from waltermeyer/test_suite

    Test suite

commit 46e2aa22379ccbab717e559081edb52caf2f6bc2
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Nov 8 16:37:00 2017 -0500

    Fix test_all script so it uses lli instead of generating .s and running
        executable

commit 559b51464f5de6eb48568edb268d0bdf5f1ab33e
Author: audreysc <audreyscopeland@gmail.com>
Date: Wed Nov 8 15:58:15 2017 -0500

    start list of tests that should pass correctly, moving to vm

commit db14d604b733c6f764dbcc62fda83f937f63f809
Author: TSamee <taimur.samee@gmail.com>
Date: Wed Nov 8 13:57:15 2017 -0500

    Added linker flags for libcurl, dummy httpget sends a request to google.com

commit d488e37efe4af58f8f3b28099d5375866b511490
Author: TSamee <taimur.samee@gmail.com>
Date: Tue Nov 7 15:34:33 2017 -0500

    Added instructions to Makefile, verified curl.h import works in C file

commit 61bf1f4aa3b9cdeaa2bd3b22c264754cda000dd1
Author: TSamee <taimur.samee@gmail.com>
Date: Tue Nov 7 15:16:41 2017 -0500

    Added httpget build code to codegen

commit 7907f3612f06311f2b401ad03da80307aa4a8f1f
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Tue Nov 7 13:17:15 2017 -0500
```

```
    Updated full test suite. Review printing format (spaces, new lines) and built-
        in functions

commit 07478a66468e8ede149ede5c362abefa803eb5d5
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Tue Nov 7 12:14:40 2017 -0500

    updated test cases until 'test_for' .. review boolean cases

commit a3514ab2de89e07a01772c54dff2c07fca8510e4
Merge: 992c712 37d5973
Author: audsc <audsc@users.noreply.github.com>
Date: Tue Nov 7 11:10:33 2017 -0500

    Merge pull request #43 from waltermeyer/test_branch

    Test branch

commit 37d59736caa4f228f31cd91b7034228c8471b3f7
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Nov 7 11:09:43 2017 -0500

    update readme to reflect new format for tests

commit eaa46ed1e95977c00475cf48bbc40ab6036bd9cf
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Nov 7 11:00:37 2017 -0500

    .out files instead of .gty.out

commit f423c0bdf5b430a9f68648231997e0f92353f4f2
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Nov 7 10:56:45 2017 -0500

    Fixes to testing generation script and output in source file for test

commit 11c9a44e919ad636896cbb2ba55a7919cda32b83
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Nov 7 10:25:33 2017 -0500

    this doesn't work for generating files

commit 0f78e6e96fb7737f99af8d92b4ad8c9cf9643bfe
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Nov 7 10:15:59 2017 -0500

    fix test format in readme
```

```
commit b33d59c22b33ce7746c5afa9016769b7e3542756
Author: audreysc <audreyscopeland@gmail.com>
Date: Fri Nov 3 08:31:22 2017 -0400

    update readme with instructions for updating tests

commit 992c7126038fedb169275140261ac1809ed4d23c
Merge: 882bc96 a49b8f8
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Nov 6 21:14:12 2017 -0500

    Merge pull request #42 from waltermeyer/codegen

    Codegen

commit a49b8f88deb984b63701f3ba8db1065057107f5
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Nov 6 21:08:25 2017 -0500

    Global declarations and subsequent instantiation in function scope working.

commit 040eba5f04d80d510d73c1468b7dada617b9e6a9
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 3 12:49:19 2017 -0500

    Regular assign (assigning to existing declaration) working for IDs. Added Expr
         to Str helper function to ast.ml to convert lvalue expression to string
        to be able to add it to function locals hash map.

commit 882bc968f73aa6266816b24623262d0afb7dbcd3
Merge: fcb0921 ccaa9dc
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 3 13:23:20 2017 -0400

    Merge pull request #40 from waltermeyer/codegen_audrey

    Codegen audrey

commit ccaa9dc8c6b612c66a48b1f8c6c74bc78aad356f
Merge: 62c29b2 9a94ac7
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 3 12:19:47 2017 -0500

    Merge branch 'codegen_audrey' of https://github.com/waltermeyer/plt into
        codegen_audrey

commit 62c29b2f2495c65c22db40b37e4a5f51c2de3c0a
Author: Walter Meyer <wgmeyer@gmail.com>
```

```
Date: Fri Nov 3 12:17:22 2017 -0500

    Var changes codegen

commit 9a94ac7bc5e1273c3a526a1ccbf78f79a8641817
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 3 12:17:22 2017 -0500

    Var changes codegen

commit fcb092150f2b3ca0bbf2a9741c03f8f5863e3728
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 3 09:34:44 2017 -0400

    Update CI for slack notifications

commit 52096e3ad6d0fc8d4553ab748b439a8206f897e4
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 3 09:31:59 2017 -0400

    Update README.md with CI status [skip ci]

commit 2db0b3b301e43f1a64941679a8b7e85901f69643
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Nov 3 09:27:53 2017 -0400

    Create .travis.yml

commit 818ecc3080e5d0318dfd8f5730867113a7e745ad
Merge: cfee2f6 7c4ad6e
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Nov 2 20:59:20 2017 -0400

    Merge pull request #39 from waltermeyer/codegen_audrey

    Codegen audrey

commit 7c4ad6e24c1c402a9551589e014a965ad33be010
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Nov 2 19:57:06 2017 -0500

    Assignment declarations with defintions and function parameters working using
        Hash Map in codegen.

commit cfee2f6dd0abd23bbd011bf935e1602fc6470589
Author: audsc <audsc@users.noreply.github.com>
Date: Thu Nov 2 19:36:03 2017 -0400
```

```
    Update README.md

commit 9e4e5bd3a4aed0e067c3181c4607401b7e0b79d4
Merge: 6ffcd78 de14440
Author: audsc <audsc@users.noreply.github.com>
Date: Thu Nov 2 19:34:07 2017 -0400

    Merge pull request #37 from waltermeyer/test_branch

    Now the test_all script works. But it doesn't clean up the files it generates

commit a6e59ce303d34b8d6060c23bb6c5182386d864b7
Author: TSamee <taimur.samee@gmail.com>
Date: Thu Nov 2 18:33:19 2017 -0500

    Commented out httpget builder lines (168-9)

commit 2fd45d0db82953247759ede4d88a28a3ca453e21
Merge: 2472e35 2967284
Author: TSamee <taimur.samee@gmail.com>
Date: Thu Nov 2 18:05:33 2017 -0500

    Merge branch 'codegen_audrey' of https://github.com/waltermeyer/plt into
        codegen_audrey

commit 2472e35f8142e4429bbbc0638bdf17058e1e6d2d
Author: TSamee <taimur.samee@gmail.com>
Date: Thu Nov 2 18:04:45 2017 -0500

    Added httpget declaration and builder

commit 2967284efa24f1afe13889d20efadbbdf6e54011
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Thu Nov 2 17:30:27 2017 -0500

    SAST first cut

commit dc1ffc7ce0664268c2759cd02e4ab7853c42f651
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Thu Nov 2 16:53:49 2017 -0500

    Starting SAST

commit ba294d061b8044af6c1c5f20911f0a51d7d31a0d
Merge: bc63230 0ad06b7
Author: TSamee <taimur.samee@gmail.com>
Date: Thu Nov 2 14:16:05 2017 -0500
```

```
    Merge branch 'codegen_audrey' of https://github.com/waltermeyer/plt into
        codegen_audrey

commit 0ad06b7e387fc76e513c6b772186dfd74f029d10
Merge: 2235cb0 7b7beba
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Nov 2 07:28:06 2017 -0500

    Merge branch 'codegen_audrey' of https://github.com/waltermeyer/plt into
        codegen_audrey

commit 2235cb01dd10f7f609fc158f32b367af5a9defe3
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Nov 2 07:15:52 2017 -0500

    Code cleanup and comment addition in codegen.ml. No useful changes other than
        the beginnings of a hashmap for dealing with func local defs during expr
        eval. Need to roll parameter locals into hashmap.

commit 7b7bebae0485b2f29ad6bd13d09d44541daa34dd
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Nov 1 00:42:11 2017 +0000

    add print float and fix print_s and print_i

commit 6ffcd781ae068a11f4ae8cc4de9d6ae253c4eb14
Merge: 984c4f4 0b0e7cf
Author: audsc <audsc@users.noreply.github.com>
Date: Tue Oct 31 19:24:35 2017 -0400

    Merge pull request #33 from waltermeyer/codegen_audrey

    Prints (string)

commit 0b0e7cf1b4c5850e9205810bb33a6a0a4d977dfa
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 31 23:21:40 2017 +0000

    prints works

commit 1645993397da4a9bfd34307f9e118a90c7dec57d
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 31 23:00:41 2017 +0000

    Fixed string literals. Printing integers from main now works.

commit af3e0c16b30379cfdfaf5863edf11ae9d9cdf3b1
Author: audsc <audreyscopeland@gmail.com>
```

```
Date: Tue Oct 31 22:40:01 2017 +0000

    Returning integers from a main function works when generating an LLVM IR
        output

commit 984c4f426fd93affdccc615f8b44bee9cb11146c
Merge: 0947403 0b66df4
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 31 18:08:45 2017 -0400

    Merge pull request #31 from waltermeyer/codegen_audrey

    Codegen audrey

commit de14440433549d5734c114a82551c40f56bae8b4
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 31 18:09:00 2017 +0000

    Now the test_all script works. But it doesn't clean up the files it generates.

commit 0b66df416ad2412db1a5ebfce542ca487c9eff9f
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 31 17:46:59 2017 +0000

    Add elif logic to codegen

commit c2dee44abb656d30b1086820dfa10ae61ba74712
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 31 15:00:33 2017 +0000

    Fixed references to fdecl.A.typ and A.Void, Added elif args to If. Still need
        to fix if logic

commit 0de13bbe9c37a19048f3f3dbf6e2113896673e92
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 31 14:43:46 2017 +0000

    function_decls is func_decls in codegen

commit fd44fccc1ebc6c632e1a44e21f5309f6e0331e18
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 31 14:08:20 2017 +0000

    Bugfixing codegen...strlit, binops, failing on assign now.

commit 9374d89ac1897ae58c5ad4824fb16a3f3be16e88
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 31 12:27:22 2017 +0000
```

```
    working on local allocation. Removed local declaration from function AST
        object and going to try and let it be taken care of in expression gen via
        AssignDecl.

commit 0947403eb98564aafc95b2d6a8ee75ffc87d908a
Merge: f9452ae 92a6dd4
Author: audsc <audsc@users.noreply.github.com>
Date: Tue Oct 31 08:16:13 2017 -0400

    Merge pull request #28 from waltermeyer/test_branch

    Generate correct output files

commit 92a6dd46bffc5aa7f32a937e4799065b53f4f8a6
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 31 12:14:26 2017 +0000

    Add script to generate correct output from the test scripts using the output
        that is commented at the bottom and put them in .err and .out files so
        that they can be compared with output generated when the tests are
        actually run by the test_all script.

commit f9452aeabe0a323c1d0f697f98ae23a89ee8d017
Merge: fa21205 9fd771d
Author: audsc <audsc@users.noreply.github.com>
Date: Mon Oct 30 18:41:05 2017 -0400

    Merge pull request #26 from waltermeyer/test_branch

    Test branch

commit 9fd771dd66dda58134a7f30174443eb023b43807
Author: audsc <audreyscopeland@gmail.com>
Date: Mon Oct 30 22:32:47 2017 +0000

    Add test_all.sh, printbig.o commented out, need to replace with our print
        string

commit bc63230849346b0c1d4f977d13a0d1a3f978d60f
Merge: 8b5d619 ec2bf58
Author: TSamee <taimur.samee@gmail.com>
Date: Mon Oct 30 22:01:19 2017 +0000

    Merge branch 'codegen_audrey' of https://github.com/waltermeyer/plt into
        codegen_audrey

commit ec2bf58aefff6902df184fc94f1e2687e6a7ed53
```

```
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Oct 30 19:53:35 2017 +0000

    Added back missing code to get compilation working. It is moving there. Need
        to resolve issue with local func variables which differ from MicroC. i.e.
        we need to determine how to translate them in their current state as decl'
        d and instantiated or copy MicroC.

commit 8b5d619a42b7fe9f418aaf8c0070e748da8b7249
Merge: 5cc67cd 53c517c
Author: TSamee <taimur.samee@gmail.com>
Date: Mon Oct 30 17:46:04 2017 +0000

    Merge branch 'codegen_audrey' of https://github.com/waltermeyer/plt into
        codegen_audrey

commit 53c517c4a29d36c66702b48ebe130bacc0d5ec09
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Oct 30 17:38:17 2017 +0000

    Forgot to add new toplevel

commit a93cb5992e3b80e3a4fa76e182bd6c549274d853
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Oct 30 17:23:24 2017 +0000

    Added makefile to attempt compilation of codegen.ml and changes to codegen

commit fa21205bd674284042188fbea5c413b068d271a5
Merge: 5c26861 c3e142c
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Oct 30 10:51:06 2017 -0400

    Merge pull request #25 from waltermeyer/ast_globals_only

    Changed AST to align with MicroC wrt global decls and function declar

commit c3e142c5089d63d300aa99ac1c613ce878805d6a
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Oct 30 14:49:52 2017 +0000

    Changed AST to align with MicroC wrt global decls and function declarations
        being the only top-level constructs/nodes on the AST.

commit bbe09a1b07c1674421416b902f68d2853233f17f
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 29 22:59:42 2017 +0000
```

```
    Added 'IF' in builder; other minor updates

commit 5cc67cdff40f3af4030837003d33bf56c1cfadbe
Merge: ae3c1d0 277d559
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Oct 29 22:49:52 2017 +0000

    fixed a merge issue

commit ae3c1d05b65fb716736ccd154f980ff696ad564e
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Oct 29 22:43:37 2017 +0000

    Edited function headers; function body still WIP

commit de8f80ede8f118b28968e002099cae46796dbe7e
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 29 22:18:58 2017 +0000

    Fixed typos, added strlit builder

commit 277d5591c9147f7941afd53e6d1829dcbb72d441
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Oct 29 21:28:51 2017 +0000

    Add array type and expression check to semant

commit 4c623eef55f5a9b739eaedfb9b1bfa5a14cd5189
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Oct 29 14:55:28 2017 +0000

    Start going through semant.ml file

commit 12643443762a9b722cfddc0e3192791e26680c03
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Oct 29 14:19:26 2017 +0000

    Add statements, fix obj/str types

commit cd620774061b7df6a6a645f97ee61ba919130573
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 28 22:48:14 2017 +0000

    Update codegen with expr builder and TODO items, initial pass through codegen
       microc

commit ba9f9299d753854cf4e875d18f3371111fd99960
Author: audsc <audreyscopeland@gmail.com>
```

```
Date: Sat Oct 28 21:18:22 2017 +0000

    Small changes to codegen

commit 43f02fad46c0ffdb1f746512b18a1bc43c6ba1f6
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 28 18:55:31 2017 +0000

    small additions to codegen adding TODOs

commit 974cbfde5bbec14b158c5701593f1a78a9df9538
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 28 18:39:04 2017 +0000

    update codegen

commit a42711fc4d013d96b6bb2b9266f7de90e60436cf
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 28 16:07:23 2017 +0000

    Start codegen, commenting questions

commit 5c268612da72be61581e3bd888a4dc321f86472d
Merge: 534e99e f466a2a
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 29 17:29:29 2017 -0400

    Merge pull request #23 from waltermeyer/arrays

    Added array as a multi-type as opposed to a single one.

commit f466a2ae783399b11b890ae314d7db22c75be183
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 29 21:28:45 2017 +0000

    Added array as a multi-type as opposed to a single one.

commit 2bd9e6bf9deb812943be463e03ef732cc551458b
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Oct 29 21:22:47 2017 +0000

    README update for tests

commit 534e99e0ca5aea21e4a50826f8fa1bac389a4d85
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 29 18:22:21 2017 +0000
```

```
    Added support for multi-dimensional array declarations using [expression]
        notation where an expression will be semantically checked for an INT value
        . That is, the INT represents the dimension of the array declaration.
        Array dimensions, unlike array size/length, will be immutable/static.

commit 50a05a427bfb773cfc19b5afaf5fbf23ac05a162
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 29 16:33:11 2017 +0000

    Removed null literal. Added support for multi-dimensional array access and
        nested object access by refactoring a few things including assignment.
        TODO: multi-dimensional array declaration.

commit 8cc0d65da6ee0522767ca7f3ea32c574a2010329
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Oct 29 14:55:28 2017 +0000

    Start going through semant.ml file

commit 25a9f8cab00aafdf78d29133775fcc25259ece00
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Oct 29 14:19:26 2017 +0000

    Add statements, fix obj/str types

commit f7bcc75fa559075070fe26234b83800d3adf89ff
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 29 03:17:57 2017 +0000

    Added (missing) object dot notation expressions. Dot-notation with nested
        objects not working. Fixed unop printing.

commit 3cdca46c06cf9df5b137e45b5bbf28e747af99fd
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 28 22:48:14 2017 +0000

    Update codegen with expr builder and TODO items, initial pass through codegen
        microc

commit 6dab932ca9afba577cd05605db695f44948bd9db
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Oct 28 22:04:02 2017 +0000

    Order of printing ast seems correct

commit aa622c1b9b7478cfa8139fe8d796e112252ec839
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Oct 28 21:29:11 2017 +0000
```

```
    Fixed func parameter printing

commit d5b393906da0c730d54c705eb53b8393ee2b5ce3
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 28 21:18:22 2017 +0000

    Small changes to codegen

commit 755009de118495aa97c860c11457b1cb2c10f272
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 28 18:55:31 2017 +0000

    small additions to codegen adding TODOs

commit ffece8142daeef320fd33e625a3c2b043fc44068
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Oct 28 18:54:51 2017 +0000

    Attempt to resolve object issue with bar notation and changing associativity
        of COLON. Simplifies previous versions

commit 08b862933650db8a3cb6d5ee8569dfd5eea861d5
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 28 18:39:04 2017 +0000

    update codegen

commit 3c33a833ec8ea747f7e017cc1334e07b790897d5
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 28 16:07:23 2017 +0000

    Start codegen, commenting questions

commit 0d2d4d613d72eec2f6164669943b826e8f05c7d0
Author: TSamee <taimur.samee@gmail.com>
Date: Fri Oct 27 21:53:52 2017 +0000

    String equality tests

commit 53429f3a73ab88301aaae46f75bd815615de8ac2
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Fri Oct 27 21:20:02 2017 +0000

    Updated all cases, some marked with 'review' in header; need to complete fail
        suite

commit c28a9b3040d42e2a65deb658b1aa33b1e91fb9e5
```

```
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Fri Oct 27 20:59:17 2017 +0000

    Updated 19 test files, next http-get

commit 73dc092c9309e3ae9517e5071698675b3a2c68b8
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Oct 27 14:54:22 2017 +0000

    Fixed bugs in conditional ast printing (dupe brackets)

commit 5707226f80c047abce323a1297726096a75ee4e8
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Oct 27 14:53:46 2017 +0000

    Fixed bug in For parsing

commit d03db255eb8e117411e4bc5dee15c546c5063f0b
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Oct 27 14:37:22 2017 +0000

    Major KeyValue changes. Updated all conditionals to use statement blocks
        surrounded by braces defined recursively within statements to simplify
        grammar for conditionals. This resulted in an ambiguity with object
        expressions that needed resolution. Moved object expressions to assignment
         expression because keyvalue blocks can only be assigned to an ID and a
        type once and then modified using dot notation. Previously the ambiguity
        was between an empty statement block {} and an empty object, which was
        resolved with contextualizing objects within assignment only.

commit 35d53ff25d2597608b91c0f2795688c32598fe51
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Thu Oct 26 19:59:07 2017 +0000

    Changes to while print, verbose yacc output in Makefile. If statements still
        not parsing even with added prec.

commit 9f9b4c92d3ec7e3b8b050e13a053f8aa8e9138cb
Merge: a750a71 8025f54
Author: audsc <audreyscopeland@gmail.com>
Date: Thu Oct 26 17:24:57 2017 +0000

    Merge branch 'test_branch' of github.com:waltermeyer/plt into test_branch

commit a750a711171059ce9248ded2819dc82afa36eb75
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 24 23:54:17 2017 +0000
```

```
    add fail tests for dcl

commit 621c2a50881ec0b9dd2b7f94fb899632adfed776
Author: TSamee <taimur.samee@gmail.com>
Date: Tue Oct 24 23:37:21 2017 +0000

    added equality tests and fail-tests

commit 50032a62379fe77ef6bbfce9f6ace45b241cdb11
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Tue Oct 24 23:19:18 2017 +0000

    Updates to test suite..in progress

commit 7a2c3207f82816307e99249ec99e04b739ae9aea
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Oct 25 20:27:16 2017 +0000

    Fixed pretty-printing for if stmts

commit cc25755fe2a97be7b90ba05cb1a4571bdba882ee
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 24 23:55:45 2017 +0000

    Fixed reversed statement declaration list

commit 8025f54e9d2e035f7d76196b20fbc646d2225493
Merge: 7d781c7 6e9bfa1
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 24 23:54:23 2017 +0000

    Merge branch 'test_branch' of github.com:waltermeyer/plt into test_branch

commit 7d781c78a7fffb445f707eaa3563d12b3e258a94
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 24 23:54:17 2017 +0000

    add fail tests for dcl

commit 81a2b5394dbee22a1605841caf9f26f17e8b84b2
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 24 23:52:38 2017 +0000

    Fixed array access/ID

commit 6e9bfa168442af30363ccf7d9e8c81b7bc1f62be
Merge: cad1b41 7e325f7
Author: TSamee <taimur.samee@gmail.com>
```

```
Date: Tue Oct 24 23:38:17 2017 +0000

    Merge branch 'test_branch' of https://github.com/waltermeyer/plt into
        test_branch

commit cad1b4188752bdc06ca68dc78646dfdf6fcf522c
Author: TSamee <taimur.samee@gmail.com>
Date: Tue Oct 24 23:37:21 2017 +0000

    added equality tests and fail-tests

commit 201ae0ff1459b8ff919b3116fb0ffea0f7814cac
Merge: 59f6fbb 7e325f7
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 24 23:20:12 2017 +0000

    Merge branch 'test_branch' of github.com:waltermeyer/plt into test_branch

commit 7e325f720a601d1b3a1ad19e4cabe5ec5a0274f3
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Tue Oct 24 23:19:18 2017 +0000

    Updates to test suite..in progress

commit 59f6fbbc4c715a3be0024f1da745cc05f498f84c
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 24 23:03:02 2017 +0000

    Fixed string literals and associated printing

commit 12092788be70cec783f34e88f37d4f768ddbfb0e
Merge: 4713211 ace8a66
Author: audsc <audsc@users.noreply.github.com>
Date: Tue Oct 24 18:22:37 2017 -0400

    Merge pull request #19 from waltermeyer/test_branch

    Test branch

commit 47132110c683eb9674e75584c0a6cd3d00593311
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 24 12:16:45 2017 -0400

    Update parser.mly

commit e3a2182ed72e0d456adff1a01a7876c1a6564aed
Merge: 7658a53 7cebc54
Author: Walter Meyer <wgmeyer@gmail.com>
```

```
Date: Tue Oct 24 12:15:14 2017 -0400

    Merge pull request #17 from waltermeyer/audrey_ast

    Parser change collapse declaration_list

commit 7cebc545f63e273774b33efed6b5307e0966f59e
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 24 16:14:25 2017 +0000

    Parser change collapse declaration_list

commit 7658a536e959f3ec870dfbe883c6d2a6a18ba8f9
Merge: 8167a67 e343f3f
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 24 10:23:58 2017 -0400

    Merge pull request #16 from waltermeyer/audrey_ast

    Merging compiling, but probably not correct AST to master to continue working.

commit e343f3fa2a865b15d01eb2efa8f668b837ec244d (origin/audrey_ast, audrey_ast)
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 24 14:21:43 2017 +0000

    Adding partial gantry top level

commit d705e3f6202246eb6bd588fe183ce90407fa3d79
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 24 14:20:56 2017 +0000

    Many changes. AST is compiling but likely needs more work. Changed CFG so that
         declaration is collaped into declaration_list. Moved assignment and
        function expressions under expressions and out of expression statements.
        Added STRLIT

commit a57a0dc1692a5bf8d49dfbf136fe780cf428d1b8
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Oct 23 19:58:15 2017 +0000

    Fix for scanner raise Failure

commit 3f33f4fa9cf8ec4d64f8e5d8a36a2a18826b8066
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Oct 23 02:13:21 2017 +0000

    Adding Makefile with tests disabled and limited functionality required for up
        to ast phase.
```

```
commit e19993cd96160cfcd08f27473be1f40362d3f984
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Oct 23 01:59:44 2017 +0000

    Checking in WIP changes to ast--non-functional

commit ace8a6626742394dab2fa43a24015f00d52b6e92 (origin/test_branch)
Merge: c1f1932 cc0ab18
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 22 22:00:44 2017 +0000

    Positive control flow test cases

commit c1f1932b6149461378b20621b86bfecbec0e04b8
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 22 21:59:00 2017 +0000

    Control loop pass test cases

commit cc0ab18d7e683e1633782bac17a1d6ae69bcc617
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Oct 22 21:56:41 2017 +0000

    test arithmetic operations for int and floats

commit 3fb075965fe70af8909caf320c015b72d2d736e9
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 22 21:37:25 2017 +0000

    Updated README

commit 9a1fb7d53ac97fc5f88530507b5ef7301aef64a2
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Oct 22 21:27:39 2017 +0000

    add initial tests

commit 04f8218dbc467ef5297fd7724588c7988ffc90e6
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Oct 22 20:52:06 2017 +0000

    Add assign for array and id

commit 00345c20c28a30641c0d1415b0635160e221186f
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 22 20:45:54 2017 +0000
```

```
    removed semis from if/else..accidentally added before

commit f1cecd6844fc5fd22aaeaa1d465fb6b9041f2679
Merge: 8bdaabb 72590a7
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 22 20:38:19 2017 +0000

    Merge branch 'audrey_ast' of github.com:waltermeyer/plt into audrey_ast

commit 8bdaabb370d42e2bfea2a4c66a1dee8b1b673611
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 22 20:38:13 2017 +0000

    merging

commit 72590a743d2b575f15b7e4a42e08a610fd89c9a5
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 22 20:32:01 2017 +0000

    Changes to parser for potential ast hooks related to assignment_expressions

commit 6054c4e8d68409f498d995f1fed32368a60caf85
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 22 20:29:41 2017 +0000

    Added semicolons in IF/WHILE

commit 3dd4992484b6f1ec40efbe9c19dc19938669f3aa
Merge: 1009663 037766d
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Oct 22 20:17:46 2017 +0000

    Merge branch 'audrey_ast' of https://github.com/waltermeyer/plt into
        audrey_ast

commit 100966327d11efad6d50924cd9cbbba842424563
Author: TSamee <taimur.samee@gmail.com>
Date: Sun Oct 22 20:16:14 2017 +0000

    Added pretty-printing for vars, funcs and programs to AST

commit 037766d8a573c1b2ace15904d233e0f46885fff5
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Sun Oct 22 20:10:59 2017 +0000

    Fixed typo bool

commit 1648ef946b6ee48ed54a86850a5d058552ede21f
```

```
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Oct 20 23:33:30 2017 +0000

    add some of the printing to the ast

commit fcaf06428e306647eebeb52845403c24ed73ac17
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Fri Oct 20 22:58:05 2017 +0000

