Dice Project Report

"Java, but worse"

Project Manager:	**David Watkins**	**djw2146**
Language Guru:	**Emily Chen**	**ec2805**
System Architect:	**Philip Schiffrin**	**pjs2186**
Tester & Verifier:	**Khaled Atef**	**kaa2168**

Contents

1. Introduction

The Dice programming language is an object-oriented, general purpose programming language. It is designed to let programmers who are more familiar with object oriented programming languages to feel comfortable with common design patterns to build useful applications. The syntax of Dice resembles the Java programming language. Dice compiles down to LLVM IR which is a cross-platform runtime environment. This allows Dice code to work on any system as long as there is an LLVM port for it, which includes Windows, Mac OS X, and Linux or various processor architectures such as x86, MIPS, and ARM[1].

Dice lays programs out the same way a Java program would. Variables and methods of a class can be declared with private scope. There is a simple to use inheritance that allows for multiple children inheriting the fields and methods of its parent. Dice also allows for convenient use of functions that exist in C, such as malloc, open, and write. This allows the user to construct objects and call c functions using those objects.

Background

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects". These objects are data structures that contain data, in the form of fields, often known as attributes. The code itself are contained within methods in the code which are compiled to varying subroutines. The most useful aspect of OOP is that these methods and fields can modify one another allowing for a rich and varied use case.

Class based OOP specifically creates instances of classes, referred to as objects, which have their values modified at runtime. There are many languages that implement their language this way including Java and C#.

Inheritance is when an object or class is based on another class using the same implementation. This allows for a class to serve as a blueprint for subclasses. Polymorphism allows an object to take on many forms. This may include an object being assigned to a type that is a class it inherits from, or being used in place of a class it inherits from.

We want to leverage these capabilities using LLVM code to produce a syntactically Java-like language but offer a cross platform solution that is simple and easy to use. Implementing inheritance and objects in a c-like context like LLVM allows for fine control over the code.

Related Work

Object-oriented programming languages have existed since the late 20th century. Java, C#, C++, Objective-C, Python, and many more languages have facilities for defining custom user classes and manipulating them at runtime.

[1]http://llvm.org/

Implementing an object-oriented paradigm using C is a well-known solution, but compiling object-oriented code down to LLVM is not publicly available. We want to contribute to the LLVM community by adding additional information regarding the creation of a compiler using OCaml that compiles to LLVM code.

Goals

Cross-Platform

Utilizing the LLVM IR we are able to compile the source once and have it work on multiple architectures without fail.

Flexibility

Allowing the user to define their own classes and offering them the ability to inherit functionality from other user defined types offer a wide range of possibilities for their programs and also saves the user time when implementing large programs.

Transparency

Using the LLVM IR allows the user to see exactly what the program is doing after the compiler is done. For a more optimal result it can then be compiled to bitcode representation using the LLVM compiler.

Familiarity

Incorporate familiar primitive data types most commonly found in languages such as C, C++, and Java such as int, char, float, and bool.

2. Language Tutorial

Environment Setup

The compiler has been built an tested using an Ubuntu 15.10 virtual machine. The ISO for downloading Ubuntu 15.10 can be found here[1]. This is followed by downloading virtualbox and following the corresponding tutorial for setting up a custom Ubuntu VM here[2]. Once inside the VM there are a series of packages that need to be installed before you can compile the compiler. Run the following commands to install the corresponding packages:

```
>sudo apt-get install m4 clang-3.7 clang-3.7-doc libclang-common-3.7-dev libclang-3.7-dev
↪  libclang1-3.7 libclang1-3.7-dbg libllvm-3.7-ocaml-dev libllvm3.7 libllvm3.7-dbg
↪  lldb-3.7 llvm-3.7 llvm-3.7-dev llvm-3.7-doc llvm-3.7-examples llvm-3.7-runtime
↪  clang-modernize-3.7 clang-format-3.7 python-clang-3.7 lldb-3.7-dev liblldb-3.7-dbg
↪  opam llvm-runtime
```

Then initialize OCaml's package manager (OPAM) in your home directory:

```
>opam init
>opam switch 4.02.1
>eval $(opam config env)
>opam install core batteries llvm yojson
```

After OPAM is initialized, go to the the directory where you want Dice installed and clone the git repository:

```
>git clone https://github.com/DavidWatkins/Dice.git
```

Using the Compiler

Inside the directory 'Dice' type **make**. This creates the dice compiler that takes in '.dice' files and compiles them to corresponding '.ll' files corresponding to LLVM IR. The syntax for running the dice executable is: **dice [optional-option] ⟨source file⟩**. There are also additional flags with respect to the compiler that allow for additional options.

