Reptile

Aileen Cano	Aviva Weinbaum	Lindsey Weiskopf	Hariti Patel
acc4440	aw3156	ltw2115	hvp2105
Test Designer	Language Guru	System Architect	Manager

Spring 2021

Contents

9.11.2.20 test-fib.rt	101
9.11.2.21 test-float.out	101
9.11.2.22 test-float.rt	101
9.11.2.23 test-gcd.out	102
9.11.2.24 test-gcd.rt	102
9.11.2.25 test-helloworld.out	102
9.11.2.26 test-helloworld.rt	102
9.11.2.27 test-if.out	102
9.11.2.28 test-if.rt	102
9.11.2.29 test-if2.out	103
9.11.2.30 test-if2.rt	103
9.11.2.31 test-if3.out	103
9.11.2.32 test-if3.rt	103
9.11.2.33 test-if4.out	103
9.11.2.34 test-if4.rt	103
9.11.2.35 test-if5.out	104
9.11.2.36 test-if5.rt	104
9.11.2.37 test-if6.out	104
9.11.2.38 test-if6.rt	104
9.11.2.39 test-not.out	104
9.11.2.40 test-not.rt	104
9.11.2.41 test-or.out	105
9.11.2.42 test-or.rt	105
9.11.2.43 test-pixel.out	105
9.11.2.44 test-pixel.rt	105

Chapter 1 - Introduction

Reptile is a programming language that is intended to support libraries that streamline the process of creating simply-coded graphics. As more children are learning computer science at a younger age, there is a demand for simple programming languages that teach computer science principles in a digestible and visual manner. Languages typically labeled for beginners like Scratch and Swift Playgrounds teach kids to code by showing immediate visual results from code – whether that is a simple square or a complex environment built upon existing code blocks. Further, libraries like Turtle graphics add novelty to simple image-building operations by showing a turtle drawing the desired shape. The goal of Reptile is to build upon the success of these “beginner” programming languages to provide immediate gratification to the coders through graphics.

Chapter 2 - Language Tutorial

2.1 Environment Setup

Begin by downloading the code for the compiler and the tests from GitHub:

https://github.com/avivaweinbaum/reptile

Then, install LLVM 6.0.0. Next, install OCaml and OPAM. OCaml version 4.05.0 is required.

Alternatively, install Docker 20.10.5. Then, open the docker container using the following command:

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

2.2 Writing "Hello World"

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

The above code is a fully compatible program:

- int main() - defines the main() function which is run automatically in any Reptile file
- print(5) - takes in an int parameter, in this case it is five, and outputs 5
- return 0 - represents the return statement, in this case it is zero, and informs the main function to end

2.3 Compiling and Running test-helloworld.rt

Follow the following steps:

- In the *reptile* folder, run **make** to compile and link to the codebase
- To compile and run test-helloworld.rt in order to output to a file, run:
    - **./reptile.native < tests/test-helloworld.rt > out.s**
- To output the result of out.s, run:
    - **lli out.s**

2.4 Compiling and Running All Tests

In the *reptile* folder, run **make** and all tests should pass. Alternatively, you can run **make** and then **./testall.sh** which will also result in all the tests passing.

Chapter 3 - Language Reference Manual

3.1 Lexical Conventions

3.1.1 Comments

The characters /\ introduce a comment for a single line. For multi-line comments, each line must begin with /\.

3.1.2 Identifiers (Names)

An identifier is a string of characters, both letters and digits, that can be used to name a variable, object, or function. The first character of the identifier must be alphabetic. Upper and lower case characters may be used along with underscores and digits to create the identifier.

3.1.3 Keywords

The following identifiers are reserved to be used as keywords, and cannot be used otherwise. The keywords are case sensitive.

Keywords	**Description**
if	Enters block if condition statement is satisfied
else	Enters block when if condition statement is not satisfied
while	Continues to repeat up until specified condition is not satisfied
true	Boolean literal for 1
false	Boolean literal for 0
return	Returns a value from a function
void	Used for functions when nothing is to be returned

3.1.4 Type Specifiers

The following are type specifiers with their descriptions and functions, if applicable. These type specifiers are case sensitive.

Type Specifiers	**Description**
int	Any signed integer

float	Any floating point decimal
boolean	Boolean value of “true” or “false”
string	Standard string
RGB	List of 3 color values ranging from 0 to 255
Canvas	Two-dimensional array of pixels which Pointers act on
Pointer	Pen used to draw pixels and move around on Canvas

3.1.5 Punctuators

A punctuator is a symbol which does not specify any specific operation to be executed, but instead, it has syntactic value to the compiler to format and parse the code.

Symbols	**Description**
;	Statement terminator
,	Separation of variables
{ }	Block of statements
( )	Condition, function declaration, and parameter specifier

3.1.6 Operators

3.1.6.1 Arithmetic/Logical Operators

Arithmetic/Logical Operators	**Description**
=	Assignment operator
+	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
==	Returns true if values are equal, false otherwise
!=	Returns true if values are not equal, false otherwise
<	Less than operator

>	Greater than operator
<=	Less than or equal to operator
>=	Greater than or equal to operator
&&	Logical AND operator
||	Logical OR operator
!	Logical NOT operator

3.1.6.2 Operator Precedence

The following operators are presented in the order of their precedence from highest to lowest.

Operator Symbols	**Description**
!	Logical NOT operator
* /	Multiplication operator, division operator
+ -	Addition operator, subtraction operator
< > <= >=	Less than operator, greater than operator, less than or equal to operator, greater than or equal to operator
== !=	Equality operator, inequality operator
&& ||	Logical AND operator, logical OR operator
=	Assignment operator

3.2 Syntax

3.2.1 Variable Declaration

Variables are defined using an identifier and an expression. Variable identifiers are strings for the name of the variable which are used to later manipulate the variable. Identifiers are any uppercase or lowercase character followed by any number of alphanumeric and underscore characters. If the variable identifier has not already been declared, it must have its type name before.

```
int a = 3;
```

3.2.2 Statements

3.2.2.1 Termination

Statements are sequenced and terminated using the semicolon. Each line must end with a semicolon in order for the next line to be taken as a separate statement, with the exception of control flow statements and loops, which utilize curly braces.

3.2.2.2 Control Flow

Conditional statements are if and else. Control flow is initialized using the “if” keyword, a boolean expression in parenthesis, and a code block enclosed in curly braces.

```
int foo = 3;
if (foo < 10) {
      /\ block of code
} else {
      /\ block of code
}
```

3.2.2.3 Loops

Reptile supports while loops.

3.2.2.3.1 While Loops

While loops consist of the “while” keyword followed by a boolean expression in parenthesis that is evaluated each time the loop is executed. When the expression is false, the loop will stop executing. An example of the syntax is as follows:

```
int foo = 3; int bar = 4;
while (foo < 10) {
      /\ block of code
}
```

3.2.3 Functions

3.2.3.1 Function Declarations

Functions are declared using the “function” keyword, a return type, an identifier, and any number of arguments enclosed in parenthesis. The function body contains a block of code,

and must return a value of the type specified using the keyword “return”. An example of the syntax is as follows:

```
int foo(int a, int b) {
    /\ block of code
    return 3;
}
```

3.2.3.2 Function Calls

Functions are called with the function identifier and the number of arguments required. Because a value is returned, the outcome of a function call can be saved in a variable. An example of the syntax is as follows:

```
int bar = foo(5,10);
```

3.3 Built-in Types and Functions

3.3.1 Canvas Manipulation

3.3.1.1 Canvas Instantiation

The canvas is the drawing slate for all external libraries and pixel manipulation in Reptile. Canvas are initialized with the pixel parameters as the width and height of the canvas. The dimensions of the canvas are immutable after instantiation.

```
Canvas canvas = Canvas(80, 100);
```

3.3.1.2 Canvas Instance Variables

Name	**Type**	**Meaning**
x	int	Length of canvas
y	int	Width of canvas

3.3.2 Pointer Manipulation

The pointer is critical to the instantiation of any drawing object created from Reptile. The pointer can “flip” or change the color of one pixel at a time.

3.3.2.1 Pointer Instantiation

The pointer can be instantiated with the following code, and the float parameter represents the angle that the pointer is facing starting from the top left corner of the canvas:

```
Pointer p = Pointer(0, 0, Rgb(255,0,0), 90.0);
```

3.3.2.2 Pointer Instance Variables

The pointer object carries the following instance variables.

Name	**Type**	**Meaning**
x	int	number of pixels from left of canvas
y	int	number of pixels from top of canvas
color	Rgb	color of pixel pointer as an RGB value
angle	float	angle from first quadrant in degrees

3.3.3 Rgb Manipulation

The Rgb type stores the red, blue, and green color values which is useful for creating colors for drawing.

3.3.3.1 Rgb Instantiation

The Rgb object can be instantiated with 3 ints:

```
Rgb rgb = Rgb(255, 255, 255);
```

3.3.3.2 Rgb Instance Variables

The Rgb object carries the following instance variables.

Name	**Type**	**Meaning**
r	int	Red value
g	int	Green value
b	int	Blue value

3.3.4 Functions

The following functions are built-in to Reptile:

Function	**Return Type**	**Purpose**
pixel(Canvas can, Rgb color, int x, int y)	Canvas	Colors the pixel at (x,y)
save(Canvas can, string filename)	void	Saves the canvas to a PNG at filename
get_rgb_r(Rgb rgb)	int	Returns the r value of the Rgb object
get_rgb_g(Rgb rgb)	int	Returns the g value of the Rgb object
get_rgb_b(Rgb rgb)	int	Returns the b value of the Rgb object
get_pointer_x(Pointer pointer)	int	Returns the x coordinate of the Pointer
get_pointer_y(Pointer pointer)	int	Returns the y coordinate of the Pointer
set_pointer_color(Pointer pointer, Rgb rgb)	Pointer	Updates the color of the Pointer
get_canvas_x(Canvas canvas)	int	Returns the length of the Canvas
get_canvas_y(Canvas canvas)	int	Returns the width of the Canvas
sine(float angle)	float	Returns the sine of the angle
cosine(float angle)	float	Returns the cosine of the angle
tangent(float angle)	float	Returns the tangent of the angle
mod(int val1, int val2)	int	Returns the product of val1%val2
floors(float val)	int	Returns the floor of val (val cast to an int)
getRise(int distance, float angle)	Int	Returns the y-difference of a line *distance* long at an *angle* angle
getRun(int distance, float angle)	Int	Returns the x-difference of a line *distance* long at an *angle* angle

3.4 Tortoise Library

With our built in functions and types, we defined the Tortoise library which provides functions and global variables that make the manipulation of a png more user friendly. The library implementation can be found in Section 3.6, simple-tort.rt. Currently, the Library must be included in all source files where the library functions are used.

The library keeps the global variables `xcur` and `ycur` to track the pointers current position.

The following functions are defined in the library:

Function	Return Type	Purpose
tortup(Canvas can, Rgb color, int distance)	int	Draws pixels for the specified direction of the length distance
tortdown(Canvas can, Rgb color, int distance)	int	Draws pixels for the specified direction of the length distance
tortleft(Canvas can, Rgb color, int distance)	int	Draws pixels for the specified direction of the length distance
tortright(Canvas can, Rgb color, int distance)	int	Draws pixels for the specified direction of the length distance
tortNE(Canvas can, Rgb color, float distance)	int	Draws pixels for the specified diagonal of the length distance
tortNW(Canvas can, Rgb color, float distance)	int	Draws pixels for the specified diagonal of the length distance
tortSE(Canvas can, Rgb color, float distance)	int	Draws pixels for the specified diagonal of the length distance
tortSW(Canvas can, Rgb color, float distance)	int	Draws pixels for the specified diagonal of the length distance
movetort(int x , int y)	int	Moves the global xcur and ycur values to the specified location

Since this library is implemented in Reptile directly, it can be changed easily by the programmer to fit their needs.

3.5 Semantics

3.5.1 Scope

3.5.1.1 Blocks

In Reptile, a block is defined by any segment of code defined within a set of curly braces. A block could be a class, a function, or segment of a control sequence. Curly braces can also be arbitrarily placed to define a block within a program.

3.5.1.2 Blocks and Scope

The scope of a variable is always available to and limited by its innermost curly braces. The only exception to this rule is instance variables defined in the standard library or belonging to objects created by the user.

3.5.2 Recursion

Recursion is extremely useful in graphics, especially when there are repeated patterns being drawn. Recursion can be used by calling a function within the function itself. See the gcd algorithm for an example.

3.6 Sample Code

gcd.rt

```
int gcd(int a, int b) {
    if (b != 0) {
        return gcd(b, mod(a, b));
    }
    else {
        return a;
    }
}
```

gcd.ll

```
; ModuleID = 'Reptile'
source_filename = "Reptile"

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%d\0A\00"

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

declare { i32, i32, { i32, i32, i32 }*, double }* @Pointer(i32, i32, { i32,
i32, i32 }*, double)

declare { i32, i32 }* @Canvas(i32, i32)

declare i32 @save({ i32, i32 }*, i8*)

declare { i32, i32 }* @pixel({ i32, i32 }*, { i32, i32, i32 }*, i32, i32)

declare i32 @get_rgb_r({ i32, i32, i32 }*)
```

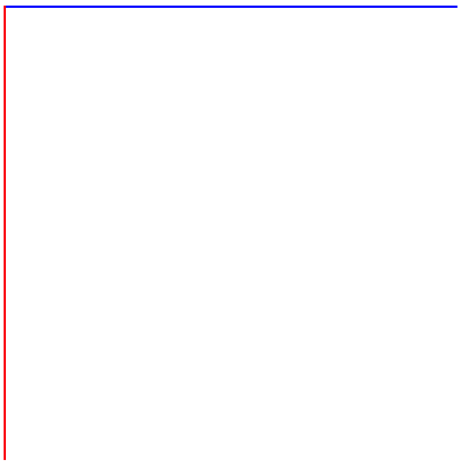
```
declare i32 @get_rgb_g({ i32, i32, i32 }*)

declare i32 @get_rgb_b({ i32, i32, i32 }*)

declare i32 @get_pointer_x({ i32, i32, { i32, i32, i32 }*, double }*)

declare i32 @get_pointer_y({ i32, i32, { i32, i32, i32 }*, double }*)

declare { i32, i32, { i32, i32, i32 }*, double }* @set_pointer_color({ i32,
i32, { i32, i32, i32 }*, double }*, { i32, i32, i32 }*)

declare i32 @get_canvas_x({ i32, i32 }*)

declare i32 @get_canvas_y({ i32, i32 }*)

declare double @sine(double)

declare double @cosine(double)

declare double @tangent(double)

declare i32 @mod(i32, i32)

declare i32 @floors(double)

declare i32 @getRun(i32, double)

declare i32 @getRise(i32, double)

define i32 @main() {
entry:
  %val = alloca i32
  %gcd_ret = call i32 @gcd(i32 60, i32 48)
  store i32 %gcd_ret, i32* %val
  %val1 = load i32, i32* %val
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x
i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %val1)
  ret i32 0
}

define i32 @gcd(i32 %a, i32 %b) {
entry:
  %a1 = alloca i32
```

```
  store i32 %a, i32* %a1
  %b2 = alloca i32
  store i32 %b, i32* %b2
  %b3 = load i32, i32* %b2
  %tmp = icmp ne i32 %b3, 0
  br i1 %tmp, label %then, label %else

merge:                                            ; No predecessors!
  ret i32 0

then:                                             ; preds = %entry
  %a4 = load i32, i32* %a1
  %b5 = load i32, i32* %b2
  %mod = call i32 @mod(i32 %a4, i32 %b5)
  %b6 = load i32, i32* %b2
  %gcd_ret = call i32 @gcd(i32 %b6, i32 %mod)
  ret i32 %gcd_ret

else:                                             ; preds = %entry
  %a7 = load i32, i32* %a1
  ret i32 %a7
}
```

simple-line.rt

```
int main() {
   /\ Make a canvas
   Canvas can = Canvas(400, 400);
   /\ Define colors
   Rgb color1 = Rgb(0,0,255);
   Rgb color2 = Rgb(255,0,0);
   int counter = 50;
   while (counter < 250) {
       pixel(can, color1, counter, 50);
       pixel(can, color2, 50, counter);
       counter = counter + 1;
   }
   /\ Save canvas to png
   save(can, "simple_line.png");
   return 0;
}
```

Output



simple-line.ll

```
; ModuleID = 'Reptile'
source_filename = "Reptile"

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@str = private unnamed_addr constant [16 x i8] c"simple_line.png\00"

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

declare { i32, i32, { i32, i32, i32 }*, double }* @Pointer(i32, i32, { i32,
i32, i32 }*, double)

declare { i32, i32 }* @Canvas(i32, i32)

declare i32 @save({ i32, i32 }*, i8*)

declare { i32, i32 }* @pixel({ i32, i32 }*, { i32, i32, i32 }*, i32, i32)

declare i32 @get_rgb_r({ i32, i32, i32 }*)
```

```
declare i32 @get_rgb_g({ i32, i32, i32 }*)

declare i32 @get_rgb_b({ i32, i32, i32 }*)

declare i32 @get_pointer_x({ i32, i32, { i32, i32, i32 }*, double }*)

declare i32 @get_pointer_y({ i32, i32, { i32, i32, i32 }*, double }*)

declare { i32, i32, { i32, i32, i32 }*, double }* @set_pointer_color({ i32,
i32, { i32, i32, i32 }*, double }*, { i32, i32, i32 }*)

declare i32 @get_canvas_x({ i32, i32 }*)

declare i32 @get_canvas_y({ i32, i32 }*)

declare double @sine(double)

declare double @cosine(double)

declare double @tangent(double)

declare i32 @mod(i32, i32)

declare i32 @floors(double)

declare i32 @getRun(i32, double)

declare i32 @getRise(i32, double)

define i32 @main() {
entry:
  %can = alloca { i32, i32 }*
  %Canvas = call { i32, i32 }* @Canvas(i32 400, i32 400)
  store { i32, i32 }* %Canvas, { i32, i32 }** %can
  %color1 = alloca { i32, i32, i32 }*
  %Rgb = call { i32, i32, i32 }* @Rgb(i32 0, i32 0, i32 255)
  store { i32, i32, i32 }* %Rgb, { i32, i32, i32 }** %color1
  %color2 = alloca { i32, i32, i32 }*
  %Rgb1 = call { i32, i32, i32 }* @Rgb(i32 255, i32 0, i32 0)
  store { i32, i32, i32 }* %Rgb1, { i32, i32, i32 }** %color2
  %counter = alloca i32
  store i32 50, i32* %counter
```

```
  br label %while

while:                                            ; preds = %while_body,
%entry
  %counter10 = load i32, i32* %counter
  %tmp11 = icmp slt i32 %counter10, 250
  br i1 %tmp11, label %while_body, label %merge

while_body:                                       ; preds = %while
  %can2 = load { i32, i32 }*, { i32, i32 }** %can
  %color13 = load { i32, i32, i32 }*, { i32, i32, i32 }** %color1
  %counter4 = load i32, i32* %counter
  %pixel = call { i32, i32 }* @pixel({ i32, i32 }* %can2, { i32, i32, i32
}* %color13, i32 %counter4, i32 50)
  %can5 = load { i32, i32 }*, { i32, i32 }** %can
  %color26 = load { i32, i32, i32 }*, { i32, i32, i32 }** %color2
  %counter7 = load i32, i32* %counter
  %pixel8 = call { i32, i32 }* @pixel({ i32, i32 }* %can5, { i32, i32, i32
}* %color26, i32 50, i32 %counter7)
  %counter9 = load i32, i32* %counter
  %tmp = add i32 %counter9, 1
  store i32 %tmp, i32* %counter
  br label %while

merge:                                            ; preds = %while
  %can12 = load { i32, i32 }*, { i32, i32 }** %can
  %save = call i32 @save({ i32, i32 }* %can12, i8* getelementptr inbounds
([16 x i8], [16 x i8]* @str, i32 0, i32 0))
  ret i32 0
}

declare { i32, i32, i32 }* @Rgb(i32, i32, i32)
```

simple-tort.rt

```
int xcur;
int ycur;


int tortup(Canvas can, Rgb color, int distance) {
   int counter = 0;
   while(counter < distance) {
       pixel(can, color, xcur, ycur-counter);
       counter = counter + 1;
```