    Added more to/changed ast and parser

commit 435b194a0ac7d2ec1eb0a1a8b390f733719c4c69
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Oct 20 22:30:27 2017 +0000

    update ast terms

commit 8f216fbd5d59e3a67b7ffea87e3d47dc38bc43cd
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Oct 20 22:06:25 2017 +0000

    update ast with function declaration fix typo in parser

commit 37c77cb8eba907e9f1fad2eec13147a6f74104d5
Author: audsc <audreyscopeland@gmail.com>
Date: Fri Oct 20 21:53:01 2017 +0000

    Starting AST and change semantic actions in parser

commit 8167a6791bf08abc83529f50d5390f89ba99bd3b
Merge: 4846f71 b11e938
Author: audsc <audsc@users.noreply.github.com>
Date: Sun Oct 15 17:51:32 2017 -0400

    Merge pull request #13 from waltermeyer/walt

    Key value keys + fix LBRACK/RBRACK errors

commit b11e9382cfa7118fd0d70566747c973939efb43f (origin/walt, walt)
Author: audsc <audreyscopeland@gmail.com>
Date: Sun Oct 15 21:49:30 2017 +0000

    Key value keys are typed. Fix erroneous LBRACEs/RBRACEs

commit eb1b54262188acbee3d8f76e73524c298b33a277
Merge: e8b254e 4c203fd
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Oct 14 15:53:17 2017 +0000
```

```
    Merge branch 'walt' of https://github.com/waltermeyer/plt into walt

commit e8b254ea01fa33f44f0ab5ecbf9e20f521921e86
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sat Oct 14 15:52:44 2017 +0000

    removed CHARLIT scanner

commit 4846f71d56adc55465c59d7cd17b1fd6fab315c6
Merge: 8b7c0ef 4c203fd
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Oct 11 21:24:18 2017 -0400

    Merge pull request #12 from waltermeyer/walt

    Walt

commit 4c203fdc9833b2eb1723fc1254ab1ef25debf7e9
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Oct 11 22:31:45 2017 +0000

    UMINUS has higher precedence than TIMES and DIVIDE

commit e1c06667c31d198e20c2914771c30124ce510081
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Oct 11 22:25:20 2017 +0000

    if statements

commit 6cad1ed3802793d074843664c232b38c22b1ae27
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Oct 11 22:14:14 2017 +0000

    add UMINUS

commit 8b7c0efca1a20dc037bbd592807b56ccbaffb9a1
Merge: 311e60a 1d3081e
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Oct 11 01:25:22 2017 -0400

    Merge pull request #10 from waltermeyer/walt

    Walt

commit 1d3081e589c0bb35058b7d468c06f7ecaa999cad
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Oct 11 05:23:08 2017 +0000
```

```
    Deleted scanner.ML from repo for now

commit 11ac4d9a3b53ec55fa7593417ea7383116466286
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Oct 11 05:19:02 2017 +0000

    Resolving reduce/reduce, shift/reduce, and never reduced conflicts. Added
        precedence for INC/DEC. Modified top-level grammar rules in yacc. Modified
         assignment expressions with ability to put arbitrary expression in array
        subscript.

commit a0389b9e3fddfa9fd345f93c7399b36b6b278c10
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Oct 11 05:15:42 2017 +0000

    Removed chars and cleanup in lexer.

commit c0d3e196cff688077e49c7d92719ae095200bbf0
Author: audsc <audreyscopeland@gmail.com>
Date: Tue Oct 10 23:54:41 2017 +0000

    Add expression_statement_opt and get rid of expression_opt,
        function_expression_opt, assignment_expression_opt

commit 85792766fcc76d4c33577aaa0491cbd417d52422
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 10 23:36:09 2017 +0000

    Updated parser errors with never reduced

commit 7960a66154f973f75d818b65251583a617e769c4
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 10 21:52:44 2017 +0000

    Fixed syntax errors, missing if-statements. Many reduction errors to fix.

commit 5060599c879ccab163042b124ea67a8c0918c0bf
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 10 20:03:49 2017 +0000

    CFG input, not tested

commit e786cfc3aa16a19379e866ce6f466d04b5c977a8
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 10 18:52:12 2017 +0000

    Additions to parser, not working/tested.
```

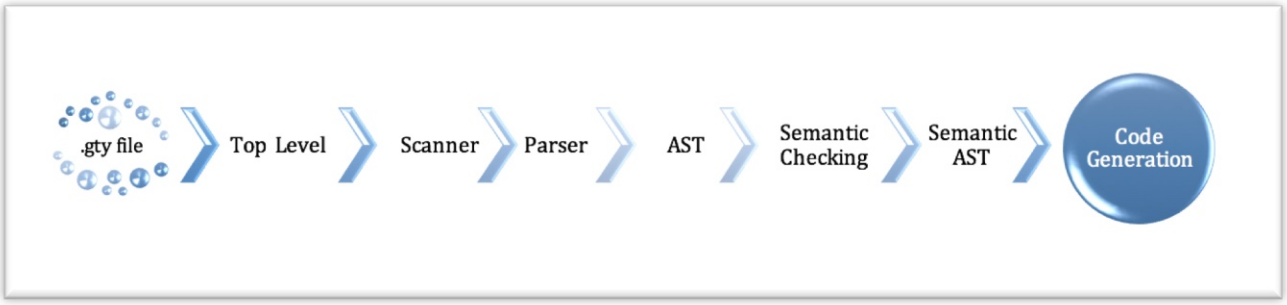
```
commit 39c73bf1595d7e5c36f09f967276a81d8c74293f
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 10 13:39:09 2017 +0000

    Added tab char to specials

commit cbfdb6b7b069786bdc973925d4618f4441e006e6
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 10 13:35:11 2017 +0000

    Committing work in progress parser.

commit fea7f9b60386c6f15524c52a7bd32ddf6724d09f
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 8 19:43:57 2017 +0000

    period

commit f618e140628a641fb00b371dfb18077436f02737
Merge: 2005db9 00d401f
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 8 19:43:20 2017 +0000

    Merge branch 'master' into walt

commit 2005db982081d28eeea85e443b0eee1dbf5342af
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 8 19:42:48 2017 +0000

    Period and update gitignore

commit 311e60afc9555d596d6a5731aed9b5fbf102ac47
Merge: b0235f9 065794c
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Sun Oct 8 12:05:54 2017 -0400

    Merge pull request #7 from waltermeyer/walt

    Walt

commit b0235f9a4c3add3efa79cd1f1efbbb6c7ed9f9f3
Author: audsc <audsc@users.noreply.github.com>
Date: Sat Oct 7 11:57:55 2017 -0400

    Update README.md

commit d4a7d08d58def2e4d58dc0611c52967af9877ea1
```

```
Author: audsc <audreyscopeland@gmail.com>
Date: Sat Oct 7 15:55:22 2017 +0000

    Start todo list for needed tests

commit 065794c17182acd4714609b049d59326517adc35
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Oct 4 16:11:40 2017 +0000

    Added colon, arr, and char to scanner.

commit 7a3c2645028644294405635e66b603be59d642cf
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 3 22:15:13 2017 +0000

    Bug fixes

commit 00d401fe1bbb6054dae501000ce0a076e48563cf
Merge: 89db63e edc1af0
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 3 14:50:42 2017 -0400

    Merge pull request #6 from waltermeyer/walt

    Walt

commit edc1af04401a9715d05b472686893ece6ee26d79
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Oct 3 18:49:23 2017 +0000

    Added string literal scanning code

commit 09daef071a303d50d5f26ae0632f03fc2f7aa874
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Mon Oct 2 14:47:34 2017 +0000

    Changes to scanner. Removed the func and arr keywors amongst other things.
        Need to add string lexing.

commit 89db63e17a8154525ccb2bf7a5deb620f03c0a35
Merge: edd3b96 9032206
Author: rms2241 <31939846+rms2241@users.noreply.github.com>
Date: Wed Sep 27 18:29:14 2017 -0400

    Merge pull request #3 from waltermeyer/walt_scanner

    Walt scanner
```

```
commit 9032206e993dff42d47a7bf365e64d162901607a (origin/walt_scanner)
Merge: 25e1e57 edd3b96
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Wed Sep 27 22:26:45 2017 +0000

    Merge branch 'master' into walt_scanner

commit edd3b96c909edc2b19bf80e1cb066aa37dc83352
Author: audsc <audreyscopeland@gmail.com>
Date: Wed Sep 27 22:21:25 2017 +0000

    Add tests directory

commit 25e1e572ff6b46b6a1c73be4a069ea8c8ed4fdb7
Author: Rizwan Syed <rms2241@columbia.edu>
Date: Wed Sep 27 22:13:50 2017 +0000

    touched README

commit 7925a877d287cd9a8f0d5ac7923745cffd06d517
Merge: 04410c7 8d87a32
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Sep 27 18:12:06 2017 -0400

    Merge pull request #1 from waltermeyer/walt_scanner

    Walt scanner

commit 8d87a32e2851f35b85865bd584bb1abc4b6a4f80
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Sep 27 22:09:19 2017 +0000

    Created gantry project directory

commit 6801718284e4968ad2dedbbfa4a964b310ed9a6a
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Wed Sep 27 21:46:37 2017 +0000

    Commit of unfinished skeleton of our Scanner

commit 04410c7dc62f98731b907d0922a8fab7f0f3f21b
Author: Walter Meyer <wgmeyer@gmail.com>
Date: Tue Sep 12 07:23:41 2017 -0400

    Initial commit
```

5 Architectural Design

5.1 Block Diagram



5.1.1 Top-Level

gantry.ml

The top-level program of the compiler "glues" all of the constituent compiler phases together. In our case, this program is straightforward and taken almost exactly from MicroC. Essentially, the top-level reads a Gantry program from standard input and then passes it through the rest of the compiler phases by way of in-order function calls to each respective component (as per the block diagram).

5.1.2 Scanner

scanner.mll
Author: Walter Meyer

The scanner takes in a stream of characters after the top-level run, and converts them into tokens of identifiers, keywords, operators, constants, and separators. The scanner discards comments.

5.1.3 Parser and AST

parser.mly and *ast.ml*
Author: Walter Meyer

The parser takes the stream of tokens from the scanner and generates a Concrete Syntax Tree (CST). This is done based on Gantry context-free grammar converted to Ocamlyacc code in parser.mly. This is then given to *ast.ml* which converts the CST into an Abstract Syntax Tree (AST).

5.1.4 Semantic Checking

semant.ml
Author: Audrey Copeland

Our Semantic Checker provides many of the same checks as MicroC. Additionally, the semantic checker:

- Keeps a symbol table for each function to check the expressions and statements in that function. We added locals to the symbol table when they were declared in the function.
- Checks that any functions that were called were either declared, or were built-in functions. We checked that the function formals matched the type of the function actuals, except in the case of some of our built-in functions which accepted void pointers, in which case we did not check the type of the parameter.
- Checks that the types match in assignments.
- Checks that identifiers in assignment declarations are not multilevel objects.
- Checks that there were no duplicate function declarations.
- Checks that Object Key value pairs are of the same type.
- Checks that Array Types are valid and enclosed array expressions are all of the same type.
- Checks that the same Object identifier is not declared multiple times.

5.1.5 Code Generation

codegen.ml
Author: Walter Meyer

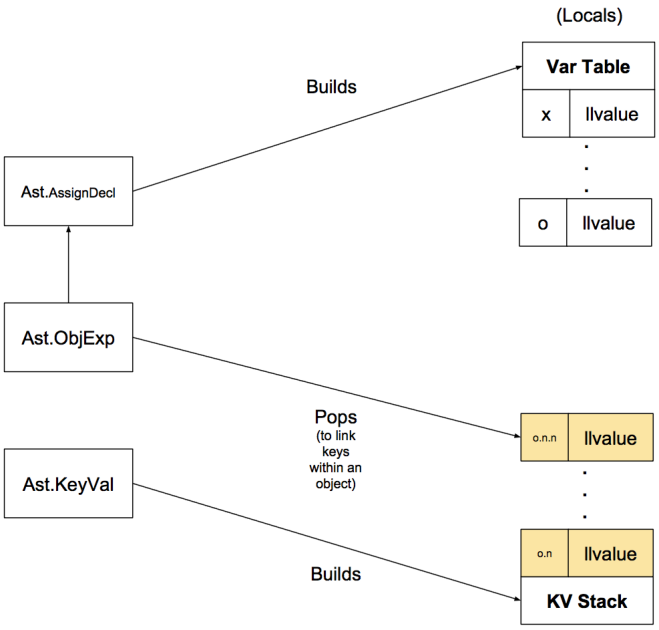
The semantically-checked AST (SAST) is passed to Codegen which generates LLVM IR from the SAST "nodes".

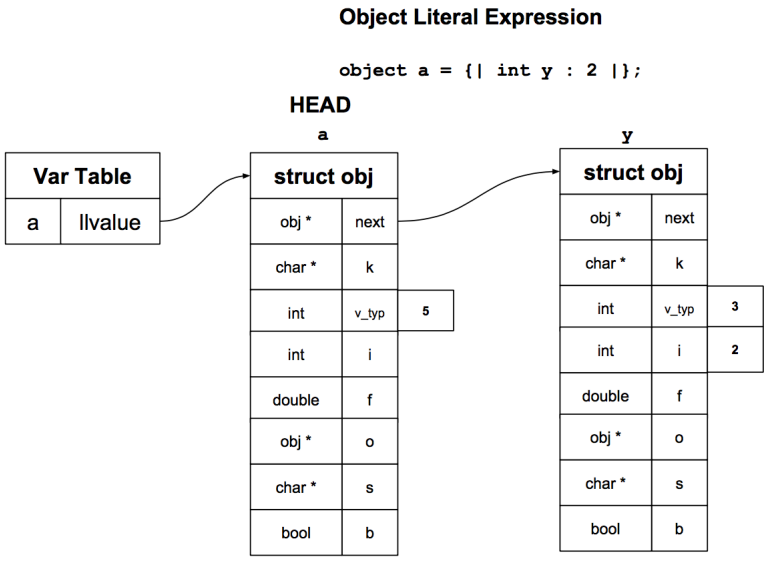
The code generation takes care of realizing the semantics of the language by translating it to the constituent IR and/or passing it off to supporting C Library functions as needed (also called in LLVM IR). The generated LLVM realizes blocks, scoping, function calls, memory allocation (locals, globals), etc. as specified by the language vis-a-vis the SAST it is being "fed".

The notable and more interesting aspects of the Codegen implementation are its per-function symbol table implemented using a HashMap and Objects, which are built in LLVM IR using Structs and an OCaml Stack data structure. The specifics of this implementation are

depicted in the graphics below.

Notice that the Structs are representative of a single key in an object (with a HEAD object being value-less to represent the beginning of an object). That is, the object key value pairs are simply a (potentially nested) linked-list that is built using a Stack in OCaml from the SAST nodes as they are fed in.





6 Test Plan

Testing occurred in multiple phases. We first wrote unit-tests which quickly became outdated as our language evolved. We wrote integration tests as new features were implemented, and other team members would supplement these tests if the features needed to be reviewed before merging. We favored more complex tests with better code coverage over many shorter tests. For example, test_math.gty tests many arithmetic operations with many different types, and test_arr_stringify.gty tests using arrays in objects, accessing and assigning objects from keys in objects, using array and object stringify, and adding objects into arrays at compile time as well as at runtime.

6.1 Sample Programs

6.1.1 Greatest Common Divisor

The *greatest common divisor* algorithm we implemented in our language demonstrates that Gantry can handle recursion as well as integer comparison and arithmetic.

Listing 42: Greatest Common Divisor - test_gcd.gty

```
1  int gcd(int x, int y){
2          while (x != y) {
3                  if ( x > y){
4                          x = x-y;
5                  }
6                  else{
7                          y = y-x;
8                  }
9          }
10         return x;
11 }
12
13
14
15 int main(){
16         int res = gcd(21, 77);
17         print_i(res);
18 }
```

Listing 43: Greatest Common Divisor - test_gcd.ll

```
1  ; ModuleID = 'Gantry'
2
3  %obj = type { %obj*, i8*, i32, i32, double, %obj*, i8*, i8, %arr_int_t**,
       %arr_flt_t**, %arr_str_t**, %arr_bool_t** }
4  %arr_int_t = type { i32, i32, i32* }
```

```
%arr_flt_t = type { i32, i32, double* }
%arr_str_t = type { i32, i32, i8** }
%arr_bool_t = type { i32, i32, i8* }

@int_fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@flt_fmt = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@str_fmt = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@int_fmt.1 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@flt_fmt.2 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@str_fmt.3 = private unnamed_addr constant [4 x i8] c"%s\0A\00"

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

declare i32 @print_b(i8, ...)

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

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

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

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

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

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

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

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

declare i32 @nap(i32, ...)

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

declare i8* @obj_stringify(%obj*, ...)

declare %obj* @obj_addkey(%obj*, i8*, i32, i8*, ...)

declare %obj* @obj_findkey(%obj*, i8*, ...)

declare i8* @obj_getkey(%obj*, ...)

declare i32 @obj_assign(%obj*, i32, i8*, ...)

declare i32 @obj_gettyp(%obj*, ...)

declare i32 @print_k(%obj*, ...)
```

```
53
54  define i32 @gcd(i32 %x, i32 %y) {
55  entry:
56    %x1 = alloca i32
57    store i32 %x, i32* %x1
58    %y2 = alloca i32
59    store i32 %y, i32* %y2
60    br label %while
61
62  while: ; preds = %merge, %entry
63    %x11 = load i32, i32* %x1
64    %y12 = load i32, i32* %y2
65    %tmp13 = icmp ne i32 %x11, %y12
66    br i1 %tmp13, label %while_body, label %merge14
67
68  while_body: ; preds = %while
69    %x3 = load i32, i32* %x1
70    %y4 = load i32, i32* %y2
71    %tmp = icmp sgt i32 %x3, %y4
72    br i1 %tmp, label %then, label %else
73
74  merge: ; preds = %else, %then
75    br label %while
76
77  then: ; preds = %while_body
78    %x5 = load i32, i32* %x1
79    %y6 = load i32, i32* %y2
80    %tmp7 = sub i32 %x5, %y6
81    store i32 %tmp7, i32* %x1
82    br label %merge
83
84  else: ; preds = %while_body
85    %y8 = load i32, i32* %y2
86    %x9 = load i32, i32* %x1
87    %tmp10 = sub i32 %y8, %x9
88    store i32 %tmp10, i32* %y2
89    br label %merge
90
91  merge14: ; preds = %while
92    %x15 = load i32, i32* %x1
93    ret i32 %x15
94  }
95
96  define i32 @main() {
97  entry:
98    %a = alloca i32
99    %gcd_result = call i32 @gcd(i32 21, i32 77)
100   store i32 %gcd_result, i32* %a
```

```
101   %b = alloca i32
102   %gcd_result1 = call i32 @gcd(i32 22, i32 110)
103   store i32 %gcd_result1, i32* %b
104   %c = alloca i32
105   %gcd_result2 = call i32 @gcd(i32 72, i32 880)
106   store i32 %gcd_result2, i32* %c
107   %a3 = load i32, i32* %a
108   %print_i = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4
          x i8]* @int_fmt.1, i32 0, i32 0), i32 %a3)
109   %b4 = load i32, i32* %b
110   %print_i5 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8],
          [4 x i8]* @int_fmt.1, i32 0, i32 0), i32 %b4)
111   %c6 = load i32, i32* %c
112   %print_i7 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8],
          [4 x i8]* @int_fmt.1, i32 0, i32 0), i32 %c6)
113   ret i32 0
114 }
```

6.1.2 Docker API calls

The small sample application included allows us to interface with the Docker API by creating and manipulating an object in the Gantry language, and then calling obj_stringify function to return a string. We then modify the initial object to make a different request to the Docker API.

Listing 44: Docker App - docker_app.gty

```
1  int main() {
2
3    string docker_host = "http://localhost";
4    string ver = httpget(docker_host ^ "/version");
5    string image = "httpd:alpine";
6
7    print_s("\n *** Starting API calls to: [" ^ docker_host ^ "] ***\n");
8
9    // Download Image
10   print_s("\n *** Pulling Container Image: [" ^ image ^ "] ***\n");
11   print_s(httppost(docker_host ^ "/images/create?fromImage=" ^ image, ""));
12
13
14   object create = {|
15                       string Image : "alpine:latest",
16                       string array Cmd : ["echo", "hello world"]
17                  |};
18
19   string s_create = obj_stringify(create);
```

```
20
21   // Create and start container
22   print_s(httppost(docker_host ^ "/containers/create", s_create));
23 }
```

Listing 45: Docker App - docker_app.ll

```
1  ; ModuleID = 'Gantry'
2  %obj = type { %obj*, i8*, i32, i32, double, %obj*, i8*, i8, %arr_int_t**,
       %arr_flt_t**, %arr_str_t**, %arr_bool_t** }
3  %arr_int_t = type { i32, i32, i32* }
4  %arr_flt_t = type { i32, i32, double* }
5  %arr_str_t = type { i32, i32, i8** }
6  %arr_bool_t = type { i32, i32, i8* }
7
8  @int_fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
9  @flt_fmt = private unnamed_addr constant [4 x i8] c"%f\0A\00"
10 @str_fmt = private unnamed_addr constant [4 x i8] c"%s\0A\00"
11 @string = private unnamed_addr constant [17 x i8] c"http://localhost\00"
12 @string.1 = private unnamed_addr constant [9 x i8] c"/version\00"
13 @string.2 = private unnamed_addr constant [13 x i8] c"httpd:alpine\00"
14 @string.3 = private unnamed_addr constant [31 x i8] c"\0A *** Starting API calls
       to: [\00"
15 @string.4 = private unnamed_addr constant [7 x i8] c"] ***\0A\00"
16 @string.5 = private unnamed_addr constant [33 x i8] c"\0A *** Pulling Container
       Image: [\00"
17 @string.6 = private unnamed_addr constant [7 x i8] c"] ***\0A\00"
18 @string.7 = private unnamed_addr constant [1 x i8] zeroinitializer
19 @string.8 = private unnamed_addr constant [26 x i8] c"/images/create?fromImage
       =\00"
20 @string.9 = private unnamed_addr constant [14 x i8] c"alpine:latest\00"
21 @key_name_str = private unnamed_addr constant [6 x i8] c"Image\00"
22 @string.10 = private unnamed_addr constant [5 x i8] c"echo\00"
23 @string.11 = private unnamed_addr constant [12 x i8] c"hello world\00"
24 @key_name_str.12 = private unnamed_addr constant [4 x i8] c"Cmd\00"
25 @string.13 = private unnamed_addr constant [19 x i8] c"/containers/create\00"
26
27 declare i32 @printf(i8*, ...)
28
29 declare i32 @print_b(i8, ...)
30
31 declare i8* @string_concat(i8*, ...)
32
33 declare i8* @slice(i8*, i32, i32, ...)
34
35 declare i32 @stringcmp(i8*, i8*, ...)
36
37 declare i8 @stringeq(i8*, i8*, ...)
```

```
38
39 declare i32 @string_length(i8*, ...)
40
41 declare i32 @arr_length(i8*, ...)
42
43 declare i8* @httpget(i8*, ...)
44
45 declare i8* @httppost(i8*, i8*, ...)
46
47 declare i32 @nap(i32, ...)
48
49 declare i8* @arr_stringify(i8*, ...)
50
51 declare i8* @obj_stringify(%obj*, ...)
52
53 declare %obj* @obj_addkey(%obj*, i8*, i32, i8*, ...)
54
55 declare %obj* @obj_findkey(%obj*, i8*, ...)
56
57 declare i8* @obj_getkey(%obj*, ...)
58
59 declare i32 @obj_assign(%obj*, i32, i8*, ...)
60
61 declare i32 @obj_gettyp(%obj*, ...)
62
63 declare i32 @print_k(%obj*, ...)
64
65 define i32 @main() {
66 entry:
67   %docker_host = alloca i8*
68   store i8* getelementptr inbounds ([17 x i8], [17 x i8]* @string, i32 0, i32 0),
        i8** %docker_host
69   %ver = alloca i8*
70   %docker_host1 = load i8*, i8** %docker_host
71   %string_concat = call i8* (i8*, ...) @string_concat(i8* %docker_host1, i8*
       getelementptr inbounds ([9 x i8], [9 x i8]* @string.1, i32 0, i32 0))
72   %httpget = call i8* (i8*, ...) @httpget(i8* %string_concat)
73   store i8* %httpget, i8** %ver
74   %image = alloca i8*
75   store i8* getelementptr inbounds ([13 x i8], [13 x i8]* @string.2, i32 0, i32
       0), i8** %image
76   %docker_host2 = load i8*, i8** %docker_host
77   %string_concat3 = call i8* (i8*, ...) @string_concat(i8* getelementptr inbounds
        ([31 x i8], [31 x i8]* @string.3, i32 0, i32 0), i8* %docker_host2)
78   %string_concat4 = call i8* (i8*, ...) @string_concat(i8* %string_concat3, i8*
       getelementptr inbounds ([7 x i8], [7 x i8]* @string.4, i32 0, i32 0))
79   %print_s = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4
        x i8]* @str_fmt, i32 0, i32 0), i8* %string_concat4)
```

```
80  %image5 = load i8*, i8** %image
81  %string_concat6 = call i8* (i8*, ...) @string_concat(i8* getelementptr inbounds
        ([33 x i8], [33 x i8]* @string.5, i32 0, i32 0), i8* %image5)
82  %string_concat7 = call i8* (i8*, ...) @string_concat(i8* %string_concat6, i8*
        getelementptr inbounds ([7 x i8], [7 x i8]* @string.6, i32 0, i32 0))
83  %print_s8 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8],
        [4 x i8]* @str_fmt, i32 0, i32 0), i8* %string_concat7)
84  %docker_host9 = load i8*, i8** %docker_host
85  %string_concat10 = call i8* (i8*, ...) @string_concat(i8* %docker_host9, i8*
        getelementptr inbounds ([26 x i8], [26 x i8]* @string.8, i32 0, i32 0))
86  %image11 = load i8*, i8** %image
87  %string_concat12 = call i8* (i8*, ...) @string_concat(i8* %string_concat10, i8*
        %image11)
88  %httppost = call i8* (i8*, i8*, ...) @httppost(i8* %string_concat12, i8*
        getelementptr inbounds ([1 x i8], [1 x i8]* @string.7, i32 0, i32 0))
89  %print_s13 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8],
        [4 x i8]* @str_fmt, i32 0, i32 0), i8* %httppost)
90  %create = alloca %obj*
91  %malloccall = tail call i8* @malloc(i32 ptrtoint (%obj* getelementptr (%obj,
        %obj* null, i32 1) to i32))
92  %obj = bitcast i8* %malloccall to %obj*
93  %key_name = getelementptr inbounds %obj, %obj* %obj, i32 0, i32 1
94  store i8* getelementptr inbounds ([6 x i8], [6 x i8]* @key_name_str, i32 0, i32
        0), i8** %key_name
95  %value_typ = getelementptr inbounds %obj, %obj* %obj, i32 0, i32 2
96  store i32 6, i32* %value_typ
97  %value = getelementptr inbounds %obj, %obj* %obj, i32 0, i32 6
98  store i8* getelementptr inbounds ([14 x i8], [14 x i8]* @string.9, i32 0, i32
        0), i8** %value
99  %malloccall14 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i1** getelementptr
        (i1*, i1** null, i32 1) to i32), i32 2))
100 %arr = bitcast i8* %malloccall14 to i8***
101 %arr15 = bitcast i8*** %arr to i8**
102 %malloccall16 = tail call i8* @malloc(i32 ptrtoint (%arr_str_t* getelementptr (
        %arr_str_t, %arr_str_t* null, i32 1) to i32))
103 %arr_struct = bitcast i8* %malloccall16 to %arr_str_t*
104 %arr_size = getelementptr inbounds %arr_str_t, %arr_str_t* %arr_struct, i32 0,
        i32 0
105 store i32 2, i32* %arr_size
106 %arr_type = getelementptr inbounds %arr_str_t, %arr_str_t* %arr_struct, i32 0,
        i32 1
107 store i32 6, i32* %arr_type
108 %arr_v = getelementptr i8*, i8** %arr15, i32 0
109 store i8* getelementptr inbounds ([5 x i8], [5 x i8]* @string.10, i32 0, i32 0)
        , i8** %arr_v
110 %arr_v17 = getelementptr i8*, i8** %arr15, i32 1
111 store i8* getelementptr inbounds ([12 x i8], [12 x i8]* @string.11, i32 0, i32
        0), i8** %arr_v17
```