- **-h** - Print help text
- **-tendl** - Prints tokens with newlines intact
- **-t** - Prints token stream
- **-p** - Pretty prints Ast as a program
- **-ast** - Prints abstract syntax tree as json
- **-sast** - Prints semantically checked syntax tree as json

[1]http://www.ubuntu.com/
[2]http://www.wikihow.com/Install-Ubuntu-on-VirtualBox

- **-c** - Compiles source and prints result to stdout
- **-f** - Compiles source to file (⟨filename⟩.⟨ext⟩ → ⟨filename⟩.ll)

The following sample dice code demonstrates the following features:

- The mandatory main function that exists within **only** one class. The syntax for a main declaration is **public void main(char[][] args)**
- Calling the built-in print function, which takes an arbitrary list of primitive values, including char[].
- A string literal with escape characters
- Defining a base class with one or more fields.

```
class example1 {
        public void main(char[][] args) {
                print("This is example 1\n");
        }
}
```

To compile the sample code above, type:

```
> ./dice example1.dice
```

The output will be a file named **example1.ll** which will run using the **lli** command:

```
>lli example1.ll
This is example 1
>
```

If you get an error: ”error: expected value token” from lli, that means your version of lli is probably set incorrectly. Run the following command to verify the version:

```
>lli --version
```

If it’s anything other than version **3.7** change it with the following commands:

```
>sudo rm \usr\bin\lli
>ln -s /usr/lib/llvm-3.7/bin/lli /usr/bin/lli
```

The basics

Primitives

All primtives are declared starting with their type followed by an identification. Dice supports the following primitives:

- integers (int)
- floating point (float)
- characters (char)
- booleans (bool)

```
class example2 {
        public void main(char[][] args) {

                (* This is a comment (* with a nested one inside *) *)
                int a; (* Declaring an integer primitive variable *)
                a = 1; (* Assigning the number one to variable a *)

                float b = 1.5; (* Combined declaration and assignment is okay *)

                (* Characters and booleans are primitives as well *)
                char c = 'c';   (* ASCII or digits only within single quotes*)
                bool d = true;  (* or 'false' *)
        }
}
```

Arrays

Arrays are indexed collections of values of a datatype (primitive or object). Dice allows for single dimension arrays only. The elements within the arrays created default to null which, like C, are implemented with zeros.

```
class example3 {
        public void main(char[][] args) {

                int[] a = new int[10]; (* int array with 10 elements set to zero *)

                a[0] = 1; (* Access the first element and assign the integer 1 to it *)

                int[] b = |0,1,2,3,4,5|; (* int array with 6 int elements *)

                print(b.length); (* prints 6 *)

                char[] c = |'h','i', 0|; (* ints are allowed to be stored in char
↪   elements *)

        }
}
```

Operators

Dice supports the following binary operators:

- Arithmetic (+ , - , * , / , %)
- Relational (== , != , ⟩, ⟩ =, ⟨=, ⟨)
- Logical (and, or)

Unary operators:

- Logical negation (not)
- Negative number (-)

```
class example4 {
        public void main(char[][] args) {

                int a = 1 + 2;      (* a is now 3 *)
                float b = 2.5 - 2;  (* 2 is promoted to float, b is now 0.5 *)
                int c = 5 + 2 * 4;  (* c is 13 due to operator precedence *)
                int d = 10 / 5 + 3; (* d is now 5 *)
                int e = 5 % 3;             (* e is now 2 *)

                bool f = true; bool g = false;
                f == f; f != g; 5 > 2; 3 >= 3; f or g;  (* all expressions evaluate to
↪   true *)
                f and g; not f; (* evaluate to false *)

                c = -a;     (* c is now -3 *)
        }
}
```

Control Flow

The statements inside source files are generally executed from top to bottom, in the order that they appear. Control flow statements, however, break up the flow of execution by employing decision making, looping, and branching, enabling your program to conditionally execute particular blocks of code. This section describes the decision-making statements (if-then, if-then-else), the looping statements (for, while), and the branching statements (break, continue, return) supported by Dice.