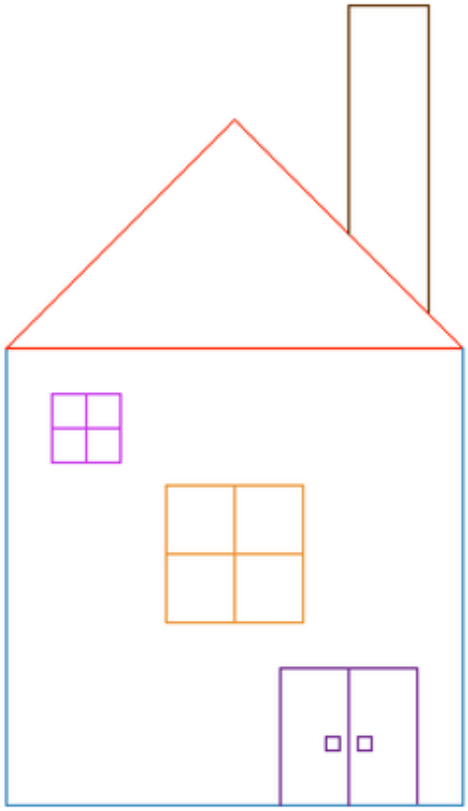
```
    }
    ycur = ycur - distance;
    return 0;
}

int tortdown(Canvas can, Rgb color, int distance) {
    int counter = 0;
    while(counter < distance) {
        pixel(can, color, xcur, ycur+counter);
        counter = counter + 1;
    }
    ycur = ycur + distance;
    return 0;
}

int tortright(Canvas can, Rgb color, int distance) {
    int counter = 0;
    while(counter < distance) {
        pixel(can, color, xcur+counter, ycur);
        counter = counter + 1;
    }
    xcur = xcur + distance;
    return 0;
}

int tortleft(Canvas can, Rgb color, int distance) {
    int counter = 0;
    while(counter < distance) {
        pixel(can, color, xcur-counter, ycur);
        counter = counter + 1;
    }
    xcur = xcur - distance;
    return 0;
}

int tortNW(Canvas can, Rgb color, float distance) {
    int counter = 0;
    float counter1 = 0.0;
    float step = distance * 0.707;
    while(counter1 < step) {
        pixel(can, color, xcur-counter, ycur-counter);
        counter1 = counter1 + 1.0;
        counter = counter + 1;
    }
    xcur = xcur - counter;
    ycur = ycur - counter;
    return 0;
}

int tortNE(Canvas can, Rgb color, float distance) {
    int counter = 0;
    float counter1 = 0.0;
```

```
    float step = distance * 0.707;
    while(counter1 < step) {
        pixel(can, color, xcur+counter, ycur-counter);
        counter1 = counter1 + 1.0;
        counter = counter + 1;
    }
    xcur = xcur + counter;
    ycur = ycur - counter;
    return 0;
}

int tortSW(Canvas can, Rgb color, float distance) {
    int counter = 0;
    float counter1 = 0.0;
    float step = distance * 0.707;
    while(counter1 < step) {
        pixel(can, color, xcur-counter, ycur+counter);
        counter1 = counter1 + 1.0;
        counter = counter + 1;
    }
    xcur = xcur - counter;
    ycur = ycur + counter;
    return 0;
}

int tortSE(Canvas can, Rgb color, float distance) {
    int counter = 0;
    float counter1 = 0.0;
    float step = distance * 0.707;
    while(counter1 < step) {
        pixel(can, color, xcur+counter, ycur+counter);
        counter1 = counter1 + 1.0;
        counter = counter + 1;
    }
    xcur = xcur + counter;
    ycur = ycur + counter;
    return 0;
}

int movetort(int x, int y) {
    xcur = x;
    ycur = y;
    return 0;
}

int main() {
    Canvas can = Canvas(600, 600);
    Rgb color = Rgb(56,137,190);

    Rgb roofcolor = Rgb(255,0,0);
    movetort(200, 200);
    tortright(can, roofcolor,200);
```

```
tortdown(can, color, 200);
tortleft(can, color, 200);
tortup(can, color, 200);

movetort(200, 200);
tortNE(can, roofcolor, 141.0);
tortSE(can, roofcolor, 141.0);

Rgb window = Rgb(201, 52, 235);
movetort(220,220);
tortright(can, window, 30);
tortdown(can, window, 30);
tortleft(can, window, 30);
tortup(can, window, 30);
movetort(235,220);
tortdown(can, window, 30);
movetort(220, 235);
tortright(can, window, 30);

Rgb win = Rgb(235, 149, 52);
movetort(270, 260);
tortright(can, win, 60);
tortdown(can, win, 60);
tortleft(can, win, 60);
tortup(can, win, 60);
movetort(300,260);
tortdown(can, win, 60);
movetort(270, 290);
tortright(can, win, 60);

Rgb chimney = Rgb(97,51,1);
movetort(350, 150);
tortup(can, chimney, 100);
tortright(can, chimney, 35);
tortdown(can, chimney, 135);

Rgb door = Rgb(114, 40, 143);
movetort(320, 400);
tortup(can, door, 60);
tortright(can, door, 60);
tortdown(can, door, 60);
movetort(350, 400);
tortup(can, door, 60);

movetort(340, 370);
tortright(can, door, 6);
tortdown(can, door, 6);
tortleft(can, door, 6);
tortup(can, door, 6);

movetort(354, 370);
tortright(can, door, 6);
```

```
    tortdown(can, door, 6);
    tortleft(can, door, 6);
    tortup(can, door, 6);

    save(can, "simple_turtle_start.png");
    return 0;
}
```

Output:



simple-tort.ll (snippet including main method, global declarations, movetort(), and torrtSE()

```
; ModuleID = 'Reptile'
source_filename = "Reptile"

@ycur = global i32 0
@xcur = global i32 0
@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@str = private unnamed_addr constant [24 x i8]
```

```
c"simple_turtle_start.png\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.3 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.4 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.5 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.6 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.7 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.8 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.9 = private unnamed_addr constant [4 x i8] c"%d\0A\00"

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

declare { i32, i32, { i32, i32, i32 }*, double }* @Pointer(i32, i32, { i32,
i32, i32 }*, double)

declare { i32, i32 }* @Canvas(i32, i32)

declare i32 @save({ i32, i32 }*, i8*)

declare { i32, i32 }* @pixel({ i32, i32 }*, { i32, i32, i32 }*, i32, i32)

declare i32 @get_rgb_r({ i32, i32, i32 }*)

declare i32 @get_rgb_g({ i32, i32, i32 }*)

declare i32 @get_rgb_b({ i32, i32, i32 }*)

declare i32 @get_pointer_x({ i32, i32, { i32, i32, i32 }*, double }*)

declare i32 @get_pointer_y({ i32, i32, { i32, i32, i32 }*, double }*)

declare { i32, i32, { i32, i32, i32 }*, double }* @set_pointer_color({ i32,
i32, { i32, i32, i32 }*, double }*, { i32, i32, i32 }*)

declare i32 @get_canvas_x({ i32, i32 }*)

declare i32 @get_canvas_y({ i32, i32 }*)

declare double @sine(double)

declare double @cosine(double)
```

```
declare double @tangent(double)

declare i32 @mod(i32, i32)

declare i32 @floors(double)

declare i32 @getRun(i32, double)

declare i32 @getRise(i32, double)

define i32 @main() {
entry:
  %can = alloca { i32, i32 }*
  %Canvas = call { i32, i32 }* @Canvas(i32 600, i32 600)
  store { i32, i32 }* %Canvas, { i32, i32 }** %can
  %color = alloca { i32, i32, i32 }*
  %Rgb = call { i32, i32, i32 }* @Rgb(i32 56, i32 137, i32 190)
  store { i32, i32, i32 }* %Rgb, { i32, i32, i32 }** %color
  %roofcolor = alloca { i32, i32, i32 }*
  %Rgb1 = call { i32, i32, i32 }* @Rgb(i32 255, i32 0, i32 0)
  store { i32, i32, i32 }* %Rgb1, { i32, i32, i32 }** %roofcolor
  %movetort_ret = call i32 @movetort(i32 200, i32 200)
  %roofcolor2 = load { i32, i32, i32 }*, { i32, i32, i32 }** %roofcolor
  %can3 = load { i32, i32 }*, { i32, i32 }** %can
  %tortright_ret = call i32 @tortright({ i32, i32 }* %can3, { i32, i32, i32
}* %roofcolor2, i32 200)
  %color4 = load { i32, i32, i32 }*, { i32, i32, i32 }** %color
  %can5 = load { i32, i32 }*, { i32, i32 }** %can
  %tortdown_ret = call i32 @tortdown({ i32, i32 }* %can5, { i32, i32, i32
}* %color4, i32 200)
  %color6 = load { i32, i32, i32 }*, { i32, i32, i32 }** %color
  %can7 = load { i32, i32 }*, { i32, i32 }** %can
  %tortleft_ret = call i32 @tortleft({ i32, i32 }* %can7, { i32, i32, i32
}* %color6, i32 200)
  %color8 = load { i32, i32, i32 }*, { i32, i32, i32 }** %color
  %can9 = load { i32, i32 }*, { i32, i32 }** %can
  %tortup_ret = call i32 @tortup({ i32, i32 }* %can9, { i32, i32, i32 }*
%color8, i32 200)
  %movetort_ret10 = call i32 @movetort(i32 200, i32 200)
  %roofcolor11 = load { i32, i32, i32 }*, { i32, i32, i32 }** %roofcolor
  %can12 = load { i32, i32 }*, { i32, i32 }** %can
  %tortNE_ret = call i32 @tortNE({ i32, i32 }* %can12, { i32, i32, i32 }*
%roofcolor11, double 1.410000e+02)
```

```
  %roofcolor13 = load { i32, i32, i32 }*, { i32, i32, i32 }** %roofcolor
  %can14 = load { i32, i32 }*, { i32, i32 }** %can
  %tortSE_ret = call i32 @tortSE({ i32, i32 }* %can14, { i32, i32, i32 }*
%roofcolor13, double 1.410000e+02)
  %window = alloca { i32, i32, i32 }*
  %Rgb15 = call { i32, i32, i32 }* @Rgb(i32 201, i32 52, i32 235)
  store { i32, i32, i32 }* %Rgb15, { i32, i32, i32 }** %window
  %movetort_ret16 = call i32 @movetort(i32 220, i32 220)
  %window17 = load { i32, i32, i32 }*, { i32, i32, i32 }** %window
  %can18 = load { i32, i32 }*, { i32, i32 }** %can
  %tortright_ret19 = call i32 @tortright({ i32, i32 }* %can18, { i32, i32,
i32 }* %window17, i32 30)
  %window20 = load { i32, i32, i32 }*, { i32, i32, i32 }** %window
  %can21 = load { i32, i32 }*, { i32, i32 }** %can
  %tortdown_ret22 = call i32 @tortdown({ i32, i32 }* %can21, { i32, i32,
i32 }* %window20, i32 30)
  %window23 = load { i32, i32, i32 }*, { i32, i32, i32 }** %window
  %can24 = load { i32, i32 }*, { i32, i32 }** %can
  %tortleft_ret25 = call i32 @tortleft({ i32, i32 }* %can24, { i32, i32,
i32 }* %window23, i32 30)
  %window26 = load { i32, i32, i32 }*, { i32, i32, i32 }** %window
  %can27 = load { i32, i32 }*, { i32, i32 }** %can
  %tortup_ret28 = call i32 @tortup({ i32, i32 }* %can27, { i32, i32, i32 }*
%window26, i32 30)
  %movetort_ret29 = call i32 @movetort(i32 235, i32 220)
  %window30 = load { i32, i32, i32 }*, { i32, i32, i32 }** %window
  %can31 = load { i32, i32 }*, { i32, i32 }** %can
  %tortdown_ret32 = call i32 @tortdown({ i32, i32 }* %can31, { i32, i32,
i32 }* %window30, i32 30)
  %movetort_ret33 = call i32 @movetort(i32 220, i32 235)
  %window34 = load { i32, i32, i32 }*, { i32, i32, i32 }** %window
  %can35 = load { i32, i32 }*, { i32, i32 }** %can
  %tortright_ret36 = call i32 @tortright({ i32, i32 }* %can35, { i32, i32,
i32 }* %window34, i32 30)
  %win = alloca { i32, i32, i32 }*
  %Rgb37 = call { i32, i32, i32 }* @Rgb(i32 235, i32 149, i32 52)
  store { i32, i32, i32 }* %Rgb37, { i32, i32, i32 }** %win
  %movetort_ret38 = call i32 @movetort(i32 270, i32 260)
  %win39 = load { i32, i32, i32 }*, { i32, i32, i32 }** %win
  %can40 = load { i32, i32 }*, { i32, i32 }** %can
  %tortright_ret41 = call i32 @tortright({ i32, i32 }* %can40, { i32, i32,
i32 }* %win39, i32 60)
  %win42 = load { i32, i32, i32 }*, { i32, i32, i32 }** %win
```

```
  %can43 = load { i32, i32 }*, { i32, i32 }** %can
  %tortdown_ret44 = call i32 @tortdown({ i32, i32 }* %can43, { i32, i32,
i32 }* %win42, i32 60)
  %win45 = load { i32, i32, i32 }*, { i32, i32, i32 }** %win
  %can46 = load { i32, i32 }*, { i32, i32 }** %can
  %tortleft_ret47 = call i32 @tortleft({ i32, i32 }* %can46, { i32, i32,
i32 }* %win45, i32 60)
  %win48 = load { i32, i32, i32 }*, { i32, i32, i32 }** %win
  %can49 = load { i32, i32 }*, { i32, i32 }** %can
  %tortup_ret50 = call i32 @tortup({ i32, i32 }* %can49, { i32, i32, i32 }*
%win48, i32 60)
  %movetort_ret51 = call i32 @movetort(i32 300, i32 260)
  %win52 = load { i32, i32, i32 }*, { i32, i32, i32 }** %win
  %can53 = load { i32, i32 }*, { i32, i32 }** %can
  %tortdown_ret54 = call i32 @tortdown({ i32, i32 }* %can53, { i32, i32,
i32 }* %win52, i32 60)
  %movetort_ret55 = call i32 @movetort(i32 270, i32 290)
  %win56 = load { i32, i32, i32 }*, { i32, i32, i32 }** %win
  %can57 = load { i32, i32 }*, { i32, i32 }** %can
  %tortright_ret58 = call i32 @tortright({ i32, i32 }* %can57, { i32, i32,
i32 }* %win56, i32 60)
  %chimney = alloca { i32, i32, i32 }*
  %Rgb59 = call { i32, i32, i32 }* @Rgb(i32 97, i32 51, i32 1)
  store { i32, i32, i32 }* %Rgb59, { i32, i32, i32 }** %chimney
  %movetort_ret60 = call i32 @movetort(i32 350, i32 150)
  %chimney61 = load { i32, i32, i32 }*, { i32, i32, i32 }** %chimney
  %can62 = load { i32, i32 }*, { i32, i32 }** %can
  %tortup_ret63 = call i32 @tortup({ i32, i32 }* %can62, { i32, i32, i32 }*
%chimney61, i32 100)
  %chimney64 = load { i32, i32, i32 }*, { i32, i32, i32 }** %chimney
  %can65 = load { i32, i32 }*, { i32, i32 }** %can
  %tortright_ret66 = call i32 @tortright({ i32, i32 }* %can65, { i32, i32,
i32 }* %chimney64, i32 35)
  %chimney67 = load { i32, i32, i32 }*, { i32, i32, i32 }** %chimney
  %can68 = load { i32, i32 }*, { i32, i32 }** %can
  %tortdown_ret69 = call i32 @tortdown({ i32, i32 }* %can68, { i32, i32,
i32 }* %chimney67, i32 135)
  %door = alloca { i32, i32, i32 }*
  %Rgb70 = call { i32, i32, i32 }* @Rgb(i32 114, i32 40, i32 143)
  store { i32, i32, i32 }* %Rgb70, { i32, i32, i32 }** %door
  %movetort_ret71 = call i32 @movetort(i32 320, i32 400)
  %door72 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can73 = load { i32, i32 }*, { i32, i32 }** %can
```

```
  %tortup_ret74 = call i32 @tortup({ i32, i32 }* %can73, { i32, i32, i32 }*
%door72, i32 60)
  %door75 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can76 = load { i32, i32 }*, { i32, i32 }** %can
  %tortright_ret77 = call i32 @tortright({ i32, i32 }* %can76, { i32, i32,
i32 }* %door75, i32 60)
  %door78 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can79 = load { i32, i32 }*, { i32, i32 }** %can
  %tortdown_ret80 = call i32 @tortdown({ i32, i32 }* %can79, { i32, i32,
i32 }* %door78, i32 60)
  %movetort_ret81 = call i32 @movetort(i32 350, i32 400)
  %door82 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can83 = load { i32, i32 }*, { i32, i32 }** %can
  %tortup_ret84 = call i32 @tortup({ i32, i32 }* %can83, { i32, i32, i32 }*
%door82, i32 60)
  %movetort_ret85 = call i32 @movetort(i32 340, i32 370)
  %door86 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can87 = load { i32, i32 }*, { i32, i32 }** %can
  %tortright_ret88 = call i32 @tortright({ i32, i32 }* %can87, { i32, i32,
i32 }* %door86, i32 6)
  %door89 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can90 = load { i32, i32 }*, { i32, i32 }** %can
  %tortdown_ret91 = call i32 @tortdown({ i32, i32 }* %can90, { i32, i32,
i32 }* %door89, i32 6)
  %door92 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can93 = load { i32, i32 }*, { i32, i32 }** %can
  %tortleft_ret94 = call i32 @tortleft({ i32, i32 }* %can93, { i32, i32,
i32 }* %door92, i32 6)
  %door95 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can96 = load { i32, i32 }*, { i32, i32 }** %can
  %tortup_ret97 = call i32 @tortup({ i32, i32 }* %can96, { i32, i32, i32 }*
%door95, i32 6)
  %movetort_ret98 = call i32 @movetort(i32 354, i32 370)
  %door99 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can100 = load { i32, i32 }*, { i32, i32 }** %can
  %tortright_ret101 = call i32 @tortright({ i32, i32 }* %can100, { i32,
i32, i32 }* %door99, i32 6)
  %door102 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can103 = load { i32, i32 }*, { i32, i32 }** %can
  %tortdown_ret104 = call i32 @tortdown({ i32, i32 }* %can103, { i32, i32,
i32 }* %door102, i32 6)
  %door105 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can106 = load { i32, i32 }*, { i32, i32 }** %can
```

```
  %tortleft_ret107 = call i32 @tortleft({ i32, i32 }* %can106, { i32, i32,
i32 }* %door105, i32 6)
  %door108 = load { i32, i32, i32 }*, { i32, i32, i32 }** %door
  %can109 = load { i32, i32 }*, { i32, i32 }** %can
  %tortup_ret110 = call i32 @tortup({ i32, i32 }* %can109, { i32, i32, i32
}* %door108, i32 6)
  %can111 = load { i32, i32 }*, { i32, i32 }** %can
  %save = call i32 @save({ i32, i32 }* %can111, i8* getelementptr inbounds
([24 x i8], [24 x i8]* @str, i32 0, i32 0))
  ret i32 0
}

define i32 @movetort(i32 %x, i32 %y) {
entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %y2 = alloca i32
  store i32 %y, i32* %y2
  %x3 = load i32, i32* %x1
  store i32 %x3, i32* @xcur
  %y4 = load i32, i32* %y2
  store i32 %y4, i32* @ycur
  ret i32 0
}

define i32 @tortSE({ i32, i32 }* %can, { i32, i32, i32 }* %color, double
%distance) {
entry:
  %can1 = alloca { i32, i32 }*
  store { i32, i32 }* %can, { i32, i32 }** %can1
  %color2 = alloca { i32, i32, i32 }*
  store { i32, i32, i32 }* %color, { i32, i32, i32 }** %color2
  %distance3 = alloca double
  store double %distance, double* %distance3
  %counter = alloca i32
  store i32 0, i32* %counter
  %counter1 = alloca double
  store double 0.000000e+00, double* %counter1
  %step = alloca double
  %distance4 = load double, double* %distance3
  %tmp = fmul double %distance4, 7.070000e-01
  store double %tmp, double* %step
  br label %while
```

```
while:                                            ; preds = %while_body,
%entry
  %counter115 = load double, double* %counter1
  %step16 = load double, double* %step
  %tmp17 = fcmp olt double %counter115, %step16
  br i1 %tmp17, label %while_body, label %merge

while_body:                                       ; preds = %while
  %can5 = load { i32, i32 }*, { i32, i32 }** %can1
  %color6 = load { i32, i32, i32 }*, { i32, i32, i32 }** %color2
  %xcur = load i32, i32* @xcur
  %counter7 = load i32, i32* %counter
  %tmp8 = add i32 %xcur, %counter7
  %ycur = load i32, i32* @ycur
  %counter9 = load i32, i32* %counter
  %tmp10 = add i32 %ycur, %counter9
  %pixel = call { i32, i32 }* @pixel({ i32, i32 }* %can5, { i32, i32, i32
}* %color6, i32 %tmp8, i32 %tmp10)
  %counter111 = load double, double* %counter1
  %tmp12 = fadd double %counter111, 1.000000e+00
  store double %tmp12, double* %counter1
  %counter13 = load i32, i32* %counter
  %tmp14 = add i32 %counter13, 1
  store i32 %tmp14, i32* %counter
  br label %while

merge:                                            ; preds = %while
  %xcur18 = load i32, i32* @xcur
  %counter19 = load i32, i32* %counter
  %tmp20 = add i32 %xcur18, %counter19
  store i32 %tmp20, i32* @xcur
  %ycur21 = load i32, i32* @ycur
  %counter22 = load i32, i32* %counter
  %tmp23 = add i32 %ycur21, %counter22
  store i32 %tmp23, i32* @ycur
  ret i32 0
}
```