```
112   %arr_struct18 = getelementptr inbounds %arr_str_t, %arr_str_t* %arr_struct, i32
          0, i32 2
113   store i8** %arr15, i8*** %arr_struct18
114   %malloccall19 = tail call i8* @malloc(i32 ptrtoint (%obj* getelementptr (%obj,
          %obj* null, i32 1) to i32))
115   %obj20 = bitcast i8* %malloccall19 to %obj*
116   %key_name21 = getelementptr inbounds %obj, %obj* %obj20, i32 0, i32 1
117   store i8* getelementptr inbounds ([4 x i8], [4 x i8]* @key_name_str.12, i32 0,
          i32 0), i8** %key_name21
118   %value_typ22 = getelementptr inbounds %obj, %obj* %obj20, i32 0, i32 2
119   store i32 10, i32* %value_typ22
120   %value23 = getelementptr inbounds %obj, %obj* %obj20, i32 0, i32 10
121   %arrinobj = alloca %arr_str_t*
122   store %arr_str_t* %arr_struct, %arr_str_t** %arrinobj
123   store %arr_str_t** %arrinobj, %arr_str_t*** %value23
124   %malloccall24 = tail call i8* @malloc(i32 ptrtoint (%obj* getelementptr (%obj,
          %obj* null, i32 1) to i32))
125   %obj25 = bitcast i8* %malloccall24 to %obj*
126   %next = getelementptr inbounds %obj, %obj* %obj25, i32 0, i32 0
127   store %obj* %obj20, %obj** %next
128   %next26 = getelementptr inbounds %obj, %obj* %obj20, i32 0, i32 0
129   store %obj* %obj, %obj** %next26
130   %next27 = getelementptr inbounds %obj, %obj* %obj, i32 0, i32 0
131   store %obj* null, %obj** %next27
132   store %obj* %obj25, %obj** %create
133   %s_create = alloca i8*
134   %create28 = load %obj*, %obj** %create
135   %obj_stringify = call i8* (%obj*, ...) @obj_stringify(%obj* %create28)
136   store i8* %obj_stringify, i8** %s_create
137   %s_create29 = load i8*, i8** %s_create
138   %docker_host30 = load i8*, i8** %docker_host
139   %string_concat31 = call i8* (i8*, ...) @string_concat(i8* %docker_host30, i8*
          getelementptr inbounds ([19 x i8], [19 x i8]* @string.13, i32 0, i32 0))
140   %httppost32 = call i8* (i8*, i8*, ...) @httppost(i8* %string_concat31, i8*
          %s_create29)
141   %print_s33 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8],
          [4 x i8]* @str_fmt, i32 0, i32 0), i8* %httppost32)
142   ret i32 0
143 }
144
145 declare noalias i8* @malloc(i32)
```

6.2 Test Automation

We automated our testing by modifying the test_all.sh script from MicroC, adding a script for output file generation, and adding a Travis CI integration to our GitHub repository to automatically compile our compiler and run our entire test suite on every push. Our test files

followed the same format, with the correct results in a commented section at the bottom, so that we could write a script that would generate the correct .out and .err files for the test and fail scripts respectively. In order to do this, we made sure that this output followed the format :

Listing 46: Test Footer

```
1 /*
2 ****TEST****
3 correct output
4 */
```

We eventually added the output generation script into the test_all.sh script, so running the test_all script would generate the appropriate output files and run all of the tests. The Travis CI integration allowed us to further automate our testing and verify that all tests were running before merging branches into the main branch. Audrey came up with the idea to auto-generate the .out and .err files, and she and Walter worked on the script to accomplish that. Walter then set up the Travis CI integration.

7 Reflections

7.1 Lessons Learned

Audrey Copeland

It took me a while to get my head around the project and the model of the class. In order to not fall behind it was necessary to start working before the project, or specific parts of the project, made sense to me. This reversal of the normal interaction between lectures and homeworks/projects was initially challenging and frustrating. That said, I found that starting to work on specific pieces of code helped demystify what was going on and make my confusion more specific, which increased the benefit of lectures. I also found that it was easy to "get started early" while putting off the truly challenging work. For example, I messed around in the Ocaml interpreter early on in the class, but it wasnt until the OCaml homework (homework 2, for us) that I forced myself to go beyond variable assignment or using logic that had already been provided by MicroC. This was a mistake on my part.

The roles were helpful in informing my understanding of the project as a whole, but if you use them to determine what you should do any given week you will (probably) get lost. My official role was testing so I didnt have a specific role in the development of our grammar, which was at the beginning of the semester. So, I was somewhat fortunate to experience this feeling of being lost, and the accompanying fear of falling further behind, early on.

This was an incredibly challenging class because the material covered was complicated and diverse, and because each team and individual had to determine their own project goals, and the day-to-day tasks necessary to achieve those goals. It was also a rewarding class

that challenged me to ask myself and my teammates better and more specific questions and figure out how to answer those questions (I am still working on both parts of this) and use a range of programming languages and technologies with which I had varying degrees of prior familiarity (Ocaml, C, Bash, LLVM, JSON, git, the Docker API).

Walter Meyer

This was one of my final CS classes at Columbia and it was one of the most difficult that I have taken during my time here, but it was also one of my favorites. In particular, discovering what was behind a lot of the "magic" of the programming languages that I have seen and used was an enlightening and humbling experience. In particular, being given a significant amount of latitude with respect to what language and features to implement in the project was both exciting and challenging. I was forced to learn and confront many of the realities of implementation that underlie programming language features that I had, up until this point, taken at face value. Overall, working on the project made me further appreciate importance of creativity, time management, and collaboration in finding solutions to technical problems.

I also feel compelled to mention that I learned a lot about functional programming by way of OCaml. At first, I found it painful to learn despite being a senior with a decent amount of programming experience. In fact, it felt like I was learning to program for the first time again! This was a difficult, but invaluable experience in terms of challenging my preconceived notions about how to think about programming and problem solving in this context. Take advantage of the fact that Professor Edwards "forces" you to learn OCaml, even if you never use it or another functional language again, the experience will make you a better programmer (and you will finally be able to contribute to those functional programming conversations at cocktail parties).

Taimur Samee

Like all the best educational experiences, this class pulled the rug out from under me. After years of treating compilers like black boxes, I was shocked at the complex mechanisms within when I cracked open the lid. Each week brought a new reason to lose sleep, and at this point the error messages of my many failures, from nonsensical IR and segfaults to failed Git merges, have etched themselves into my brain like ghosted images on a monitor. But after learning the myriad ways we could fail, discovering the path to success was all the more satisfying.

The technical challenge was only half the battle, though. Tackling a group project of this scale gave me priceless insight into how to break down large problems, collaborate effectively, and keep each other active and motivated through the semester. I quickly found that my usual approach to work was a poor fit for the project, and am thankful to my teammates for checking me and showing me how to contribute effectively while Gantry was still in its early stages. More personally, I also learned that I make a pretty poor systems architect, but can turn smaller features and fixes around quickly and effectively, so I prioritised the many

tasks within our libraries and code generation where I knew I would perform. All in all, this journey has been about learning to ask the right questions: when faced with a new language, the 'how?' is almost as important as the 'what?', and when facing a mammoth task, it's best to seek the stages of the process where you can work your hardest. I'm grateful to my supportive and talented teammates, our ever-helpful TA Kai-Zhan, and Professor Edwards for teaching me to ask better questions.

For future teams reading this, I'll keep my advice short and sweet.

- Read ahead. To keep a manageable timeline without too many sleepless nights, you will need to be reading and **implementing** the material from the next three lectures at any given time
- Compromise when you need to. This is your language, and you should be ready to cut features that you're attached to in order to meet deadlines. Pouring a week of work into a feature just to throw out your code is painful, but it will help you stay humble and on-track
- Actively seek new sources of information. If you start early, and assess what works best for you personally, you will build a list of bookmarks that will save your project at crunch-time

Rizwan Syed

Working on this project was like running with ankle weights. It was difficult, but now that the weights are off, I feel remarkably empowered. This was the first time I worked with a team on a major programming project from conception to execution, and in addition to gaining a rooted knowledge of how compilers work, I am especially grateful for the skills I learned in project management. I also developed a working proficiency with Git (far beyond what we learned in Advanced Programming) that has already proven invaluable in my work beyond this class.

This project has also helped me appreciate how people are good at different things – and to use an analogy from class, forcing square pegs into round holes does not tend to end very well. In particular, while I made modest contributions to our language grammar and code, I found my teammates to be far better programmers than myself, and that I could spend my time best by focusing on our project management and documentation. I found it important to make efforts in many directions, but to reflect often to iterate effectively.

And finally, I learned the importance of having a great team. I was not only humbled by the sheer brilliance of all three teammates, but calmed by our cohesion and civility. If I could make one recommendation to future teams, it would be to find people who complement your skills and with whom you will love to work throughout the course of your project. If you get this part down, you are already halfway to greatness.

7.2 Advice for Future Teams

The nature of the project is such that it will challenge your ability to think creatively as much, if not more so, than it will challenge your technical abilities. Often CS assignments and projects give you fairly formalized and detailed rubrics about how to approach solving a problem. While that is often intentional, especially for introductory courses, having this security blanket taken away can be a bit of a shock. However, Professor Edwards does provide what amounts to an excellent rubric for you to start from in MicroC. However, by the time you are able to wrap your head around how MicroC works, you have already submitted your Language Proposal and Language Reference Manual that have very likely promised polymorphic unicorns and garbage-collected rainbows. This is where things get interesting. You will be forced to make compromises with respect to features and hopefully, push yourself to implement something that matches your finite time and productivity constraints, which are directly proportional to how early and often you work on the project throughout the semester.

That is, even if you consider yourself a fast or talented coder, the point of the project is not to simply crank out something that "works", but rather to explore ideas - often ones that turn out to be incredibly bad. It became clear to us that the identification of choices and design decisions as bad ideas and undergoing revisions as a result were markers for progress, not just our mistakes.

Stay Focused
We found that it was helpful to refer back to our original application-specific goal frequently to keep things in perspective. Specifically, when you are in the depths of your language implementation and hoping that your tears will fix that shift reduce-conflict or properly dereference that LLVM struct pointer, it is easy to lose sight of what you originally intended to accomplish. Ultimately, you will need to stem-the-tide of "feature-creep" by discarding or delaying features to stay focused on what are core-features. It seemed to us that Professor Edwards suggests that you implement a domain-specific language, in part, to limit the scope and complexity of your language and to keep you focused on a concrete deliverable. We kept on task by writing a program in our language that was able to interface with the Docker API. This helped us stay focused instead of wasting time implementing non-critical features.

Prioritize This Class
Build your schedule around this class because completing the project requires a significant time contribution from 4-5 people. We estimate that our team spent 50+ hours per week working on this project (not including class and homework). If you cannot make this time contribution, you are doing a disservice to yourself and to your team. Also note that the project time requirements are static and that your teammates' weekly availability may not be. So, be prepared to hold everyone accountable and step in for a team member that may have to pull back at a certain time because of an academic commitment or simply disappear into the wilderness to go on a vision-quest before they interview for a position at a Machine Learning cupcake Start-Up. Also, take note of past projects that had only one or two mem-

bers that were still completed.[7]

Roles

In our team, the assigned roles did not align with our work. To put it bluntly, had we all stuck solely to our assigned roles, we would not have been able to realize any of the initial goals we outlined in our proposal. Professor Edwards says in class that the roles should not be taken literally, but they can be tempting boundaries for limiting your individual contribution. Resist the urge to stay confined within your role. In our experience, it is impossible to code in a vacuum for this project. The reality is that while implementing a compiler has high-level discrete components that need to be implemented, each person cannot isolate themselves within these technical confines while working on a team. The project requires that each individual understand and contribute to parts of the project that are well outside of the scope of their role. With that said, the roles were helpful as a framework for understanding the responsibilities the team had to take on as a whole, especially at the beginning of the semester, when we were still trying to figure out what the project would entail.

[7] SetC

8 Appendix

8.1 Code Listing

8.1.1 Introduction

Listing 47: README

```
The Gantry Compiler README

----

Requirements:

OCaml 4.02.3 (or higher)
LLVM 3.8
m4 1.4.17 (or higher)
gcc 5.4 (or higher)
make 4.1 (or higher)
libcurl4
pkg-config

Other versions other than the aforementioned may work but have not been tested.
Additionally, you should be able to get Gantry compiled on the Operating System
of choice provided you have the prerequisites installed.

-----

Installation on Ubuntu 16.04 LTS:

sudo apt-get update -y
sudo apt-get install -y ocaml opam m4 pkg-config llvm-3.8 libcurl4-gnutls-dev
opam init --auto-setup
opam install -y ocamlfind llvm.3.8

----

Compiling the Compiler:

From the gantry working directory (likely where this README is) run:

make

This will produce the compiler binary 'gantry.native'.

----

Running Compiler Tests:
```

```
41
42 make clean && make
43 ./test_all.sh
44
45 This will compile and execute all tests inside of the tests/ directory.
46 If these work, you can assume the compiler is working on your system
47 and you can begin compiling your own Gantry (.gty) programs.
48
49 ----
50
51 Compiling a Gantry Program to a native binary:
52
53 Gantry provides a convenient wrapper script that will compile your source (.gty)
54 program and perform the appropriate linking with libraries that Gantry uses.
55
56 ./gantry_comp.sh your_program.gty
57
58 ----
59
60 Compiling a Gantry program to LLVM IR:
61
62 If you'd like to see the LLVM IR output from
63 the compiler you can run the following to send
64 it to stdout:
65
66 ./gantry.native < program.gty
```

8.1.2 Core Code

Makefile

Listing 48: Makefile Code

```
# Make sure ocamlbuild can find opam-managed packages: first run
#
# eval `opam config env`

# Easiest way to build: using ocamlbuild, which in turn uses ocamlfind


all : gantry.native gantrylib_string.o gantrylib_http.o gantrylib_obj.o

gantry.native :
	ocamlbuild -use-ocamlfind -pkgs str,llvm,llvm.analysis -cflags -w,+a-4 \
		gantry.native

# "make clean" removes all generated files

.PHONY : clean
clean :
	ocamlbuild -clean
	rm -rf testall.log *.diff gantry scanner.ml parser.ml parser.mli
	rm -rf printbig
	rm -rf string_concat string_concat.o
	rm -rf slice slice.o
	rm -rf gantrylib_http
	rm -rf *.cmx *.cmi *.cmo *.cmx *.o *.s *.ll tests/*.out *.exe *.err

# More detailed: build using ocamlc/ocamlopt + ocamlfind to locate LLVM

OBJS = ast.cmx codegen.cmx parser.cmx scanner.cmx semant.cmx gantry.cmx

gantry : $(OBJS)
	ocamlfind ocamlopt -linkpkg -package str,llvm,llvm.analysis $(OBJS) -o
	    gantry

scanner.ml : scanner.mll
	ocamllex scanner.mll

parser.ml parser.mli : parser.mly
	ocamlyacc -v parser.mly

%.cmo : %.ml
	ocamlc -c $<

%.cmi : %.mli
```

```
43 	ocamlc -c $<
44 
45 %.cmx : %.ml
46 	ocamlfind ocamlopt -c -package llvm $<
47 
48 CC=gcc
49 CFLAGS=-Wall -c
50 LDFLAGS=-L/usr/lib/x86_64-linux-gnu -lcurl
51 
52 gantrylib_http.o : gantrylib_http.c
53 	$(CC) $(CFLAGS) gantrylib_http.c
54 
55 # Testing the "printbig" example
56 
57 # printbig : printbig.c
58 # cc -o printbig -DBUILD_TEST printbig.c
59 
60 # Testing the "stringconcat" example
61 
62 gantrylib_string : gantrylib_string.c
63 	cc -o gantrylib_string -DBUILD_TEST gantrylib_string.c
64 
65 string_concat : string_concat.c
66 	cc -o string_concat -DBUILD_TEST string_concat.c
67 
68 slice : slice.c
69 	cc -o slice -DBUILD_TEST slice.c
70 
71 gantrylib_obj: gantrylib_obj.c
72 	cc -o gantrylib_obj -DBUILD_TEST gantrylib_obj.c
73 
74 ### Generated by "ocamldep *.ml *.mli" after building scanner.ml and parser.ml
75 ast.cmo :
76 ast.cmx :
77 codegen.cmo : ast.cmo
78 codegen.cmx : ast.cmx
79 gantry.cmo : semant.cmo scanner.cmo parser.cmi codegen.cmo ast.cmo
80 gantry.cmx : semant.cmx scanner.cmx parser.cmx codegen.cmx ast.cmx
81 parser.cmo : ast.cmo parser.cmi
82 parser.cmx : ast.cmx parser.cmi
83 scanner.cmo : parser.cmi
84 scanner.cmx : parser.cmx
85 semant.cmo : ast.cmo
86 semant.cmx : ast.cmx
87 parser.cmi : ast.cmo
88 
89 # Building the tarball
90 
```

```
#TESTS = add1 arith1 arith2 arith3 fib for1 for2 func1 func2 func3 \
    func4 func5 func6 func7 func8 gcd2 gcd global1 global2 global3 \
    hello if1 if2 if3 if4 if5 local1 local2 ops1 ops2 var1 var2 \
    while1 while2 printbig

#FAILS = assign1 assign2 assign3 dead1 dead2 expr1 expr2 for1 for2 \
    for3 for4 for5 func1 func2 func3 func4 func5 func6 func7 func8 \
    func9 global1 global2 if1 if2 if3 nomain return1 return2 while1 \
    while2

#TESTFILES = $(TESTS:%=test-%.mc) $(TESTS:%=test-%.out) \
           $(FAILS:%=fail-%.mc) $(FAILS:%=fail-%.err)

#TARFILES = ast.ml codegen.ml Makefile microc.ml parser.mly README scanner.mll \
       semant.ml testall.sh $(TESTFILES:%=tests/%) printbig.c arcade-font.pbm \
       font2c

#microc-llvm.tar.gz : $(TARFILES)
# cd .. && tar czf microc-llvm/microc-llvm.tar.gz \
              $(TARFILES:%=microc-llvm/%)
```

gantry.ml

Listing 49: Top-Level Code

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

type action = Ast | LLVM_IR | Compile

let _ =
  let action =
    if Array.length Sys.argv > 1 then
      List.assoc Sys.argv.(1)
        [("-a", Ast); (* Print the AST only *)
         ("-l", LLVM_IR); (* Generate LLVM, don't check *)
         ("-c", Compile)] (* Generate, check LLVM IR *)
    else Compile in
      let lexbuf = Lexing.from_channel stdin in
        let ast = Parser.program Scanner.token lexbuf in
          Semant.check ast;
          match action with
            | Ast -> print_string (Ast.string_of_program ast)
            | LLVM_IR ->
                print_string (Llvm.string_of_llmodule (Codegen.translate ast))
            | Compile ->
                let m = Codegen.translate ast in
                       Llvm_analysis.assert_valid_module m;
                       print_string (Llvm.string_of_llmodule m);;
```

scanner.mll

Listing 50: Scanner Code

```
(*
 * Gantry: Scanner
 * Author: Walter Meyer
 *)

{ open Parser }

let identifier = ['a'-'z' 'A'-'Z' '0'-'9']['a'-'z' 'A'-'Z' '0'-'9' '_' '.']*

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*" { comment lexbuf } (* Comments *)
| "//" { new_comment lexbuf } (* New-Style Comments *)
| '(' { LPAREN }
| ')' { RPAREN }
| '{' { LBRACE }
| '}' { RBRACE }
| '[' { LBRACK }
| ']' { RBRACK }
| ';' { SEMI }
| ',' { COMMA }
| ':' { COLON }
| '.' { PERIOD }
| '|' { BAR }

(* Arithmetic Operators (Binary and Unary) *)
| '+' { PLUS }
| '-' { MINUS }
| "++" { INCREM }
| "--" { DECREM }

(* Arithmetic Operators (Binary) *)
| '*' { TIMES }
| '/' { DIVIDE }
| '=' { ASSIGN }
| "==" { EQ }
| "!=" { NEQ }

(* Relational Operators *)
| '<' { LT }
| "<=" { LEQ }
| ">" { GT }
| ">=" { GEQ }

```

```
45 | (* Logical Operators *)
46 | | "&&" { AND }
47 | | "||" { OR }
48 | | "!" { NOT }
49 |
50 | (* String Concatenation *)
51 | | "^" { CONCAT }
52 |
53 | (* Flow Control *)
54 | | "if" { IF }
55 | | "else" { ELSE }
56 | | "for" { FOR }
57 | | "while" { WHILE }
58 | | "continue" { CONTINUE }
59 | | "break" { BREAK }
60 | | "return" { RETURN }
61 |
62 | (* Keywords *)
63 | | "int" { INT }
64 | | "float" { FLOAT }
65 | | "bool" { BOOL }
66 | | "null" { NULL }
67 | | "true" { TRUE }
68 | | "false" { FALSE }
69 | | "object" { OBJECT }
70 | | "array" { ARRAY }
71 | | "string" { STRING }
72 |
73 | (* Strings *)
74 | | '"' { read_string (Buffer.create 10) lexbuf }
75 |
76 | (* Literals *)
77 | | ['0'-'9']+ as lxm { INTLIT(int_of_string lxm) }
78 | | ['0'-'9']+['.']['0'-'9']+ as lxm { FLOATLIT(float_of_string lxm) }
79 |
80 | (* Identifiers *)
81 | | identifier as lxm { ID(lxm) }
82 |
83 | (* EOF and Error Handling *)
84 | | eof { EOF }
85 | | _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }
86 |
87 | (* Comments *)
88 | and comment = parse
89 |   "*/" { token lexbuf }
90 | | _ { comment lexbuf }
91 |
92 | and new_comment = parse
```

```
  '\n' { token lexbuf }
| _ { new_comment lexbuf }

(* String Literals
 * Recursive read_string modified from
 * https://realworldocaml.org/v1/en/html/parsing-with-ocamllex-and-menhir.html
 * accept '\r', '\t', '\n', '\b', '\f', '\"', '\\'
 *)
and read_string buf =
  parse
  | '"' { STRLIT (Buffer.contents buf) }
  | '\\' 'r' { Buffer.add_char buf '\r'; read_string buf lexbuf }
  | '\\' 't' { Buffer.add_char buf '\t'; read_string buf lexbuf }
  | '\\' 'n' { Buffer.add_char buf '\n'; read_string buf lexbuf }
  | '\\' 'b' { Buffer.add_char buf '\b'; read_string buf lexbuf }
  | '\\' 'f' { Buffer.add_char buf '\012'; read_string buf lexbuf }
  | '\\' '"' { Buffer.add_char buf '\"'; read_string buf lexbuf }
  | '\\' '\\' { Buffer.add_char buf '\\'; read_string buf lexbuf }
  | [^ '"' '\\']+
    { Buffer.add_string buf (Lexing.lexeme lexbuf);
      read_string buf lexbuf
    }
  | _ { raise (Failure ("Illegal string character: " ^ Lexing.lexeme lexbuf)) }
  | eof { raise (Failure ("String is not terminated")) }
```

parser.mly

Listing 51: Parser Code

```
/*
 * Ocamlyacc parser for Gantry
 * Author: Walter Meyer
 * Contributor: Audrey Copeland
 */

%{
open Ast
%}

/* Tokens / Terminals */
%token LPAREN RPAREN LBRACE RBRACE LBRACK RBRACK SEMI COLON COMMA PERIOD BAR
%token QUOTE
%token PLUS MINUS INCREM DECREM
%token TIMES DIVIDE ASSIGN EQ NEQ
%token LT LEQ GT GEQ
%token AND OR NOT
%token CONCAT
%token IF ELSE FOR WHILE CONTINUE BREAK RETURN
%token INT FLOAT OBJECT ARRAY STRING BOOL NULL
%token TRUE FALSE
%token <int> INTLIT
%token <float> FLOATLIT
%token <string> ID STRLIT
%token EOF

/* Precedence Rules */
%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN COLON
%left OR
%left AND
%left INCREM DECREM
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%nonassoc UMINUS
%right NOT
%left CONCAT
%left PERIOD LBRACK

%start program
%type <Ast.program> program
```

```
45
46 %%
47
48 /* CFG */
49 program:
50     declaration_list EOF { (List.rev (fst $1)), (List.rev (snd $1)) }
51
52 /* Build up a tuple of ordered lists for stmts and fdecls for the AST */
53 declaration_list:
54     /* empty */ { [], [] }
55     | declaration_list global_declaration { $2 :: fst $1, snd $1 }
56     | declaration_list function_declaration { fst $1, $2 :: snd $1 }
57
58 global_declaration:
59     type_spec ID SEMI { ($1, $2) }
60
61 function_declaration:
62     type_spec ID LPAREN func_param_list_opt RPAREN LBRACE statement_list RBRACE
63       { { type_spec = $1;
64           f_id = $2;
65           f_params = $4;
66           f_statements = List.rev $7;
67       } }
68
69 type_spec:
70     INT { Int }
71     | FLOAT { Float }
72     | OBJECT { Object }
73     | STRING { String }
74     | BOOL { Bool }
75     | NULL { Null }
76     | INT ARRAY { Int_Array }
77     | FLOAT ARRAY { Float_Array }
78     | OBJECT ARRAY { Object_Array }
79     | STRING ARRAY { String_Array }
80     | BOOL ARRAY { Bool_Array }
81
82 func_param_list_opt:
83      /* nothing */ { [] }
84     | func_param_list { List.rev $1 }
85
86 func_param_list:
87     type_spec ID { [($1, $2)] }
88     | func_param_list COMMA type_spec ID { ($3, $4) :: $1 }
89
90 function_expression:
91     | ID LPAREN expression_list_opt RPAREN { FunExp($1, $3) }
92
```

```
93  statement_list:
94      /* empty */ { [] }
95      | statement_list statement { $2 :: $1 }
96
97  statement:
98      for_statement { $1 }
99      | if_statement { $1 }
100     | while_statement { $1 }
101     | jump_statement { $1 }
102     | expression_statement { $1 }
103     | LBRACE statement_list RBRACE { Block(List.rev $2) }
104
105 expression_statement:
106     expression SEMI { Expr($1) }
107
108 expression:
109     ID { Id($1) }
110     | access_expression { $1 }
111     | constant { $1 }
112     | array_expression { $1 }
113     | object_expression { $1 }
114     | arithmetic_expression { $1 }
115     | comparison_expression { $1 }
116     | logical_expression { $1 }
117     | string_concat_expression { $1 }
118     | assignment_expression { $1 }
119     | function_expression { $1 }
120
121 array_expression:
122     LBRACK expression_list_opt RBRACK { ArrExp($2) }
123
124 access_expression:
125     expression LBRACK expression RBRACK { ArrAcc($1, $3) }
126
127 assignment_expression:
128     expression ASSIGN expression { Assign($1, $3) }
129     | type_spec ID ASSIGN expression { AssignDecl($1, $2, $4) }
130     | type_spec ID COLON expression { KeyVal($1, $2, $4) }
131
132 object_expression:
133     LBRACE BAR expression_list_opt BAR RBRACE
134         { ObjExp($3) }
135
136 expression_list_opt:
137     /* empty */ { [] }
138     | expression_list { List.rev $1 }
139
140 expression_list:
```

```
141     expression { [$1] }
142     | expression_list COMMA expression { $3 :: $1 }
143
144 arithmetic_expression:
145     expression PLUS expression { Binop($1, Add, $3) }
146     | expression MINUS expression { Binop($1, Sub, $3) }
147     | expression TIMES expression { Binop($1, Mult, $3) }
148     | expression DIVIDE expression { Binop($1, Div, $3) }
149     | expression INCREM { Unop(Inc, $1) }
150     | expression DECREM { Unop(Dec, $1) }
151     | MINUS expression %prec UMINUS { Unop(Neg, $2) }
152
153 comparison_expression:
154     expression LT expression { Binop($1, Lt, $3) }
155     | expression GT expression { Binop($1, Gt, $3) }
156     | expression LEQ expression { Binop($1, Leq, $3) }
157     | expression GEQ expression { Binop($1, Geq, $3) }
158     | expression EQ expression { Binop($1, Eq, $3) }
159     | expression NEQ expression { Binop($1, Neq, $3) }
160
161 logical_expression:
162     expression AND expression { Binop($1, And, $3) }
163     | expression OR expression { Binop($1, Or, $3) }
164     | NOT expression { Unop(Not, $2) }
165
166 string_concat_expression:
167     expression CONCAT expression { Binop($1, Conc, $3) }
168
169 for_statement:
170     FOR LPAREN expression SEMI expression SEMI expression RPAREN statement
171       { For($3, $5, $7, $9) }
172
173 if_statement:
174     IF LPAREN expression RPAREN statement %prec NOELSE
175       { If($3, $5, Block([])) }
176     | IF LPAREN expression RPAREN statement ELSE statement
177       { If($3, $5, $7) }
178
179 while_statement:
180     WHILE LPAREN expression RPAREN statement
181       { While($3, $5) }
182
183 jump_statement:
184       RETURN SEMI { Return(Noexpr) }
185     | RETURN expression SEMI { Return($2) }
186
187 constant:
188     TRUE { BoolLit(true) }
```

```
189     | FALSE { BoolLit(false) }
190     | literal { $1 }
191
192 literal:
193     INTLIT { IntLit($1) }
194     | FLOATLIT { FloatLit($1) }
195     | STRLIT { StrLit($1) }
```