Branching

```
class example5 {
        public void main(char[][] args) {
                int a;
                if (true)
                        a = 1;
                else
                        a = 0;
                (* a is now 1 *)

                int b;
                if (false){
                        b = 2; a = 3;
                }
                else {
                        b = 0; a = 0;
                }
                (* b and a are now 0 *)

                int c;
                if(false){ a = 1; b = 1; c = 1;}
                else if(true) { a = 5; b = 5; c = 5;}
                else { a = 0; b = 0; c = 0;}
                (* a,b,c are now set to 5 *)
        }
}
```

Loops

The two types of loops that Dice supports are 'for' and 'while' loops. The for statement allows the programmer the iterate over a range of values. The while statement executes a user-defined block of statements as long as a particular conditional expression evaluates to true.

```
class example6 {
        public void main(char[][] args) {
                int a = 0;
                int i;                (* The loop counter must be declared outside the for loop *)
                 for (i = 0 ; i < 5 ; i = i + 1) {
                     a = a + 2;
                 }
                 (* a is now set to 10 *)

                 int b = 0;
                 int j;
                 for (j = 0 ; j < 5 ; j = j + 1) {
                         a = a + 2;
                     if(a >= 14){
                             break;                (* will break out of the parent for loop *)
                     }
                     else { continue; } (* will skip the remaining code and start the next iteration *)
                     b = b + 10;
                 }
                 (* b is still zero, a is 14 *)

                 while(b<5){
                         b = b + 1;
                }

                (* b is now 4 *)
        }
}
```

Defining methods

Dice supports methods that return a datatype after execution or simply execute without returing anything. Methods can accept arguments which are computed in an applicative order. Each method must also contain a scope (public/private) which determine access for outside classes. The following example will show two kinds of methods:

```
class example7 {
        public int p(int i){
                print(i);
                return i;
        }

        public void q(int a, int b, int c){
                int total = a ;
                print(b);
                total = total + c ;
        }

        public void main(char[][] args) {
                this.q( this.p(1), 2, this.p(3));
        }
}
```

The output of this program is:

```
132
```

Classes and Inheritance

Since Dice is an Object Oriented language, you can create custom classes that can serve as datatypes. A class contains three sections:

- Fields
- Constructors
- Methods

These sections may be written in any order desired. You may also mix them up if desired. For example, a constructor may be added inbetween field declarations if desired. If no constructors are defined, Dice will use a default constructor to instantiate objects. A parent class can also be assigned any class that is a descendant of it as shown below:

```
class shape {
        public int xCoord;    (* Fields *)
        public int yCoord;

        constructor(){                  (* Constructor *)
                  this.xCoord = 0;
                  this.yCoord = 0;
        }

        (* Constructor with a different signature due to the two arguments *)
        constructor(int x, int y){
                  this.xCoord = x;
                  this.yCoord = y;
        }

        public void myAction(){  (* Method *)
                  print("shape");
        }
        }

        class circle extends shape {
        public int radius;          (* Field unique to circle *)

        constructor(){
                  this.xCoord = 0;        (* xCoord and yCoord from parent class 'shape'
↪   *)
                  this.yCoord = 0;
                  this.radius = 0;
        }

        constructor(int x, int y, int r){
                  this.xCoord = x;
                  this.yCoord = y;
                  this.radius = r;
        }

        public void myAction(){   (* This method overrides the one defined in parent
↪   class *)
                  print("circle\n");
                  print(this.radius);

        }
}

class example8 {
        public void main(char[][] args) {
           class circle a = new circle(1, 2, 3);
           class circle[] b = new class circle[10];
           b[0] = a;
           print(b[0].radius,"\n");

```

```
            class shape c = new circle(4, 5, 6);  (* Inheritance in action! *)
            c.myAction();
            print("\n");
        }
}
```

The output for example8 is:

```
3
circle
6
```

3. Language Reference Manual

Introduction

Dice is a general purpose, object-oriented programming language. The principal is simplicity, pulling many themes of the language from Java. Dice is a high level language that utilizes LLVM IR to abstract away hardware implementation of code. Utilizing the LLVM as a backend allows for automatic garbage collection of variables as well.
Dice is a strongly typed programming language, meaning that at compile time the language will be type-checked, thus preventing runtime errors of type.
This language reference manual is organized as follows:

- Section 2 Describes types, values, and variables, subdivided into primitive types and reference types
- Section 3 Describes the lexical structure of Dice, based on Java. The language is written in the ASCII character set
- Section 4 Describes the expressions and operators that are available to be used in the language
- Section 5 Describes different statements and how to invoke them
- Section 6 Describes the structure of a program and how to determine scope
- Section 7 Describes classes, how they are defined, fields of classes or their variables, and their methods

The syntax of the language is meant to be reminescent of Java, thereby allowing ease of use for the programmer.