Chapter 4 - Project Plan

4.1 Development Process

4.1.1 Planning

Our team met every Wednesday evening in order to check-in on our progress and ensure that we all were on track to fulfill the upcoming milestones. If we had any concerns that we were not able to address during our meetings, we would attend the office hours of our T.A. Jianan Yao or Professor Edwards. After getting feedback from T.A.s and Professor Edwards, we would often meet on Fridays, Saturdays, and Mondays in order to continue making progress and meeting deadlines. In order to have a communication channel running, we used Facebook Messenger for discussion of pressing issues and scheduling times to meet.

4.1.2 Specification and Development

During our meetings, we would review what we accomplished since the last meeting, any blockers in the code and plan, as well as the next steps to achieve by the following Wednesday. We also worked on multiple parts of the language collaboratively to resolve bugs. We planned out the architectural aspects of our compiler in order to better understand how expressions or statements would pass through the scanner, parser, semantic checker, and code generation. Every week, we would either divide up tasks to accomplish before the next meeting or schedule another meeting during the week to collaboratively work on a component. At the end of every meeting, we would all have a clear understanding of the next steps by documenting them in our README.

4.1.3 Testing

Every time we added a component in the scanner, parser, AST, SAST, semantic checker, and code generator, we made robust test cases to ensure the component is working as intended. We created .rt files where we wrote the Reptile program, .out files that have the expected output, and .err files that have the exceptions or error messages. When we distributed the work of adding different components, each individual was responsible for adding test cases for the component(s) they worked on. During the meetings, we ensured that we covered all the test cases needed for the components we had implemented up until that point.

4.2 Programming Style and Conventions

Our team utilized the following guidelines to maintain consistent code structure and readability:

- For positive integration tests and output, use test-*.rt and test-*.out
- For positive integration tests and errors, use fail-*.rt and fail-*.err
- Use spaces instead of tabs, so that different editors render them as intended.
- Any nested code should be indented by two spaces in OCaml, and four spaces in C.
- Use snake case for function names and camelCase for variable names.
- Any expressions and statements that are recursively evaluated, and if bound locally, should take the name of the original argument, along with an apostrophe added (i.e. let t' = expr locals t).
- Functions which are added should include a comment above the signature describing what the function does, and the context(s) in which it would be called.
- Function calls or lines which extend past 80 characters should be broken up into multiple lines, and those lines should be indented four spaces beyond the first line, to indicate that those lines are part of one expression, and not a different, nested expression.

4.3 Project Timeline

Date	Description
January 28	Brainstorm ideas for language and get feedback on top ideas
February 3	Complete Project Proposal
February 10	Revisit Proposal and incorporate feedback
February 24	Complete Scanner, Parser, AST, and Language Reference Manual
March 29	Complete Hello World Program
April 10	Add remaining statements
April 17	Add Standard Library Functions
April 23	Complete PNG Library Integration
April 25	Complete Final Report
April 25	Complete Final Presentation

4.4 Roles and Responsibilities

Team Member	Role
Aileen Cano	Test Designer

Aviva Weinbaum	Language Guru
Lindsey Weiskopf	System Architect
Hariti Patel	Manager

Although each member had a specified role, we all collaboratively worked on all aspects of the proposal, Language Reference Manual, compiler, test suite, and report. We split up several components in our project based on the types we specified in the AST. Every member was expected to ensure that they implement the assigned type all the way through code generation and create test cases for it. We used our meeting times to implement more complex components and resolve difficult bugs.

4.5 Software Development Environment

Libraries and Languages:

- OCaml version 4.05.0
- OCaml LLVM version 6.0.0
- OCamlyacc version 4.05.0
- Ocamllex version 4.05.0
- C & C libraries:
    - stdio.h
    - stdlib.h
    - string.h
    - math.h
    - libattopng.h
        - We used this library to implement all of our png manipulation functionalities. The library, created by Michael Schwarz, includes simple png manipulation functions including file I/O, memory handling, color conversions, and pixel bit-flipping.

Software:

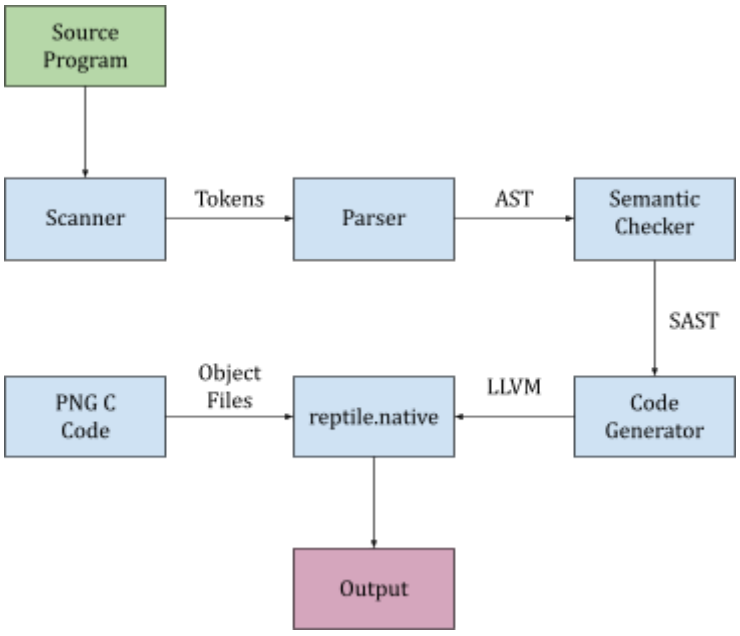
We used a Git repository on GitHub for version control. We all utilized Visual Studio Code to develop our code and the built in terminal to compile and run our files. We also used Google Docs and LaTeX to manage our proposal, Language Reference Manual, and final report.

OS: MacOS Big Sur 11.2.3 with Docker 20.10.5

4.6 Project Log

See Appendix 9.13 Git Log for all our GitHub commit messages indicating our course of development.

Chapter 5 - Architectural Design



5.1 Interfaces

The diagram above depicts the components of our compiler as well as the interfaces between them. An input Reptile (.rt) program is fed into the scanner (scanner.mll), tokenized, and is then fed to the parser (parser.mly), which creates the abstract syntax tree (AST - ast.ml). The AST is fed into the semantic checker (semant.ml) to generate a semantically-checked AST (SAST - sast.ml), which is essentially an AST with types associated with each expression. The SAST is then passed to the code generator (codegen.ml), which creates the LLVM of the source program. The provided reptile.native compiles the LLVM IR into a .s native assembly code file, and ultimately links it with the PNG C code object files (including the structs, the libattopng library, and the built-in functions within pixelator.c), which are compiled when *make* is run. Lastly, an executable file is created and run by the native along with the created .ll and .s files, which are all removed, and finally, an output file is created.

5.2 Error Checking

Checking for errors is crucial for any language and compiler. When the source program has invalid tokens, the scanner will throw an error. If there are syntactically invalid statements or expressions in the source program, the parser will throw an error. If there are semantically invalid statements or expressions in the source program, the semantic checker will throw an error.

5.3 Authorship

The scanner, parser, ast, sast, semantic checker, code generator, and C libraries were all written collectively by all four team members. Although we mainly worked together during meetings, we occasionally split up working on components. Aviva worked on assignment, while loops, the pixel function, and adding tests for those. Hariti worked on if/else functionality, adding string types, changing the complex types to be pointers, and adding tests for those along with the boolean operators. Aileen worked on adding tests for functions, return, if, and ensuring that we have test coverage for all the components we have added. Lindsey worked on adding the complex types like RGB, access methods for complex types, adding extra C structs, starting the library integration, and producing the PNG file after we collectively integrated the library. The files png.c and png.h were taken directly from the Libattopng Library (see here: https://github.com/misc0110/libattopng). We also adapted a lot of code building up from MicroC and inspiration from SSOL (2018).

Chapter 6 - Test Plan

All of the features in our language were tested with passing and failing tests. Features such as the built-in functions, structs, and recursion were tested individually and in conjunction with other features to ensure there were no problems or conflicts with other features. The end-to-end integration tests can be found in the *test/* directory.

6.1 Unit Testing

For our project, we wrote our tests concurrently with our code. We made sure that as operators, functions, and keywords were added to our parser, our parser would be able to recognize them through tests. We also individually tested the parser to ensure that our language has a context-free grammar and is free of any shift/reduce conflicts.

6.2 Integration Testing

We created an end-to-end integration test suite in order to rigorously test the functionalities of our language. The test suite also helped us ensure that we did not break any of our previously working features after we added our new ones.

For all of the features we implemented, we wrote test programs in .rt files, our chosen extension, using the naming pattern of test-*.rt, and made sure tests that we expected to pass had matching output to a corresponding .out file, using the naming pattern of test-*.out. If the test was expected to fail, we compared the output of our negative test programs, using the naming pattern of fail-*.rt, with the corresponding .err files, using the naming pattern of fail-*.err, and made sure the correct compiler error messages were output. Aileen, the test designer, created general tests to check for syntax, but all team members created tests for the features they developed.

A complete set of tests can be found in Appendix 9.11.1 Integration Test Files - Negative Tests and Appendix 9.11.2 Integration Test Files - Positive Tests. These tests were parallely implemented with the functionalities they tested, meaning that they were chosen to test and check the implemented functionalities and their failure cases.

6.2.1 Test Automation and Scripts

In terms of automation for our integration tests, we made use of the test all script (testall.sh), provided in the MicroC compiler, to compile, run through, and check the output of all our tests. We modified the script to match our language. The script essentially works by running all of the fail-*.rt and test-*.rt programs and comparing the errors or outputs, respectively, to check whether they match the expected value. In the case that the output or

error matches what is expected, the script prints the test name and an “OK” next to each test name. Conversely, in the case that the output or error does not match what is expected, the script prints the test name, “FAILED” next to each test name, and the error message. To find a more detailed message, one can check the testall.log.

6.3 Loads Tests

The test below includes approximately 61,000 lines of code and pixel function calls. We use this as a load test to make sure that our language is allocating and deallocating the appropriate structures on the heap. Though this program takes several seconds to compile and run, it is successful.

test-edwards.rt

```
int main() {
        Canvas can = Canvas(300, 400);
        Rgb color = Rgb(0, 0, 0);
        pixel(can, color, 0, 0);
        pixel(can, color, 0, 1);
        ...
        save(can, "edwardstest.png");
}
```

6.4 Example Test Programs

6.4.1 Unit Testing

fail-dead.rt

```
int main()
{
        int i = 15;
        return i;
        i = 32;
}
```

6.4.2 Integration Testing

test-simple-tort.rt

```
int main() {
        Canvas can = Canvas (600, 600);
        Rgb color = Rgb(56, 137, 190);
        Rgb roofcolor = Rgb(255, 0, 0);
        movetort(200, 200);  /\This is an example of a user-defined function.
        tortright(can, roofcolor, 200); /\This is an example of a user-defined function.
```

```
        tortdown(can, color, 200); /\This is an example of a user-defined function.
        tortleft(can, color, 200); /\This is an example of a user-defined function.
        tortup(can, color, 200); /\This is an example of a user-defined function.
        ...
        save(can, "simple_turtle_start.png");
        return 0;
}
```

6.5 Division of Labor for Testing

Our team strived for a test-driven development of our project. To that end, general tests for syntax, recursion, and arithmetic were created first while tests for the structs and built in functions were created by the individuals responsible for those features. The specific authorship of tests is listed in the authorship section.

Chapter 7 - Conclusions & Lessons Learned

7.1 Aileen

This semester, I learned a lot about functional programming and compilers. Throughout my time as a computer science student, I had never really thought about what happened after you hit the run button on the terminal. However, after this class and project, I have a greater appreciation for compilers and the work they do to help us code really cool programs like the project I worked on. One of the biggest lessons I learned from this project is the importance of being accountable for your work in terms of time and contributions. My group immediately set up a time that worked for us and met each week religiously, even if there wasn't much work to do. This really helped us keep a steady timeline and make sure we made good progress. As diligent as we were in our schoolwork, I think my teammates will agree with me when I say that even though we kept up with the lectures, the uniqueness of our project presented certain roadblocks that we were often uncertain on how to approach. By doing research on possible solutions, going to office hours, and working at our project, we were able to solve the problems we faced. There were moments in which it seemed like we had bitten off more than we could chew, but we always managed to find a solution, which felt really great.

7.2 Aviva

I learned a lot throughout the semester, about OCaml and functional programming, making a compiler, and working with a team. I was not aware of functional programming before this class, and although it took a lot of work at the beginning to understand the full extent of OCaml's capabilities, by the end of developing our language it made more sense and I understood how easy it made it to make a compiler. My team went into the project with very ambitious plans for how to make our language very robust and how we could add a lot of functionality with a standard library. However, it ended up being more important to spend a lot of time getting the basic language working before we could build off of it. It is more important to spend a lot of time planning out the small additions than to jump in with large implementation plans and get bogged down with errors and roadblocks and have to backtrack. I would also recommend spending more time dividing up tasks and discussing the development timeline so that everyone is not working on the same tasks at the same time. Our group did a really good job of communicating what we were working on so our additions never overlapped, but we probably could have been more efficient with delegation.

7.3 Lindsey

When we first started working on this project, I felt as though I was being thrown into completely unknown territory. When it came to my knowledge of programming languages, I knew that I liked certain aspects of Java, other aspects of Python, and very few aspects of C (and OCaml at the time). That being said, I did not have the vocabulary to express those features in terms of building a language. As I learned more about grammars, intermediate representations, and memory, the project started to clarify. What started as the jumbled puzzle pieces of MicroC, turned into a functioning programming language that has some pretty cool graphical features! At times during the course, I was worried that I would never fully "get it." But, after getting my hands a little dirty, I felt comfortable enough with the structure to explain what was going on to someone else. I would say that my biggest piece of advice is to create a plan for the project, early on, that is clear enough in your own mind that you would be able to explain it to someone else who wasn't taking PLT. Throughout the process, it was most frustrating for me when I didn't understand the full picture. But as I connected the dot between scanning/parsing, semantics, and LLVM, the process became rewarding. There is no better feeling than watching your language compile and your tests pass, so build yourself a strong foundation to get yourself there!

7.4 Hariti

Prior to this class, I had not been introduced to the power of functional programming. Although I found it to be frustrating during the initial phases of this class and project, I realized that it is a unique way to programmatically solve problems with such few lines of code. It was especially interesting to understand how lessons from some core classes of the CS curriculum, namely Computer Science Theory and Fundamentals of Computer Systems, fit into the puzzle of Programming Languages & Translators. Compilers were something I took for granted in the previous classes. However, this class and this project has made me realize the value of understanding the structure and inner workings of compiler design. I am very grateful to have had the opportunity to understand what happens when a source program is fed and ultimately generates an output after executing numerous intermediate steps as depicted in our architectural design diagram. One advice that I would like to give to future teams is that starting early and continually making progress makes a tremendous difference. You will undoubtedly face roadblocks and may feel like giving up, but the beauty of functional programming is that you keep trying and trying until you get your program and language to compile, afterwhich it most likely works as intended. It is important to incrementally code and develop components to achieve small goals that will help you accomplish your ultimate goal.

Chapter 8 - Acknowledgements

Our team would like to specially thank Professor Edwards, our TA advisor Jianan Yao, as well as the other TAs of this course for providing us with support and guidance throughout this project. Additionally, we are grateful to have referenced projects like MicroC, SSOL (2018), Crayon (2017), and Michael Schwarz's Libattopng Library to seek inspiration in our development process.