ast.ml

Listing 52: AST Code

```
(*
 * Gantry: Abstract Syntax Tree
 * Author: Walter Meyer
 * Contributor: Audrey Copeland
 *)

type op =
  | Add | Sub | Mult | Div
  | Eq | Neq | Geq | Leq | Gt | Lt
  | And | Or | Conc

type uop =
  Not | Neg | Inc | Dec

type typ =
    Int | Float | Object | String | Bool | Null |
    Int_Array | Float_Array | Object_Array | String_Array | Bool_Array

type typ_bind = typ * string

type expression =
         IntLit of int
       | FloatLit of float
       | StrLit of string
       | BoolLit of bool
       | Id of string
       | ArrAcc of expression * expression
       | Binop of expression * op * expression
       | Unop of uop * expression
       | Assign of expression * expression
       | AssignDecl of typ * string * expression
       | FunExp of string * expression list
       | KeyVal of typ * string * expression
       | ArrExp of expression list
       | ObjExp of expression list
       | Noexpr

type statement =
         Block of statement list
       | Expr of expression
       | Return of expression
       | If of expression * statement * statement
       | For of expression * expression * expression * statement
       | While of expression * statement
```

```
45
46 type function_decl = {
47    type_spec : typ;
48    f_id : string;
49    f_params : typ_bind list;
50    f_statements : statement list;
51  }
52
53 type program = typ_bind list * function_decl list
54
55 (* Expression to String for Variable Resolution in Codegen *)
56 let expr_to_str = function
57          Id(s) -> s
58        | _ -> ""
59
60 (* Pretty-printing functions *)
61
62 let string_of_op = function
63          Add -> "+"
64        | Sub -> "-"
65        | Mult -> "*"
66        | Div -> "/"
67        | Eq -> "=="
68        | Neq -> "!="
69        | Lt -> "<"
70        | Leq -> "<="
71        | Gt -> ">"
72        | Geq -> ">="
73        | And -> "&&"
74        | Or -> "||"
75        | Conc -> "^"
76
77 let string_of_uop = function
78          Neg -> "-"
79        | Not -> "!"
80        | Inc -> "++"
81        | Dec -> "--"
82
83 let string_of_typ = function
84          Int -> "int"
85        | Float -> "float"
86        | Object -> "object"
87        | String -> "string"
88        | Bool -> "bool"
89        | Null -> "null"
90        | Int_Array -> "int array"
91        | Float_Array -> "float array"
92        | Object_Array -> "object array"
```

```
93        | String_Array -> "string array"
94        | Bool_Array -> "bool array"
95
96  let rec string_of_expression = function
97           IntLit(i) -> string_of_int i
98        | FloatLit(f) -> string_of_float f
99        | StrLit(s) -> "\"" ^ s ^ "\""
100       | BoolLit(true) -> "true"
101       | BoolLit(false) -> "false"
102       | Id(s) -> s
103       | ArrAcc(e1, e2) -> string_of_expression e1 ^ "[" ^ string_of_expression
            e2 ^ "]"
104       | Binop(e1, o, e2) -> string_of_expression e1 ^ " " ^ string_of_op o ^ " "
             ^ string_of_expression e2
105       | Unop(o, e) -> (match o with
106                    Inc | Dec -> string_of_expression e ^ string_of_uop o
107                    | _ -> string_of_uop o ^ string_of_expression e)
108       | KeyVal(t, k, e) -> " " ^ string_of_typ t ^ " " ^ k ^ " : " ^
            string_of_expression e
109       | ArrExp(el) -> "[ " ^ String.concat ", " (List.map string_of_expression
            el) ^ " ]"
110       | ObjExp(el) -> "{| " ^ String.concat ", " (List.map string_of_expression
            el) ^ " |}"
111       | Assign(e1, e2) -> string_of_expression e1 ^ " = " ^ string_of_expression
             e2
112       | AssignDecl(t, v, e) -> string_of_typ t ^ " " ^ v ^ " = " ^
            string_of_expression e
113       | FunExp(f, el) -> f ^ "(" ^ String.concat ", " (List.map
            string_of_expression el) ^ ")"
114       | Noexpr -> ""
115
116 let rec string_of_statement = function
117          Block(statements) ->
118               "{\n" ^ String.concat "" (List.map string_of_statement statements)
                    ^ "}\n"
119       | Expr(expression) -> string_of_expression expression ^ ";\n";
120       | Return(expression) -> "return " ^ string_of_expression expression ^ ";\n
            ";
121       | If(e1, s1, Block([])) -> "if (" ^ string_of_expression e1 ^ ")\n" ^
122                                                  string_of_statement s1 ^ "\n"
123       | If(e1, s1, s2) -> "if (" ^ string_of_expression e1 ^ ")\n" ^
124                                           string_of_statement s1 ^ "\n" ^
125                                           "else\n" ^ string_of_statement s2 ^ "\n
                                                "
126       | For(e1, e2, e3, s) -> "for (" ^ string_of_expression e1 ^ " ; " ^
            string_of_expression e2 ^ " ; "
127                                      ^ string_of_expression e3 ^ ") " ^
                                         string_of_statement s
```

```
128         | While(e, s) -> "while (" ^ string_of_expression e ^ ")\n" ^
              string_of_statement s ^ "\n"
129
130 let string_of_global (t, id) = string_of_typ t ^ " " ^ id ^ ";\n"
131
132 let string_of_typ_bind (t, id) = string_of_typ t ^ " " ^ id
133
134 let string_of_fdecl fdecl =
135         string_of_typ fdecl.type_spec ^ " "
136         ^ fdecl.f_id ^ " " ^ "(" ^ String.concat ", " (List.map string_of_typ_bind
              fdecl.f_params) ^ ")\n{\n"
137         ^ String.concat "" (List.map string_of_statement fdecl.f_statements)
138         ^ "}\n"
139
140 let string_of_program (globals, fdecls) =
141         String.concat "" (List.map string_of_global globals) ^ "\n"
142         ^ String.concat "\n" (List.map string_of_fdecl fdecls)
```

semant.ml

Listing 53: Semant Code

```
(*
 * Gantry: Semantic AST
 * Author: Audrey Copeland
 * Contributors: Walter Meyer, Taimur Samee
 *)

open Ast

module StringMap = Map.Make(String)


(* Create Hash Table for symbol table *)
let symbols : (string, Ast.typ) Hashtbl.t = Hashtbl.create 10;;

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

   Checks each global _statement_ , then check each function *)

let check (globals, functions) =

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

  (* Check not void, used for functions and variables *)
  let check_not_null exceptf = function
      (Null, n) -> raise (Failure (exceptf n))
    | _ -> ()
  in

  (* Are two sides of assignment compatible? *)
  let check_assign lvaluet rvaluet err =
     match lvaluet, rvaluet with
       (Object, _) -> lvaluet
     | (_, Object) -> lvaluet
     | (_, _) -> if lvaluet == rvaluet then lvaluet else raise err
  in

```

```
45  let check_kv_assign lvaluet rvaluet err =
46     match lvaluet, rvaluet with
47       (_, _) -> if lvaluet == rvaluet then Object else raise err
48  in
49
50  let add_local (t, n) =
51    ignore(Hashtbl.add symbols n t);
52  in
53
54  let check_decl lt lv err =
55     if lt == Object then
56        (if (Hashtbl.mem symbols lv) then (raise err)
57        else
58          add_local(lt, lv))
59     else
60        add_local(lt, lv);
61  in
62
63  (**** Checking Global Variables ****)
64  List.iter (check_not_null (fun n -> "illegal null global " ^ n)) globals;
65
66  report_duplicate (fun n -> "duplicate global " ^ n) (List.map snd globals);
67
68  (**** Checking Functions ****)
69  (* Checks to make sure print functions are defined *)
70  if List.mem "print_s" (List.map (fun fd -> fd.f_id) functions)
71  then raise (Failure ("function print string may not be defined")) else ();
72
73  if List.mem "print_i" (List.map (fun fd -> fd.f_id) functions)
74  then raise (Failure ("function print integer may not be defined")) else ();
75
76  if List.mem "print_d" (List.map (fun fd -> fd.f_id) functions)
77  then raise (Failure ("function print float may not be defined")) else ();
78
79  if List.mem "print_k" (List.map (fun fd -> fd.f_id) functions)
80  then raise (Failure ("function print key may not be defined")) else ();
81
82  report_duplicate (fun n -> "duplicate function " ^ n)
83    (List.map (fun fd -> fd.f_id) functions);
84
85  (* Function declaration for a named function *)
86
87  (* Built-ins: print, arrify, objectify, jsonify, length, slice, tostring,
       httpget, httppost *)
88  let built_in_decls = StringMap.add "print_s"
89     { type_spec = Null; f_id = "print_s"; f_params = [(String, "x")] ;
         f_statements = [] }
90     (StringMap.add "print_i"
```

```
91      { type_spec = Null; f_id = "print_i"; f_params = [(Int, "x")] ; f_statements
           = [] }
92      (StringMap.add "print_d"
93      { type_spec = Null; f_id = "print_d"; f_params = [(Float, "x")] ;
           f_statements = [] }
94      (StringMap.add "print_b"
95      { type_spec = Null; f_id = "print_b"; f_params = [(Bool, "x")] ; f_statements
            = [] }
96      (StringMap.add "print_k"
97      { type_spec = Null; f_id = "print_k"; f_params = [(Object, "x")] ;
           f_statements = [] }
98      (StringMap.add "get_string"
99      { type_spec = String; f_id = "get_string"; f_params = [(String, "x")] ;
           f_statements = [] }
100     (StringMap.add "stoint"
101     { type_spec = Int; f_id = "length"; f_params = [(String, "x")] ; f_statements
            = [] }
102     (StringMap.add "length"
103     { type_spec = Int; f_id = "length"; f_params = [(String, "x")] ; f_statements
            = [] }
104     (StringMap.add "slice"
105     { type_spec = String; f_id = "slice"; f_params = [(String, "x"); (Int, "y");
           (Int, "z")] ; f_statements = [] }
106     (StringMap.add "stringcmp"
107     { type_spec = Int; f_id = "stringcmp"; f_params = [(String, "x"); (String, "y
           ")] ; f_statements = [] }
108     (StringMap.add "string_length"
109     { type_spec = Int; f_id = "string_length"; f_params = [(String, "x")] ;
           f_statements = [] }
110     (StringMap.add "arr_length"
111     { type_spec = Int; f_id = "arr_length"; f_params = [(String, "x")] ;
           f_statements = [] }
112     (StringMap.add "tostring"
113     { type_spec = String; f_id = "tostring"; f_params = [(String, "x")] ;
           f_statements = [] }
114     (StringMap.add "nap"
115     { type_spec = Int; f_id = "nap"; f_params = [(Int, "x")] ; f_statements = []
           }
116     (StringMap.add "httpget"
117     { type_spec = String; f_id = "httpget"; f_params = [(String, "x")] ;
           f_statements = [] }
118     (StringMap.add "arr_stringify"
119     { type_spec = String; f_id = "arr_stringify"; f_params = [(String , "x")] ;
           f_statements = [] }
120     (StringMap.add "obj_stringify"
121     { type_spec = String; f_id = "obj_stringify"; f_params = [(Object, "x")] ;
           f_statements = [] }
122     (StringMap.add "obj_addkey"
```

```
123     { type_spec = Object; f_id = "obj_addkey"; f_params = [(Object, "x"); (String
            , "y"); (Int, "z"); (String, "a")] ; f_statements = [] }
124     (StringMap.singleton "httppost"
125     { type_spec = String; f_id = "httppost"; f_params = [(String, "x");(String, "
            x")] ; f_statements = [] }))))))))))))))))))
126   in
127
128   (* Add built in functions to list of function declaration list *)
129   let function_decls = List.fold_left (fun m fd -> StringMap.add fd.f_id fd m)
          built_in_decls functions
130
131   in
132
133   (* Check that function exists in function declaration list *)
134   let function_decl s = try StringMap.find s function_decls
135        with Not_found -> raise (Failure ("unrecognized function " ^ s))
136   in
137
138   (* Ensure "main is defined" *)
139   let _ = function_decl "main" in
140
141   let check_function func =
142     Hashtbl.clear symbols;
143
144     (* Check that function does not have null parameters *)
145     List.iter (check_not_null (fun n -> "illegal null formal " ^ n ^
146       " in " ^ func.f_id)) func.f_params;
147
148     (* Report if there are duplicate function parameters *)
149     report_duplicate (fun n -> "duplicate formal " ^ n ^ " in " ^ func.f_id)
150       (List.map snd func.f_params);
151
152     (* Add globals and function parameters to symbol table *)
153     List.iter (fun (a, b) -> Hashtbl.add symbols b a) (globals @ func.f_params);
154
155     let type_of_identifier s =
156         try Hashtbl.find symbols s
157         with Not_found -> raise (Failure ("undeclared identifier " ^ s))
158     in
159
160     (* Return the type of an expression or throw an exception *)
161     let rec expression = function
162          IntLit _ -> Int
163        | FloatLit _ -> Float
164        | StrLit _ -> String
165        | BoolLit _ -> Bool
166        | Id s -> if not (String.contains s '.') then (type_of_identifier s) else
             (Object)
```

```
167         | Binop(e1, op, e2) as e -> let t1 = expression e1 and t2 = expression e2
              in
168          (match op with
169             Add | Sub | Mult | Div when t1 = Int && t2 = Int -> Int
170           | Add | Sub | Mult | Div when t1 = Float && t2 = Float -> Float
171           | Eq | Neq when t1 = t2 -> Bool
172           | Lt | Leq | Gt | Geq when t1 = Int && t2 = Int -> Bool
173           | Lt | Leq | Gt | Geq when t2 = Float && t2 = Float -> Bool
174           | And | Or when t1 = Bool && t2 = Bool -> Bool
175           | Conc when t1 = String && t2 = String -> String
176           | _ -> raise (Failure ("illegal binary operator " ^ string_of_typ t1 ^ "
                  " ^ string_of_op op ^ " " ^ string_of_typ t2 ^ " in " ^
                string_of_expression e))
177         )
178         | Unop (op, e) as ex -> let t = expression e in
179          (match op with
180           Neg when t = Int -> Int
181           | Neg when t = Float -> Float
182           | Not when t = Bool -> Bool
183           | Inc when t = Int -> Int
184           | Dec when t = Int -> Int
185           | _ -> raise (Failure ("Illegal unary operator " ^ string_of_uop op ^
                string_of_typ t ^ " in " ^ string_of_expression ex)))
186         | Assign(e1, e2) as ex -> let lt = expression e1
187                                   and rt = expression e2 in
188               check_assign lt rt (Failure("illegal assignment " ^ string_of_typ
                    lt ^
189               " = " ^ string_of_typ rt ^ " in " ^ string_of_expression ex))
190         | AssignDecl (t, n, e) as ex ->
191               check_decl t n (Failure("Object " ^ n ^ " declared twice"));
192               let lt = type_of_identifier n
193               and rt = expression e in
194               if (String.contains n '.') then
195                     (raise (Failure ("Can not declare multi-level variable " ^ n
                          ^ " in expression " ^ string_of_expression ex)))
196               else
197                     (check_assign lt rt (Failure("illegal assignment " ^
                        string_of_typ lt ^
198               " = " ^ string_of_typ rt ^ " in " ^ string_of_expression ex)))
199         | FunExp(f_id, actuals) as funexp ->
200               let fd = function_decl f_id in
201               if List.length actuals != List.length fd.f_params then
202               raise (Failure ("expecting " ^ string_of_int
203                  (List.length fd.f_params) ^ " arguments in " ^
                      string_of_expression funexp))
204               else
205                 List.iter2
206                   (fun (ft, _) e -> let et = expression e in
```

```
207                     let obj_addkey = Str.regexp "obj_addkey" in
208                     let arr_length = Str.regexp "arr_length" in
209                     let arr_stringify = Str.regexp "arr_stringify" in
210                     if not (String.contains (string_of_expression e) '.' )
211                     && not (Str.string_match obj_addkey f_id 0)
212                     && not (Str.string_match arr_stringify f_id 0)
213                     && not (Str.string_match arr_length f_id 0) then (
214                     ignore (check_assign ft et
215                       (Failure ("illegal actual argument found " ^ string_of_typ
                              et ^
216                       " expected " ^ string_of_typ ft ^ " in " ^
                              string_of_expression e)))
217                     )
218                   )
219                   fd.f_params actuals;
220             fd.type_spec
221         | KeyVal (lt, _, e) as ex ->
222             let rt = expression e in
223             check_kv_assign lt rt (Failure("Key " ^ string_of_typ lt ^
224             " has different type from value " ^ string_of_typ rt ^ " in " ^
                    string_of_expression ex))
225         | ArrExp el -> let arr_typ e =
226                        match e with
227                          Int -> Int_Array
228                        | Float -> Float_Array
229                        | Object -> Object_Array
230                        | String -> String_Array
231                        | Bool -> Bool_Array
232                        | _ -> raise (Failure("Invalid array type"))
233                      in
234                      let t = List.fold_left
235                              (fun e1 e2 ->
236                                if (e1 == expression e2) then
237                                  e1
238                                else raise
239                                  (Failure("Multiple types inside an array of type
                                         " ^ string_of_typ e1))
240                              )
241                              (expression (List.hd el)) (List.tl el)
242                      in
243                      arr_typ t
244         | ArrAcc (e1, _) -> (match (expression e1) with
245                               Int_Array -> Int
246                             | Float_Array -> Float
247                             | Object_Array -> Object
248                             | String_Array -> String
249                             | Bool_Array -> Bool
250                             | Object -> Object
```

```
                          | _ -> raise (Failure("Invalid array access on " ^
                                                string_of_expression e1)))
      | ObjExp (el) -> (match (List.hd (List.map expression el)) with
                        Object -> Object
                      | _ -> raise (Failure("Invalid object expression")))
      | Noexpr -> Null
    in

    let check_bool_expression e = if expression e != Bool
      then raise (Failure ("expected Boolean expression in " ^
          string_of_expression e))
      else ()
      in

    let rec statement = function
      Block sl -> let rec check_block = function [Return _ as s] -> statement s
       | Return _ :: _ -> raise (Failure "nothing may follow a return")
       | Block sl :: ss -> check_block (sl @ ss)
       | s :: ss -> statement s ; check_block ss
       | [] -> ()
      in check_block sl
    | Expr e -> ignore (expression e)
    | Return e ->
       let t = expression e in
       if t = func.type_spec then ()
       else if (String.contains (string_of_expression e) '.') then raise(Failure
            ("trying to access object key " ^ string_of_expression e ^ " in
           return of function expecting " ^ string_of_typ func.type_spec ^ "
           please assign to variable and then return"))
       else raise (Failure ("return gives " ^ string_of_typ t ^ " expected " ^
                       string_of_typ func.type_spec ^ " in " ^
                          string_of_expression e))
    | If(p, s1, s2) -> check_bool_expression p; statement s1; statement s2;
    | For(e1, e2, e3, st) -> ignore (expression e1); check_bool_expression e2;
       ignore(expression e3); statement st
    | While(p, s) -> check_bool_expression p; statement s
   in

   statement (Block func.f_statements)

in
List.iter check_function functions
```