Types

There are two kinds of types in the Dice programming language: primitive types and non-primitive types. There are, correspondingly, two kinds of data values that can be stored in variables, passed as arguments, returned by methods, and operated on: primitive values and non-primitive values.

```
Type:
PrimitiveType
NonprimitiveType
```

There is also a special null type, the type of the expression *null*, which has no name. Because the null type has no name, it is impossible to declare a variable of the null type. The null reference is the only possible value of an expression of null type. The null reference can always undergo a widening reference conversion to any reference type. In practice, the programmer can ignore the null type and just pretend that *null* is merely a special literal that can be of any reference type.

Primitive Types and Values

A primitive type is predefined by the Dice programming language and named by its reserved keyword.

```
PrimitiveType:
NumericType
bool
NumericType:
IntegralType
float
IntegralType: one of
int char
```

int

A value of type *int* is stored as a 32-bit signed two's-complement integer. The *int* type can hold values ranging from -2,147,483,648 to 2,147,483,647, inclusive.

float

The float type stores the given value in 64 bits. The *float* type can hold values ranging from 1e-37 to 1e37. Since all values are represented in binary, certain floating point values must be approximated.

char

The *char* data type is a 8-bit ASCII character. A *char* value maps to an integral ASCII code. The decimal values 0 through 31, and 127, represent non-printable control characters. All other characters can be printed by the computer, i.e. displayed on the screen or printed on printers, and are called printable characters. The character 'A' has the code value of 65, 'B' has the value 66, and so on. The ASCII values of letters 'A' through 'Z' are in a contiguous increasing numeric sequence. The values of the lower case letters 'a' through 'z' are also in a contiguous increasing sequence starting at the code value 97. Similarly, the digit symbol characters '0' through '9' are also in an increasing contiguous sequence starting at the code value 48.

bool

A variable of type *bool* can take one of two values, *true* or *false*. A bool could also be *null*.

Non-Primitive Types

Non-primitive types include arrays and classes.

Arrays

An array stores one or more values of the same type contiguously in memory. The type of an array can be any primitive or an array type. This allows the creation of an n-dimensional array, the members of which can be accessed by first indexing to the desired element of the outermost array, which is of type *array*, and then accessing into the desired element of the immediately nested array, and continuing n-1 times.

Classes

Classes are user-defined types. See chapter 7 to learn about their usage.

Casting

Casting is not supported in this language. There are behaviors between ints and float defined in the section on operators that imitate casting, but there is no syntax to support casting between types directly.

Lexical Conventions

This chapter describes the lexical elements that make up Dice source code. These elements are called tokens. There are six types of tokens: identifiers, keywords, literals, separators, and operators. White space, sometimes required to separate tokens, is also described in this chapter.

Identifiers

Identifiers are sequences of characters used for naming variables, functions and new data types. Valid identifier characters include ASCII letters, decimal digits, and the underscore character '_'. The first character must be alphabetic.
An identifier cannot have the same spelling (character sequence) as a keyword, boolean or null literal, a compile-time error occurs. Lowercase letters and uppercase letters are distinct, such that foo and Foo are two different identifiers.

```
ID = "['a'-'z' 'A'-'Z'](['a'-'z' 'A'-'Z']|['0'-'9']|'\_')*"
```

Keywords

Keywords are special identifiers reserved for use as part of the programming language itself. You cannot use them for any other purpose. Dice recognizes the following keywords:

if	else	for	while	
break	continue	return		
int	float	bool	char	void
null	true	false	class	constructor
public	private	extends	include	this

Literals

A literal is the source code representation of a value of a primitive type or the null type.

Integer Literals

An integer literal is expressed in decimal (base 10). It is represented with either the single ASCII digit 0, representing the integer zero, or an ASCII digit from 1 to 9 optionally followed by one or more ASCII digits from 0 to 9.

```
INT = "['0'-'9']+"
```

Float Literals

A float literal has the following parts: an integer part, a decimal point (represented by an ASCII period character), and a fraction part. The integer and fraction parts are defined by a single digit 0 or one digit from 1-9 followed by more ASCII digits from 0 to 9.

```
FLOAT = "['0'-'9']+ ['.'] ['0'-'9']+"
```

Boolean Literals

The boolean type has two values, represented by the boolean literals true and false, formed from ASCII letters.

```
BOOL = "true|false"
```

Character Literals

A character literal is always of type *char*, and is formed by an ascii character appearing between two single quotes. The following characters are represented with an escape sequence, which consists of a backslash and another character:

- '\\' - backslash
- '\"' - double-quote
- '\'' - single-quote
- '\n' - newline
- '\r' - carriage return
- '\t' - tab character