Chapter 9 - Appendix

9.1 Scanner - scanner.mll

```
(* Ocamllex scanner for Reptile *)
(* Aviva Weinbaum, Lindsey Weiskopf, Hariti Patel, Aileen Cano *)

{ open Parser }

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

rule token = parse
 [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/\\"     { comment lexbuf }           (* Comments *)
| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| ';'      { SEMI }
| ','      { COMMA }
| '='      { ASSIGN }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| "=="     { EQ }
| "!="     { NEQ }
| '<'      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "&&"     { AND }
| "||"     { OR }
| "!"      { NOT }
| "if"     { IF }
| "else"   { ELSE }
| "int"    { INT }
| "string" { STRING }
| "float"  { FLOAT }
| "void"   { VOID }
| "bool"   { BOOL }
| "Rgb"    { RGB }
| "Canvas" { CANVAS }
| "Pointer" { POINTER }
| "true"   { BLIT(true) }
| "false"   { BLIT(false) }
| "while"  { WHILE }
| "return" { RETURN }
| digits '.'  digit* ( ['e' 'E'] ['+' '-']? digits )? as lxm { FLIT(lxm) }
| digits as lxm { LITERAL(int_of_string lxm) }
```

```
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']*     as lxm { ID(lxm) }
| '"' ([^ '"']* as str) '"' { SLIT(str) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

and comment = parse
 "\n" { token lexbuf }
| _    { comment lexbuf }
```

9.2 Parser - parser.mly

```
/* Ocamlyacc parser for Reptile */
/* Inspo from SSOL (2018), Crayon (2017), and MicroC */
/* Aviva Weinbaum, Lindsey Weiskopf, Hariti Patel, Aileen Cano */

%{
open Ast
%}

%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA
%token PLUS MINUS TIMES DIVIDE ASSIGN
%token TRUE FALSE
%token NOT EQ NEQ LT LEQ GT GEQ AND OR
%token INT STRING VOID BOOL FLOAT
%token RGB CANVAS POINTER
%token RETURN WHILE
%token RETURN IF ELSE
%token <int> LITERAL
%token <bool> BLIT
%token <string> ID SLIT FLIT
%token EOF

%start program
%type <Ast.program> program

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%left EXP
%right NOT

%%
```

```
program:
 decls EOF { $1 }

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

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

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

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

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

vdecl_stmt:
   typ ID SEMI { Var($1,$2) }
 | typ ID ASSIGN expr SEMI { VarAssign($1,$2,$4)}

typ:
   STRING { String }
 | INT { Int }
 | VOID { Void }
 | BOOL { Bool }
 | FLOAT { Float }
 | RGB { Rgb }
 | CANVAS { Canvas }
 | POINTER { Pointer }

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

stmt:
   expr SEMI                               { Expr $1   }
 | RETURN expr_opt SEMI                    { Return $2 }
 | LBRACE stmt_list RBRACE                 { Block(List.rev $2) }
 | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
 | IF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7)        }
 | vdecl_stmt                              { $1 }
```

```
 | WHILE LPAREN expr RPAREN stmt           { While($3, $5) }

expr_opt:
         { Noexpr }
 | expr  { $1 }

expr:
   LITERAL          { Literal($1)            }
 | BLIT             { BoolLit($1)            }
 | FLIT             { Fliteral($1)           }
 | SLIT             { Sliteral($1)             }
 | ID               { Id($1) }
 | expr PLUS   expr { Binop($1, Add,   $3)   }
 | expr MINUS  expr { Binop($1, Sub,   $3)   }
 | expr TIMES  expr { Binop($1, Mul,  $3)   }
 | expr DIVIDE expr { Binop($1, Div,   $3)   }
 | expr EQ     expr { Binop($1, Equal, $3)   }
 | expr NEQ    expr { Binop($1, Neq,   $3)   }
 | expr LT     expr { Binop($1, Less,  $3)   }
 | expr LEQ    expr { Binop($1, Leq,   $3)   }
 | expr GT     expr { Binop($1, Greater, $3) }
 | expr GEQ    expr { Binop($1, Geq,   $3)   }
 | expr AND    expr { Binop($1, And,   $3)   }
 | expr OR     expr { Binop($1, Or,    $3)   }
 | MINUS expr %prec NOT { Unop(Neg, $2)      }
 | NOT expr         { Unop(Not, $2)          }
 | ID ASSIGN expr   { Assign($1, $3)         }
 | ID LPAREN args_opt RPAREN { Call($1, $3)  }
 | typ LPAREN args_opt RPAREN        { Call((string_of_typ $1), $3) }
 | LPAREN expr RPAREN { $2                   }

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

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

9.3 AST - ast.ml

```
(* AST for Reptile *)
(* Aviva Weinbaum, Lindsey Weiskopf, Hariti Patel, Aileen Cano *)

type operator = Add | Sub | Mul | Div | Equal | Neq | Less | Greater | Geq | Leq |
         And | Or

type unoperator = Not | Neg
```

```
type typ = Int | String | Void | Bool | Float | Rgb | Canvas | Pointer

type bind = typ * string

type expr =
   Binop of expr * operator * expr
 | Unop of unoperator * expr
 | Literal of int
 | Fliteral of string
 | Sliteral of string
 | BoolLit of bool
 | Id of string
 | Call of string * expr list
 | Assign of string * expr
 | Noexpr

type stmt =
   Expr of expr
 | Block of stmt list
 | Return of expr
 | If of expr * stmt * stmt
 | Var of typ * string
 | VarAssign of typ * string * expr
 | While of expr * stmt

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

type program = bind list * func_decl list


(* Pretty-printing functions *)

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

let string_of_typ = function
```

```
 Int -> "int"
| Bool -> "bool"
| Void -> "void"
| Float -> "float"
| String -> "string"
| Rgb -> "Rgb"
| Canvas -> "Canvas"
| Pointer -> "Pointer"

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

let rec string_of_expr = function
 Literal(l) -> string_of_int l
| Fliteral(f) -> f
| Sliteral(s) -> s
| Id(s) -> s
| Binop(e1, o, e2) ->
 string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
| Unop(o, e) -> string_of_uop o ^ string_of_expr e
| BoolLit(true) -> "true"
| BoolLit(false) -> "false"
| Call(f, el) ->
 f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
| Assign(v, e) -> v ^ " = " ^ string_of_expr e
| Noexpr -> ""

let rec string_of_stmt = function
Block(stmts) ->
 "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
| Expr(expr) -> string_of_expr expr ^ ";\n";
| Var(typ, var) -> string_of_typ typ ^ " " ^ var ^ "\n"
| VarAssign(typ, var, ex) -> string_of_typ typ ^ " " ^ var ^ " = " ^ string_of_expr ex ^
"\n"
| Return(expr) -> "return " ^ string_of_expr expr ^ ";\n"
| While(expr, stmt) -> "while (" ^ string_of_expr expr ^ ") " ^ string_of_stmt stmt ^ "\n"
| If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
| If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
     string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2

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

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

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

```
 String.concat "\n" (List.map string_of_fdecl funcs)
```

9.4 SAST - sast.ml

```
(* SAST for Reptile *)
(* Aviva Weinbaum, Lindsey Weiskopf, Hariti Patel, Aileen Cano *)

open Ast

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

type sstmt =
   SExpr of sexpr
 | SBlock of sstmt list
 | SReturn of sexpr
 | SIf of sexpr * sstmt * sstmt
 | SVar of typ * string
 | SVarAssign of typ * string * sexpr
 | SWhile of sexpr * sstmt

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

 type sprogram = bind list * sfunc_decl list


(* Pretty-printing functions *)

let rec string_of_sexpr (t, e) =
 "(" ^ string_of_typ t ^ " : " ^ (match e with
   SLiteral(l) -> string_of_int l
 | SFliteral(f) -> f
 | SSliteral(s) -> s
 | SBoolLit(true) -> "true"
```

```
 | SBoolLit(false) -> "false"
 | SId(s) -> s
 (* | SString(s) -> s *)
 | SBinop(e1, o, e2) ->
     string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
 | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
 | SCall(f, el) ->
     f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
 (* | SListAccess(arr, index) ->
   arr ^ "[" ^ string_of_sexpr index ^ "]"
 | SListLit(args) -> "[" ^ (List.map string_of_sexpr args) ^ "]" *)
 | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e
 | SNoexpr -> ""
        ) ^ ")"

let rec string_of_sstmt = function
   SBlock(stmts) ->
     "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "}\n"
 | SExpr(expr) -> string_of_sexpr expr ^ ";\n";
 | SVar(typ, var) -> string_of_typ typ ^ " " ^ var ^ "\n"
 | SVarAssign(typ, var, ex) -> string_of_typ typ ^ " " ^ var ^ " = " ^ string_of_sexpr ex ^
"\n"
 | SReturn(expr) -> "return " ^ string_of_sexpr expr ^ ";\n"
 | SWhile(expr, stmt) -> "while (" ^ string_of_sexpr expr ^ ") " ^ string_of_sstmt stmt ^
"\n"
 | SIf(e, s, SBlock([])) ->
     "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s
 | SIf(e, s1, s2) ->  "if (" ^ string_of_sexpr e ^ ")\n" ^
     string_of_sstmt s1 ^ "else\n" ^ string_of_sstmt s2

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

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

9.5 Semantic Checker - semant.ml

```
(* Semantic checking for Reptile compiler *)
(* Aviva Weinbaum, Lindsey Weiskopf, Hariti Patel, Aileen Cano *)

open Ast
open Sast

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

```
(* Semantic checking of the AST. Returns an SAST if successful,
  throws an exception if something is wrong.

  Check each global variable, then check each function *)

let check (globals, functions) =

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

 (**** Check global variables ****)

 check_binds "global" globals;

 (**** Check functions ****)

 (* Collect function declarations for built-in functions: no bodies *)
 let built_in_decls =
   let add_bind map (name, rety, tys) = StringMap.add name {
     typ = rety;
     fname = name;
     formals = tys;
     body = [] } map
   in List.fold_left add_bind StringMap.empty [ ("print", Void, [(Int, "x")]);
     ("Rgb", Rgb, [(Int, "r"); (Int, "g"); (Int, "b")]);
     ("Pointer", Pointer, [(Int, "x"); (Int, "y"); (Rgb, "color"); (Float, "angle")]);
     ("Canvas", Canvas, [(Int, "x"); (Int, "y")]);
     ("save", Void, [(Canvas, "can"); (String, "filename")]);
     ("pixel", Canvas, [(Canvas, "can"); (Rgb, "color"); (Int, "x"); (Int, "y")]);
     ("get_rgb_r", Int, [(Rgb, "rgb");]);
     ("get_rgb_g", Int, [(Rgb, "rgb");]);
     ("get_rgb_b", Int, [(Rgb, "rgb");]);
     ("get_pointer_x", Int, [(Pointer, "pointer")]);
     ("get_pointer_y", Int, [(Pointer, "pointer")]);
     ("set_pointer_color", Pointer, [(Pointer, "pointer"); (Rgb, "rgb")]);
     ("get_canvas_x", Int, [(Canvas, "canvas")]);
     ("get_canvas_y", Int, [(Canvas, "canvas")]);
     ("sine", Float, [(Float, "angle");]);
     ("cosine", Float, [(Float, "angle");]);
     ("tangent", Float, [(Float, "angle");]);
     ("mod", Int, [(Int, "val1"); (Int, "val2");]);
     ("floors", Int, [(Float, "val");]);
     ("getRise", Int, [(Int, "distance"); (Float, "angle");]);
     ("getRun", Int, [(Int, "distance"); (Float, "angle");])
```

```
    ];
in

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

(* Collect all function names into one symbol table *)
let function_decls = List.fold_left add_func built_in_decls functions
in
 (* Return a function from our symbol table *)
let find_func s =
  try StringMap.find s function_decls
  with Not_found -> raise (Failure ("unrecognized function " ^ s))
in

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

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

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

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

  (* Return a variable from our local symbol table *)
  let type_of_identifier locals s =
    try StringMap.find s locals
    with Not_found -> raise (Failure ("undeclared identifier " ^ s))
  in

  (* Return a semantically-checked expression, i.e., with a type *)
  let rec expr locals = function
      Literal  l -> (Int, SLiteral l)
    | Fliteral f -> (Float, SFliteral f)
    | Sliteral s -> (String, SSliteral s)
```

```
    | BoolLit  l -> (Bool, SBoolLit l)
    | Noexpr     -> (Void, SNoexpr)
    | Id s       -> (type_of_identifier locals s, SId s)
    | Unop(op, e) ->
        let (t, e') = expr locals e in
        let ty = match op with
          Neg when t = Int -> t
        | Not when t = Bool -> Bool
        | _ -> raise (Failure ("illegal unary operator"))
        in (ty, SUnop(op, (t, e')))
    | Binop(e1, op, e2) ->
        let (t1, e1') = expr locals e1
        and (t2, e2') = expr locals e2 in
        (* All binary operators require operands of the same type *)
        let same = t1 = t2 in
        (* Determine expression type based on operator and operand types *)
        let ty = match op with
          Add | Sub | Mul | Div when same && t1 = Int   -> Int
        | Add | Sub | Mul | Div when same && t1 = Float -> Float
        | Equal | Neq            when same               -> Bool
        | Less | Leq | Greater | Geq
                   when same && ((t1 = Int) || (t1 = Float)) -> Bool
        | And | Or when same && t1 = Bool -> Bool
        | _ -> raise (
      Failure ("illegal binary operator "))
        in (ty, SBinop((t1, e1'), op, (t2, e2')))
    | Call(fname, args) ->
        let fd = find_func fname in
        let param_length = List.length fd.formals in
        if List.length args != param_length then
          raise (Failure ("calling failure"))
        else let check_call (ft, _) e =
          let (et, e') = expr locals e in
          let err = "illegal argument found" ^ string_of_typ et ^
            " expected " ^ string_of_typ ft ^ " in " ^ string_of_expr e
          in (check_assign ft et err, e')
        in
        let args' = List.map2 check_call fd.formals args
        in (fd.typ, SCall(fname, args'))
    | Assign(var, e) as ex ->
        let lt = type_of_identifier locals var
        and (rt, e') = expr locals e in
        let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
          string_of_typ rt ^ " in " ^ string_of_expr ex
        in (check_assign lt rt err, SAssign(var, (rt, e')))
  in

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

```
   (* Return a semantically-checked statement i.e. containing sexprs *)
   let rec check_stmt locals = function
       Expr e -> SExpr (expr locals e)
     | If(p, b1, b2) -> SIf(check_bool_expr locals p, check_stmt locals b1, check_stmt
locals b2)
     | Return e -> let (t, e') = expr locals e in
       if t = func.typ then SReturn (t, e')
       else raise (Failure ("return failure"))

     (* A block is correct if each statement is correct and nothing
        follows any Return statement.  Nested blocks are flattened. *)
     | Block sl ->
         let rec check_stmt_list block_locals ssl = function
             [Return _ as s] -> [check_stmt block_locals s]
           | Return _ :: _   -> raise (Failure "nothing may follow a return")
           | Block sl :: ss -> [check_stmt block_locals (Block sl)]
                               @ (check_stmt_list block_locals ssl ss)
           | s :: ss ->
             (match s with
               Var(t,name) ->
                 (match t with
                     Void -> raise(Failure ("illegal void local "^name))
                   | _ -> let block_locals = StringMap.add name t block_locals
                          in [check_stmt block_locals s] @ check_stmt_list block_locals ssl
ss)
             | VarAssign(t,name,e) ->
               if t == Void then raise(Failure ("illegal void local "^name) )
               else
               let sx = expr block_locals e in
               let typ = (if fst(sx) == t
                          then fst(sx)
                          else raise(Failure("illegal assignment"))) in
               let block_locals = StringMap.add name typ block_locals in
                 [check_stmt block_locals s] @ check_stmt_list block_locals ssl ss
             | _ -> [check_stmt block_locals s] @ check_stmt_list block_locals ssl ss)
           | []              -> ssl
         in SBlock(check_stmt_list locals [] sl)
     | Var (ty,id) -> SVar(ty,id)
     | VarAssign(_,s,e) ->
       let sx = expr locals e in
       let ty = type_of_identifier locals s in
       SVarAssign(ty,s,sx)
     | While(e, s) -> SWhile(check_bool_expr locals e, check_stmt locals s)
     in
   { styp = func.typ;
     sfname = func.fname;
     sformals = func.formals;
     sbody = match check_stmt symbols (Block func.body) with
       SBlock(sl) -> sl
     | _ -> raise (Failure ("internal error: block didn't become a block"))
   }
```

```
 in (globals, List.map check_function functions)
```

9.6 Code Generator - codegen.ml

```
(* Code generation for Reptile compiler *)
(* Aviva Weinbaum, Lindsey Weiskopf, Hariti Patel, Aileen Cano *)

module L = Llvm
module A = Ast
open Sast

module StringMap = Map.Make(String)

(* translate : Sast.program -> Llvm.module *)
let translate (globals, functions) =
 let context    = L.global_context () in

 (* Create the LLVM compilation module into which
    we will generate code *)
 let the_module = L.create_module context "Reptile" in

 (* Get types from the context *)
 let i32_t      = L.i32_type    context
 and i8_t       = L.i8_type     context
 and i1_t       = L.i1_type     context
 and void_t     = L.void_type   context
 and float_t    = L.double_type context
 and string_t   = L.pointer_type (L.i8_type context) in
 let rgb_t      = L.pointer_type(L.struct_type context [| i32_t ; i32_t; i32_t |]) in
 let pointer_t  = L.pointer_type(L.struct_type context [| i32_t ; i32_t ; rgb_t ; float_t
|]) in
 let canvas_t   = L.pointer_type(L.struct_type context [| i32_t ; i32_t |]) in

(* Return the LLVM type for a Reptile type *)
 let ltype_of_typ = function
     A.Int   -> i32_t
   | A.Bool  -> i1_t
   | A.Void  -> void_t
   | A.Float -> float_t
   | A.String -> string_t
   | A.Rgb -> rgb_t
   | A.Pointer -> pointer_t
   | A.Canvas -> canvas_t
 in

 let global_vars : L.llvalue StringMap.t =
   let global_var m (t, n) =
     let init = match t with
         A.Float -> L.const_float (ltype_of_typ t) 0.0
```

```
      | _ -> L.const_int (ltype_of_typ t) 0
    in StringMap.add n (L.define_global n init the_module) m in
  List.fold_left global_var StringMap.empty globals in

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

let ptrcons_t : L.lltype =
    L.function_type pointer_t [| i32_t ; i32_t ; rgb_t ; float_t |] in
let ptrcons_fun : L.llvalue =
    L.declare_function "Pointer" ptrcons_t the_module in

let canvascons_t : L.lltype =
    L.function_type canvas_t  [| i32_t ; i32_t |]  in
let canvascons_fun : L.llvalue =
    L.declare_function "Canvas" canvascons_t the_module in

let savecons_t : L.lltype =
    L.function_type i32_t  [| canvas_t ; string_t |] in
let savecons_fun : L.llvalue =
    L.declare_function "save" savecons_t the_module in

let pixelcons_t : L.lltype =
    L.function_type canvas_t  [| canvas_t ; rgb_t; i32_t ; i32_t |] in
let pixelcons_fun : L.llvalue =
    L.declare_function "pixel" pixelcons_t the_module in

let sinecons_t : L.lltype =
    L.function_type float_t [|float_t;|] in
let sinecons_fun : L.llvalue =
    L.declare_function "sine" sinecons_t the_module in

let cosinecons_t : L.lltype =
    L.function_type float_t [|float_t;|] in
let cosinecons_fun : L.llvalue =
    L.declare_function "cosine" cosinecons_t the_module in

let tangentcons_t : L.lltype =
    L.function_type float_t [|float_t;|] in
let tangentcons_fun : L.llvalue =
    L.declare_function "tangent" tangentcons_t the_module in

let modcons_t : L.lltype =
    L.function_type i32_t [|i32_t; i32_t;|] in
let modcons_fun : L.llvalue =
    L.declare_function "mod" modcons_t the_module in

let floorscons_t : L.lltype =
    L.function_type i32_t [|float_t;|] in
let floorscons_fun : L.llvalue =
```

```
    L.declare_function "floors" floorscons_t the_module in

 let getRuncons_t : L.lltype =
     L.function_type i32_t [|i32_t; float_t;|] in
 let getRuncons_fun : L.llvalue =
     L.declare_function "getRun" getRuncons_t the_module in

 let getRisecons_t : L.lltype =
     L.function_type i32_t [|i32_t; float_t;|] in
 let getRisecons_fun : L.llvalue =
     L.declare_function "getRise" getRisecons_t the_module in

(* Define each function (arguments and return type) so we can
    call it even before we've created its body *)
 let function_decls : (L.llvalue * sfunc_decl) StringMap.t =
   let function_decl m fdecl =
     let name = fdecl.sfname
     and formal_types =
       Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) fdecl.sformals)
     in let ftype = L.function_type (ltype_of_typ fdecl.styp) formal_types in
     StringMap.add name (L.define_function name ftype the_module, fdecl) m in
   List.fold_left function_decl StringMap.empty functions in