codegen.ml

Listing 54: Codegen Code

```
(*
 * Gantry: Code Generation
 * Author: Walter Meyer
 * Contributors: Audrey Copeland, Taimur Samee
 *)

module L = Llvm
module A = Ast

module StringMap = Map.Make(String)

(* Hash Tables for our variable bindings *)
let g_var_tbl : (string, L.llvalue) Hashtbl.t = Hashtbl.create 10;;
let f_var_tbl : (string, L.llvalue) Hashtbl.t = Hashtbl.create 10;;

(* Stack and Buffer for building Objects *)
let kv_stack = Stack.create ();;
let key_buf = Buffer.create 10;;

let translate (globals, functions) =

  let context = L.global_context () in
  let the_module = L.create_module context "Gantry"

    (* LLVM Types *)
    and i1_t = L.i1_type context
    and i32_t = L.i32_type context
    and i8_t = L.i8_type context
    and b_t = L.i8_type context
    and str_t = L.pointer_type (L.i8_type context)
    and flt_t = L.double_type context
    and void_t = L.void_type context in
    (* Array Types *)
    let arr_int_t = L.named_struct_type context "arr_int_t" in
               let body =
                 [|
                   i32_t; (* arr length *)
                   i32_t; (* arr type *)
                   L.pointer_type i32_t;
           |] in
                 ignore (L.struct_set_body arr_int_t body false);
    let arr_flt_t = L.named_struct_type context "arr_flt_t" in
               let body =
                 [|
```

```
                i32_t; (* arr length *)
                i32_t; (* arr type *)
                L.pointer_type flt_t;
        |] in
              ignore (L.struct_set_body arr_flt_t body false);
  let arr_str_t = L.named_struct_type context "arr_str_t" in
            let body =
              [|
                i32_t; (* arr length *)
                i32_t; (* arr type *)
                L.pointer_type str_t;
        |] in
              ignore (L.struct_set_body arr_str_t body false);
  let arr_bool_t = L.named_struct_type context "arr_bool_t" in
            let body =
              [|
                i32_t; (* arr length *)
                i32_t; (* arr type *)
                L.pointer_type b_t;
        |] in
              ignore (L.struct_set_body arr_bool_t body false);
  (* Object Type *)
  let obj_t = L.named_struct_type context "obj" in
            let body =
              [|
                L.pointer_type obj_t; (* next *)
                str_t; (* key *)
                (* values *)
         i32_t; (* value type *)
                i32_t; (* int *)
                flt_t; (* float *)
                L.pointer_type obj_t; (* child (object) *)
                str_t; (* string *)
                b_t; (* bool *)
         (* Arrays *)
         L.pointer_type (L.pointer_type arr_int_t);
         L.pointer_type (L.pointer_type arr_flt_t);
         L.pointer_type (L.pointer_type arr_str_t);
         L.pointer_type (L.pointer_type arr_bool_t);
              |] in
              ignore (L.struct_set_body obj_t body false);
  let arr_obj_t = L.named_struct_type context "arr_obj_t" in
            let body =
              [|
                i32_t; (* arr length *)
                i32_t; (* arr type *)
                L.pointer_type (L.pointer_type obj_t);
        |] in
```

```
93                ignore (L.struct_set_body arr_obj_t body false);
94
95    (* AST to LLVM types *)
96    let ltype_of_typ = function
97        A.Int -> i32_t
98      | A.Float -> flt_t
99      | A.Object -> L.pointer_type obj_t
100     | A.String -> str_t
101     | A.Bool -> b_t
102     | A.Null -> void_t
103     (* Array Types *)
104     | A.Int_Array -> L.pointer_type arr_int_t
105     | A.Float_Array -> L.pointer_type arr_flt_t
106     | A.Object_Array -> L.pointer_type arr_obj_t
107     | A.String_Array -> L.pointer_type arr_str_t
108     | A.Bool_Array -> L.pointer_type arr_bool_t
109   in
110
111   (* Global Declarations *)
112   let add_global (t, n) =
113   let init = L.const_null (ltype_of_typ t) in
114   Hashtbl.add g_var_tbl n (L.define_global n init the_module) in
115   ignore(List.iter add_global globals);
116
117   (* Printf Built-in *)
118   let printf_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
119   let printf_func = L.declare_function "printf" printf_t the_module in
120
121   (* Print bool *)
122   let printb_t = L.var_arg_function_type i32_t [| b_t |] in
123   let printb_func = L.declare_function "print_b" printb_t the_module in
124
125   (* Get user input *)
126   let get_string_t = L.var_arg_function_type str_t [| L.pointer_type i8_t |] in
127   let get_string = L.declare_function "get_string" get_string_t the_module in
128
129   let stoint_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
130   let stoint = L.declare_function "stoint" stoint_t the_module in
131
132   (* String Concatenation *)
133   let string_concat_t = L.var_arg_function_type str_t [| L.pointer_type i8_t |]
        in
134   let string_concat = L.declare_function "string_concat" string_concat_t
        the_module in
135
136   (* String Slice *)
137   let slice_t = L.var_arg_function_type str_t [| L.pointer_type i8_t ; i32_t ;
        i32_t |] in
```

```
138     let slice = L.declare_function "slice" slice_t the_module in
139
140     (* String Comparison *)
141     let stringcmp_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t ; L.
            pointer_type i8_t |] in
142     let stringcmp = L.declare_function "stringcmp" stringcmp_t the_module in
143
144     (* String Equal *)
145     let stringeq_t = L.var_arg_function_type i8_t
146             [| L.pointer_type i8_t ; L.pointer_type i8_t |] in
147     let stringeq = L.declare_function "stringeq" stringeq_t the_module in
148
149     (* String Length *)
150     let string_length_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t |]
            in
151     let string_length = L.declare_function "string_length" string_length_t
            the_module in
152
153     (* Array Length *)
154     let arr_length_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
155     let arr_length = L.declare_function "arr_length" arr_length_t the_module in
156
157     (* HTTP GET library function *)
158     let httpget_t = L.var_arg_function_type str_t [| L.pointer_type i8_t |] in
159     let httpget_func = L.declare_function "httpget" httpget_t the_module in
160
161     (* HTTP POST library function *)
162     let httppost_t = L.var_arg_function_type str_t
163             [| L.pointer_type i8_t ; L.pointer_type i8_t |] in
164     let httppost_func = L.declare_function "httppost" httppost_t the_module in
165
166     (* Nap (sleep wrapper) library function *)
167     let nap_t = L.var_arg_function_type i32_t [| i32_t |] in
168     let nap_func = L.declare_function "nap" nap_t the_module in
169
170     (* Array Stringify *)
171     let arr_stringify_t = L.var_arg_function_type (L.pointer_type i8_t)
172             [| L.pointer_type i8_t |] in
173     let arr_stringify = L.declare_function "arr_stringify"
174             arr_stringify_t the_module in
175
176     (* Object Library Runtime *)
177     let obj_stringify_t = L.var_arg_function_type (L.pointer_type i8_t)
178             [| L.pointer_type obj_t |] in
179     let obj_stringify = L.declare_function "obj_stringify"
180             obj_stringify_t the_module in
181
182     let obj_addkey_t = L.var_arg_function_type (L.pointer_type obj_t)
```

```
183           [| L.pointer_type obj_t ; (L.pointer_type i8_t) ; i32_t; (L.
                 pointer_type i8_t) |] in
184     let obj_addkey = L.declare_function "obj_addkey"
185           obj_addkey_t the_module in
186
187     let obj_findkey_t = L.var_arg_function_type (L.pointer_type obj_t)
188           [| L.pointer_type obj_t ; L.pointer_type i8_t |] in
189     let obj_findkey_func = L.declare_function "obj_findkey"
190           obj_findkey_t the_module in
191
192     let obj_getkey_t = L.var_arg_function_type (L.pointer_type i8_t)
193           [| L.pointer_type obj_t |] in
194     let obj_getkey_func = L.declare_function "obj_getkey"
195           obj_getkey_t the_module in
196
197     let obj_assign_t =
198         L.var_arg_function_type i32_t
199         [| L.pointer_type obj_t ; i32_t ; (L.pointer_type i8_t) |] in
200     let obj_assign_func = L.declare_function "obj_assign" obj_assign_t the_module
        in
201
202     let obj_gettyp_t = L.var_arg_function_type i32_t [| L.pointer_type obj_t |] in
203     let obj_gettyp_func = L.declare_function "obj_gettyp" obj_gettyp_t the_module
        in
204
205     let printk_t = L.var_arg_function_type i32_t [| L.pointer_type obj_t |] in
206     let printk_func = L.declare_function "print_k" printk_t the_module in
207
208     (* Function Declarations *)
209     let func_decls =
210       let func_decl m fdecl =
211       let name = fdecl.A.f_id
212       and param_types =
213       (* Just grab parameter types *)
214       Array.of_list (List.map (fun (t, _) -> ltype_of_typ t) fdecl.A.f_params) in
215       (* Get Function Return and Parameter Types *)
216       let ftype = L.function_type (ltype_of_typ fdecl.A.type_spec) param_types in
217       (* Define Function with LLVM module and store in StringMap *)
218       StringMap.add name (L.define_function name ftype the_module, fdecl) m in
219       (* Put Functions from AST into StringMap *)
220       List.fold_left func_decl StringMap.empty functions in
221
222       (* Function Body *)
223       let build_function_body fdecl =
224       (*
225        * Search our func_decls StringMap for fdecl.A.f_id then
226        * grab: L.define_function name ftype the_module for entry
227        *)
```

```
228       let (the_function, _) = StringMap.find fdecl.A.f_id func_decls in
229       (* Create 'entry' block for the function and track using builder *)
230       let builder = L.builder_at_end context (L.entry_block the_function) in
231
232       (* Global Printf Formats *)
233       let int_format_str = L.build_global_stringptr "%d\n" "int_fmt" builder
234       and flt_format_str = L.build_global_stringptr "%f\n" "flt_fmt" builder
235       and str_format_str = L.build_global_stringptr "%s\n" "str_fmt" builder in
236
237     (* Construct the function's "locals": formal arguments and locally
238        declared variables. Allocate each on the stack, initialize their
239        value, if appropriate, and remember their values in the "locals" map *)
240
241       (* Function Parameters *)
242       let add_param (t, n) p =
243         (* Set (optional, but friendly) name of the parameter value *)
244         L.set_value_name n p;
245         (* Create an alloca(tion) instruction of type t to store n on stack *)
246         let local = L.build_alloca (ltype_of_typ t) n builder in
247         (* Insert instruction that stores parameter in a new function local *)
248         ignore (L.build_store p local builder);
249         (* Add formal to f_var_tbl Hash Map *)
250     Hashtbl.add f_var_tbl n local in
251     (* Generate and Add Function Parameters *)
252         ignore(List.iter2 add_param fdecl.A.f_params
253           (Array.to_list (L.params the_function)));
254
255       (* Function Locals *)
256       let add_local (t, n) builder =
257     let local = L.build_alloca (ltype_of_typ t) n builder in
258     ignore(Hashtbl.add f_var_tbl n local);
259       in
260
261       (* Variable Lookup *)
262       let lookup builder n =
263         (* Local Lookup *)
264         try Hashtbl.find f_var_tbl n with
265         Not_found ->
266         (* Global Lookup *)
267     try Hashtbl.find g_var_tbl n with
268     Not_found ->
269     (* Object Key Lookup *)
270     let l = Str.split (Str.regexp_string ".") n in
271       (* Get 'base' object to search *)
272       let obj = List.hd l in
273       (* Get object ptr from locals *)
274       let obj_p =
275            try Hashtbl.find f_var_tbl obj with
```

```
276             Not_found -> Hashtbl.find g_var_tbl obj
277       in
278       (* Build runtime call to find key *)
279       let keys = String.concat "." (List.tl l) in
280       let key = L.build_global_stringptr keys "keys_find" builder in
281       (* Dereference parent object pointer *)
282       let obj_p = L.build_load obj_p "tmp" builder in
283       (* Now build runtime call to find the key *)
284       L.build_call obj_findkey_func [| obj_p ; key |]
285       "obj_findkey" builder
286       in
287
288     (* Construct code for an expression and return the value *)
289     let rec expr builder = function
290         A.IntLit i -> L.const_int i32_t i
291       | A.FloatLit f -> L.const_float flt_t f
292       | A.StrLit s -> L.build_global_stringptr s "string" builder
293       | A.BoolLit b -> L.const_int b_t (if b then 1 else 0)
294       | A.Noexpr -> L.const_int i32_t 0
295       | A.Id s -> if (String.contains s '.') then
296             (lookup builder s)
297           else
298             (L.build_load (lookup builder s) s builder)
299       | A.Binop (e1, op, e2) ->
300         let e1' = expr builder e1
301         and e2' = expr builder e2 in
302     let typ = L.string_of_lltype (L.type_of e1') in
303     (match typ with
304           "i32" -> (match op with
305             A.Add -> L.build_add e1' e2' "tmp" builder
306           | A.Sub -> L.build_sub e1' e2' "tmp" builder
307           | A.Mult -> L.build_mul e1' e2' "tmp" builder
308           | A.Div -> L.build_sdiv e1' e2' "tmp" builder
309       | A.Eq -> L.build_intcast (L.build_icmp L.Icmp.Eq e1' e2' "tmp" builder)
310             i8_t "tmp" builder
311       | A.Neq -> L.build_intcast (L.build_icmp L.Icmp.Ne e1' e2' "tmp" builder)
312             i8_t "tmp" builder
313       | A.Lt -> L.build_intcast (L.build_icmp L.Icmp.Slt e1' e2' "tmp" builder)
314             i8_t "tmp" builder
315       | A.Leq -> L.build_intcast (L.build_icmp L.Icmp.Sle e1' e2' "tmp" builder)
316             i8_t "tmp" builder
317       | A.Gt -> L.build_intcast (L.build_icmp L.Icmp.Sgt e1' e2' "tmp" builder)
318             i8_t "tmp" builder
319       | A.Geq -> L.build_intcast (L.build_icmp L.Icmp.Sge e1' e2' "tmp" builder)
320             i8_t "tmp" builder
321       | _ -> raise (Failure ("Invalid integer binary operation"))
322       )
323         | "double" -> (match op with
```

```
324             A.Add -> L.build_fadd e1' e2' "tmp" builder
325           | A.Sub -> L.build_fsub e1' e2' "tmp" builder
326           | A.Mult -> L.build_fmul e1' e2' "tmp" builder
327           | A.Div -> L.build_fdiv e1' e2' "tmp" builder
328       | A.Eq -> L.build_intcast (L.build_fcmp L.Fcmp.Oeq e1' e2' "tmp" builder)
329             i8_t "tmp" builder
330       | A.Neq -> L.build_intcast (L.build_fcmp L.Fcmp.One e1' e2' "tmp" builder)
331             i8_t "tmp" builder
332       | A.Lt -> L.build_intcast (L.build_fcmp L.Fcmp.Ult e1' e2' "tmp" builder)
333             i8_t "tmp" builder
334       | A.Leq -> L.build_intcast (L.build_fcmp L.Fcmp.Ole e1' e2' "tmp" builder)
335             i8_t "tmp" builder
336       | A.Gt -> L.build_intcast (L.build_fcmp L.Fcmp.Ogt e1' e2' "tmp" builder)
337             i8_t "tmp" builder
338       | A.Geq -> L.build_intcast (L.build_fcmp L.Fcmp.Oge e1' e2' "tmp" builder)
339             i8_t "tmp" builder
340       | _ -> raise (Failure ("Invalid float binary operation"))
341       )
342         | "i8" -> (match op with
343             A.And -> L.build_intcast (L.build_and
344                     (L.build_intcast e1' i1_t "tmp" builder)
345                     (L.build_intcast e2' i1_t "tmp" builder)
346                     "tmp" builder)
347           | A.Or -> L.build_intcast (L.build_or
348                     (L.build_intcast e1' i1_t "tmp" builder)
349                     (L.build_intcast e2' i1_t "tmp" builder)
350                     "tmp" builder)
351           | A.Eq -> L.build_intcast (L.build_icmp L.Icmp.Eq
352                     (L.build_intcast e1' i1_t "tmp" builder)
353                     (L.build_intcast e2' i1_t "tmp" builder)
354                     "tmp" builder)
355           | A.Neq -> L.build_intcast (L.build_icmp L.Icmp.Ne
356                     (L.build_intcast e1' i1_t "tmp" builder)
357                     (L.build_intcast e2' i1_t "tmp" builder)
358                     "tmp" builder)
359           | _ -> raise (Failure ("Invalid Boolean binary operation"))
360          ) i8_t "tmp" builder
361        | "i8*" -> (match op with
362            A.Eq -> L.build_intcast (L.build_icmp L.Icmp.Eq
363                     (L.build_call stringeq [| (e1') ; (e2') |] "stringeq"
                            builder)
364                     (L.const_int b_t 1) "tmp" builder ) i8_t "tmp" builder
365
366         | A.Neq -> L.build_intcast (L.build_icmp L.Icmp.Eq
367                     (L.build_call stringeq [| (e1') ; (e2') |] "stringeq"
                            builder)
368                     (L.const_int b_t 0) "tmp" builder ) i8_t "tmp" builder
```

```
        | A.Conc -> L.build_call string_concat [| e1'; e2'|] "string_concat"
            builder
      | _ -> raise (Failure ("Invalid string binary operation")))
    | _ -> raise (Failure ("Invalid binary operation"))
      )
      | A.Unop(op, e) ->
       let e' = expr builder e in
       (match op with
           A.Neg -> L.build_neg e' "tmp" builder
         | A.Not -> L.build_intcast (L.build_not
                    (L.build_intcast e' i1_t "tmp" builder)
                    "tmp" builder) i8_t "tmp" builder
     | A.Inc ->
         let n = lookup builder (A.expr_to_str e) in
         let tmp = L.build_load n "tmp" builder in
         let tmp = L.build_add (L.const_int i32_t 1) tmp "tmp" builder in
         L.build_store tmp n builder
     | A.Dec ->
         let n = lookup builder (A.expr_to_str e) in
         let tmp = L.build_load n "tmp" builder in
         let tmp = L.build_sub tmp (L.const_int i32_t 1) "tmp" builder in
         L.build_store tmp n builder
    )
      | A.ArrExp(el) ->
    let vl = List.map (expr builder) el in
    (* We infer type by first expression - semantic should handle this *)
    let typ = L.pointer_type (L.type_of (List.hd vl)) in
    let size = L.const_int i32_t (List.length vl) in
    let arr_typ = (match L.string_of_lltype typ with
             "i32*" -> arr_int_t
           | "double*" -> arr_flt_t
           | "%obj**" -> arr_obj_t
           | "i8**" -> arr_str_t
           | "i8*" -> arr_bool_t
           | _ -> raise (Failure ("Invalid array type"))) in
    (* Declare Array *)
    let arr = L.build_array_malloc typ size "arr" builder in
    let arr = L.build_pointercast arr typ "arr" builder in
    (* Declare Struct *)
    let arr_struct = L.build_malloc arr_typ "arr_struct" builder in
    (* Set array size field *)
    let arr_l = L.build_struct_gep arr_struct 0 "arr_size" builder in
    ignore(L.build_store (L.const_int i32_t
                (List.length vl)) arr_l builder);
    (* Set Array Type in Struct *)
    let arr_t = L.build_struct_gep arr_struct 1 "arr_type" builder in
    let sidx_of_typ = function
        "i32*" -> 3
```

```
416       | "double*" -> 4
417       | "%obj**" -> 5
418       | "i8**" -> 6
419       | "i8*" -> 7
420       | _ -> raise (Failure ("Invalid array type"))
421     in
422     let t = sidx_of_typ (L.string_of_lltype typ) in
423     ignore(L.build_store (L.const_int i32_t t) arr_t builder);
424     (* For each value, store in array *)
425     let fill i v =
426       let vp =
427       L.build_gep arr [| L.const_int i32_t i |] "arr_v" builder in
428       ignore(L.build_store v vp builder);
429     in
430     (* Populate Array with Values *)
431     List.iteri fill vl;
432     (* Store Array in Struct *)
433     let arr_p = L.build_struct_gep arr_struct 2 "arr_struct" builder in
434     ignore(L.build_store arr arr_p builder);
435     arr_struct
436       | A.ArrAcc(e1, e2) ->
437         let e1_str = A.expr_to_str e1
438         and idx = expr builder e2 in
439     let arr =
440           if (String.contains e1_str '.') then
441         (lookup builder e1_str)
442       else
443         (L.build_load (lookup builder e1_str) "arracc" builder)
444     in
445     (* Get actual array from struct *)
446     let arr = L.build_struct_gep arr 2 "arr_v" builder in
447     let arr = L.build_load arr "arr_v" builder in
448     let e1' = L.build_gep arr [| idx |] "arracc_e" builder in
449     let e1' = L.build_load e1' "arracc" builder in
450     e1'
451       | A.KeyVal(t, n, e) ->
452     (* Resolve struct index and 'value_typ' in struct *)
453     let sidx_of_typ = function
454         A.Int -> 3
455       | A.Float -> 4
456       | A.Object -> 5
457       | A.String -> 6
458       | A.Bool -> 7
459       (* Arrays *)
460       | A.Int_Array -> 8
461       | A.Float_Array -> 9
462       | A.String_Array -> 10
463       | A.Bool_Array -> 11
```

```
464       | _ -> raise (Failure ("Invalid or unsupported object key type"))
465         in
466     (* Build the data structure for a key *)
467     let e' = expr builder e in
468     let k = L.build_malloc obj_t "obj" builder in
469     (* Set key name *)
470     let name = L.build_global_stringptr n "key_name_str" builder in
471     ignore(L.build_store name
472     (L.build_struct_gep k 1 "key_name" builder) builder);
473     (* Set value type *)
474     ignore(L.build_store (L.const_int i32_t (sidx_of_typ t))
475     (L.build_struct_gep k 2 "value_typ" builder) builder);
476     (* Set value *)
477     let value = L.build_struct_gep k (sidx_of_typ t) "value" builder in
478     let e' =
479       if ((sidx_of_typ t) > 7) then (
480       (* Cast ptr to RHV to void ptr to store in Object *)
481       let e'' = L.build_alloca (L.type_of e') "arrinobj" builder in
482       ignore(L.build_store e' e'' builder);
483         e''
484       )
485       else (
486         e'
487       )
488     in
489     ignore(L.build_store e' value builder);
490     (* Add the key value pair to a stack *)
491     ignore(Stack.push (sidx_of_typ t, n, k) kv_stack);
492        e'
493       | A.ObjExp(el) ->
494     let null = (L.const_pointer_null (L.pointer_type obj_t)) in
495     ignore(Stack.push (-1, "NULL", null) kv_stack);
496     (* Set next for a key or object *)
497     let set_next k =
498     try
499       let (t, _, next_k) = Stack.top kv_stack in
500       if t > -1 then
501         (ignore(L.build_store next_k
502               (L.build_struct_gep k 0 "next" builder) builder);)
503       else
504         (ignore(L.build_store
505            (L.const_pointer_null (L.pointer_type obj_t))
506                (L.build_struct_gep k 0 "next" builder) builder);
507            ignore(Stack.pop kv_stack);)
508     with
509       Stack.Empty -> ()
510     in
511     (* Resolve the list of key value exprs (putting them into a stack) *)
```

```
512     let kv = List.map (expr builder) el in
513     (* The Enclosing (parent) Object *)
514     let parent = L.build_malloc obj_t "obj" builder in
515     (* Connect parent to first key *)
516     ignore(set_next parent);
517     (* Connect each key in this object *)
518     let build_obj _ =
519       let (_, _, k) = Stack.pop kv_stack in
520       ignore(set_next k);
521     in
522     (* Build out all of the key values within this object *)
523     List.iter build_obj kv;
524     (* Return the pointer to the enclosing object *)
525     parent
526       | A.AssignDecl(t, n, e) ->
527         (* First add this declaration to f_var_tbl hash map *)
528         ignore (add_local (t, n) builder);
529     (* Non-Object on RHS *)
530     if (not (String.contains (A.expr_to_str e) '.')) then (
531           let e' = expr builder e in
532       let n = lookup builder n in
533           ignore (L.build_store e' n builder);
534           e'
535     )
536     (* Object on RHS *)
537     else (
538       let e1_str = n
539       and e2_str = A.expr_to_str e in
540       let e1' = lookup builder e1_str
541       and e2' = lookup builder e2_str in
542       let sidx_of_typ = function
543           "i32*" -> 3
544         | "double*" -> 4
545         | "%obj**" -> 5
546         | "i8**" -> 6
547         | "i8*" -> 7
548         (* Arrays *)
549         | "%arr_int_t**" -> 8
550         | "%arr_flt_t**" -> 9
551         | "%arr_str_t**" -> 10
552         | "%arr_bool_t**" -> 11
553         | _ -> raise (Failure ("Invalid object assignment"))
554       in
555       (* Get type of primitive on LHS *)
556       let t = sidx_of_typ (L.string_of_lltype (L.type_of e1')) in
557       let t_e1' = L.const_int i32_t t in
558       (* Get Value of Object on RHS (based on e1' type) *)
559       let v_e2' = L.build_call obj_getkey_func
```

```
560             [| e2' ; t_e1' |] "obj_getkey" builder in
561       (* Cast void* RHV to ptr of LHV type *)
562       let v_e2' = L.build_bitcast v_e2' (L.type_of e1') "cst" builder in
563       let v_e2' = L.build_load v_e2' "loadcst" builder in
564       ignore(L.build_store v_e2' e1' builder);
565       v_e2'
566     )
567       | A.Assign(e1, e2) ->
568     (* Object Assignment *)
569     let e1_str = A.expr_to_str e1
570     and e2_str = A.expr_to_str e2 in
571     (* Object LHV and RHV: enforce type of RHV *)
572     if ((String.contains e1_str '.') && (String.contains e2_str '.')) then (
573       (* Get Objects on LHS and RHS *)
574       let e1' = lookup builder e1_str
575       and e2' = lookup builder e2_str in
576       (* Get type of Object on RHS *)
577       let t_e2' = L.build_call obj_gettyp_func
578             [| e2' |] "obj_gettyp" builder in
579       (* Get Value of Object on RHS *)
580       let v_e2' = L.build_call obj_getkey_func
581             [| e2' ; t_e2' |] "obj_getkey" builder in
582       (* Store that value in Object on LHS *)
583       ignore (L.build_call obj_assign_func
584       [| e1' ; t_e2' ; v_e2' |] "obj_assign" builder);
585       e2'
586     )
587     (* Object LHV Only: enforce type of RHV *)
588     else if (String.contains e1_str '.') then (
589       let e1' = lookup builder e1_str
590       and e2' = expr builder e2 in
591       let sidx_of_typ = function
592           "i32" -> 3
593         | "double" -> 4
594         | "%obj*" -> 5
595         | "i8*" -> 6
596         | "i8" -> 7
597         (* Arrays *)
598         | "%arr_int_t*" -> 8
599         | "%arr_flt_t*" -> 9
600         | "%arr_str_t*" -> 10
601         | "%arr_bool_t*" -> 11
602         | _ -> raise (Failure ("Invalid object assignment"))
603       in
604       (* Get type of primitive on RHS *)
605       let t = sidx_of_typ (L.string_of_lltype (L.type_of e2')) in
606       let t_e2' = L.const_int i32_t t in
607       (* Cast ptr to RHV to void ptr to store in Object *)
```

```
608       let p_e2' = L.build_alloca (L.type_of e2') "pcst" builder in
609       ignore(L.build_store e2' p_e2' builder);
610       let v_e2' = L.build_bitcast p_e2' (L.pointer_type i8_t)
611             "cst" builder in
612       (* Store value from primitive on RHS in Object on LHS *)
613       ignore (L.build_call obj_assign_func
614       [| e1' ; t_e2' ; v_e2' |] "obj_assign" builder);
615       e2'
616     )
617     (* Object RHV Only: try type of RHV (runtime error on failure) *)
618     else if (String.contains e2_str '.') then (
619       let e1' = lookup builder e1_str
620       and e2' = lookup builder e2_str in
621       let sidx_of_typ = function
622           "i32*" -> 3
623         | "double*" -> 4
624         | "%obj**" -> 5
625         | "i8**" -> 6
626         | "i8*" -> 7
627         (* Arrays *)
628         | "%arr_int_t**" -> 8
629         | "%arr_flt_t**" -> 9
630         | "%arr_str_t**" -> 10
631         | "%arr_bool_t**" -> 11
632         | _ -> raise (Failure ("Invalid object assignment"))
633       in
634       (* Get type of primitive on LHS *)
635       let t = sidx_of_typ (L.string_of_lltype (L.type_of e1')) in
636       let t_e1' = L.const_int i32_t t in
637       (* Get Value of Object on RHS (based on e1' type) *)
638       let v_e2' = L.build_call obj_getkey_func
639             [| e2' ; t_e1' |] "obj_getkey" builder in
640       (* Cast void* RHV to ptr of LHV type *)
641       let v_e2' = L.build_bitcast v_e2' (L.type_of e1') "cst" builder in
642       let v_e2' = L.build_load v_e2' "loadcst" builder in
643       ignore(L.build_store v_e2' e1' builder);
644       v_e2'
645     )
646     (* Primitive Assignment *)
647     else (
648       let e2' = expr builder e2 in
649       let e1' = lookup builder e1_str in
650       ignore (L.build_store e2' e1' builder);
651       e2'
652     )
653       | A.FunExp("print_i", [e]) ->
654     L.build_call printf_func [| int_format_str ; (expr builder e) |]
655       "print_i" builder
```

```
656       | A.FunExp("print_s", [e]) ->
657     L.build_call printf_func [| str_format_str ; (expr builder e) |]
658       "print_s" builder
659       | A.FunExp("print_d", [e]) ->
660     L.build_call printf_func [| flt_format_str ; (expr builder e) |]
661       "print_d" builder
662       | A.FunExp("print_b", [e]) ->
663         L.build_call printb_func [| (expr builder e) |]
664           "print_b" builder
665       | A.FunExp("print_k", [e]) ->
666     L.build_call printk_func [| (expr builder e) |]
667       "print_k" builder
668       | A.FunExp("get_string", [e]) ->
669       L.build_call get_string [|(expr builder e)|]
670       "get_string" builder
671       | A.FunExp("stoint", [e]) ->
672       L.build_call stoint [|(expr builder e)|]
673       "stoint" builder
674       | A.FunExp("httpget", [e]) ->
675         L.build_call httpget_func [| (expr builder e) |]
676           "httpget" builder
677       | A.FunExp("httppost", [e; e2]) ->
678           let e2' = expr builder e2 in
679             L.build_call httppost_func [| (expr builder e) ; (e2') |]
680             "httppost" builder
681       | A.FunExp("nap", [e]) ->
682             L.build_call nap_func [| (expr builder e) |]
683             "nap" builder
684       | A.FunExp("string_length", [e]) ->
685         L.build_call string_length [| (expr builder e ) |]
686         "string_length" builder
687       | A.FunExp("arr_length", [e]) ->
688         let e' = expr builder e in
689         (* Cast array to void * to send to length function *)
690         let p_e' = L.build_alloca (L.type_of e') "pcst" builder in
691         ignore(L.build_store e' p_e' builder);
692         let e' = L.build_bitcast p_e' (L.pointer_type i8_t) "cs" builder in
693         L.build_call arr_length [| (e') |] "arr_length" builder
694       | A.FunExp("slice", [e; e1; e2]) ->
695           let e1' = expr builder e1
696           and e2' = expr builder e2 in
697         L.build_call slice [| (expr builder e) ; (e1') ; (e2') |]
698         "slice" builder
699       | A.FunExp("stringcmp", [e1; e2]) ->
700           let e1' = expr builder e1
701           and e2' = expr builder e2 in
702         L.build_call stringcmp [| (e1') ; (e2') |]
703         "stringcmp" builder
```

```
704       | A.FunExp("arr_stringify", [e]) ->
705       let e' = expr builder e in
706       (* Cast e to void ptr *)
707       let p_e' = L.build_alloca (L.type_of e') "pcst" builder in
708       ignore(L.build_store e' p_e' builder);
709       let v_e' = L.build_bitcast p_e' (L.pointer_type i8_t)
710             "cst" builder in
711       (* Build call to obj_addkey *)
712       L.build_call arr_stringify [| (v_e') |]
713         "arr_stringify" builder
714       | A.FunExp("obj_stringify", [e]) ->
715       let e' = expr builder e in
716         L.build_call obj_stringify [|(e')|]
717         "obj_stringify" builder
718       | A.FunExp("obj_addkey", [e; e1; e2; e3]) ->
719       let e' = expr builder e
720       and e1' = expr builder e1
721       and e2' = expr builder e2
722       and e3' = expr builder e3 in
723       (* Cast e3 to void ptr to store in Object *)
724       let p_e3' = L.build_alloca (L.type_of e3') "pcst" builder in
725       ignore(L.build_store e3' p_e3' builder);
726       let v_e3' = L.build_bitcast p_e3' (L.pointer_type i8_t)
727             "cst" builder in
728       (* Build call to obj_addkey *)
729       L.build_call obj_addkey [| (e'); (e1'); (e2'); (v_e3') |]
730         "obj_addkey" builder
731       | A.FunExp(f, act) ->
732         let (fdef, fdecl) =
733     StringMap.find f func_decls in
734     (* If Object is actual, cast its value to formal's type *)
735     let resolve_acts form_t act =
736     if (String.contains (A.expr_to_str act) '.') then (
737       let e' = lookup builder (A.expr_to_str act) in
738       let sidx_of_typ = function
739           "int" -> 3
740         | "float" -> 4
741         | "object" -> 5
742         | "string" -> 6
743         | "bool" -> 7
744         (* Arrays *)
745         | "int array" -> 8
746         | "float array" -> 9
747         | "string array" -> 10
748         | "bool array" -> 11
749         | _ -> raise (Failure ("Invalid object function actual parameter"))
750       in
751       (* Get type of formal *)
```

```
752       let t = sidx_of_typ (A.string_of_typ form_t) in
753       let t = L.const_int i32_t t in
754       (* Get Value of Object on RHS (based on formal type) *)
755       let v_e' = L.build_call obj_getkey_func [| e' ; t |] "obj_getkey_param"
              builder in
756       (* Cast void* RHV to ptr of LHV type *)
757       let v_e' = L.build_bitcast v_e' (L.pointer_type (ltype_of_typ form_t)) "cst"
              builder in
758       let v_e' = L.build_load v_e' "loadcst" builder in
759       v_e'
760     )
761     else (
762       expr builder act
763     ) in
764     let formals = List.map (fun (t, _) -> t) fdecl.A.f_params in
765         let actuals = List.rev (List.map2 (resolve_acts) (List.rev formals) (List.
              rev act)) in
766         let result = (match fdecl.A.type_spec with
767                 A.Null -> ""
768                       | _ -> f ^ "_result") in
769         L.build_call fdef (Array.of_list actuals) result builder
770       in
771
772
773     let add_terminal builder f =
774       match L.block_terminator (L.insertion_block builder) with
775         Some _ -> ()
776       | None -> ignore (f builder) in
777
778     (* Build the code for the given statement; return the builder for
779        the statement's successor *)
780     let rec stmt builder = function
781         A.Block sl -> List.fold_left stmt builder sl
782       | A.Expr e -> ignore (expr builder e); builder
783       | A.Return e -> ignore (match fdecl.A.type_spec with
784         A.Null -> L.build_ret_void builder
785       | _ -> L.build_ret (expr builder e) builder); builder
786       | A.If (predicate, then_stmt, else_stmt) ->
787         let bool_val = expr builder predicate in
788     let typ = L.string_of_lltype (L.type_of bool_val) in
789     let bv =
790     if ((String.compare typ "i8") == 0) then
791         (L.build_intcast bool_val i1_t "tmp" builder)
792     else
793         bool_val in
794
795         let merge_bb = L.append_block context "merge" the_function in
796
```

```
797         let then_bb = L.append_block context "then" the_function in
798         add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
799         (L.build_br merge_bb);
800
801         let else_bb = L.append_block context "else" the_function in
802         add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
803         (L.build_br merge_bb);
804
805         ignore (L.build_cond_br bv then_bb else_bb builder);
806         L.builder_at_end context merge_bb
807       | A.While (predicate, body) ->
808         let pred_bb = L.append_block context "while" the_function in
809         ignore (L.build_br pred_bb builder);
810
811         let body_bb = L.append_block context "while_body" the_function in
812         add_terminal (stmt (L.builder_at_end context body_bb) body)
813         (L.build_br pred_bb);
814
815         let pred_builder = L.builder_at_end context pred_bb in
816         let bool_val = expr pred_builder predicate in
817     let typ = L.string_of_lltype (L.type_of bool_val) in
818     let bv =
819     if ((String.compare typ "i8") == 0) then
820         (L.build_intcast bool_val i1_t "tmp" pred_builder)
821     else
822         bool_val
823     in
824
825         let merge_bb = L.append_block context "merge" the_function in
826         ignore (L.build_cond_br bv body_bb merge_bb pred_builder);
827         L.builder_at_end context merge_bb
828       | A.For (e1, e2, e3, body) -> stmt builder
829       ( A.Block [A.Expr e1 ; A.While (e2, A.Block [body ; A.Expr e3]) ] )
830       in
831
832       (* Build the code for each statement in the function *)
833       let builder = stmt builder (A.Block fdecl.A.f_statements) in
834
835       (* Add a return if the last block falls off the end *)
836       add_terminal builder (match fdecl.A.type_spec with
837         A.Null -> L.build_ret_void
838       | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
839       in
840
841     (* Populate Function Bodies *)
842     List.iter build_function_body functions;
843     the_module
```