 (* Fill in the body of the given function *)
 let build_function_body fdecl =
   let (the_function, _) = StringMap.find fdecl.sfname function_decls in
   let builder = L.builder_at_end context (L.entry_block the_function) in

   let int_format_str = L.build_global_stringptr "%d\n" "fmt" builder in

   let local_vars =
     let add_formal m (t, n) p =
       L.set_value_name n p;
       let local = L.build_alloca (ltype_of_typ t) n builder in
       ignore (L.build_store p local builder);
        StringMap.add n local m
     in
   List.fold_left2 add_formal StringMap.empty fdecl.sformals
     (Array.to_list (L.params the_function)) in

   let lookup n locals = try StringMap.find n locals
                 with Not_found -> StringMap.find n global_vars
   in

   (* Construct code for an expression; return its value *)
   let rec expr builder locals ((_, e) : sexpr) = match e with
       SLiteral i  -> L.const_int i32_t i
     | SFliteral f -> L.const_float_of_string float_t f
     | SSliteral s -> L.build_global_stringptr s "str" builder
     | SBoolLit b  -> L.const_int i1_t (if b then 1 else 0)
     | SNoexpr     -> L.const_int i32_t 0
     | SId s       -> L.build_load (lookup s locals) s builder
```

```
    | SAssign (s, e) -> let e' = expr builder locals e in
                        ignore(L.build_store e' (lookup s locals) builder); e'
    | SBinop ((A.Float,_ ) as e1, op, e2) ->
      let e1' = expr builder locals e1
      and e2' = expr builder locals e2 in
      (match op with
        A.Add     -> L.build_fadd
      | A.Sub     -> L.build_fsub
      | A.Mul     -> L.build_fmul
      | A.Div     -> L.build_fdiv
      | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
      | A.Neq     -> L.build_fcmp L.Fcmp.One
      | A.Less    -> L.build_fcmp L.Fcmp.Olt
      | A.Leq     -> L.build_fcmp L.Fcmp.Ole
      | A.Greater -> L.build_fcmp L.Fcmp.Ogt
      | A.Geq     -> L.build_fcmp L.Fcmp.Oge
      | _         -> raise (Failure ("illegal usage of operator " ^
        (A.string_of_op op) ^ " on float"))
      ) e1' e2' "tmp" builder
    | SBinop (e1, op, e2) ->
      let e1' = expr builder locals e1
      and e2' = expr builder locals e2 in
      (match op with
        A.Add     -> L.build_add
      | A.Sub     -> L.build_sub
      | A.Mul     -> L.build_mul
      | A.Div     -> L.build_sdiv
      | A.And     -> L.build_and
      | A.Or      -> L.build_or
      | A.Equal   -> L.build_icmp L.Icmp.Eq
      | A.Neq     -> L.build_icmp L.Icmp.Ne
      | A.Less    -> L.build_icmp L.Icmp.Slt
      | A.Leq     -> L.build_icmp L.Icmp.Sle
      | A.Greater -> L.build_icmp L.Icmp.Sgt
      | A.Geq     -> L.build_icmp L.Icmp.Sge
      ) e1' e2' "tmp" builder
    | SUnop(op, ((t, _) as e)) ->
        let e' = expr builder locals e in
        (match op with
            A.Neg when t = A.Float -> L.build_fneg
          | A.Neg                  -> L.build_neg
          | A.Not                  -> L.build_not)
        e' "tmp" builder
    | SCall ("print", [e]) ->
        L.build_call printf_func [| int_format_str ; (expr builder locals e) |]
        "printf" builder
    | SCall ("Rgb", [r;g;b]) ->
        let build_t : L.lltype =
          L.function_type rgb_t [|i32_t; i32_t; i32_t;|] in
            let build_func : L.llvalue =
              L.declare_function "Rgb" build_t the_module in
        L.build_call build_func [| expr builder locals r;
```

```
            expr builder locals g; expr builder locals b; |]
          "Rgb" builder
  | SCall("Pointer", [x;y;color;angle]) ->
      let x' = expr builder locals x
      and y' = expr builder locals y
      and color' = expr builder locals color
      and angle' = expr builder locals angle in
      L.build_call ptrcons_fun [| x' ; y' ; color' ; angle' |]
          "Pointer" builder
  | SCall("Canvas", [x;y]) ->
      let x' = expr builder locals x
      and y' = expr builder locals y in
      L.build_call canvascons_fun [| x' ; y' |]
          "Canvas" builder
  | SCall("save", [can;filename]) ->
      let can' = expr builder locals can
      and filename' = expr builder locals filename in
      L.build_call savecons_fun [| can' ; filename' |]
          "save" builder
  | SCall("pixel", [can;color;x;y]) ->
    let can' = expr builder locals can
    and color' = expr builder locals color
    and x' = expr builder locals x
    and y' = expr builder locals y in
    L.build_call pixelcons_fun [| can';color';x';y' |]
        "pixel" builder
  | SCall ("get_rgb_r", [rgb;]) ->
    let build_t : L.lltype =
      L.function_type i32_t [|rgb_t;|] in
        let build_func : L.llvalue =
          L.declare_function "get_rgb_r" build_t the_module in
      L.build_call build_func [| expr builder locals rgb|]
        "get_rgb_r" builder
  | SCall ("get_rgb_g", [rgb;]) ->
    let build_t : L.lltype =
      L.function_type i32_t [|rgb_t;|] in
        let build_func : L.llvalue =
          L.declare_function "get_rgb_g" build_t the_module in
      L.build_call build_func [| expr builder locals rgb|]
        "get_rgb_g" builder
  | SCall ("get_rgb_b", [rgb;]) ->
    let build_t : L.lltype =
      L.function_type i32_t [|rgb_t;|] in
        let build_func : L.llvalue =
          L.declare_function "get_rgb_b" build_t the_module in
      L.build_call build_func [| expr builder locals rgb|]
        "get_rgb_b" builder
  | SCall ("get_pointer_x", [pointer;]) ->
    let build_t : L.lltype =
      L.function_type i32_t [|pointer_t;|] in
        let build_func : L.llvalue =
          L.declare_function "get_pointer_x" build_t the_module in
```

```
      L.build_call build_func [| expr builder locals pointer|]
        "get_pointer_x" builder
  | SCall ("get_pointer_y", [pointer;]) ->
    let build_t : L.lltype =
      L.function_type i32_t [|pointer_t;|] in
        let build_func : L.llvalue =
          L.declare_function "get_pointer_y" build_t the_module in
      L.build_call build_func [| expr builder locals pointer|]
        "get_pointer_y" builder
  | SCall ("set_pointer_color", [pointer;rgb]) ->
    let build_t : L.lltype =
      L.function_type pointer_t [|pointer_t;rgb_t|] in
        let build_func : L.llvalue =
          L.declare_function "set_pointer_color" build_t the_module in
      L.build_call build_func [| expr builder locals pointer ; expr builder locals rgb |]
        "set_pointer_xy" builder
  | SCall ("get_canvas_x", [canvas;]) ->
    let build_t : L.lltype =
      L.function_type i32_t [|canvas_t;|] in
        let build_func : L.llvalue =
          L.declare_function "get_canvas_x" build_t the_module in
      L.build_call build_func [| expr builder locals canvas|]
        "get_canvas_x" builder
  | SCall ("get_canvas_y", [canvas;]) ->
    let build_t : L.lltype =
      L.function_type i32_t [|canvas_t;|] in
        let build_func : L.llvalue =
          L.declare_function "get_canvas_y" build_t the_module in
      L.build_call build_func [| expr builder locals canvas|]
        "get_canvas_y" builder
  | SCall ("sine", [angle;]) ->
    let angle' = expr builder locals angle in
    L.build_call sinecons_fun [| angle';|]
        "sine" builder
  | SCall ("cosine", [angle;]) ->
    let angle' = expr builder locals angle in
    L.build_call cosinecons_fun [| angle';|]
        "cosine" builder
  | SCall ("tangent", [angle;]) ->
    let angle' = expr builder locals angle in
    L.build_call tangentcons_fun [| angle';|]
        "tangent" builder
  | SCall ("mod", [val1; val2;]) ->
    let val1' = expr builder locals val1
    and val2' = expr builder locals val2 in
    L.build_call modcons_fun [|val1';val2'|]
        "mod" builder
  | SCall ("floors", [val1;]) ->
    let val1' = expr builder locals val1 in
    L.build_call floorscons_fun [|val1';|]
        "floors" builder
  | SCall ("getRise", [distance;angle;]) ->
```

```
      let distance' = expr builder locals distance
      and angle' = expr builder locals angle in
      L.build_call getRisecons_fun [|distance';angle';|]
          "getRise" builder
    | SCall ("getRun", [distance;angle;]) ->
      let distance' = expr builder locals distance
      and angle' = expr builder locals angle in
      L.build_call getRuncons_fun [|distance';angle';|]
          "getRun" builder
    | SCall (fname, args) ->
      let (ldev, sfd) = StringMap.find fname function_decls in
      let actuals = List.rev (List.map (fun e -> expr builder locals e)
          (List.rev args)) in
      let ret = (match sfd.styp with
          A.Void -> ""
        | _-> fname^"_ret") in
      L.build_call ldev (Array.of_list actuals) ret builder
  in

(* LLVM insists each basic block end with exactly one "terminator"
     instruction that transfers control.  This function runs "instr builder"
     if the current block does not already have a terminator.  Used,
     e.g., to handle the "fall off the end of the function" case. *)
  let add_terminal builder instr =
    match L.block_terminator (L.insertion_block builder) with
      Some _ -> ()
    | None -> ignore (instr builder) in

  (* Build the code for the given statement; return the builder for
     the statement's successor (i.e., the next instruction will be built
     after the one generated by this call) *)

  let rec stmt builder locals = function
      SBlock sl -> List.fold_left (fun (b, lv) s -> stmt b lv s) (builder,locals) sl
    | SVar (ty, id) ->
      let local_var = L.build_alloca (ltype_of_typ ty) id builder in
      let locals = StringMap.add id local_var locals in
      (builder, locals)
    | SVarAssign (ty, id, v) ->
      let local_var = L.build_alloca (ltype_of_typ ty) id builder in
      let locals = StringMap.add id local_var locals in
        ignore (expr builder locals (ty,SAssign(id, v))); (builder, locals)
    | SExpr e -> ignore(expr builder locals e); (builder,locals)
    | SReturn e -> ignore(match fdecl.styp with
                          (* Special "return nothing" instr *)
                          A.Void -> L.build_ret_void builder
                          (* Build return statement *)
                        | _ -> L.build_ret (expr builder locals e) builder );
                  (builder,locals)
    | SWhile (predicate, body) ->
      let pred_bb = L.append_block context "while" the_function in
      ignore(L.build_br pred_bb builder);
```

```
      let body_bb = L.append_block context "while_body" the_function in
      add_terminal (fst (stmt (L.builder_at_end context body_bb) locals body))
      (L.build_br pred_bb);
      let pred_builder = L.builder_at_end context pred_bb in
      let bool_val = expr pred_builder locals predicate in
      let merge_bb = L.append_block context "merge" the_function in
      ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
      (L.builder_at_end context merge_bb, locals)
    | SIf (predicate, then_stmt, else_stmt) ->
    let bool_val = expr builder locals predicate in
    let merge_bb = L.append_block context "merge" the_function in
    let build_br_merge = L.build_br merge_bb in (* partial function *)
    let then_bb = L.append_block context "then" the_function in
      add_terminal (fst (stmt (L.builder_at_end context then_bb) locals then_stmt))
        build_br_merge;
    let else_bb = L.append_block context "else" the_function in
      add_terminal (fst (stmt (L.builder_at_end context else_bb) locals else_stmt))
        build_br_merge;
    ignore(L.build_cond_br bool_val then_bb else_bb builder);
    (L.builder_at_end context merge_bb, locals)
  in


  (* Build the code for each statement in the function *)
  let (builder,_) = stmt builder local_vars (SBlock fdecl.sbody) in
  (* Add a return if the last block falls off the end *)
    add_terminal builder (match fdecl.styp with
        A.Void -> L.build_ret_void
      | A.Float -> L.build_ret (L.const_float float_t 0.0)
      | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
in

List.iter build_function_body functions;
the_module
```

9.7 Makefile

```
# "make test" Compiles everything and runs the regression tests

.PHONY : test
test : all testall.sh
   ./testall.sh

# "make all" builds the executable as well as the "printbig" library designed
# to test linking external code

.PHONY : all
all : reptile.native rgb canvas pointer png pixelator trig# printbig.o
```

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

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

# "make clean" removes all generated files

.PHONY : clean
clean :
   ocamlbuild -clean
   rm -rf testall.log ocamlllvm *.diff *.o

# Testing the "printbig" example

# printbig : printbig.c
#   cc -o printbig -DBUILD_TEST printbig.c

rgb : rgb.c
   gcc -c rgb.c -lm

canvas: canvas.c
   gcc -c canvas.c -lm

pointer: pointer.c
   gcc -c pointer.c -lm

png: png.c
   gcc -c png.c -lm

pixelator: pixelator.c
   gcc -c pixelator.c -lm

trig: trig.c
   gcc -c trig.c -lm

# Building the tarball

TESTS = \
   helloworld float void assign arithmetic if if2 if3 if4 if5 if6 \
   else string comments create_save fib field pixel pointer rgb \
   while and or not canvas

FAILS = \
   helloworld float if dead dead2 func func2 func3 func4 func5 func6 \
   func7 expr global if2 nomain return
```

```
TESTFILES = $(TESTS:%=test-%.rt) $(TESTS:%=test-%.out) \
       $(FAILS:%=fail-%.rt) $(FAILS:%=fail-%.err)

TARFILES = ast.ml sast.ml codegen.ml Makefile _tags reptile.ml parser.mly \
   README scanner.mll semant.ml testall.sh \
   printbig.c \
   Dockerfile \
   $(TESTFILES:%=tests/%)

reptile.tar.gz : $(TARFILES)
   cd .. && tar czf reptile/reptile.tar.gz \
       $(TARFILES:%=reptile/%)
```

9.8 Test All Script File - testall.sh

```
#!/bin/sh

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

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

# Path to the LLVM compiler
LLC="llc"

# Path to the C compiler
CC="cc"

# Path to the reptile compiler.  Usually "./reptile.native"
# Try "_build/reptile.native" if ocamlbuild was unable to create a symbolic link.
REPTILE="./reptile.native"
#REPTILE="_build/reptile.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] [.rt files]"
```

```
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}

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

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

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

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

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

```
   echo -n "$basename..."

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

   generatedfiles=""

   generatedfiles="$generatedfiles ${basename}.ll ${basename}.s ${basename}.exe
${basename}.out" &&
   Run "$REPTILE" "$1" ">" "${basename}.ll" &&
   Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">" "${basename}.s" &&
   Run "$CC" "-o" "${basename}.exe" "${basename}.s" "rgb.o" "pointer.o" "canvas.o"
"pixelator.o" "png.o" "trig.o" "-lm" &&
   Run "./${basename}.exe" > "${basename}.out" &&
   Compare ${basename}.out ${reffile}.out ${basename}.diff

   # Report the status and clean up the generated files

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

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

   echo -n "$basename..."

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

   generatedfiles=""

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

   # Report the status and clean up the generated files

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

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

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

shift `expr $OPTIND - 1`

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

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

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

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

```
done

exit $globalerror
```

9.9 C Files

9.9.1 rgb.c

```
#include<stdio.h>
#include <stdlib.h>
#include<math.h>
#include "types.h"

struct Rgb {
  int r;
  int g;
  int b;
};

struct rgb* Rgb(int r, int g, int b) {
   struct rgb *color = malloc(sizeof(struct rgb));
   color->r = r;
   color->g = g;
   color->b = b;
   return color;
}

int get_rgb_r(struct rgb* rgb) {
   return rgb->r;
}
int get_rgb_g(struct rgb* rgb) {
   return rgb->g;
}
int get_rgb_b(struct rgb* rgb) {
   return rgb->b;
}
```

9.9.2 pointer.c

```
#include<stdio.h>
#include<math.h>
#include <stdlib.h>
#include "types.h"

struct Pointer {
   int x;
   int y;
```

```
   struct rgb* color;
   float angle;
};

struct pointer* Pointer(int x, int y, struct rgb* color, float angle) {
   struct pointer *point = malloc(sizeof(struct pointer));
   point->x = x;
   point->y = y;
   point->color = *color;
   point->angle = angle;
   return point;
}

int get_pointer_x(struct pointer* pointer) {
   return pointer->x;
}

int get_pointer_y(struct pointer* pointer) {
   return pointer->y;
}

struct pointer* set_pointer_color(struct pointer* pointer, struct rgb* rgb) {
   return Pointer(pointer->x, pointer->y, rgb, pointer->angle);
}
```

9.9.3 canvas.c

```
#include<stdio.h>
#include<math.h>
#include <stdlib.h>
#include "types.h"
#include "png.h"

struct Canvas {
   int x;
   int y;
   libattopng_t *png;
};

struct canvas* Canvas(int x, int y) {
   struct canvas *can = malloc(sizeof(struct canvas));
   can->x = x;
   can->y = y;
   can->png = libattopng_new(x, y, PNG_RGBA);
   return can;
}

int get_canvas_x(struct canvas* canvas) {
   return canvas->x;
}
```

```
int get_canvas_y(struct canvas* canvas) {
   return canvas->y;
}
```

9.9.4 types.h

```
#include<stdio.h>
#include<math.h>
#include "png.h"


typedef struct rgb {
   int r;
   int g;
   int b;
} rgb;

typedef struct pointer {
   int x;
   int y;
   struct rgb color;
   float angle;
} pointer;

struct canvas {
   int x;
   int y;
   libattopng_t* png;
};

struct rgb* Rgb(int r, int g, int b);
struct canvas* Canvas(int x, int y);
struct pointer* Pointer(int x, int y, struct rgb* color, float angle);
int get_rgb_r(struct rgb* rgb);
int get_rgb_g(struct rgb* rgb);
int get_rgb_b(struct rgb* rgb);
int get_pointer_x(struct pointer* pointer);
int get_pointer_y(struct pointer* pointer);
struct pointer* set_pointer_color(struct pointer* pointer, struct rgb* rgb);
int get_canvas_x(struct canvas* canvas);
int get_canvas_y(struct canvas* canvas);
```