8.1.3 Library Functions

gantrylib_http.c

Listing 55: HTTP

```
1  /*
2   * Gantry: HTTP Library Module
3   * Author: Taimur Samee
4   */
5
6  #include <stdio.h>
7  #include <string.h>
8  #include <stdlib.h>
9  #include <curl/curl.h>
10
11 // struct to avoid strlen calls
12 struct string {
13         char *contents;
14         size_t len;
15 };
16
17 // curl uses this to write page data to a buffer
18 size_t writer(void *ptr, size_t size, size_t nmemb, struct string *userdata) {
19
20         size_t bufflen = userdata->len + size*nmemb;
21         if ((userdata->contents = realloc(userdata->contents, bufflen + 1)) ==
               NULL) {
22
23                 fprintf(stderr, "httpget: realloc() failed\n");
24                 exit(EXIT_FAILURE);
25         }
26
27
28         memcpy(userdata->contents+userdata->len, ptr, size*nmemb);
29         userdata->contents[bufflen] = '\0';
30         userdata->len = bufflen;
31
32         return size*nmemb;
33 }
34
35 /* sends a GET request to the target URL, returns the server response
36    Currently leaks memory; where do we free the return pointer? */
37 char* httpget(char *url) {
38
39         CURL *curl;
40         CURLcode res; // Error codes
41         struct string data; // Holds file contents
42         struct string *s = &data;
```

```
43          s->len = 0;
44          s->contents = malloc(s->len+1);
45          s->contents[0] = '\0';
46
47          curl_global_init(CURL_GLOBAL_DEFAULT);
48          curl = curl_easy_init();
49
50          curl_easy_setopt(curl, CURLOPT_URL, url);
51          curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, writer);
52          curl_easy_setopt(curl, CURLOPT_WRITEDATA, s);
53
54          res = curl_easy_perform(curl);
55          if (res != CURLE_OK) {
56                  fprintf(stderr, "httpget: curl_easy_perform() failed: %s\n",
57                          curl_easy_strerror(res));
58          }
59
60          curl_easy_cleanup(curl);
61          curl_global_cleanup();
62          // free(s->contents);
63          return s->contents;
64  }
65
66  /* POSTs the provided data to the target URL, returns the server response
67     Currently leaks memory; where do we free the return pointer? */
68  char* httppost(char* target, char *tosend) {
69
70          CURL *curl;
71          CURLcode res;
72          struct curl_slist *headers = NULL;
73          headers = curl_slist_append(headers, "Content-Type: application/json");
74          struct string data;
75          struct string *s = &data;
76          s->len = 0;
77          s->contents = malloc(s->len+1);
78          s->contents[0] = '\0';
79
80          curl_global_init(CURL_GLOBAL_DEFAULT);
81          curl = curl_easy_init();
82
83          curl_easy_setopt(curl, CURLOPT_URL, target);
84          curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, -1L); // strlen calculates
                size
85
86          // COPYPOSTFIELDS will copy the data before sending so altering during
                sending is ok
87          curl_easy_setopt(curl, CURLOPT_COPYPOSTFIELDS, tosend);
88          // Set HEADERS for POST to Content-Type: application/json
```

```
89          curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
90
91          // Put any responses in a dummy array instead of printing to stdout
92          curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, writer);
93          curl_easy_setopt(curl, CURLOPT_WRITEDATA, s);
94
95          res = curl_easy_perform(curl);
96          if (res != CURLE_OK) {
97                  fprintf(stderr, "httppost: curl_easy_perform() failed: %s\n",
98                          curl_easy_strerror(res));
99          }
100
101         curl_easy_cleanup(curl);
102         curl_global_cleanup();
103
104         return s->contents;
105 }
106
107
108 // Main for testing
109 #ifdef BUILD_TEST
110 int main() {
111
112         // GET request
113         char* page = httpget("https://requestb.in/1jhotzd1");
114         printf("%s\n",page);
115
116         // POST
117         char* req = "{ \"name\" : \"Jeff\" }";
118         char* url = "https://requestb.in/1jhotzd1";
119         printf("%s\n",httppost(url, req));
120
121         return 0;
122 }
123 #endif
```

gantrylib_nap.c

Listing 56: Nap

```
/*
 * Gantry: nap Library Module
 * Author: Taimur Samee
 */

// Wrapper function for sleep() in C
#include<unistd.h>

int nap(int x) {

   return sleep(x);

}

#ifdef BUILD_TEST
int main() {

    x = sleep(5);
    printf("%d", x);
    return 0;
}
#endif
```

gantrylib_obj.c

Listing 57: Obj

```
/*
 * Gantry: Object and Array Library Module
 * Author (Object Runtime Functions): Walter Meyer
 * Author (Stringify Functions): Audrey Copeland
 */

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


// The Object Struct
typedef struct obj {
  // metadata
  struct obj *next; // next
  char *k; // key name
  int v_typ; // value type

  // values
  int i; // int
  double f; // float
  struct obj *o; // object
  char *s; // string
  bool b; // bool
  struct arr_int *i_a;
  struct arr_flt *f_a;
  struct arr_str *s_a;
  struct arr_bool *b_a;
} obj;

// The arr int
typedef struct arr_int {
  int len;
  int typ;
  int *i_a;
} arr_int;

// The arr float
typedef struct arr_flt {
  int len;
  int typ;
  double *f_a;
} arr_flt;
```

```
45
46  // The arr string
47  typedef struct arr_str {
48    int len;
49    int typ;
50    char **s_a;
51  } arr_str;
52
53
54  typedef struct arr_obj {
55    int len;
56    int typ;
57    obj **o_a;
58  } arr_obj;
59
60
61  // The arr bool
62  typedef struct arr_bool {
63    int len;
64    int typ;
65    bool *b_a;
66  } arr_bool;
67
68  char *obj_stringify(obj *);
69
70  // Object Key Printing
71  int print_k(obj *o) {
72    if (o == NULL)
73      return 1;
74    switch(o->v_typ) {
75      case 3: printf("%d\n", o->i); break;
76      case 4: printf("%f\n", o->f); break;
77      case 5: printf("object key [%p]\n", &o); break;
78      case 6: printf("%s\n", o->s); break;
79      case 7: printf("%s", o->b ? "true\n" : "false\n"); break;
80      default: printf("object [%p]\n", &o); break;
81    };
82    return 0;
83  }
84
85  /* Checks realloc and makes sure memory null */
86  char *xrealloc(char *ptr, size_t sz){
87          char *temp = (char *)realloc(ptr, sz);
88          if (temp == NULL){
89                  printf("Runtime Error: Failed to realloc");
90          }
91          else {
92                  ptr = temp;
```

```
93                memset(ptr, 0 , sz);
94        }
95        return ptr;
96  }
97
98
99  /* Grows buff by appended cpy_buff */
100 char *fill_buff(char *buff, char *cpy_buff){
101       size_t cpy_len;
102       size_t buff_sz;
103       char *old_buff;
104
105       cpy_len = sizeof(char) *(strlen(cpy_buff) + 1);
106
107       buff_sz = sizeof(char) *(strlen(buff)+1);
108       old_buff = (char *)malloc(buff_sz);
109       memcpy(old_buff, buff, buff_sz);
110       buff = (char *)xrealloc(buff, buff_sz + cpy_len);
111
112       memcpy(buff,old_buff, buff_sz);
113       free(old_buff);
114
115       memcpy(buff + buff_sz -1, cpy_buff, cpy_len);
116       return buff;
117 }
118
119
120 char *arr_stringify(void *arr){
121       /* Cast to each arr type */
122       struct arr_int *arr_i = *(arr_int **)(arr);
123       struct arr_flt *arr_f = *(arr_flt **)(arr);
124       struct arr_str *arr_s = *(arr_str **)(arr);
125       struct arr_bool *arr_b = *(arr_bool **)(arr);
126       struct arr_obj *arr_o = *(arr_obj **)(arr);
127
128       int len = arr_i->len;
129       int typ = arr_i->typ;
130
131       char *buff;
132
133       char *cpy_buff;
134       size_t cpy_len;
135
136       cpy_buff = (char *)malloc(sizeof(char));
137
138       buff = (char *)malloc(sizeof(char));
139       memcpy(buff, "", 1);
140
```

```
141         cpy_len = 2;
142         cpy_buff = xrealloc(cpy_buff, sizeof(char)*(cpy_len+1));
143         memcpy(cpy_buff, "[ ", cpy_len+1);
144         buff = fill_buff(buff, cpy_buff);
145
146         obj *o;
147         int i;
148         switch(typ){
149                 case 3:
150                         for (i=0; i< len; i++){
151                                 cpy_len = snprintf(NULL, 0 , "%d" , (arr_i->i_a[i]))
                                        ;
152                                 cpy_buff = xrealloc(cpy_buff, sizeof(char)*(cpy_len
                                        +1));
153                                 snprintf(cpy_buff,cpy_len+1, "%d", (arr_i->i_a[i]));
154                                 buff = fill_buff(buff, cpy_buff);
155                                 if (i!=len-1){
156                                         buff = fill_buff(buff, " , ");
157                                 }
158                         }
159                         break;
160                 case 4:
161                         for (i=0; i< len; i++){
162                                 cpy_len = snprintf(NULL, 0 , "%lf" , (arr_f->f_a[i])
                                        );
163                                 cpy_buff = xrealloc(cpy_buff, sizeof(char)*(cpy_len
                                        +1));
164                                 snprintf(cpy_buff,cpy_len+1, "%lf", (arr_f->f_a[i]))
                                        ;
165                                 buff = fill_buff(buff, cpy_buff);
166                                 if (i!=len-1){
167                                         buff = fill_buff(buff, " , ");
168                                 }
169                         }
170                         break;
171                 case 5:
172                         for (i=0; i<len; i++){
173                                 o = arr_o->o_a[i];
174                                 cpy_buff = obj_stringify(o);
175                                 buff=fill_buff(buff,cpy_buff);
176                                 if (i!=len-1){
177                                         buff = fill_buff(buff, ", ");
178                                 }
179                         }
180                         break;
181                 case 6:
182                         for (i=0; i< len; i++){
```

```
183                             cpy_len = snprintf(NULL, 0 , "\"%s\"" , (arr_s->s_a[
                                    i]));
184                             cpy_buff = xrealloc(cpy_buff, sizeof(char)*(cpy_len
                                    +1));
185                             snprintf(cpy_buff,cpy_len+1, "\"%s\"", (arr_s->s_a[i
                                    ]));
186                             buff = fill_buff(buff,cpy_buff);
187                             if (i!=len-1){
188                                     buff = fill_buff(buff, " , ");
189                             }
190                     }
191                     break;
192             case 7:
193                     for (i=0; i< len; i++){
194                             cpy_len= snprintf(NULL, 0 , "%s" , (arr_b->b_a[i]) ?
                                     "true" : "false");
195                             cpy_buff = xrealloc(cpy_buff, sizeof(char)*(cpy_len
                                    +1));
196                             snprintf(cpy_buff, cpy_len+1,"%s", (arr_b->b_a[i]) ?
                                     "true" : "false");
197                             buff = fill_buff(buff, cpy_buff);
198                             if (i!=len-1){
199                                     buff = fill_buff(buff, " , ");
200                             }
201                     }
202                     break;
203     }
204
205     cpy_len = 2;
206     cpy_buff = xrealloc(cpy_buff, sizeof(char)*(cpy_len+1));
207     memcpy(cpy_buff, " ]", cpy_len+1);
208     buff = fill_buff(buff, cpy_buff);
209     free(cpy_buff);
210     return buff;
211 }
212
213
214 /* Populates buffer with properly formatted key from object */
215 char *string_key(obj *o, char *buff){
216     char *cpy_buff;
217     char *k;
218     int len;
219
220     len = 2;
221     cpy_buff = malloc(sizeof(char)*len);
222     cpy_buff = memcpy(cpy_buff, "\"", len);
223     buff = fill_buff(buff, cpy_buff);
224     free(cpy_buff);
```

```
225
226         k = o->k;
227         len = strlen(k);
228         cpy_buff = malloc(sizeof(char)*(len+1));
229         memcpy(cpy_buff, k, len+1);
230         buff = fill_buff(buff, cpy_buff);
231         free(cpy_buff);
232
233         len = 2;
234         cpy_buff = malloc(sizeof(char)*len);
235         cpy_buff = memcpy(cpy_buff, "\"", len);
236         buff = fill_buff(buff, cpy_buff);
237         free(cpy_buff);
238
239         len = 4;
240         cpy_buff = malloc(sizeof(char)*len);
241         cpy_buff = memcpy(cpy_buff, " : ", len);
242         buff = fill_buff(buff, cpy_buff);
243         free(cpy_buff);
244
245         return buff;
246 }
247
248 /* Populates buff with properly formatted value from object */
249 char *string_val(obj *o, char *buff){
250         char *cpy_buff;
251         int len;
252
253         void *arr;
254
255         switch(o->v_typ) {
256           case 3:
257                len = snprintf(NULL, 0 , "%d" , o->i);
258                cpy_buff = malloc(sizeof(char)*(len+1));
259                snprintf(cpy_buff,len+1, "%d\n", o->i);
260                break;
261           case 4:
262                len = snprintf(NULL, 0 , "%f" , o->f);
263                cpy_buff = malloc(sizeof(char)*(len+1));
264                snprintf(cpy_buff,len+1, "%f", o->f);
265                break;
266           case 6:
267                len = snprintf(NULL, 0 , "\"%s\"" , o->s);
268                cpy_buff = malloc(sizeof(char)*(len+1));
269                snprintf(cpy_buff,len+1, "\"%s\"", o->s);
270                break;
271           case 7:
272                len= snprintf(NULL, 0 , "%s" , o->b ? "true" : "false");
```

```
273                 cpy_buff = malloc(sizeof(char)*(len+1));
274                 snprintf(cpy_buff, len+1,"%s", o->b ? "true" : "false");
275                 break;
276           case 8:
277                 arr = (void **) o->i_a;
278                 cpy_buff = arr_stringify(arr);
279                 break;
280           case 9:
281                 arr = (void **) o->f_a;
282                 cpy_buff = arr_stringify(arr);
283                 break;
284           case 10:
285                 arr = (void **) o->s_a;
286                 cpy_buff = arr_stringify(arr);
287                 break;
288           case 11:
289                 arr = (void **) o->b_a;
290                 cpy_buff = arr_stringify(arr);
291                 break;
292           default:
293                 cpy_buff = malloc(sizeof(char));
294         };
295         buff = fill_buff(buff, cpy_buff);
296         free(cpy_buff);
297         len = 4;
298         cpy_buff = malloc(sizeof(char)*len);
299         memcpy(cpy_buff, " , ", len);
300         buff = fill_buff(buff, cpy_buff);
301         free(cpy_buff);
302
303         return buff;
304 }
305
306
307 /* Appends properly formatted string object to buff */
308 char *rec_stringify(obj *o, char *buff){
309
310         size_t cpy_len;
311         const char *cpy_buff;
312         size_t buff_len;
313         char *old_buff;
314
315         o = o->next;
316         buff_len = (strlen(buff)+1)*sizeof(char);
317         old_buff = malloc(buff_len);
318         memset(old_buff, 0, buff_len);
319         old_buff = memcpy(old_buff, buff, buff_len);
320
```

```
321         cpy_buff = "{ ";
322         cpy_len = (strlen(cpy_buff) + 1)*sizeof(char);
323         buff = xrealloc(buff, (buff_len + cpy_len)*sizeof(char));
324         memcpy(buff, old_buff, buff_len);
325         memcpy(buff + buff_len -1, cpy_buff, cpy_len);
326
327         free(old_buff);
328
329         while (o != NULL) {
330                 if (o->v_typ == 5){
331                         //printf("==== Object before key ======\n %s \n ==========",
                                buff);
332                         if(o->k){
333                                 buff = string_key(o, buff);
334                         }
335                         //printf("==== Object after key ======\n %s \n ==========",
                            buff);
336                         buff = rec_stringify(o->o, buff);
337                         //printf("==== Object after rec_stringify ======\n %s \n
                            ==========", buff);
338                 }
339                 else{
340                         buff = string_key(o, buff);
341                         buff = string_val(o, buff);
342                 }
343                 o = o->next;
344         }
345
346         /* Remove comma after last value in object */
347         //printf("==== Object before removing comma returns =======\n %s \n
                ==========", buff);
348         buff_len = (strlen(buff))*sizeof(char);
349
350         old_buff = malloc(buff_len-1);
351         memset(old_buff, 0, buff_len-1);
352         old_buff = memcpy(old_buff, buff, buff_len-2);
353
354         //printf("==== Object after removing comma returns =======\n %s \n
                ==========", old_buff);
355         /* Add closing parenthesis of object */
356         buff_len = (strlen(old_buff)+1)*sizeof(char);
357
358         cpy_buff = "} , ";
359         cpy_len = (strlen(cpy_buff) + 1)*sizeof(char);
360
361         buff = xrealloc(buff, (buff_len + cpy_len));
362         memcpy(buff, old_buff, buff_len);
363         memcpy(buff + buff_len - sizeof(char), cpy_buff, cpy_len);
```

```
364
365         free(old_buff);
366         //printf("==== Object before rec_stringify returns =======\n %s \n
                ==========", buff);
367
368         return buff;
369 }
370
371 /*
372  * Called in Gantry to stringify an object.
373  * Calls rec_stringify to obtain string, cleans up string
374  */
375 char *obj_stringify(obj *o){
376         size_t buff_len;
377         char *buff;
378         char *pre_buff;
379
380         char *temp_buff = (char *)malloc(sizeof(char));
381         memcpy(temp_buff, "", 1);
382         pre_buff = rec_stringify(o, temp_buff);
383
384         //printf("==== Object before cleaning up =======\n %s \n ==========",
                pre_buff);
385
386         /* Remove comma after Object */
387         buff_len = (strlen(pre_buff))*sizeof(char);
388
389         buff = malloc(buff_len-1);
390         memset(buff, 0 , buff_len-1);
391
392         buff = memcpy(buff, pre_buff, buff_len-2);
393         //printf("==== Object after cleaning up =======\n %s \n ==========", buff)
                ;
394         free(pre_buff);
395
396         return buff;
397 }
398
399
400 /* Add key and value to the beginning of an object
401  * Accepts: key, typ, value
402  * Returns : Object with added key
403  */
404 obj *obj_addkey(obj *o, char *key, int typ, void *val) {
405         /* Save what o currently sexts as next */
406         obj *o_next = o->next;
407         /* Malloc object */
408         obj *o_new = (obj *) malloc(sizeof(obj));
```

```
409         o_new->k = key;
410         o_new->v_typ = typ;
411         o_new->next = o_next;
412 
413         o->next = o_new;
414 
415         int l;
416         char *s;
417         switch(o_new->v_typ) {
418           case 3: o_new->i = *((int *) val); break;
419           case 4: o_new->f = *((double *) val); break;
420           case 5: o_new->o = (obj *) val; break;
421           case 6:
422                  l = strlen((const char *)(*(void **)val)) + 1;
423                  s = (char *)malloc(sizeof(l));
424                  memcpy(s, (const char *)(*(void **)val), l);
425                  o_new->s = s;
426                  break;
427           case 7: o_new->b = *((bool *) val); break;
428           case 8: o_new->i_a = (arr_int *) val; break;
429           case 9: o_new->f_a = (arr_flt *) val; break;
430           case 10: o_new->s_a = (arr_str *)val; break;
431           case 11: o_new->b_a = (arr_bool *)val; break;
432         };
433         return o;
434 }
435 
436 
437 /*
438  * Recursively search for a key in an object
439  */
440 obj *obj_findkey(obj *o, char *keys) {
441   char *k, *keys_dup;
442   keys_dup = strdup(keys);
443 
444   // parse a key to search for
445   k = strsep(&keys_dup, ".");
446 
447   // search for keys within an object
448   o = o->next; // get first key from head struct
449   while (o != NULL) {
450     if (strcmp(o->k, k) == 0) {
451       // found our key
452       if (strlen(k) == strlen(keys)) {
453         free(k);
454         return o;
455       }
456       // continue nested key search
```

```
457       else if (o->v_typ == 5) {
458         o = obj_findkey(o->o, keys_dup);
459         free(k);
460         return o;
461       }
462     }
463     o = o->next;
464   }
465 
466   // not found
467   free(k);
468   return NULL;
469 }
470 
471 /*
472  * Assign a value to an object
473  */
474 int obj_assign(obj *o, int t, void *v) {
475   int l;
476   char *s;
477   if (o != NULL) {
478     // Set new key type
479     o->v_typ = t;
480     // Set key value
481     switch(o->v_typ) {
482       case 3: o->i = *(int *)v; break;
483       case 4: o->f = *(double *)v; break;
484       case 5: o->o = (obj *)v; break;
485       case 6:
486         // malloc space for the string and store
487         l = strlen((const char *)(*(void **)v)) + 1;
488         s = (char *)malloc(sizeof(l));
489         strcpy(s, (const char *)(*(void **)v));
490         o->s = s;
491         break;
492       case 7: o->b = *(bool *)v; break;
493       case 8: o->i_a = (arr_int *)v; break;
494       case 9: o->f_a = (arr_flt *)v; break;
495       case 10: o->s_a = (arr_str *)v; break;
496       case 11: o->b_a = (arr_bool *)v; break;
497     };
498   }
499   return 0;
500 }
501 
502 
503 /*
504  * Get a key value from an Object
```

```
505  */
506 void *obj_getkey(obj *o, int t) {
507   /*
508    * Throw a runtime error if the type
509    * requested does not match actual type
510    */
511   if (o->v_typ != t) {
512     printf("Runtime Error: Invalid type requested from Object ID [%s] [%p]\n",
513            o->k, &o);
514     exit(2);
515   }
516   switch(o->v_typ) {
517     case 3: return (void *)&o->i;
518     case 4: return (void *)&o->f;
519     case 5: return (void *)o->o;
520     case 6: return (void *)&o->s;
521     case 7: return (void *)&o->b;
522     case 8: return (void *)o->i_a;
523     case 9: return (void *)o->f_a;
524     case 10: return (void *)o->s_a;
525     case 11: return (void *)o->b_a;
526   };
527   return NULL;
528 }
529
530 /*
531  * Get a key value's type from an Object
532  */
533 int obj_gettyp(obj *o) {
534   return o->v_typ;
535 }
536
537 /*
538  * Get the length of an array
539  */
540 int arr_length(void *arr) {
541   struct arr_int *arr_i = *(arr_int **)(arr);
542   return arr_i->len;
543 }
544
545
546 #ifdef BUILD_TEST
547 int main () {
548   obj *o = (obj *) malloc(sizeof(obj)); // head
549   obj *o2 = (obj *) malloc(sizeof(obj));
550   obj *o3 = (obj *) malloc(sizeof(obj));
551   obj *o4_head = (obj *) malloc(sizeof(obj));
552   obj *o4 = (obj *) malloc(sizeof(obj));
```

```
  obj *o5_head = (obj *) malloc(sizeof(obj));
  obj *o5 = (obj *) malloc(sizeof(obj));
  obj *o6 = (obj *) malloc(sizeof(obj));

  /*
   * Test data structure represented below:
   * object o = {| int key2 : 3,
   * object key3 :
   * {| object nestedkey4 :
   * {| string nestedkey5 : "test",
   * float nestedkey6 : 30.65
   * |}
   * |}
   * |}
   */

  o->k = NULL;
  o->next = o2;

  o2->k = "key2";
  o2->i = 2;
  o2->v_typ = 3;
  o2->next = o3;

  o3->k = "key3";
  o3->o = o4_head;
  o3->v_typ = 5;
  o3->next = NULL;

  o4_head->next = o4;
  o4->k = "nestedkey4";
  o4->o = o5_head;
  o4->v_typ = 5;

  o5_head->next = o5;
  o5->k = "nestedkey5";
  o5->s = "test";
  o5->v_typ = 6;
  o5->next = o6;

  o6->k = "nestedkey6";
  o6->f = 30.65;
  o6->v_typ = 4;
  o6->next = NULL;

  obj *tmp;
  void *v;
  tmp = obj_findkey(o, "key2");
```

```
601   v = obj_getkey(tmp, tmp->v_typ);
602   printf("%d\n", *(int *)v);
603   double t = 42.24;
604   obj_assign(tmp, 4, (void *)&t);
605   tmp = obj_findkey(o, "key2");
606   v = obj_getkey(tmp, tmp->v_typ);
607   printf("%lf\n", *(double *)v);
608
609   obj_findkey(o, "key3.nestedkey4.nestedkey5");
610
611   obj_findkey(o, "key3.nestedkey4.nestedkey6");
612   obj_findkey(o, "key3.nestedkey4");
613   obj_findkey(o, "keyfail.nestedkey4");
614
615   char *buff;
616   buff = obj_stringify(o);
617   printf("====THE OBJECT=====\n\n %s \n\n ===============\n" , buff);
618   free(buff);
619
620   obj *o_new = (obj *) malloc(sizeof(obj)); // head
621   obj *o2_new = (obj *) malloc(sizeof(obj));
622   o_new->k = NULL;
623   o_new->next = o2_new;
624
625   o2_new->k = "key2";
626   o2_new->i = 2;
627   o2_new->v_typ = 3;
628   o2_new->next = NULL;
629
630   buff = obj_stringify(o_new);
631   printf("====THE OBJECT NEW=====\n\n %s \n\n ===============\n" , buff);
632
633   /* Array Stringify */
634   printf("Before arr_int \n");
635   struct arr_int *ia = (arr_int *) malloc(sizeof(arr_int));
636   printf("After arr_int \n");
637   //ia->len = 4;
638
639   //ia->typ = 3;
640   printf("After setting typ and len \n");
641   //
642   int *arr = (int *) malloc(4*sizeof(int));
643
644   arr[0] = 1;
645   arr[1] = 2;
646   arr[2] = 3;
647   arr[3] = 4;
648
```

```
649   ia->len = 4;
650   ia->typ = 3;
651   ia->i_a = arr;
652
653   printf("%d \n " , arr[1]);
654
655   //int array
656
657   //printf("%d\n" , ia->len);
658   //printf("Strinfied array %s \n", s);
659   //free(s);
660   //free(ia->i_a);
661   //void *int_array = *(void **)(ia);
662   //char *s = arr_stringify(int_array);
663   //printf("%s\n", s);
664
665   free(int_array);
666   free(s);
667   free(ia->i_a);
668   free(ia);
669   free(buff);
670
671   free(o_new);
672   free(o2_new);
673
674   free(o);
675   free(o2);
676   free(o3);
677   free(o4);
678   free(o4_head);
679   free(o5);
680   free(o5_head);
681   free(o6);
682 }
683 #endif
```

gantrylib_string.c

Listing 58: String

```
/*
 * Author: Audrey Copeland
 * Contributor: Taimur Samee
 */

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

char *get_string(char* req) {
       char s[100];
       char *ret;
       printf("%s", req);
       scanf("%s", s);
       ret = malloc(strlen(s))+1;
       memcpy(ret, s, strlen(s)+1);
       return ret;
}

int stoint(char *s){
       int ret = atoi(s);
       return ret;
}

char *slice(char *src, int begin, int end){
       int len = end - begin;
       char *dest = malloc(len + 1);
       int i;
       if (begin>end){
              printf("Runtime Error: Slice begin integer %d is greater than end
                  integer %d,", begin, end);
       }
       int dest_i = 0;
       for (i=begin;i<end&&src[i]!='\0' ; i++){
              dest[dest_i] = src[i];
              dest_i++;
       }
       for (; i <= end; i++){
              dest[dest_i] = '\0';
       }
       return dest;
}