9.10 PNG Rendering

9.10.1 png.c

```
#include "png.h"
#include <stdlib.h>
#include <string.h>

#define LIBATTOPNG_ADLER_BASE 65521

static const uint32_t libattopng_crc32[256] = {
	0x00000000, 0x77073096, 0xee0e612c, 0x990951ba, 0x076dc419, 0x706af48f, 0xe963a535,
0x9e6495a3, 0x0edb8832,
	0x79dcb8a4, 0xe0d5e91e, 0x97d2d988, 0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91,
0x1db71064, 0x6ab020f2,
	0xf3b97148, 0x84be41de, 0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7, 0x136c9856,
0x646ba8c0, 0xfd62f97a,
	0x8a65c9ec, 0x14015c4f, 0x63066cd9, 0xfa0f3d63, 0x8d080df5, 0x3b6e20c8, 0x4c69105e,
0xd56041e4, 0xa2677172,
	0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b, 0x35b5a8fa, 0x42b2986c, 0xdbbbc9d6,
0xacbcf940, 0x32d86ce3,
	0x45df5c75, 0xdcd60dcf, 0xabd13d59, 0x26d930ac, 0x51de003a, 0xc8d75180, 0xbfd06116,
0x21b4f4b5, 0x56b3c423,
	0xcfba9599, 0xb8bda50f, 0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924, 0x2f6f7c87,
0x58684c11, 0xc1611dab,
	0xb6662d3d, 0x76dc4190, 0x01db7106, 0x98d220bc, 0xefd5102a, 0x71b18589, 0x06b6b51f,
0x9fbfe4a5, 0xe8b8d433,
	0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818, 0x7f6a0dbb, 0x086d3d2d, 0x91646c97,
0xe6635c01, 0x6b6b51f4,
	0x1c6c6162, 0x856530d8, 0xf262004e, 0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457,
0x65b0d9c6, 0x12b7e950,
	0x8bbeb8ea, 0xfcb9887c, 0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65, 0x4db26158,
0x3ab551ce, 0xa3bc0074,
	0xd4bb30e2, 0x4adfa541, 0x3dd895d7, 0xa4d1c46d, 0xd3d6f4fb, 0x4369e96a, 0x346ed9fc,
0xad678846, 0xda60b8d0,
	0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd0d7cc9, 0x5005713c, 0x270241aa, 0xbe0b1010,
0xc90c2086, 0x5768b525,
	0x206f85b3, 0xb966d409, 0xce61e49f, 0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4,
0x59b33d17, 0x2eb40d81,
	0xb7bd5c3b, 0xc0ba6cad, 0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a, 0xead54739,
0x9dd277af, 0x04db2615,
	0x73dc1683, 0xe3630b12, 0x94643b84, 0x0d6d6a3e, 0x7a6a5aa8, 0xe40ecf0b, 0x9309ff9d,
0x0a00ae27, 0x7d079eb1,
	0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x806567cb, 0x196c3671,
0x6e6b06e7, 0xfed41b76,
	0x89d32be0, 0x10da7a5a, 0x67dd4acc, 0xf9b9df6f, 0x8ebeeff9, 0x17b7be43, 0x60b08ed5,
0xd6d6a3e8, 0xa1d1937e,
	0x38d8c2c4, 0x4fdff252, 0xd1bb67f1, 0xa6bc5767, 0x3fb506dd, 0x48b2364b, 0xd80d2bda,
0xaf0a1b4c, 0x36034af6,
	0x41047a60, 0xdf60efc3, 0xa867df55, 0x316e8eef, 0x4669be79, 0xcb61b38c, 0xbc66831a,
0x256fd2a0, 0x5268e236,
	0xcc0c7795, 0xbb0b4703, 0x220216b9, 0x5505262f, 0xc5ba3bbe, 0xb2bd0b28, 0x2bb45a92,
0x5cb36a04, 0xc2d7ffa7,
	0xb5d0cf31, 0x2cd99e8b, 0x5bdeae1d, 0x9b64c2b0, 0xec63f226, 0x756aa39c, 0x026d930a,
0x9c0906a9, 0xeb0e363f,
	0x72076785, 0x05005713, 0x95bf4a82, 0xe2b87a14, 0x7bb12bae, 0x0cb61b38, 0x92d28e9b,
```

```
0xe5d5be0d, 0x7cdcefb7,
       0x0bdbdf21, 0x86d3d2d4, 0xf1d4e242, 0x68ddb3f8, 0x1fda836e, 0x81be16cd, 0xf6b9265b,
0x6fb077e1, 0x18b74777,
       0x88085ae6, 0xff0f6a70, 0x66063bca, 0x11010b5c, 0x8f659eff, 0xf862ae69, 0x616bffd3,
0x166ccf45, 0xa00ae278,
       0xd70dd2ee, 0x4e048354, 0x3903b3c2, 0xa7672661, 0xd06016f7, 0x4969474d, 0x3e6e77db,
0xaed16a4a, 0xd9d65adc,
       0x40df0b66, 0x37d83bf0, 0xa9bcae53, 0xdebb9ec5, 0x47b2cf7f, 0x30b5ffe9, 0xbdbdf21c,
0xcabac28a, 0x53b39330,
       0x24b4a3a6, 0xbad03605, 0xcdd70693, 0x54de5729, 0x23d967bf, 0xb3667a2e, 0xc4614ab8,
0x5d681b02, 0x2a6f2b94,
       0xb40bbe37, 0xc30c8ea1, 0x5a05df1b, 0x2d02ef8d
};

/* ------------------------------------------------------------------------ */
libattopng_t *libattopng_new(size_t width, size_t height, libattopng_type_t type) {
   libattopng_t *png;
   if (SIZE_MAX / 4 / width < height) {
       /* ensure no type leads to an integer overflow */
       return NULL;
   }
   png = (libattopng_t *) calloc(sizeof(libattopng_t), 1);
   png->width = width;
   png->height = height;
   png->capacity = width * height;
   png->palette_length = 0;
   png->palette = NULL;
   png->out = NULL;
   png->out_capacity = 0;
   png->out_pos = 0;
   png->type = type;
   png->stream_x = 0;
   png->stream_y = 0;

   if (type == PNG_PALETTE) {
       png->palette = (uint32_t *) calloc(256, sizeof(uint32_t));
       if (!png->palette) {
           free(png);
           return NULL;
       }
       png->bpp = 1;
   } else if (type == PNG_GRAYSCALE) {
       png->bpp = 1;
   } else if (type == PNG_GRAYSCALE_ALPHA) {
       png->capacity *= 2;
       png->bpp = 2;
   } else if (type == PNG_RGB) {
       png->capacity *= 4;
       png->bpp = 3;
   } else if (type == PNG_RGBA) {
       png->capacity *= 4;
       png->bpp = 4;
```

```
    }

    png->data = (char *) calloc(png->capacity, 1);
    if (!png->data) {
        free(png->palette);
        free(png);
        return NULL;
    }
    return png;
}

/* ---------------------------------------------------------------------- */
int libattopng_set_palette(libattopng_t *png, uint32_t *palette, size_t length) {
    if (length > 256) {
        return 1;
    }
    memmove(png->palette, palette, length * sizeof(uint32_t));
    png->palette_length = length;
    return 0;
}

/* ---------------------------------------------------------------------- */
void libattopng_set_pixel(libattopng_t *png, size_t x, size_t y, uint32_t color) {
    if (!png || x >= png->width || y >= png->height) {
        return;
    }
    if (png->type == PNG_PALETTE || png->type == PNG_GRAYSCALE) {
        png->data[x + y * png->width] = (char) (color & 0xff);
    } else if (png->type == PNG_GRAYSCALE_ALPHA) {
        ((uint16_t *) png->data)[x + y * png->width] = (uint16_t) (color & 0xffff);
    } else {
        ((uint32_t *) png->data)[x + y * png->width] = color;
    }
}

/* ---------------------------------------------------------------------- */
uint32_t libattopng_get_pixel(libattopng_t* png, size_t x, size_t y) {
    uint32_t pixel = 0;
    if (!png || x >= png->width || y >= png->height) {
        return pixel;
    }
    if (png->type == PNG_PALETTE || png->type == PNG_GRAYSCALE) {
        pixel = (uint32_t)(png->data[x + y * png->width] & 0xff);
    } else if (png->type == PNG_GRAYSCALE_ALPHA) {
        pixel = (uint32_t)(((uint16_t *) png->data)[x + y * png->width] & 0xffff);
    } else {
        pixel = ((uint32_t *) png->data)[x + y * png->width];
    }
    return pixel;
}

/* ---------------------------------------------------------------------- */
```

```
void libattopng_start_stream(libattopng_t* png, size_t x, size_t y) {
   if(!png || x >= png->width || y >= png->height) {
       return;
   }
   png->stream_x = x;
   png->stream_y = y;
}

/* ---------------------------------------------------------------------- */
void libattopng_put_pixel(libattopng_t* png, uint32_t color) {
   size_t x, y;
   if(!png) {
       return;
   }
   x = png->stream_x;
   y = png->stream_y;
   if (png->type == PNG_PALETTE || png->type == PNG_GRAYSCALE) {
       png->data[x + y * png->width] = (char) (color & 0xff);
   } else if (png->type == PNG_GRAYSCALE_ALPHA) {
       ((uint16_t *) png->data)[x + y * png->width] = (uint16_t) (color & 0xffff);
   } else {
       ((uint32_t *) png->data)[x + y * png->width] = color;
   }
   x++;
   if(x >= png->width) {
       x = 0;
       y++;
       if(y >= png->height) {
           y = 0;
       }
   }
   png->stream_x = x;
   png->stream_y = y;
}

/* ---------------------------------------------------------------------- */
static uint32_t libattopng_swap32(uint32_t num) {
   return ((num >> 24) & 0xff) |
          ((num << 8) & 0xff0000) |
          ((num >> 8) & 0xff00) |
          ((num << 24) & 0xff000000);
}

/* ---------------------------------------------------------------------- */
static uint32_t libattopng_crc(const unsigned char *data, size_t len, uint32_t crc) {
   size_t i;
   for (i = 0; i < len; i++) {
       crc = libattopng_crc32[(crc ^ data[i]) & 255] ^ (crc >> 8);
   }
   return crc;
}
```

```
/* ---------------------------------------------------------------------- */
static void libattopng_out_raw_write(libattopng_t *png, const char *data, size_t len) {
   size_t i;
   for (i = 0; i < len; i++) {
       png->out[png->out_pos++] = data[i];
   }
}

/* ---------------------------------------------------------------------- */
static void libattopng_out_raw_uint(libattopng_t *png, uint32_t val) {
   *(uint32_t *) (png->out + png->out_pos) = val;
   png->out_pos += 4;
}

/* ---------------------------------------------------------------------- */
static void libattopng_out_raw_uint16(libattopng_t *png, uint16_t val) {
   *(uint16_t *) (png->out + png->out_pos) = val;
   png->out_pos += 2;
}

/* ---------------------------------------------------------------------- */
static void libattopng_out_raw_uint8(libattopng_t *png, uint8_t val) {
   *(uint8_t *) (png->out + png->out_pos) = val;
   png->out_pos++;
}

/* ---------------------------------------------------------------------- */
static void libattopng_new_chunk(libattopng_t *png, const char *name, size_t len) {
   png->crc = 0xffffffff;
   libattopng_out_raw_uint(png, libattopng_swap32((uint32_t) len));
   png->crc = libattopng_crc((const unsigned char *) name, 4, png->crc);
   libattopng_out_raw_write(png, name, 4);
}

/* ---------------------------------------------------------------------- */
static void libattopng_end_chunk(libattopng_t *png) {
   libattopng_out_raw_uint(png, libattopng_swap32(~png->crc));
}

/* ---------------------------------------------------------------------- */
static void libattopng_out_uint32(libattopng_t *png, uint32_t val) {
   png->crc = libattopng_crc((const unsigned char *) &val, 4, png->crc);
   libattopng_out_raw_uint(png, val);
}

/* ---------------------------------------------------------------------- */
static void libattopng_out_uint16(libattopng_t *png, uint16_t val) {
   png->crc = libattopng_crc((const unsigned char *) &val, 2, png->crc);
   libattopng_out_raw_uint16(png, val);
}

/* ---------------------------------------------------------------------- */
```

```
static void libattopng_out_uint8(libattopng_t *png, uint8_t val) {
   png->crc = libattopng_crc((const unsigned char *) &val, 1, png->crc);
   libattopng_out_raw_uint8(png, val);
}

/* ------------------------------------------------------------------------ */
static void libattopng_out_write(libattopng_t *png, const char *data, size_t len) {
   png->crc = libattopng_crc((const unsigned char *) data, len, png->crc);
   libattopng_out_raw_write(png, data, len);
}

/* ------------------------------------------------------------------------ */
static void libattopng_out_write_adler(libattopng_t *png, unsigned char data) {
   libattopng_out_write(png, (char *) &data, 1);
   png->s1 = (uint16_t) ((png->s1 + data) % LIBATTOPNG_ADLER_BASE);
   png->s2 = (uint16_t) ((png->s2 + png->s1) % LIBATTOPNG_ADLER_BASE);
}

/* ------------------------------------------------------------------------ */
static void libattopng_pixel_header(libattopng_t *png, size_t offset, size_t bpl) {
   if (offset > bpl) {
       /* not the last line */
       libattopng_out_write(png, "\0", 1);
       libattopng_out_uint16(png, (uint16_t) bpl);
       libattopng_out_uint16(png, (uint16_t) ~bpl);
   } else {
       /* last line */
       libattopng_out_write(png, "\1", 1);
       libattopng_out_uint16(png, (uint16_t) offset);
       libattopng_out_uint16(png, (uint16_t) ~offset);
   }
}

/* ------------------------------------------------------------------------ */
char *libattopng_get_data(libattopng_t *png, size_t *len) {
   size_t index, bpl, raw_size, size, p, pos, corr;
   unsigned char *pixel;
   if (!png) {
       return NULL;
   }
   if (png->out) {
       /* delete old output if any */
       free(png->out);
   }
   png->out_capacity = png->capacity + 4096 * 8 + png->width * png->height;
   png->out = (char *) calloc(png->out_capacity, 1);
   png->out_pos = 0;
   if (!png->out) {
       return NULL;
   }

   libattopng_out_raw_write(png, "\211PNG\r\n\032\n", 8);
```

```
    /* IHDR */
    libattopng_new_chunk(png, "IHDR", 13);
    libattopng_out_uint32(png, libattopng_swap32((uint32_t) (png->width)));
    libattopng_out_uint32(png, libattopng_swap32((uint32_t) (png->height)));
    libattopng_out_uint8(png, 8); /* bit depth */
    libattopng_out_uint8(png, (uint8_t) png->type);
    libattopng_out_uint8(png, 0); /* compression */
    libattopng_out_uint8(png, 0); /* filter */
    libattopng_out_uint8(png, 0); /* interlace method */
    libattopng_end_chunk(png);

    /* palette */
    if (png->type == PNG_PALETTE) {
        char entry[3];
        size_t s = png->palette_length;
        if (s < 16) {
            s = 16; /* minimum palette length */
        }
        libattopng_new_chunk(png, "PLTE", 3 * s);
        for (index = 0; index < s; index++) {
            entry[0] = (char) (png->palette[index] & 255);
            entry[1] = (char) ((png->palette[index] >> 8) & 255);
            entry[2] = (char) ((png->palette[index] >> 16) & 255);
            libattopng_out_write(png, entry, 3);
        }
        libattopng_end_chunk(png);

        /* transparency */
        libattopng_new_chunk(png, "tRNS", s);
        for (index = 0; index < s; index++) {
            entry[0] = (char) ((png->palette[index] >> 24) & 255);
            libattopng_out_write(png, entry, 1);
        }
        libattopng_end_chunk(png);
    }

    /* data */
    bpl = 1 + png->bpp * png->width;
    if(bpl >= 65536) {
        fprintf(stderr, "[libattopng] ERROR: maximum supported width for this type of PNG is
%d pixel\n", (int)(65535 / png->bpp));
        return NULL;
    }
    raw_size = png->height * bpl;
    size = 2 + png->height * (5 + bpl) + 4;
    libattopng_new_chunk(png, "IDAT", size);
    libattopng_out_write(png, "\170\332", 2);

    pixel = (unsigned char *) png->data;
    png->s1 = 1;
    png->s2 = 0;
```

```
    index = 0;
    if (png->type == PNG_RGB) {
        corr = 1;
    } else {
        corr = 0;
    }
    for (pos = 0; pos < png->width * png->height; pos++) {
        if (index == 0) {
            /* line header */
            libattopng_pixel_header(png, raw_size, bpl);
            libattopng_out_write_adler(png, 0); /* no filter */
            raw_size--;
        }

        /* pixel */
        for (p = 0; p < png->bpp; p++) {
            libattopng_out_write_adler(png, *pixel);
            pixel++;
        }
        pixel += corr;

        raw_size -= png->bpp;
        index = (index + 1) % png->width;
    }
    /* checksum */
    png->s1 %= LIBATTOPNG_ADLER_BASE;
    png->s2 %= LIBATTOPNG_ADLER_BASE;
    libattopng_out_uint32(png, libattopng_swap32((uint32_t) ((png->s2 << 16) | png->s1)));
    libattopng_end_chunk(png);

    /* end of image */
    libattopng_new_chunk(png, "IEND", 0);
    libattopng_end_chunk(png);

    if (len) {
        *len = png->out_pos;
    }
    return png->out;
}

/* ------------------------------------------------------------------------ */
int libattopng_save(libattopng_t *png, const char *filename) {
    size_t len;
    FILE* f;
    char *data = libattopng_get_data(png, &len);
    if (!data) {
        return 1;
    }
    f = fopen(filename, "wb");
    if (!f) {
        return 1;
    }
```

```
    if (fwrite(data, len, 1, f) != 1) {
        fclose(f);
        return 1;
    }
    fclose(f);
    return 0;
}

/* ---------------------------------------------------------------------- */
void libattopng_destroy(libattopng_t *png) {
    if (!png) {
        return;
    }
    free(png->palette);
    png->palette = NULL;
    free(png->out);
    png->out = NULL;
    free(png->data);
    png->data = NULL;
    free(png);
}
```

9.10.2 png.h

```
/**
* @file libattopng.h
* @brief A minimal C library to write uncompressed PNG files.
*
* libattopng is a minimal C library to create uncompressed PNG images.
* It is cross-platform compatible, has no dependencies and a very small footprint.
* The library supports palette, grayscale as well as raw RGB images all with and without
transparency.
*
* @author Michael Schwarz
* @date 29 Jan 2017
*/

#ifndef _PNG_H_
#define _PNG_H_

#ifdef __cplusplus
extern "C" {
#endif

#include <stdio.h>
#include <stdint.h>

/**
* @brief PNG type.
*
```

```
* The type of PNG image. It determines how the pixels are stored.
*/
typedef enum {
   PNG_GRAYSCALE = 0,          /**< 256 shades of gray, 8bit per pixel  */
   PNG_RGB = 2,                /**< 24bit RGB values */
   PNG_PALETTE = 3,            /**< Up to 256 RGBA palette colors, 8bit per pixel */
   PNG_GRAYSCALE_ALPHA = 4,    /**< 256 shades of gray plus alpha channel, 16bit per pixel
*/
   PNG_RGBA = 6                /**< 24bit RGB values plus 8bit alpha channel */
} libattopng_type_t;


/**
* @brief Reference to a PNG image
*
* This struct holds the internal state of the PNG. The members should never be used
directly.
*/
typedef struct {
   libattopng_type_t type;      /**< File type */
   size_t capacity;             /**< Reserved memory for raw data */
   char *data;                  /**< Raw pixel data, format depends on type */
   uint32_t *palette;           /**< Palette for image */
   size_t palette_length;       /**< Entries for palette, 0 if unused */
   size_t width;                /**< Image width */
   size_t height;               /**< Image height */

   char *out;                   /**< Buffer to store final PNG */
   size_t out_pos;              /**< Current size of output buffer */
   size_t out_capacity;         /**< Capacity of output buffer */
   uint32_t crc;                /**< Currecnt CRC32 checksum */
   uint16_t s1;                 /**< Helper variables for Adler checksum */
   uint16_t s2;                 /**< Helper variables for Adler checksum */
   size_t bpp;                  /**< Bytes per pixel */

   size_t stream_x;             /**< Current x coordinate for pixel streaming */
   size_t stream_y;             /**< Current y coordinate for pixel streaming */
} libattopng_t;


/**
* @function libattopng_new
*
* @brief Create a new, empty PNG image to be used with all other functions.
*
* @param width The width of the image in  pixels
* @param height The height of the image in pixels
* @param type The type of image. Possible values are
*                   - PNG_GRAYSCALE (8bit grayscale),
*                   - PNG_GRAYSCALE_ALPHA (8bit grayscale with 8bit alpha),
*                   - PNG_PALETTE (palette with up to 256 entries, each 32bit RGBA)
*                   - PNG_RGB (24bit RGB values)
```

```
*                  - PNG_RGBA (32bit RGB values with alpha)
* @return reference to a PNG image to be used with all other functions or NULL on error.
*          Possible errors are:
*               - Out of memory
*               - Width and height combined exceed the maximum integer size
* @note It's the callers responsibility to free the data structure.
*       See @ref libattopng_destroy
*/
libattopng_t *libattopng_new(size_t width, size_t height, libattopng_type_t type);


/**
* @function libattopng_destroy
*
* @brief Destroys the reference to a PNG image and free all associated memory.
*
* @param png Reference to the image
*
*/
void libattopng_destroy(libattopng_t *png);


/**
* @function libattopng_set_palette
*
* @brief Sets the image's palette if the image type is \ref PNG_PALETTE.
*
* @param png      Reference to the image
* @param palette  Color palette, each entry contains a 32bit RGBA value
* @param length   Number of palette entries
* @return 0 on success, 1 if the palette contained more than 256 entries
*/
int libattopng_set_palette(libattopng_t *png, uint32_t *palette, size_t length);


/**
* @function libattopng_set_pixel
*
* @brief Sets the pixel's color at the specified position
*
* @param png    Reference to the image
* @param x      X coordinate
* @param y      Y coordinate
* @param color The pixel value, depending on the type this is
*               - the 8bit palette index (\ref PNG_PALETTE)
*               - the 8bit gray value (\ref PNG_GRAYSCALE)
*               - a 16bit value where the lower 8bit are the gray value and
*                 the upper 8bit are the opacity (\ref PNG_GRAYSCALE_ALPHA)
*               - a 24bit RGB value (\ref PNG_RGB)
*               - a 32bit RGBA value (\ref PNG_RGBA)
* @note If the coordinates are not within the bounds of the image,
*       the functions does nothing.
```

```
*/
void libattopng_set_pixel(libattopng_t *png, size_t x, size_t y, uint32_t color);


/**
* @function libattopng_get_pixel
*
* @brief Returns the pixel's color at the specified position
*
* @param png   Reference to the image
* @param x     X coordinate
* @param y     Y coordinate
* @return      The pixel value, depending on the type this is
*              - the 8bit palette index (\ref PNG_PALETTE)
*              - the 8bit gray value (\ref PNG_GRAYSCALE)
*              - a 16bit value where the lower 8bit are the gray value and
*                the upper 8bit are the opacity (\ref PNG_GRAYSCALE_ALPHA)
*              - a 24bit RGB value (\ref PNG_RGB)
*              - a 32bit RGBA value (\ref PNG_RGBA)
*              - 0 if the coordinates are out of bounds
*/
uint32_t libattopng_get_pixel(libattopng_t *png, size_t x, size_t y);


/**
* @function libattopng_start_stream
*
* @brief Set the start position for a batch of pixels
*
* @param png  Reference to the image
* @param x    X coordinate
* @param y    Y coordinate
*
* @see libattopng_put_pixel
*/
void libattopng_start_stream(libattopng_t *png, size_t x, size_t y);


/**
* @function libattopng_put_pixel
*
* @brief Sets the pixel of the current pixel within a stream and advances to the next pixel
*
* @param png   Reference to the image
* @param color The pixel value, depending on the type this is
*              - the 8bit palette index (\ref PNG_PALETTE)
*              - the 8bit gray value (\ref PNG_GRAYSCALE)
*              - a 16bit value where the lower 8bit are the gray value and
*                the upper 8bit are the opacity (\ref PNG_GRAYSCALE_ALPHA)
*              - a 24bit RGB value (\ref PNG_RGB)
*              - a 32bit RGBA value (\ref PNG_RGBA)
*/
```

```
void libattopng_put_pixel(libattopng_t *png, uint32_t color);


/**
* @function libattopng_get_data
*
* @brief Returns the image as PNG data stream
*
* @param png  Reference to the image
* @param len  The length of the data stream is written to this output parameter
* @return A reference to the PNG output stream
* @note The data stream is free'd when calling \ref libattopng_destroy and
*       must not be free'd be the caller
*/
char *libattopng_get_data(libattopng_t *png, size_t *len);


/**
* @function libattopng_save
*
* @brief Saves the image as a PNG file
*
* @param png      Reference to the image
* @param filename Name of the file
* @return 0 on success, 1 on error
*/
int libattopng_save(libattopng_t *png, const char *filename);


#ifdef __cplusplus
}
#endif
#endif
```

9.10.3 pixelator.c

```
#include "png.h"
#include "types.h"
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define RGBA(r, g, b, a) ((r) | ((g) << 8) | ((b) << 16) | ((a) << 24))

struct canvas* pixel(struct canvas* can, struct rgb* color, int x, int y) {

   libattopng_set_pixel(can->png, x, y, RGBA(get_rgb_r(color) & 255, get_rgb_g(color) & 255,
get_rgb_b(color) & 255, (255 )));
   return can;
```

```
}

void save(struct canvas* can, char *filename) {
   libattopng_save(can->png, filename);
   libattopng_destroy(can->png);
}
```

9.10.4 pixelator.h

```
#include <stdio.h>
#include <stdint.h>
#include "types.h"
#include "png.h"

#ifndef _PIXELATOR_H_
#define _PIXELATOR_H_
#endif


struct canvas* pixel(struct canvas* can, struct rgb* color, int x, int y);

void save(struct canvas* can, char *filename);
```

9.10.5 trig.c

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

#define PI 3.14159265


double sine(double angle) {
   double val = PI / 180.0;
   double sinval = sin(angle*val);
   return sinval;
}

double cosine(double angle) {
   double val = PI / 180.0;
   double cosval = cos(angle*val);
   return cosval;
}

double tangent(double angle) {
   double val = PI / 180.0;
   double tanval = tanh(angle*val);
```

```
   return tanval;
}

int mod(int val1, int val2) {
   return val1 % val2;
}

int floors(float val1) {
   int my_int;
   my_int = (int)val1;
   return my_int;
}

int getRise(int distance, double angle) {
   return (int) (distance * sin(angle));
}

int getRun(int distance, double angle) {
   return (int) (distance * cos(angle));
}
```

9.10.6 trig.h

```
#include <stdio.h>
#include <stdint.h>
#include <math.h>

#ifndef _TRIG_H_
#define _TRIG_H_
#endif

double sine(double angle);
double cosine(double angle);
double tangent(double angle);
int mod(int val1, int val2);
int floors(float val1);
int getRise(int distance, double angle);
int getRun(int distance, double angle)
```

9.11 Integration Test Suite

9.11.1 Integration Test Files - Negative Tests

Below are the failure cases of our integration tests. The expected error is listed prior to the test program which would produce the said error or output.

9.11.1.1 fail-dead.err

```
Fatal error: exception Failure("nothing may follow a return")
```

9.11.1.2 fail-dead.rt

```
int main()
{
 int i = 15;
 return i;
 i = 32;
}
```

9.11.1.3 fail-dead2.err

```
Fatal error: exception Failure("nothing may follow a return")
```

9.11.1.4 fail-dead2.rt

```
int main()
{
 int i = 15;
 return i;
 i = 32;
}
```

9.11.1.5 fail-expr.err

```
Fatal error: exception Failure("illegal binary operator ")
```

9.11.1.6 fail-expr.rt

```
void foo(int c, float d)
{
 int a = 0;
 float b = 0.0;
 int d = 1;
 float e = 1.1;
 b + a;
}

int main()
{
```

```
 return 0;
}
```

9.11.1.7 fail-float.err

```
Fatal error: exception Failure("return failure")
```

9.11.1.8 fail-float.rt

```
float main() {
   print(5);
   return 5;
}
```

9.11.1.9 fail-func.err

```
Fatal error: exception Failure("duplicate function bar")
```

9.11.1.10 fail-func.rt

```
int foo() {}

int bar() {}

int baz() {}

void bar() {}

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

9.11.1.11 fail-func2.err

```
Fatal error: exception Failure("duplicate formal a")
```

9.11.1.12 fail-func2.rt

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

void bar(int a, bool b, int a) {}

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

9.11.1.13 fail-func3.err

```
Fatal error: exception Failure("function print may not be defined")
```

9.11.1.14 fail-func3.rt

```
int foo() {}

void bar() {}

int print() {}

void baz() {}

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

9.11.1.15 fail-func4.err

```
Fatal error: exception Failure("illegal void local b")
```

9.11.1.16 fail-func4.rt

```
int foo() {}

int bar() {
 int a;
 void b;
 bool c;

 return 0;
}
```

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

9.11.1.17 fail-func5.err

```
Fatal error: exception Failure("calling failure")
```

9.11.1.18 fail-func5.rt

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

int main()
{
 foo(42, true);
 foo(42);
}
```

9.11.1.19 fail-func6.err

```
Fatal error: exception Failure("illegal argument foundvoid expected bool in bar()")
```

9.11.1.20 fail-func6.rt

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

void bar()
{
}

int main()
{
 foo(42, true);
 foo(42, bar());
}
```

9.11.1.21 fail-func7.err

```
Fatal error: exception Failure("illegal argument foundint expected bool in 42")
```

9.11.1.22 fail-func7.rt

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

int main()
{
 foo(42, true);
 foo(42, 42);
}
```

9.11.1.23 fail-global.err

```
Fatal error: exception Failure("duplicate global b")
```

9.11.1.24 fail-global.rt

```
int b;
bool c;
int a;
int b;

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

9.11.1.25 fail-helloworld.err

```
Fatal error: exception Failure("int_of_string")
```

9.11.1.26 fail-helloworld.rt

```
int main() {
   print(9876456789999999999999);
   return 0;
```

```
}
```

9.11.1.27 fail-if.err

```
Fatal error: exception Failure("expected Boolean expression in 42")
```

9.11.1.28 fail-if.rt

```
int main() {
   if (42) {
       print(4);
   }
   return 0;
}
```

9.11.1.29 fail-if2.err

```
Fatal error: exception Failure("undeclared identifier foo")
```

9.11.1.30 fail-if2.rt

```
int main()
{
 if (true) {
   foo;
 }
}
```

9.11.1.31 fail-nomain.err

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

9.11.1.32 fail-nomain.rt

```

```

9.11.1.33 fail-return.err

```
Fatal error: exception Failure("return failure")
```

9.11.1.34 fail-return.rt

```
int main()
{
 return 0.0;
}
```

9.11.2 Integration Test Files - Positive Tests

Below are the positive cases of our integration tests. The expected output is listed prior to the test program.

9.11.2.1 test-access.out

```
100
200
1
2
3
4
5
```

9.11.2.2 test-access.rt

```
int main() {
   Canvas can = Canvas(100,200);
   print(get_canvas_x(can));
   print(get_canvas_y(can));

   Rgb color = Rgb(1,2,3);
   int r = get_rgb_r(color);
   print(get_rgb_r(color));
   print(get_rgb_g(color));
   print(get_rgb_b(color));

   Pointer ptr = Pointer(4,5,color,5.5);
   int px = get_pointer_x(ptr);
   int py = get_pointer_y(ptr);
   print(px);
   print(py);
```

```
    Pointer ptr = set_pointer_color(ptr, Rgb(10,20,10));

    return 0;
}
```

9.11.2.3 test-and.out

```
9
```

9.11.2.4 test-and.rt

```
int main() {
    if (true && false) {
        print(4);
    }
    else {
        print(9);
    }
    return 0;
}
```

9.11.2.5 test-arithmetic.out

```
33
```

9.11.2.6 test-arithmetic.rt

```
int main() {
    print(30+3);
    return 0;
}
```

9.11.2.7 test-assign.out

```
3
5
```

9.11.2.8 test-assign.rt

```
int main() {
   int i = 3;
   print(i);
   i = i+2;
   print(i);
   return 0;
}
```

9.11.2.9 test-canvas.out

```

```

9.11.2.10 test-canvas.rt

```
int main() {
   Canvas can = Canvas(2, 3);
   return 0;
}
```

9.11.2.11 test-comments.out

```
5
```

9.11.2.12 test-comments.rt

```
int main() {
   /\ print(3);
   print(5);
   return 0;
}
```

9.11.2.13 test-create_save.out

```

```

9.11.2.14 test-create_save.rt

```
int main() {
   Canvas can = Canvas(800, 800);
```

```
    save(can, "myfile.png");
    return 0;
}
```

9.11.2.15 test-edwards.out (Refer to 9.12.1 edwardstest.png for PNG Output file)

```
```

9.11.2.16 test-edwards.rt

```
int main() {
    Canvas can = Canvas(300, 400);
    Rgb color = Rgb(0,0,0);
    pixel(can, color,0, 0);
    pixel(can, color,0, 1);
    pixel(can, color,0, 2);
    pixel(can, color,0, 3);
    pixel(can, color,0, 4);
    pixel(can, color,0, 5);

    ...

    pixel(can, color,399, 295);
    pixel(can, color,399, 296);
    pixel(can, color,399, 297);
    pixel(can, color,399, 298);
    pixel(can, color,399, 299);

    save(can, "edwardstest.png");
    return 0;
}
```

9.11.2.17 test-else.out

```
5
```

9.11.2.18 test-else.rt

```
int main() {
    if (0 > 1) {
        print(4);
    }
    else {
```

```
        print(5);
    }
    return 0;
}
```

9.11.2.19 test-fib.out

```
34
```

9.11.2.20 test-fib.rt

```
int main() {
    int m = fib(9);
    print(m);
    return 0;
}

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

9.11.2.21 test-float.out

```
5
```

9.11.2.22 test-float.rt

```
float f(float x) {
    return x;
}
int main() {
    print(5);
    f(5.5);
    return 0;
}
```

9.11.2.23 test-gcd.out

```
12
```

9.11.2.24 test-gcd.rt

```
int gcd(int a, int b) {
   if (b != 0) {
       return gcd(b, mod(a, b));
   }
   else {
       return a;
   }
}

int main() {
   int val = gcd(60, 48);
   print(val);
}
```

9.11.2.25 test-helloworld.out

```
5
```

9.11.2.26 test-helloworld.rt

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

9.11.2.27 test-if.out

```
4
```

9.11.2.28 test-if.rt

```
int main() {
   if (0 < 1) {
```

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

9.11.2.29 test-if2.out

```
4
```

9.11.2.30 test-if2.rt

```
int main() {
    if (1 > 0) {
        print(4);
    }
    return 0;
}
```

9.11.2.31 test-if3.out

```
4
```

9.11.2.32 test-if3.rt

```
int main() {
    if (1 == 1) {
        print(4);
    }
    return 0;
}
```

9.11.2.33 test-if4.out

```
4
```

9.11.2.34 test-if4.rt

```
int main() {
```

```
    if (1 != 2) {
        print(4);
    }
    return 0;
}
```

9.11.2.35 test-if5.out

```
4
```

9.11.2.36 test-if5.rt

```
int main() {
    if (1 <= 1) {
        print(4);
    }
    return 0;
}
```

9.11.2.37 test-if6.out

```
4
```

9.11.2.38 test-if6.rt

```
int main() {
    if (1 >= 1) {
        print(4);
    }
    return 0;
}
```

9.11.2.39 test-not.out

```
5
```

9.11.2.40 test-not.rt

```
int main() {
   if (!true) {
       print(4);
   }
   else {
       print(5);
   }
   return 0;
}
```

9.11.2.41 test-or.out

```
4
```

9.11.2.42 test-or.rt

```
int main() {
   if (true || false) {
       print(4);
   }
   return 0;
}
```

9.11.2.43 test-pixel.out

```

```

9.11.2.44 test-pixel.rt

```
int main() {
   Canvas can = Canvas(400, 400);
   Rgb color = Rgb(0,0,0);
   pixel(can, color, 200, 200);
   pixel(can, color, 201, 200);
   pixel(can, color, 202, 200);
   pixel(can, color, 203, 200);
   pixel(can, color, 204, 200);
   pixel(can, color, 205, 200);
   save(can, "pixeltest.png");
   return 0;
}
```

9.11.2.45 test-pointer.out

9.11.2.46 test-pointer.rt

```
int main() {
   Rgb color = Rgb(1,2,5);
   Pointer pt = Pointer(50, 60, color, 90.0);
   return 0;
}
```

9.11.2.47 test-rgb.out

9.11.2.48 test-rgb.rt

```
int main() {
   Rgb color = Rgb(1,2,3);
   return 0;
}
```

9.11.2.49 test-simple-tort.out (Refer to 9.12.2 simple_turtle_start.png for PNG Output file)

9.11.2.50 test-simple-tort.rt

```
int xcur;
int ycur;


int tortup(Canvas can, Rgb color, int distance) {
   int counter = 0;
   while(counter < distance) {
       pixel(can, color, xcur, ycur-counter);
       counter = counter + 1;
   }
   ycur = ycur - distance;
```

```
    return 0;
}

int tortdown(Canvas can, Rgb color, int distance) {
    int counter = 0;
    while(counter < distance) {
        pixel(can, color, xcur, ycur+counter);
        counter = counter + 1;
    }
    ycur = ycur + distance;
    return 0;
}

int tortright(Canvas can, Rgb color, int distance) {
    int counter = 0;
    while(counter < distance) {
        pixel(can, color, xcur+counter, ycur);
        counter = counter + 1;
    }
    xcur = xcur + distance;
    return 0;
}

int tortleft(Canvas can, Rgb color, int distance) {
    int counter = 0;
    while(counter < distance) {
        pixel(can, color, xcur-counter, ycur);
        counter = counter + 1;
    }
    xcur = xcur - distance;
    return 0;
}

int tortNW(Canvas can, Rgb color, float distance) {
    int counter = 0;
    float counter1 = 0.0;
    float step = distance * 0.707;
    while(counter1 < step) {
        pixel(can, color, xcur-counter, ycur-counter);
        counter1 = counter1 + 1.0;
        counter = counter + 1;
    }
    xcur = xcur - counter;
    ycur = ycur - counter;
    return 0;
}

int tortNE(Canvas can, Rgb color, float distance) {
    int counter = 0;
    float counter1 = 0.0;
    float step = distance * 0.707;
    while(counter1 < step) {
```

```
        pixel(can, color, xcur+counter, ycur-counter);
        counter1 = counter1 + 1.0;
        counter = counter + 1;
    }
    xcur = xcur + counter;
    ycur = ycur - counter;
    return 0;
}

int tortSW(Canvas can, Rgb color, float distance) {
    int counter = 0;
    float counter1 = 0.0;
    float step = distance * 0.707;
    while(counter1 < step) {
        pixel(can, color, xcur-counter, ycur+counter);
        counter1 = counter1 + 1.0;
        counter = counter + 1;
    }
    xcur = xcur - counter;
    ycur = ycur + counter;
    return 0;
}

int tortSE(Canvas can, Rgb color, float distance) {
    int counter = 0;
    float counter1 = 0.0;
    float step = distance * 0.707;
    while(counter1 < step) {
        pixel(can, color, xcur+counter, ycur+counter);
        counter1 = counter1 + 1.0;
        counter = counter + 1;
    }
    xcur = xcur + counter;
    ycur = ycur + counter;
    return 0;
}

int movetort(int x, int y) {
    xcur = x;
    ycur = y;
    return 0;
}

int main() {
    Canvas can = Canvas(600, 600);
    Rgb color = Rgb(56,137,190);
    Rgb roofcolor = Rgb(255,0,0);
    movetort(200, 200);
    tortright(can, roofcolor,200);
    tortdown(can, color, 200);
    tortleft(can, color, 200);
    tortup(can, color, 200);
```

```
movetort(200, 200);
tortNE(can, roofcolor, 141.0);
tortSE(can, roofcolor, 141.0);

Rgb window = Rgb(201, 52, 235);
movetort(220,220);
tortright(can, window, 30);
tortdown(can, window, 30);
tortleft(can, window, 30);
tortup(can, window, 30);
movetort(235,220);
tortdown(can, window, 30);
movetort(220, 235);
tortright(can, window, 30);

Rgb win = Rgb(235, 149, 52);
movetort(270, 260);
tortright(can, win, 60);
tortdown(can, win, 60);
tortleft(can, win, 60);
tortup(can, win, 60);
movetort(300,260);
tortdown(can, win, 60);
movetort(270, 290);
tortright(can, win, 60);

Rgb chimney = Rgb(97,51,1);
movetort(350, 150);
tortup(can, chimney, 100);
tortright(can, chimney, 35);
tortdown(can, chimney, 135);

Rgb door = Rgb(114, 40, 143);
movetort(320, 400);
tortup(can, door, 60);
tortright(can, door, 60);
tortdown(can, door, 60);
movetort(350, 400);
tortup(can, door, 60);

movetort(340, 370);
tortright(can, door, 6);
tortdown(can, door, 6);
tortleft(can, door, 6);
tortup(can, door, 6);

movetort(354, 370);
tortright(can, door, 6);
tortdown(can, door, 6);
tortleft(can, door, 6);
tortup(can, door, 6);
```

```
    save(can, "simple_turtle_start.png");
    return 0;
}
```