```

```
44 /* Returns 0 if equal, -1 if a<b, 1 if a>b */
45 int stringcmp(const char *a, const char *b){
46         int x = strcmp(a, b);
47         int res = 0;
48         if (x == 0){
49                 res = 0;;
50         }
51         else if (x < 0){
52                 res = -1;
53         }
54         else {
55                 res = 1;
56         }
57         return res;
58 }
59
60 /* Returns 1 if equal , returns 0 if not equal */
61 bool stringeq(const char *a , const char *b){
62         int x = strcmp(a, b);
63         bool res = 0;
64         if (x == 0){
65                 res = 1;
66         }
67         else {
68                 res = 0;
69         }
70         return res;
71 }
72
73 /* Get passed two strings, malloc space for concatenation */
74 char *string_concat(char *a, char *b){
75         int len_a = strlen(a);
76         int len_b = strlen(b);
77         char *c = malloc(len_a + len_b + 1);
78         memcpy(c, a, len_a);
79         memcpy(c + len_a, b, len_b + 1);
80         return c;
81 }
82
83 int string_length(char *a){
84         return strlen(a);
85 }
86
87 int test_slice(){
88         char *src = "this is a string";
89         char *dest = slice(src, 1, 3);
90         printf("%s\n", dest);
91         dest = slice(src, 0, 10);
```

```
92          printf("%s\n", dest);
93          dest = slice(src, 4, 20);
94          printf("%s\n", dest);
95          return 0;
96  }
97
98  int test_string_concat(){
99          char a[] = "foo";
100         char b[] = "bar";
101         char *c = string_concat(a, b);
102         printf("%s", c);
103         free(c);
104         return 0;
105 }
106
107 int test_stringcmp(){
108         char a [] = "foo";
109         char b [] = "bar";
110         char c [] = "foo";
111         int x = stringcmp(a, b);
112         int y = stringcmp(a, c);
113         printf("These are different %d \n", x);
114         printf("These are the same %d \n", y);
115         return 0;
116 }
117
118
119 int test_string_length(){
120         char a [] = "foo";
121         char b [] = "foobar";
122         int x = string_length(a);
123         int y = string_length(b);
124         printf("This should be 3 : %d \n", x);
125         printf("This should be 6 : %d \n", y);
126         return 0;
127 }
128
129 #ifdef BUILD_TEST
130 int main(){
131         printf("=== Testing String Comparison ===\n");
132         test_stringcmp();
133         printf("=== Testing String Concat ===\n");
134         test_string_concat();
135         printf("=== Testing String Length ===\n");
136         test_string_length();
137         printf("=== Testing String Slice ===\n");
138         test_slice();
139         char *s = get_string("Please enter input : ");
```

```
140         printf("%s\n", s);
141         char *si = "4";
142         int i = stoint(si);
143         printf("%d\n", i);
144         return 0;
145 }
146 #endif
```

gantrylib_printb.c

Listing 59: Printb

```
/*
 * Gantry: Print Boolean Function
 * Author: Taimur Samee
 */

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

void print_b (bool b) {

    printf("%s\n", b ? "true" : "false");

}
```

gantry_comp.sh

Listing 60: Gantry_Comp

```
#!/bin/bash

# Author: Walter Meyer
# A wrapper script for compiling Gantry programs

if [ $# -ne 1 ]
then
        echo "Gantry Compiler Script Usage:"
        echo "./gantry_comp.sh program.gty"
        exit 1
fi

program=${1::-4}
./gantry.native < $1 \
               > $program.ll && llc $program.ll \
               > $program.s && gcc -o $program $program.s \
                 gantrylib_string.o gantrylib_http.o gantrylib_obj.o \
                 gantrylib_nap.o -L/usr/lib/x86_64-linux-gnu -lcurl
```

8.2 Test Suite

8.2.1 Test Script

Listing 61: Test Script

```
#!/bin/bash

# Testing script taken from MicroC
# with minor changes for automation by
# Audrey Copeland and Walter Meyer

# Regression testing script for Gantry
# 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="gcc"
LDFLAGS="-L/usr/lib/x86_64-linux-gnu -lcurl"

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

```
44
45 SignalError() {
46     if [ $error -eq 0 ] ; then
47         echo "FAILED"
48         error=1
49     fi
50     echo " $1"
51 }
52
53 # Compare <outfile> <reffile> <difffile>
54 # Compares the outfile with reffile. Differences, if any, written to difffile
55 Compare() {
56     generatedfiles="$generatedfiles $3"
57     echo diff -b $1 $2 ">" $3 1>&2
58     diff -b "$1" "$2" > "$3" 2>&1 || {
59         SignalError "$1 differs"
60         echo "FAILED $1 differs from $2" 1>&2
61     }
62 }
63
64 # Run <args>
65 # Report the command, run it, and report any errors
66 Run() {
67     echo $* 1>&2
68     eval $* || {
69         SignalError "$1 failed on $*"
70         return 1
71     }
72 }
73
74 # RunFail <args>
75 # Report the command, run it, and expect an error
76 RunFail() {
77     echo $* 1>&2
78     eval $* && {
79         SignalError "failed: $* did not report an error"
80         return 1
81     }
82     return 0
83 }
84
85 Check() {
86     error=0
87     basename=‘echo $1 | sed ’s/.*\\///
88                             s/.gty//’‘
89     reffile=‘echo $1 | sed ’s/.gty$//’‘
90     basedir="‘echo $1 | sed ’s/\/[^\/]*$//’‘/."
91
```

```
92      echo -n "$basename..."
93
94      echo 1>&2
95      echo "###### Testing $basename" 1>&2
96
97      generatedfiles=""
98      generatedfiles="$generatedfiles ${basename}.ll ${basename}.s ${basename}.exe $
            {basename}.out" &&
99
100     Run "$GANTRY" "<" "$1" ">" "${basename}.ll" &&
101     Run "$LLC" "${basename}.ll" ">" "${basename}.s" &&
102     Run "$CC" "-o" "${basename}.exe" "${basename}.s" \
103           "gantrylib_http.o" "gantrylib_string.o" \
104           "gantrylib_obj.o" "gantrylib_nap.o" \
105           "gantrylib_printb.o" "$LDFLAGS" &&
106     Run "./${basename}.exe" > "${basename}.out" &&
107     Compare ${basename}.out ${reffile}.out ${basename}.diff
108
109     # Report the status and clean up the generated files
110
111     if [ $error -eq 0 ] ; then
112         if [ $keep -eq 0 ] ; then
113             rm -f $generatedfiles
114         fi
115         echo "OK"
116         echo "###### SUCCESS" 1>&2
117     else
118         echo "###### FAILED" 1>&2
119         globalerror=$error
120     fi
121 }
122
123 CheckFail() {
124     error=0
125     basename=‘echo $1 | sed ’s/.*\\///
126                             s/.gty//’‘
127     reffile=‘echo $1 | sed ’s/.gty$//’‘
128     basedir="‘echo $1 | sed ’s/\/[^\/]*$//’‘/."
129
130     echo -n "$basename..."
131
132     echo 1>&2
133     echo "###### Testing $basename" 1>&2
134
135     generatedfiles=""
136
137     generatedfiles="$generatedfiles ${basename}.ll ${basename}.s ${basename}.exe $
            {basename}.err ${basename}.diff" &&
```

```
138     RunFail "$GANTRY" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
139     Compare ${basename}.err ${reffile}.err ${basename}.diff
140 
141     if [ $error -eq 0 ] ; then
142         if [ $keep -eq 0 ] ; then
143             rm -f $generatedfiles
144         fi
145         echo "OK"
146         echo "###### SUCCESS" 1>&2
147     else
148         echo "###### FAILED" 1>&2
149         globalerror=$error
150     fi
151 }
152 
153 while getopts kdpsh c; do
154     case $c in
155         k) # Keep intermediate files
156             keep=1
157             ;;
158         h) # Help
159             Usage
160             ;;
161     esac
162 done
163 
164 shift `expr $OPTIND - 1`
165 
166 LLIFail() {
167   echo "Could not find the LLVM interpreter \"$LLI\"."
168   echo "Check your LLVM installation and/or modify the LLI variable in testall.sh
        "
169   exit 1
170 }
171 
172 which "$LLI" >> $globallog || LLIFail
173 
174 if [ $# -ge 1 ]
175 then
176     files=$@
177 else
178     files="tests/test_*.gty tests/fail_*.gty"
179 fi
180 
181 for file in ./tests/test*.gty; do
182         if [[ `grep "****TEST****" $file` ]]; then
183           sed -n -e '/\*\*\*\*TEST\*\*\*\*/,$p' $file | tail -n +2 | head -n -1 >
              ${file%.*}.out
```

```
          echo "Generated $(echo $file | cut -c 3-).out"
        fi
done

for file in ./tests/fail*.gty; do
        if [[ `grep "****TEST****" $file` ]]; then
          sed -n -e '/\*\*\*\*TEST\*\*\*\*/,$p' $file | tail -n +2 | head -n -1 >
              ${file%.*}.err
          echo "Generated $(echo $file | cut -c 3-).err"
        fi
done


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
```

8.2.2 Test Cases

Due to the large number of tests, we list all test code in a single block, with each test separated by headers denoting the file name. For individual test files, see the source code.

```
==> fail_array_multtype.gty <==
int main(){
    int array z = [1, "foo",3,4];
    return 0;
}

/*
****TEST****
Fatal error: exception Failure("Multiple types inside an array of type int")
*/

==> fail_assign_declaration.gty <==
int main() {
    object a.x = {| string foo: "bar" |};
}

/*
****TEST****
Fatal error: exception Failure("Can not declare multi-level variable a.x in
    expression object a.x = {| string foo : "bar" |}")
*/

==> fail_for1.gty <==
int main() {
    int x = 1;

    // This should fail beceause y is undefined
    for(y = 1; x > 0; x--) {
           print_i(x);
    }

    return 0;
}

/*
****TEST****
Fatal error: exception Failure("undeclared identifier y")
*/

==> fail_for2.gty <==
int main() {
    int y = 1;
```

```
42
43      // This should fail beceause x is undefined
44      for(y = 1; x > 0; x--) {
45              print_i(y);
46      }
47
48      return 0;
49  }
50
51  /*
52  ****TEST****
53  Fatal error: exception Failure("undeclared identifier x")
54  */
55
56  ==> fail_for3.gty <==
57  int main() {
58      int x = 1;
59
60      // This should fail beceause x is undefined
61      for(x= 1; x > 0; y--) {
62              print_i(x);
63      }
64
65      return 0;
66  }
67
68  /*
69  ****TEST****
70  Fatal error: exception Failure("undeclared identifier y")
71  */
72
73  ==> fail_if1.gty <==
74  int main(){
75      bool x = true;
76      if(x){
77          int y = 2;
78      }
79      if(y){
80      }
81      return 0;
82  }
83
84  /*
85  ****TEST****
86  Fatal error: exception Failure("expected Boolean expression in y")
87  */
88
89  ==> fail_if2.gty <==
```

```
int main(){
    if(true){
        print_i(x);
    }
    return 0;
}

/*
****TEST****
Fatal error: exception Failure("undeclared identifier x")
*/

==> fail_if3.gty <==
int main(){
    if(true){
        foo();
    }
    return 0;
}

/*
****TEST****
Fatal error: exception Failure("unrecognized function foo")
*/

==> fail_increment.gty <==
int main(){
    bool v = true;
    bool w = false;
    w++;
    w = !w;
    return 0;
}

/*
****TEST****
Fatal error: exception Failure("Illegal unary operator ++bool in w++")
*/

==> fail_invalid_array_typ.gty <==
int main(){
    string array y = [["foo"], ["bar"]];
    return 0;
}

/*
****TEST****
Fatal error: exception Failure("Invalid array type")
```

```
*/

==> fail_invalid_assign_array.gty <==
int main(){
    int array four = [1,2,3,4];
    string z = four[3];
    return 0;
}

/*
****TEST****
Fatal error: exception Failure("illegal assignment string = int in string z =
    four[3]")
*/

==> fail_multilevel_declaration.gty <==
int main(){
    object o.z = {|
           string s = "foo"
            |};
    return 0;
}

/*
****TEST****
Fatal error: exception Failure("Can not declare multi-level variable o.z in
    expression object o.z = {| string s = "foo" |}")
*/

==> fail_null_function_return.gty <==
null foo(){
}

null bar(){
    return;
}

null foobar(){
    return 0;
}


int main(){
    foo();
    print_s("Got past null function without return");
    bar();
    print_s("Got past return in null function");
    foobar();
```

```
184     print_s("Got here, oops");
185     return 0;
186 }
187 
188 /*
189 ****TEST****
190 Fatal error: exception Failure("return gives int expected null in 0")
191 */
192 
193 ==> fail_object_double_declaration.gty <==
194 int main(){
195     /* This should be fine */
196     int y = 2;
197     int y = 4;
198     /* This should not work */
199     object x = {| int y : 2 |};
200     object x = {| int z : 3 |};
201 }
202 
203 /*
204 ****TEST****
205 Fatal error: exception Failure("Object x declared twice")
206 */
207 
208 ==> fail_print_int.gty <==
209 int main(){
210     string s = "foo";
211     print_i(s);
212     return 0;
213 }
214 
215 /*
216 ****TEST****
217 Fatal error: exception Failure("illegal actual argument found string expected int
        in s")
218 */
219 
220 ==> fail_stringcmp.gty <==
221 int main(){
222     int x = 4;
223     int y = 5;
224     int z = stringcmp(4, 5);
225     return 0;
226 }
227 
228 /*
229 ****TEST****
230 Fatal error: exception Failure("illegal actual argument found int expected string
```

```
     in 4")
*/

==> fail_unrecognized_function.gty <==
int main(){
    string s = "foobar";
    string split = string_slice(2, 10, s);
    return 0;
}

/*
****TEST****
Fatal error: exception Failure("unrecognized function string_slice")
*/

==> fail_while.gty <==
int main(){

    int x = 10;

    while (x < 20){
        print_i(x);
        x = x+1;
    }

    while (x){
        print_i(x);
        x = x+1;
    }
    return 0;
}

/*
****TEST****
Fatal error: exception Failure("expected Boolean expression in x")
*/

==> test_add_int.gty <==
// pass - add two integers

int main(){
    int x = 4;
    int y = 6;

    int z = x + y;
    print_i(z);
    return 0;
}
```

```
278
279 /*
280 ****TEST****
281 10
282 */
283
284 ==> test_addkey.gty <==
285 int main() {
286     object o = {|
287             string foo : "foo",
288             string bar : "bar"
289            |};
290
291     object b = obj_addkey(o, "biz", 3, 3);
292     string s = obj_stringify(b);
293     print_s(s);
294
295     string y = "baz value";
296     object c = obj_addkey(b, "baz", 6, y);
297     string sc = obj_stringify(c);
298     print_s(sc);
299
300
301     object d = obj_addkey(c, "blue", 7, true);
302     string sd = obj_stringify(d);
303     print_s(sd);
304
305
306     object e = obj_addkey(d, "yellow", 4, 3.14);
307     string se = obj_stringify(e);
308     print_s(se);
309
310     float test_float = e.yellow;
311     int test_int = e.biz;
312     string test_string = e.baz;
313     bool test_bool = e.blue;
314
315     print_d(test_float);
316     print_i(test_int);
317     print_s(test_string);
318     print_b(test_bool);
319
320     return 0;
321 }
322
323 /*
324 ****TEST****
325 { "biz" : 3 , "bar" : "bar" , "foo" : "foo" }
```

```
{ "baz" : "baz value" , "biz" : 3 , "bar" : "bar" , "foo" : "foo" }
{ "blue" : true , "baz" : "baz value" , "biz" : 3 , "bar" : "bar" , "foo" : "foo"
    }
{ "yellow" : 3.140000 , "blue" : true , "baz" : "baz value" , "biz" : 3 , "bar" :
    "bar" , "foo" : "foo" }
3.140000
3
baz value
true
*/

==> test_array.gty <==
int main () {
  int array x = [1, 2, 3, 4, 5, 6, 7, 8];
  int i = x[0] + x[2];
  print_i(i);
  print_i(x[1+2] + x[x[1] + x[2]]);
  string array y = [ "Hello", "Arrays" ];
  print_s(y[0] ^ " " ^ y[1]);
  bool array b = [true, false];

  if (b[0] == b[0]) {
    print_s("woop");
  }
  if (b[0] != b[1]) {
    print_s("woop");
  }

}
/*
****TEST****
4
10
Hello Arrays
woop
woop
*/

==> test_arr_stringify.gty <==
int main() {
    int array x = [1,2,3];
    string s = arr_stringify(x);
    print_s(s);

    float array y = [1.00, 3.14, 4.00, 9.88];
    string s = arr_stringify(y);
    print_s(s);

```

```
372     bool array b = [true, false];
373     string s = arr_stringify(b);
374     print_s(s);
375 
376     string array c = ["this", "is", "a", "string", "array", "yay"];
377     string s = arr_stringify(c);
378     print_s(s);
379     object o1 = {|
380         string Image : "alpine:latest", int x : 3, int array z : [1,2,3]
381          |};
382 
383     int array test = o1.z;
384     print_i(test[1]);
385 
386     int array Cmd = [1,2];
387     string array arr2 = ["foo", "barr", "zoo"];
388     float array arr3 = [1.22, 3.000, 4.2, 5.3];
389     bool array arr4 = [true];
390     object o2 = obj_addkey(o1, "foo", 8, Cmd);
391     object o3 = obj_addkey(o2, "key1", 10, arr2);
392     object o4 = obj_addkey(o3, "key3", 9, arr3);
393     object o5 = obj_addkey(o4, "key4", 11, arr4);
394     string s = obj_stringify(o1);
395     print_s(s);
396     return 0;
397 }
398 
399 /*
400 ****TEST****
401 [ 1 , 2 , 3 ]
402 [ 1.000000 , 3.140000 , 4.000000 , 9.880000 ]
403 [ true , false ]
404 [ "this" , "is" , "a" , "string" , "array" , "yay" ]
405 2
406 { "key4" : [ true ] , "key3" : [ 1.220000 , 3.000000 , 4.200000 , 5.300000 ] , "
        key1" : [ "foo" , "barr" , "zoo" ] , "foo" : [ 1 , 2 ] , "z" : [ 1 , 2 , 3 ] ,
         "x" : 3 , "Image" : "alpine:latest" }
407 */
408 
409 ==> test_bool_val.gty <==
410 int main(){
411     bool a = true;
412     bool b = false;
413     if (true){
414         print_s("true");
415     }
416     if (a != b){
417         print_s("true != false");
```

```
418     }
419     if (!a == b){
420         print_s("false == false");
421     }
422     int x = 2;
423     int y = 3;
424     if (x<y){
425         print_s("2 is less than 3");
426     }
427     if (!b){
428         print_s("!b is true");
429     }
430 }
431 
432 /*
433 ****TEST****
434 true
435 true != false
436 false == false
437 2 is less than 3
438 !b is true
439 */
440 
441 ==> test_comp_float.gty <==
442 int main(){
443     float a = 3.14;
444     float b = 5.00;
445     if (a<b){
446         print_s("hello");
447     }
448     return 0;
449 }
450 
451 /*
452 ****TEST****
453 hello
454 */
455 
456 ==> test_concat.gty <==
457 
458 string str_conc(string a, string b){
459     string c = a ^ b;
460     return c;
461 }
462 
463 int main(){
464     string a = "foo";
465     string b = "bar";
```

```
    string c = str_conc(a , b);
    print_s(c);
    return 0;
}


/*
****TEST****
foobar
*/

==> test_control_flow.gty <==
int main(){
    int i = 10;
    bool v = true;
    for (i = 0; i < 4; i++){
        if (i==2){
            print_i(i);
        }
    }
    bool x = false;
    while(x == false){
        i++;
        if (i>5 && v == true){
            x = true;
        }
        else{
            print_b(x);
        }
    }
    return 0;
}

/*
****TEST****
2
false
*/

==> test_div_int.gty <==
// divide integers

int main(){
    int x = 4;
    int y = 6;
    int z = y/x;

    print_i(z);
```

```
514     return 0;
515 }
516
517 /*
518 ****TEST****
519 1
520 */
521
522 ==> test_float_add.gty <==
523 // Adding two floats
524
525 int main() {
526
527     float x = 1.51;
528     float y = 2.7;
529     float z = x + y;
530
531     print_d(z);
532     return 0;
533 }
534
535 /*
536 ****TEST****
537 4.210000
538 */
539
540 ==> test_float_div.gty <==
541 // Dividing one float by another
542
543 int main() {
544
545     float x = 8.75;
546     float y = 3.5;
547     float z = x / y;
548
549     print_d(z);
550     return 0;
551
552 }
553
554 /*
555 ****TEST****
556 2.500000
557 */
558
559 ==> test_float_eq.gty <==
560 // pass - check float eq and neq operators
561
```

```
562 int main() {
563 
564     float x = 1.5;
565     float y = 1.5;
566     float z = 2.51;
567 
568     if (x == y) {
569         print_s("equal");
570     }
571 
572     if (y == x) {
573         print_s("equal");
574     }
575 
576     if (x != z) {
577         print_s("not equal");
578     }
579 
580 
581     if (z != y) {
582         print_s("not equal");
583     }
584 
585     return 0;
586 
587 }
588 
589 /*
590 ****TEST****
591 equal
592 equal
593 not equal
594 not equal
595 */
596 
597 ==> test_float_gtgeq.gty <==
598 // pass - check float greater-than and geq operators
599 
600 int main() {
601 
602     float x = 4.5679;
603     float y = 4.5678;
604 
605     if (x > y) {
606         print_s("greater than");
607     }
608 
609     if (y >= y) {
```

```
        print_s("geq");
    }

    return 0;
}

/*
****TEST****
greater than
geq
*/

==> test_float_ltleq.gty <==
// pass - check float less-than and leq operators

int main() {

    float x = 4.5679;
    float y = 4.5678;

    if (y < x) {
        print_s("less than");
    }

    if (y <= y) {
        print_s("leq");
    }

    return 0;
}

/*
****TEST****
less than
leq
*/

==> test_float_mult.gty <==
// Multiplying two floats

int main() {

    float x = 2.5;
    float y = 3.5;
    float z = x * y;

    print_d(z);

```

```
}

/*
****TEST****
8.750000
*/

==> test_float_sub.gty <==
// pass - subtract floats

int main() {

    float x = 4.217;
    float y = 7.4591;
    float z = x - y;

    print_d(z);
    return 0;

}

/*
****TEST****
-3.242100
*/

==> test_for_dec_inc.gty <==
// pass - for loop

int main() {
    int x = 1;
    x--;
    print_i(x);
    x--;
    print_i(x);

    for(x = 5; x > 0; x--) {
           print_i(x);
    }

    for(int x = 0; x < 5; x++) {
           print_i(x);
    }

    print_i(x);
    return 0;
}

```

```
/*
****TEST****
0
-1
5
4
3
2
1
0
1
2
3
4
5
*/

==> test_gcd.gty <==
int gcd(int x, int y){
    while (x != y) {
        if ( x> y){
            x = x-y;
        }
        else{
            y = y-x;
        }
    }
    return x;
}



int main(){
    int a = gcd(21, 77);
    int b = gcd(22, 110);
    int c = gcd(72, 880);
    print_i(a);
    print_i(b);
    print_i(c);
}

/*
****TEST****
7
22
8
*/
```

```
==> test_global.gty <==
int x;
int y;

int main(){
    x = 0;
    y = 2;
    if (x==0){
        print_s("Global x is 0");
    }
    if (y==2){
        print_s("Global y is 2");
    }
    int y = 3;
    if (y==3){
        print_s("Local y is 3, no conflict on reassigning in this function.");
    }
    if (y==2){
        print_s("y is still 2");
    }
    return 0;
}

/*
****TEST****
Global x is 0
Global y is 2
Local y is 3, no conflict on reassigning in this function.
*/

==> test_hello_world.gty <==
// Tests printing Hello World!