9.11.2.51 test-string.out

```

```

9.11.2.52 test-string.rt

```
int main() {
    string s = "hey";
    return 0;
}
```

9.11.2.53 test-void.out

```
3
```

9.11.2.54 test-void.rt

```
void p(int a) {
    print(a);
}
int main() {
    p(3);
    return 0;
}
```

9.11.2.55 test-while.out

```
5
6
7
8
9
```

9.11.2.56 test-while.rt

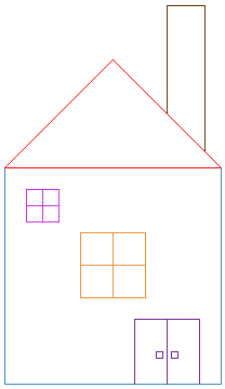
```
int main() {
```

```
    int i = 5;
    while(i < 10) {
        print(i);
        i = i+1;
    }
    return 0;
}
```

9.12 PNG Output Files

9.12.1 edwardstest.png

9.12.2 simple_turtle_start.png



9.13 Git Log

```
commit 7ce37cbb48ecba6780358da271ad7546a193e531 (HEAD -> main, origin/main, origin/HEAD)
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sun Apr 25 14:20:04 2021 -0400

    making functions uniform in codegen

commit 412fbc72e710b2339385b91188fcdd911284e2c0
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Sun Apr 25 13:58:14 2021 -0400

    cleaned up test files and file structure

commit 5268693fb2b47207b24d98c8d491f4d71fad5d94
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sun Apr 25 10:46:25 2021 -0400

    getting rid of executables

commit 508fbe408c1848cdd8541075fdc11ccf6e9ffd5a
Merge: 6567e5a b9192f7
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sun Apr 25 10:05:46 2021 -0400
:...skipping...
commit 7ce37cbb48ecba6780358da271ad7546a193e531 (HEAD -> main, origin/main, origin/HEAD)
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sun Apr 25 14:20:04 2021 -0400

    making functions uniform in codegen

commit 412fbc72e710b2339385b91188fcdd911284e2c0
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Sun Apr 25 13:58:14 2021 -0400

    cleaned up test files and file structure

commit 5268693fb2b47207b24d98c8d491f4d71fad5d94
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sun Apr 25 10:46:25 2021 -0400

    getting rid of executables

commit 508fbe408c1848cdd8541075fdc11ccf6e9ffd5a
Merge: 6567e5a b9192f7
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sun Apr 25 10:05:46 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 6567e5ab20f6c29a0de1845127d6797d799b9e6d
Author: Aviva Weinbaum <aw3156@barnard.edu>
```

```
Date:   Sun Apr 25 10:05:41 2021 -0400

    moree cleanup

commit b9192f7faad5dd7f42058eb5f8fe9932573ff145
Author: Hariti Patel <45344801+haritipatel@users.noreply.github.com>
Date:   Sat Apr 24 23:49:24 2021 -0400

    Updated README

commit 581f170e32edf979ca2d496288766efb0c216d6e (HEAD -> main, origin/main,
origin/HEAD)
Merge: 84c15d0 df9d4de
Author: Aviva Weinbaum <44536250+avivaweinbaum@users.noreply.github.com>
Date:   Sat Apr 24 23:16:55 2021 -0400

    Merge pull request #1 from avivaweinbaum/start_library

    Adding beginning of library integration - NOT WORKING

commit df9d4de83fd7252500c6e0b3e105b0f0c1f494cc (origin/start_library,
start_library)
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sat Apr 24 21:39:27 2021 -0400

    cleanup

commit 2df9428836ce5aa3333e6137a46c9e9f9cdc8681
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
:...skipping...
commit 581f170e32edf979ca2d496288766efb0c216d6e (HEAD -> main, origin/main,
origin/HEAD)
commit 581f170e32edf979ca2d496288766efb0c216d6e (HEAD -> main, origin/main,
origin/HEAD)
Merge: 84c15d0 df9d4de
Author: Aviva Weinbaum <44536250+avivaweinbaum@users.noreply.github.com>
Date:   Sat Apr 24 23:16:55 2021 -0400

    Merge pull request #1 from avivaweinbaum/start_library

    Adding beginning of library integration - NOT WORKING

commit df9d4de83fd7252500c6e0b3e105b0f0c1f494cc (origin/start_library,
start_library)
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sat Apr 24 21:39:27 2021 -0400

    cleanup
```

```
commit 2df9428836ce5aa3333e6137a46c9e9f9cdc8681
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Sat Apr 24 21:30:04 2021 -0400

    Add gcd and house

commit f826f4a2fc6ecb3289ba9e064b47a419b505cf3e
Merge: eb39eee a393f48
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sat Apr 24 20:54:50 2021 -0400

    Merge branch 'start_library' of https://github.com/avivaweinbaum/reptile into
start_library

commit eb39eeef25fa08122093e789f480c9f7398e0261
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sat Apr 24 20:39:24 2021 -0400

    cleanup

commit a393f48680f7ed4e6fb40a52e998f007321bc285
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Sat Apr 24 20:38:43 2021 -0400

    reorganized file structure

commit 8e3ca3e8631644e611d6b991c3e462fab3472534
Merge: 694fbf4 9d49556
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Sat Apr 24 20:24:10 2021 -0400

    Resolve merge conflicts

commit 694fbf456a1ef5e91510bf43d52246e23bff1ba0
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Sat Apr 24 20:23:21 2021 -0400

    add gcd function test and set rgb values

commit 9d49556db69476577a13248e98a1895c41748e16
Merge: 75cb6b4 06da4d7
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Sat Apr 24 20:21:32 2021 -0400

    Merge branch 'start_library' of https://github.com/avivaweinbaum/reptile into
start_library
```

```
commit 75cb6b4937887ffd587e9cdddcc4f8533c1955c4
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Sat Apr 24 20:21:20 2021 -0400

    added test for canvas

commit 06da4d78fc472ebb1506ff33fba029ee339ac80e
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sat Apr 24 20:21:11 2021 -0400

    getting rid of access

commit 13a71b6d5a4118e81b377c40981b2f65a1d77ec3
Merge: e716608 d08e7cd
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Sat Apr 24 18:44:09 2021 -0400

    resolve merge

commit e716608dd90c685c76c3e2b10ef777a6e04e2752
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Sat Apr 24 18:39:32 2021 -0400

    Add turtle line functionality with diagonals

commit d08e7cd1ac129c38b895f0badd35b8db6317fd30
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sat Apr 24 18:07:09 2021 -0400

    finishing access functions

commit 6d7c30f1b783af0eba562489e64e3144e8fb31fd
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sat Apr 24 11:29:07 2021 -0400



commit 5e2d568d1a4420ae8a781b28cb2899497a708664
Merge: ee08ce6 946c75e
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Sat Apr 24 00:08:58 2021 -0400

    Resolve merge conflicts

commit ee08ce62fcaef563130bdbd159fa03f162706e50
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
```

```
Date:   Sat Apr 24 00:06:44 2021 -0400

    Add beginning of Turtle library - not working

commit 946c75e0f16e2dcee4d5f2ffc96d102f889b5b40
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Fri Apr 23 22:21:42 2021 -0400

    resolving warnings

commit a7c37e3db4aa03390f0d4df9a517235a9ed2e6b4
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Fri Apr 23 22:04:23 2021 -0400

    deleting file

commit 3d4de6752a65ac8bf2e7ead260f4737050365c00
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Fri Apr 23 17:23:42 2021 -0400

    save canvas class and repush

commit f343f6523912507dab02b2bab02b1ed031552287
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Fri Apr 23 17:23:20 2021 -0400

    Change canvas logic and parameters to eliminate pointer for now

commit 42c0e94b429e2da5cf1c330c9794a4b523f176a2
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Fri Apr 23 16:21:49 2021 -0400

    added more tests for add, or, and not, removed file, mod, and exp

commit 6f648dd0d8a7f1ce6b1c416a79a58655c3f0460c
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Fri Apr 23 14:53:06 2021 -0400

    Add fib test and structure plan

commit 981618c580414956a8b3790fcfc8a9f1d4594eac
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 22 22:55:56 2021 -0400

    Add test to create profs face

commit 09eb196d00ab9993fe8eee8e1c9f0ed82fd9efc2
```

```
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 22 21:11:11 2021 -0400

    Added other access methods and modified pointers to produce image

commit ef2290fe67f38fa5846fd2bc9af3ec668beba659
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Wed Apr 21 21:17:31 2021 -0400

    Changed user-defined types to be pointers and deleted sub miniReptile inside
miniReptile

commit 94f9606113fcacf2b787f7ba276269526117a737
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Wed Apr 21 18:58:58 2021 -0400

    Changes from office hours --> TODO change to pointers

commit 0e12ed317e729de5e93ad8d291513bdc748092a0
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Mon Apr 19 16:19:22 2021 -0400

    adding pixel function

commit bcfde9160b96f4af98323beac4e9d40c12e514b1
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Sun Apr 18 14:33:58 2021 -0400

    added string type

commit ff7f7a0a3e05a0fd45ec781f8ab2dea8e3754a8d
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Wed Apr 14 17:53:49 2021 -0400

    Adding beginning of library integration

commit 84c15d0be56b6b468e9cdb4e8ec6e6dedc620a9d
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Mon Apr 12 09:18:16 2021 -0400

    updating types and adding access

commit 57cffee6d390037c820c02a1c89e7970d940ae5f
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sat Apr 10 12:06:38 2021 -0400

    adding access
```

```
commit a8b2cd9dbcbf703e0aa0f08476a0d952dc766ac7
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Fri Apr 9 14:23:41 2021 -0400

    add extra c structs

commit 0675e290b144dfa82cfeeda940a214892e7d8120
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 8 20:30:02 2021 -0400

    change testall

commit c2ca830c3ed265142d369d5166ac1f2f24ddf3e8
Merge: 2ea1407 dab0ee8
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 8 20:28:55 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 2ea1407c180a94ad2fac3ccde38ddbcafa031663
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 8 20:28:46 2021 -0400

    makefile change

commit dab0ee8dffd38d16a5744fcabf94403bdf733013
Merge: 1b06c4e 3a26c4c
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 8 20:27:28 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 1b06c4e8529fe79501a4f329eebf1c4da189ed48
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 8 20:26:55 2021 -0400

    codegen maybe

commit 3a26c4c761d952d9ee4875c102e907c4d5d75e14
Merge: 8d83b5f b45ff96
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 8 20:03:35 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 8d83b5f4d8224a5311a7769528dd666f7d0c01f0
```

```
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 8 20:02:23 2021 -0400

    Add rgb

commit b45ff96a5ecec1747782ac148c6219b652c72d8b
Merge: 554f175 139df34
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 8 19:51:52 2021 -0400

    Merge conflicts

commit 554f175e088f8d75dff0c8db929dc1addc5346c7
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Thu Apr 8 19:50:35 2021 -0400

    Got rgb working

commit 139df34e9c4c7fa15534ab029524a4e4d0d68228
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Wed Apr 7 16:50:09 2021 -0400

    changed from float and add assign to parser

commit 29852d691e3ca26f1139dcf20559463668b70d85
Author: Aileen Cano <aileen4622@gmail.com>
Date:   Tue Apr 6 21:52:54 2021 -0700

    Resolving some warnings related to Blit

commit add292d1d26af7ccbe632ceee97debae19a85085
Author: Aileen Cano <aileen4622@gmail.com>
Date:   Tue Apr 6 21:30:04 2021 -0700

    adding tests for funcs, return, if, etc

commit c065e0041532243493bcc8efd2ee449c3e203065
Merge: d076bee d1696be
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Mon Apr 5 19:15:34 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit d076beeb42526df5dcaab58c4d34f791f8254856
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Mon Apr 5 19:14:14 2021 -0400
```

```
    added while loops

commit d1696be6486c9bf6f015a0e3904ed977a69c4325
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Mon Apr 5 17:37:10 2021 -0400

    added if/else functionality

commit cef1ebe7d8ab793ac9dbf8a3e2db5130bb3ad842
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Mon Apr 5 12:48:48 2021 -0400

    more assign

commit 2212f10d824f42a8eb88a9858f10506b6aceda1c
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Fri Apr 2 16:00:39 2021 -0400

    adding some assign

commit aed8aa0229d2d3f22aff497acb290559f6f7f602
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Fri Apr 2 13:21:48 2021 -0400

    adding types

commit 8f62677d13f00189e6eff0e85c9caf6e8275a644
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Wed Mar 24 17:41:53 2021 -0400

    updating readme

commit e3a20f1e5d25e1e01de7923b50a4db63d9b3fbc7
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Wed Mar 24 17:29:48 2021 -0400

    miniReptile

commit 487e28ed950a4554c3091804e4f8f56713bae361
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Wed Mar 24 15:06:28 2021 -0400

    fixing errors with ast, scanner, parser, codegen, semant

commit 3c1be781db9a47aafbc8d839d72d8991a5be215f
Merge: e48d7a2 8259033
Author: haritipatel <haritipatel2000@gmail.com>
```

```
Date:   Wed Mar 24 13:03:22 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 82590336b8d6b102d4ed689737194c5cee403d42
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Wed Mar 24 13:00:14 2021 -0400

    updated miniReptile

commit e313d1cd4db2193db6c53bdbd23a37656b3a930f
Merge: 285f667 a886237
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Tue Mar 23 17:12:02 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 285f667c57949b6109a4dc53cc76f4e2fc679ec1
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Tue Mar 23 17:11:50 2021 -0400

    Adding minireptile

commit e48d7a2aa1726c2ee0c5a99951317d48bf2060a7
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Tue Mar 23 17:02:54 2021 -0400

    worked with main codegen.ml

commit a8862377847feec250b807632d0181e7a6ba3937
Author: Hariti Patel <45344801+haritipatel@users.noreply.github.com>
Date:   Tue Mar 23 17:02:34 2021 -0400

    added miniReptile

commit ad9467469fc040228bd9d2b45aa46c6a266ecd25
Merge: 230e938 a16a3f9
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Mon Mar 22 21:10:11 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 230e9387eabf2c168694ff89354cc031379c1a75
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Mon Mar 22 21:10:07 2021 -0400

    updated codegen.ml
```

```
commit ed84532103ddcc6dfaa04807545b80b05b721fc6
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Mon Mar 22 16:08:13 2021 -0400

    added printbig.c and Dockerfile and edited Makefile

commit a16a3f978ea2f82bceb4f3cdbee5a013e07187e9
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Mon Mar 22 16:00:28 2021 -0400

    Added _tags

commit 55e1e1b6632291f0d4b26e2d5097ffbd3327fcb7
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Mon Mar 22 15:54:45 2021 -0400

    added simplified reptile version

commit 0d2e446a0daec7cebf30f59479876b84ce4ed8cb
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Mon Mar 22 14:46:07 2021 -0400

    Added tests

commit 470fbf0c013d722cd06258e468db4fdd5512d187
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Mon Mar 22 14:45:52 2021 -0400

    Updated codegen and makefile

commit e9ba8c3844b333b13c98bc966b798ca89ae9e217
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Mon Mar 22 14:26:55 2021 -0400

    Create stupid codegen

commit 57a53363c5a9f6d0d164971742d31260a2aa1efa
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Sun Mar 21 23:35:06 2021 -0400

    started codegen

commit 275b73f6c6620a6e07e3ae5846a3d0a00a217e5c
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Sun Mar 21 23:07:56 2021 -0400
```

```
    Add codegen file

commit 23b67b71ecbf0903dd9cd414d122d4cbec244d91
Merge: b97b032 30401cc
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Wed Mar 17 17:13:51 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit b97b032c6da0049292ec37cd0a13ef046d7b877b
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Wed Mar 17 17:13:44 2021 -0400

    Updated testall.sh

commit 30401cc721903a22faadc3f84f6d71be7b5a4827
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Wed Mar 17 17:01:48 2021 -0400

    adding reptile.ml

commit 18a7ea4790f04435d3366a723377206774b44b46
Merge: ff8f9cd 1bba1aa
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Wed Mar 17 17:00:33 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit ff8f9cd574fbeab57615afaec1801e8783eae0b0
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Wed Mar 17 17:00:27 2021 -0400

    added sast and edited ast

commit 1bba1aa2310cda64c1fda417cdabe677a8f0ecd0
Merge: 0522071 5c16197
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Wed Mar 17 16:48:18 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 0522071a3f70a439e67b9265629610d77f075deb
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Wed Mar 17 16:47:59 2021 -0400

    Changed file extensions
```

```
commit 5c16197f01d68a9b81598fcfca14bd124f38ca55
Merge: 0ac230b 74d6fdd
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Wed Mar 17 16:46:38 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 0ac230b4e495c2690e0f429701127139fe8fe462
Author: haritipatel <haritipatel2000@gmail.com>
Date:   Wed Mar 17 16:46:29 2021 -0400

    added testall.sh

commit 74d6fdd2bbbc2ffc4632552894d0ddc8da6c8e90
Merge: 98925c9 4d7f2e7
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Wed Mar 17 16:45:56 2021 -0400

    Merge branch 'main' of https://github.com/avivaweinbaum/reptile into main

commit 98925c9ce222aa9b0c1a89ebb0be7fefc9bacc65
Author: Lindsey Weiskopf <ltweiskopf@gmail.com>
Date:   Wed Mar 17 16:45:23 2021 -0400

    Create makefile from Microc skeleton

commit 4d7f2e73a732478b6df8cf0e96afdadeca202858
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Wed Mar 17 16:40:46 2021 -0400

    updating ast

commit 37035ac7b942fb1b474b4c966c5be8627345bb2a
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Wed Feb 24 11:30:11 2021 -0500

    adding inc

commit 2f122e235357645c120f4a644a9912f3216caeed
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sun Feb 21 14:03:21 2021 -0500

    adding ast

commit 2ffb8bd8ba0e8a592bdac254006c3d4bf14d824b
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Sat Feb 20 12:37:38 2021 -0500
```

```
    adding scanner

commit c20a3eefb5145114ab834ae82d34f599e2e9291d
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Fri Feb 19 14:29:12 2021 -0500

    adding parser

commit e4ba47404a2d96bf4d2f4b131b4aeac33e00d9ba
Author: Aviva Weinbaum <aw3156@barnard.edu>
Date:   Wed Feb 17 17:20:21 2021 -0500

    initial commit

commit cdaab9bdc977a17f72d846c6c6453c51c8129827
Author: Aviva Weinbaum <44536250+avivaweinbaum@users.noreply.github.com>
Date:   Wed Feb 17 17:18:58 2021 -0500

    Initial commit
(END)
```