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

/*
****TEST****
Hello World!
*/

==> test_httpget.gty <==
// pass - send GET and POST requests to RequestBin

int main() {

```

```
    string page = "https://httpbin.org/status/418";
    print_s(httpget(page));

    return 0;
}

/*
****TEST****

    -=[ teapot ]=-

       _....._
     .' _ _ `.
    | ."` ^ `". _,
    \_;`"---"`|//
      | ;/
      \_ _/
        `"""`

*/

==> test_httppost.gty <==
// pass - send a POST request to a mockbin page. Go to <page>/view to see
    requests

int main() {

    string page = "http://google.com";
    string req = "{ \"name\" : \"Mr Teatime\" }";
    print_s(httppost(page, req));

    return 0;
}

/*
****TEST****
<!DOCTYPE html>
<html lang=en>
  <meta charset=utf-8>
  <meta name=viewport content="initial-scale=1, minimum-scale=1, width=device-
      width">
  <title>Error 405 (Method Not Allowed)!!1</title>
  <style>
    *{margin:0;padding:0}html,code{font:15px/22px arial,sans-serif}html{background
        :#fff;color:#222;padding:15px}body{margin:7% auto 0;max-width:390px;min-
        height:180px;padding:30px 0 15px}* > body{background:url(//www.google.com/
        images/errors/robot.png) 100% 5px no-repeat;padding-right:205px}p{margin
        :11px 0 22px;overflow:hidden}ins{color:#777;text-decoration:none}a img{
```

```
       border:0}@media screen and (max-width:772px){body{background:none;margin-
       top:0;max-width:none;padding-right:0}}#logo{background:url(//www.google.
       com/images/branding/googlelogo/1x/googlelogo_color_150x54dp.png) no-repeat
       ;margin-left:-5px}@media only screen and (min-resolution:192dpi){#logo{
       background:url(//www.google.com/images/branding/googlelogo/2x/
       googlelogo_color_150x54dp.png) no-repeat 0% 0%/100% 100%;-moz-border-image
       :url(//www.google.com/images/branding/googlelogo/2x/
       googlelogo_color_150x54dp.png) 0}}@media only screen and (-webkit-min-
       device-pixel-ratio:2){#logo{background:url(//www.google.com/images/
       branding/googlelogo/2x/googlelogo_color_150x54dp.png) no-repeat;-webkit-
       background-size:100% 100%}}#logo{display:inline-block;height:54px;width
       :150px}
  </style>
  <a href=//www.google.com/><span id=logo aria-label=Google></span></a>
  <p><b>405.</b> <ins>Thats an error.</ins>
  <p>The request method <code>POST</code> is inappropriate for the URL <code>/</
      code>. <ins>Thats all we know.</ins>

*/

==> test_if.gty <==
// pass - test if, elif, else

int main(){
    int x = 1;

    if(x == 0){
        //test fails
        print_s("If/else failed");
    }
    else {
        //test passes
        print_s("If/else works!");
    }

return 0;
}

/*
****TEST****
If/else works!
*/

==> test_int_dcl.gty <==
// pass - declare an integer

int main(){
    int y = 42;
```

```
    print_i(y);
    return 0;
}

/*
****TEST****
42
*/

==> test_logical_and_multiple.gty <==
//pass - test if multiple logical ANDs work in a condition

int main() {

  int x = 0;
  int y = 1;
  int z = 3;
  float a = 4.0;

  if (x == 0 && y == 1 && z == 3 && a == 4.0) {
    print_s("x = 0 and y = 1 and z = 3 and a = 4.0");
  }

  return 0;
}

/*
****TEST****
x = 0 and y = 1 and z = 3 and a = 4.0
*/

==> test_logical_and_or_mixed.gty <==
//pass - test if multiple logical ANDs work in a condition

int main() {

  int x = 0;
  int y = 0;
  int z = 2;
  float a = 4.0;

  if (x == 0 || y == 1 && z == 3 || a == 4.0) {
    print_s("x = 0 or y = 1 and z = 3 or a = 4.0");
  }

  if (x == 0 && a == 5.5 || z == 2) {
    print_s("x = 0 and a = 5.5 or z = 2");
  }
```

```
927
928   if (x == 1 || a == 5.5 && z == 2) {
929      print_s("this shouldn't work, dummy!");
930   }
931
932   return 0;
933 }
934
935 /*
936 ****TEST****
937 x = 0 or y = 1 and z = 3 or a = 4.0
938 x = 0 and a = 5.5 or z = 2
939 */
940
941 ==> test_logical_and_single.gty <==
942 //pass - test if a single logical AND works in a condition
943
944 int main () {
945
946   int x = 0;
947   int y = 1;
948
949
950   if (x == 0 && y == 1) {
951     print_s("x = 0 and y = 1");
952   }
953
954   return 0;
955 }
956 /*
957 ****TEST****
958 x = 0 and y = 1
959 */
960
961 ==> test_logical_or_multiple.gty <==
962 // pass - test if multiple logical ORs work in a condition
963
964 int main () {
965
966   int x = 0;
967   float y = 1.0;
968   bool z = true;
969
970   if (x == 1 || y == 0.0 || z) {
971     print_s("x = 1 or y = 0.0 or z = true");
972   }
973
974 }
```

```
975
976 /*
977 ****TEST****
978 x = 1 or y = 0.0 or z = true
979 */
980
981 ==> test_logical_or_single.gty <==
982 //pass - test if a single logical OR works in a condition
983
984 int main () {
985
986   int x = 0;
987   int y = 0;
988
989
990   if (x == 0 || y == 1) {
991     print_s("x = 0 or y = 1");
992   }
993
994   return 0;
995 }
996 /*
997 ****TEST****
998 x = 0 or y = 1
999 */
1000
1001 ==> test_math.gty <==
1002 int main(){
1003     int x = -19;
1004     int y = 10;
1005     int d = 4;
1006     int c = -192;
1007     int m = 3;
1008     print_i(x + y);
1009     print_i(x * y);
1010     print_i(c/2);
1011     print_i(m/2);
1012     print_i(c - x);
1013     if (true == true){
1014         print_s("This is true");
1015     }
1016     if (true != false){
1017         print_s("This is false");
1018     }
1019     if (c < x) {
1020         print_s("-192 is less than -19");
1021         if ( d > c){
1022             print_s("4 is greater than -192");
```

```
        }
        else {
            print_s("nope");
        }
        if (3/4 < 1){
            print_s("3/4 is less than 1");
        }
        else {
            print_s("int division then comp doesn't work");
        }
    }

    return 0;
}

/*
****TEST****
-9
-190
-96
1
-173
This is true
This is false
-192 is less than -19
4 is greater than -192
3/4 is less than 1
*/

==> test_mult_int.gty <==
// pass - multiply integers

int main(){
    int x = 4;
    int y = 6;

    int z = x * y;
    print_i(z);

    return 0;
}

/*
****TEST****
24
*/

==> test_nap.gty <==
```

```
// Test nap() function (sleep wrapper)

int main() {

    int x = nap(5);
    print_i(x);
    return 0;
}

/*
****TEST****
0
*/

==> test_obj_arr_stringify.gty <==
int main(){
    object array x = [ {| int x: 4, int y: 10, int z: 15 |}, {|string s: "foo",
        string foo:"bar"|} , {| float z:2.00 |}, {|bool a: true |}];
    string s = arr_stringify(x);
    print_s(s);
    return 0;
}


/*
****TEST****
[ { "z" : 15 , "y" : 10 , "x" : 4 } , { "foo" : "bar" , "s" : "foo" } , { "z" :
    2.000000 } , { "a" : true } ]
*/

==> test_object2.gty <==
int fun(object o) {
  print_k(o.x);
}

int main () {
  object a = {|
        string test_key : "Hello",
        int x : 123,
        object b : {| string hello : "World!" |}
         |};

  object b = {|
        string test_key : "Hello...",
        object b : {| string hello : "Mars!", int x : 111 |},
        int x : 456,
        bool a : true
         |};
```

```
1117
1118   print_k(a.test_key);
1119   print_k(a.b.hello);
1120   print_k(b.x);
1121   print_k(b.a);
1122
1123   // reassign keys to new values/types from primitives
1124
1125   int i = 42;
1126   bool z = true;
1127   string s = "Meaning of life";
1128   float f = 97.1;
1129
1130   a.test_key = i;
1131   a.b.hello = z;
1132   b.x = s;
1133   b.a = f;
1134   a.x = 42.42;
1135
1136   print_k(a.test_key);
1137   print_k(a.b.hello);
1138   print_k(b.x);
1139   print_k(b.a);
1140   print_k(a.x);
1141
1142   a.x = b.x;
1143   print_k(a.x);
1144   a.x = b.b;
1145   print_k(a.x.hello);
1146
1147   /*
1148   b.x = {| string sokey : "much value!" |};
1149   print_k(b.x);
1150   print_k(b.x.sokey);
1151   */
1152
1153   int i = 42;
1154   i = b.b.x;
1155   print_i(i);
1156   int x = b.b.x;
1157   print_i(x);
1158 }
1159 /*
1160 ****TEST****
1161 Hello
1162 World!
1163 456
1164 true
```

```
42
true
Meaning of life
97.100000
42.420000
Meaning of life
Mars!
111
111
*/

==> test_object_array2.gty <==
int main () {
  object array o = [ {| int x : 2 |}, {| int x : 3 |} ];
  object test = o[0];
  int x = test.x;
  print_i(x);

  object o2 = {| string array Cmd : ["echo", "Hello World!"] |};
  string array a = ["echo", "Hello Mars!", "why", "oh", "why"];
  int array c = [42, 420];
  o2.Cmd = a;
  string array b = o2.Cmd;
  print_s(b[0] ^ " " ^ b[1]);

  print_i(arr_length(b));
  print_i(arr_length(c));

  for(int i = 0; i < arr_length(b); i++) {
    print_s(b[i]);
  }

}
/*
****TEST****
2
echo Hello Mars!
5
2
echo
Hello Mars!
why
oh
why
*/

==> test_object_array.gty <==
int fun(int x, string y) {
```

```
  print_i(x);
  print_s(y);
}

int main () {
  string create = "{\"Image\": \"alpine:latest\", \"Cmd\": [\"echo\", \"hello
      world\"]}";

  object o = {|
        string Image : "alpine:latest", string array Cmd : ["echo", "Hello World
           "], int x : 3
         |};

  fun(o.x, o.Image);

}
/*
****TEST****
3
alpine:latest
*/

==> test_object.gty <==
int fun(object o) {
  print_k(o.x);
}

int main () {
  object a = {|
        string test_key : "Hello",
        int x : 123,
        object b : {| string hello : "World!" |}
         |};

  object b = {|
        string test_key : "Hello...",
        object b : {| string hello : "Mars!", int x : 111 |},
        int x : 456,
        bool a : true
         |};

  print_k(a.test_key);
  print_k(a.b.hello);
  print_k(a.x);

  print_k(b.test_key);
  print_k(b.b.hello);
  print_k(b.x);
```

```
  print_k(b.b.x);
  print_k(b.a);

  print_k(a.test_key);
  print_k(a.b.hello);
  print_k(a.x);
  print_k(a.z);
// print_k(a);

  fun(a);

}
/*
****TEST****
Hello
World!
123
Hello...
Mars!
456
111
true
Hello
World!
123
123
*/

==> test_object_stringify.gty <==
int main () {
  object a = {|
        string test_key : "Hello",
        int x : 123,
        object b : {| string hello : "World!" |}
         |};

  object b = {|
        string z : "foo",
        int y : 4
         |};

  object c = {|
        bool t: true,
        float f: 4.000
         |};

  string s = obj_stringify(a);
  print_s(s);
```

```
1307
1308   string objb = obj_stringify(b);
1309   print_s(objb);
1310
1311   string objc = obj_stringify(c);
1312   print_s(objc);
1313
1314   return 0;
1315 }
1316
1317 /*
1318 ****TEST****
1319 { "b" : { "hello" : "World!" } , "x" : 123 , "test_key" : "Hello" }
1320 { "y" : 4 , "z" : "foo" }
1321 { "f" : 4.000000 , "t" : true }
1322 */
1323
1324 ==> test_object_stringify_simple.gty <==
1325 int main(){
1326     object o = {|
1327         string name: "Joe",
1328         int age: 3,
1329         string array pets : ["Cat", "Dog", "Pig"]
1330          |};
1331     string to_string = obj_stringify(o);
1332     print_s(to_string);
1333 }
1334
1335 /*
1336 ****TEST****
1337 { "pets" : [ "Cat" , "Dog" , "Pig" ] , "age" : 3 , "name" : "Joe" }
1338 */
1339
1340 ==> test_obj_nested_stringify.gty <==
1341 int main(){
1342     string image = "foo";
1343     object exec =
1344         {|
1345             string Image : image,
1346             bool Tty : true,
1347             object HostConfig :
1348               {|
1349                  object PortBindings : {| int x : 2 |}
1350               |}
1351         |};
1352     string o = obj_stringify(exec);
1353     print_s(o);
1354     return 0;
```

```
}

/*
****TEST****
{ "HostConfig" : { "PortBindings" : { "x" : 2 } } , "Tty" : true , "Image" : "foo
    " }
*/

==> test_print_bool.gty <==
// Test bool printing

int main() {

    print_b(true);
    print_b(false);
    return 0;

}

/*
****TEST****
true
false
*/

==> test_print_int.gty <==
// Test integer printing

int main(){
    print_i(4);
    return 0;
}

/*
****TEST****
4
*/

==> test_stoint.gty <==
int main(){
    string x = "3";
    int y = stoint(x);
    print_i(y);

    x = "412";
    y = stoint(x);
    print_i(y);
}
```

```
1402
1403
1404 /*
1405 ****TEST****
1406 3
1407 412
1408 */
1409
1410 ==> test_str_eq.gty <==
1411 int main(){
1412     string x = "foo";
1413     string y = "bar";
1414     if (x != y) {
1415       print_s("not equal");
1416     }
1417     string x = "bar";
1418     if (x == y) {
1419       print_s("equal");
1420     }
1421     return 0;
1422 }
1423
1424 /*
1425 ****TEST****
1426 not equal
1427 equal
1428 */
1429
1430 ==> test_stringcmp.gty <==
1431 int main(){
1432     string a = "a";
1433     string b = "boy";
1434     int comp = stringcmp(a, b);
1435     print_i(comp);
1436     comp = stringcmp(b,a);
1437     print_i(comp);
1438     comp = stringcmp(b,b);
1439     print_i(comp);
1440     return 0;
1441 }
1442
1443 /*
1444 ****TEST****
1445 -1
1446 1
1447 0
1448 */
1449
```

```
==> test_string_length.gty <==
int main(){
    string a = "foo";
    string b = "bar";
    int x = string_length(a);
    int y = string_length(b);
    print_i(x);
    print_i(y);
    return 0;
}

/*
****TEST****
3
3
*/

==> test_string_slice.gty <==
// pass - test string slice

int main(){
    string s = "foobar";
    string sliced = slice(s, 2, 5);
    print_s(sliced);
}

/*
****TEST****
oba
*/

==> test_sub_int.gty <==
// pass - subtract integers

int main(){
    int x = 4;
    int y = 6;

    int z = x - y;
    print_i(z);
}

/*
****TEST****
-2
*/

==> test_while.gty <==
```

```
// pass - while

int main(){

    int x = 10;
    while (x < 12){

        print_i(x);
        x = x+1;
    }
return 0;
}

/*
****TEST****
10
11
*/
```

8.3 Applications

8.3.1 blackjack.gty

Listing 62: Blackjack

```
/* Author : Audrey
 * A game of Black Jack in the Gantry Language:
 * No betting.
 * Requires two players, or you can play against yourself.
 * No splitting
 * No dealer
 */

string find_id(string res, string key, int gap, int l){
	string value = "";
	int len = string_length(key);
	int res_len = string_length(res);
	string temp = "";
	for(int i = 0; i < res_len; i++){
		temp = slice(res, i, i+len);
		if (temp == key){
			value = slice(res, i+len+gap, i+len+gap+l);
		}
	}
	return value;
}



/* Get black jack value of card */
int match_card(string value){
	int val = 0;
	val = stoint(value);
	if (value == "A"){
		val = 1;
	}
	if (value == "J" || value == "Q" || value == "K" ) {
		val = 10;
	}
	return val;
}

/* Draw a card */
int hit(string draw_url){
	string new_card = httpget(draw_url);
	print_s("Here is the card :");
	print_s(new_card);
	int len = string_length(new_card);
```

```
44          int val = 0;
45          for (int i = 0; i<len; i++){
46                  int x = i+4;
47                  string temp = slice(new_card, i, x);
48                  if (temp == "code"){
49                          int index = x + 4;
50                          string value = slice(new_card, index, index + 1);
51                          string suite = slice(new_card, index + 1, index + 2);
52                          val = match_card(value);
53                          print_s("With a value of : ");
54                          print_i(val);
55                  }
56          }
57          return val;
58  }
59
60  /* Give player option to hit (1) or stay (other) */
61  int turn(string draw_url){
62          int new = 0;
63          string req = "Enter 1 to draw, anything else to stay ";
64          string res = get_string(req);
65          if (res == "1"){
66                  //while (new == 0){
67                          new = hit(draw_url);
68                  //}
69          }
70          else {
71                  new = 0;
72          }
73          return new;
74  }
75
76  /* Set aces based on new draw */
77  object set_aces(int new, object p){
78          int aces = p.aces;
79          /* If current card is an ace */
80          if (new == 1){
81                  p.aces = aces + 1;
82          }
83          return p;
84  }
85
86
87  /* Set hasten based on new draw */
88  object set_hasten(int new, object p){
89          if (new == 10){
90                  p.hasten = true;
91          }
```

```
92          return p;
93  }
94
95
96  /* Set done to true based on done conditions */
97  object set_done(object p){
98          bool blackjack = p.blackjack;
99          bool twentyone = p.twentyone;
100         bool bust = p.bust;
101         if (blackjack == true || twentyone == true || bust == true){
102                 p.done = true;
103         }
104         return p;
105 }
106
107 /* If player goes over 21, they bust */
108 object set_bust(object p){
109         int total = p.total_na;
110         if (total > 21){
111                 p.bust = true;
112         }
113         return p;
114 }
115
116 /* Check if total with ace is 21 */
117 object check_total_wa(object p){
118         int total_wa = p.total_wa;
119         int aces = p.aces;
120         bool hasten = p.hasten;
121
122         if (total_wa == 21){
123                 p.twentyone = true;
124                 if (aces > 0 && hasten == true){
125                         p.blackjack = true;
126                 }
127         }
128
129         return p;
130 }
131
132 /* Check total without aces */
133 object check_total_na(object p) {
134         int total_na = p.total_na;
135         if (total_na == 21){
136                 p.twentyone = true;
137         }
138         return p;
139 }
```

```
140
141 /* Compute score with ace as 11 */
142 object score_aces(object p){
143         int total = p.total_na;
144         int aces = p.aces;
145         int diff = 21 - total;
146
147         if (aces > 0 && diff > 9){
148                 p.total_wa = total + 10;
149         }
150         else {
151                 p.total_wa = total;
152         }
153
154         return p;
155 }
156
157 /* Ace and ten in hand is a blackjack */
158 object set_blackjack(object p){
159
160         return p;
161 }
162
163 object score(object p, int new){
164         bool blackjack = p.blackjack;
165         int total = p.total_na;
166         /* Total with aces gets recomputed each time */
167         p.total_wa = total;
168
169         p = set_aces(new, p);
170         p = set_hasten(new, p);
171
172         int aces = p.aces;
173         bool hasten = p.hasten;
174
175         p = set_blackjack(p);
176
177         /* Treat an ace as a 1 */
178         p.total_na = total + new;
179         total = p.total_na;
180         /* Score with 11 */
181         p = score_aces(p);
182
183         p = set_bust(p);
184         p = check_total_na(p);
185         p = check_total_wa(p);
186         p = set_done(p);
187         return p;
```

```
188 }
189
190 bool check_done(object p){
191         bool done = p.done;
192         return done;
193 }
194
195 int print_bust(object p){
196         bool bust = p.bust;
197         if (bust == true) {
198                 print_s("BUSTED!!!");
199         }
200         return 0;
201 }
202
203 int print_win(object p){
204         bool twentyone = p.twentyone;
205         bool blackjack = p.blackjack;
206         if (twentyone || blackjack) {
207                 print_s("WINNER!!!");
208         }
209         return 0;
210 }
211
212 int determine_diff(object p){
213         int diff = 0;
214         int total = p.total_wa;
215         diff = 21 - total;
216         return diff;
217 }
218
219 int determine_winner(object p1, object p2){
220         bool p1_done = p1.done;
221         bool p2_done = p2.done;
222         int p1_diff = 0;
223         int p2_diff = 0;
224         if (p1_done == false && p2_done == false){
225                 p1_diff = determine_diff(p1);
226                 p2_diff = determine_diff(p2);
227                 if (p1_diff < p2_diff) {
228                         print_s("Player 1 Won");
229                 }
230                 else {
231                         print_s("Player 2 Won");
232                 }
233         }
234         return 0;
235 }
```

```
236
237
238 int main(){
239         object p1 = {|
240                         string name : "Player 1",
241                         bool blackjack : false,
242                         bool twentyone : false,
243                         bool bust : false,
244                         bool done : false,
245                         bool hasten : false,
246                         int total_na : 0,
247                         int total_wa : 0,
248                         int aces : 0
249                     |};
250         object p2 = {|
251                         string name : "Player 2",
252                         bool blackjack : false,
253                         bool twentyone : false,
254                         bool bust : false,
255                         bool done : false,
256                         bool hasten : false,
257                         int total_na : 0,
258                         int total_wa : 0,
259                         int aces : 0
260                     |};
261         string url = "https://deckofcardsapi.com/api/deck/";
262         string new_deck = httpget(url ^ "new/shuffle/");
263         string deck_id = find_id(new_deck,"deck_id", 4, 12);
264         int len = string_length(new_deck);
265         string draw_url = url ^ deck_id ^ "/draw/?count=1";
266         bool p1_done = p1.done;
267         bool p2_done = p2.done;
268         int new = 0;
269         int stay_p1 = 0;
270         int stay_p2 = 0;
271         bool both_stay = false;
272         while (p1_done == false && p2_done == false && both_stay == false){
273                 print_s("===Player 1===");
274                 new = turn(draw_url);
275                 if (new == 0){
276                         stay_p1++;
277                 }
278                 p1 = score(p1, new);
279
280                 print_s("===Player 2===");
281                 new = turn(draw_url);
282                 if (new == 0){
283                         stay_p2++;
```

```
                }
                p2 = score(p2, new);

                p1_done = check_done(p1);
                p2_done = check_done(p2);
                if (stay_p1 >0 && stay_p2 > 0){
                        both_stay = true;
                }
        }
        print_s("+++++ GAME OVER +++++");
        string p1_res = obj_stringify(p1);
        string p2_res = obj_stringify(p2);
        print_s("===Player 1===");
        print_win(p1);
        print_bust(p1);
        print_s(p1_res);
        print_s("===Player 2===");
        print_win(p2);
        print_bust(p2);
        print_s(p2_res);
        determine_winner(p1, p2);
        return 0;
}
```

8.3.2 docker.gty

Listing 63: Docker

```
/*
 *
 * Docker Orchestration Application
 *
 * The following demostrates some typical use cases
 * for the Gantry language with respect to interacting
 * with a JSON API using Gantry data structures and
 * built-in functions.
 *
 * Utilized Docker API Reference:
 * https://docs.docker.com/engine/api/v1.32/
 *
 * Author: Walter Meyer
 */

string docker_host;

// Get one line from logs
int watch_logs(int polls, int wait, string cid, string tail) {
  // Poll stdout of running container
  int i = 0;
  while (i < polls) {
    string out = httpget(docker_host ^ "/containers/" ^ cid ^ "/logs?stdout=1&tail
        =" ^ tail);
    print_s(out);
    nap(wait);
    i++;
  }
}

int main() {
  // The Docker API Host
  docker_host = "http://localhost";

  // Our container image and the process/cmd to launch
  object exec =
    {|
        string Image : "ubuntu:14.04",
        bool Tty : true,
        string array Cmd :
        [ "bash", "-c", "bash -c \\\"while true; do echo Hello World && uname -a
            && date; sleep 3; done\\\"" ]
    |};

```

```
  // Download Image
  string image = exec.Image;
  print_s("\n *** Pulling Container Image: [" ^ image ^ "] ***\n");
  print_s("Sending JSON:\n" ^ obj_stringify(exec) ^ "\n");
  httppost(docker_host ^ "/images/create?fromImage=" ^ image, "");

  // Create and Start Container from Image
  string exec_json = obj_stringify(exec);
  string container_id = httppost(docker_host ^ "/containers/create", exec_json);
  print_s("JSON Response:");
  print_s(container_id);

  // Grab sliced CID
  string cid = slice(container_id, 7, 17);

  // Start the container
  print_s("\n *** Starting Container: [" ^ cid ^ "] ***\n");
  httppost(docker_host ^ "/containers/" ^ cid ^ "/start", "");
  print_s("\n *** Container Logs: [" ^ cid ^ "] ***\n");
  watch_logs(5, 3, cid, "3");

}
```

8.3.3 docker-run.gty

Listing 64: Docker-run

```
/*
 * Docker Orchestration Application 2
 *
 * Contains a function to run a container and
 * handle port binding/exposure.
 *
 * The following demostrates some typical use cases
 * for the Gantry language with respect to interacting
 * with a JSON API using Gantry data structures and
 * built-in functions.
 *
 * Utilized Docker API Reference:
 * https://docs.docker.com/engine/api/v1.32/
 *
 * Author: Walter Meyer
 */

string docker_host;

// Get one line from logs
int watch_logs(int polls, int wait, string cid, string tail) {
  // Poll stdout of running container
  int i = 0;
  while (i < polls) {
    string out = httpget(docker_host ^ "/containers/" ^ cid ^ "/logs?stdout=1&tail
        =" ^ tail);
    print_s(out);
    nap(wait);
    i++;
  }
}

string format_str(string exec_json) {
  exec_json = slice(exec_json, 0, 41) ^ "/" ^ slice(exec_json, 42, string_length(
      exec_json));
  exec_json = slice(exec_json, 0, 51) ^ slice(exec_json, 52, string_length(
      exec_json));
  exec_json = slice(exec_json, 0, 74) ^ slice(exec_json, 76, string_length(
      exec_json));
  return exec_json;
}

// Create and start an arbitrary container with port bindings
int start(string image, string port) {
```

```
41    // Pull image
42    print_s("\n *** Pulling Container Image: [" ^ image ^ "] ***\n");
43    httppost(docker_host ^ "/images/create?fromImage=" ^ image, "");
44
45    // Array of Port Bindings
46    object port_bind = {| string HostPort : port |};
47    string port = obj_stringify(port_bind);
48    print_s(port);
49
50    // Start Container
51    object exec =
52      {|
53          string Image : image,
54          bool Tty : true,
55          object HostConfig :
56            {|
57               object PortBindings : {| string array 80_tcp : [ port ] |}
58            |}
59      |};
60
61    string exec_json = obj_stringify(exec);
62    exec_json = format_str(exec_json);
63    print_s(exec_json);
64    print_s("\n *** Creating Container: [" ^ image ^ "] ***\n");
65    string container_id = httppost(docker_host ^ "/containers/create", exec_json);
66    print_s(container_id);
67    string cid = slice(container_id, 7, 17);
68
69    // Create and Start Apache Container
70    print_s("\n *** Starting Container: [" ^ cid ^ "] ***");
71    container_id = httppost(docker_host ^ "/containers/" ^ cid ^ "/start", "");
72    print_s(container_id);
73
74    // Watch container logs
75    print_s("\n *** Tailing Last Line of Container Logs: [" ^ cid ^ "] ***");
76    watch_logs(100, 3, cid, "1");
77  }
78
79  int main() {
80    // The Docker API Host
81    docker_host = "http://localhost";
82
83    // Spawn Apache Container
84    start("httpd:alpine", "8080");
85
86    return 0;
87  }
```

8.3.4 test_string_input.gty

Listing 65: Test_String_Input

```
1 int main(){
2         string y = "Please enter a string : ";
3         string s = get_string(y);
4         return 0;
5 }
```