It is a compile-time error for the character following the character literal to be other than a single-quote character $'$.

```
CHAR = "\' ( ([' '-'!' '#'-'[' ']'-'~'] | '\\' [ '\\' '\"' 'n' 'r' 't' ]) )\' "
```

String Literals

A string literal is always of type char[] and is initialized with zero or more characters or escape sequences enclosed in double quotes.

```
char[] x = "abcdef\n";
```

```
STRING = "\"( ([' '-'!' '#'-'[' ']'-'~'] | '\\' [ '\\' '\"' 'n' 'r' 't' ]) )*\""
```

Separators

A separator separates tokens. White space is a separator but it is not a token. The other separators are all single-character tokens themselves: ( ) [ ] ; , .

```
'('      { LPAREN }
')'      { RPAREN }
'{'      { LBRACE }
'}'      { RBRACE }
';'      { SEMI }
','      { COMMA }
'['      { LBRACKET }
']'      { RBRACKET }
'.'      { DOT }
```

Operators

The following operators are reserved lexical elements in the language. See the expression and operators section for more detail on their defined behavior.

+	-	*	/	%
=	==	!=		
<	<= >	>=		
and	or	not		
new	delete			

White Space

White space refers to one or more of the following characters:

- the ASCII SP character, also known as ”space”
- the ASCII HT character, also known as ”horizontal tab”
- the ASCII FF character, also known as ”form feed”
- LineTerminator

White space is ignored, except when it is used to separate tokens. Aside from its use in separating tokens, it is optional.

```
WHITESPACE = "[’ ’ ’\t’ ’\r’ ’\n’]"
```

Comments

The characters (∗ introduce a comment, which terminates with the characters ∗). Comments may be nested within each other.

```
COMMENT = "(\* [^ \*)]* \*)"
```

Expressions and Operators

Syntax Notation

In the syntax notation used in this manual, syntactic categories are indicated by *italic* type and literal words are indicated in **bold** type.
$\{expression\}$ indicates a required expression in braces.

An optional terminal or non-terminal symbol has the subscript $_{opt}$ appended, so that $\{expression_{opt}\}$ indicates an optional expression in braces.

Operator Precedence

The precedence of expression operators is the same as the order of the major subsections of this section (highest precedence first). Within each subsection, the operators have the same precedence. Left- or right-associativity is specified in each subsection for the operators discussed therein.

Primary Expressions

Primary expressions involving . , subscripting, and function calls group left to right.

identifier
An identifier is a primary expression, provided it has been suitably declared as discussed below. Its type is specified by its declaration.

constant
A constant of any of the primitive types discussed in Chapter 3 is a primary expression.

(*expression*)
A parenthesized expression is a primary expression whose type and value are identical to those of the unadorned expression. The presence of parentheses does not affect whether the expression is an lvalue.

Array Literal

$|expression_{opt}|$
| *expression-list* |
A string, which originally has the type "array of **char**", is a primary expresion. An array literal storing another type is also a primary expression.

Array Access

primary-expression[*expression*]
A primary expression followed by an expression in square brackets is a primary expression. The intuitive meaning is that of a subscript. The primary expression has type array of . . . and the type of the result is . . . . The type of the subscript expression must be a type that is convertible to an integral type, or a compile-time error occurs.

Function Call

primary-expression ($expression - list_{opt}$)
A function call is a primary expression followed by parentheses containing a possibly empty, comma-separated list of expressions which constitute the actual arguments to the function. The result of the function call is the function's return type. Recursive calls to any function are permissible.

Object Member Access

primary-lvalue . *r-value*
primary-lvalue: *identifier* | **this** | (*expression*) | *primary-expression*[*expression*]
primary-rvalue: *identifier* | *primary-expression* ($expression - list_{opt}$)
An lvalue expression followed by a dot followed by the name of a class member is a primary expression. The object referred to by the lvalue is assumed to be an instance of the class defining the class member. The given lvalue can be an instance of any user-defined class.

Unary Operations

unary-operator expression
unary-operator: **not** | **-**
Expressions with unary operators group right-to-left.

Logical Not

not *expression*
The result of the logical negation operator **not** is **true** if the value of the expression is **false**, **false** if the value of the expression is **true**. The type of the result is **bool**. This operator is applicable only to operands that evaluate to **bool**.

Negation

-*constant* | **-**(*expression*)
The result is the negative of the expression, and has the same type. The type of the expression must be **char**, **int**, or **float**.

Dynamic Memory Management

The **new** operator is used to allocate dynamic memory in two scenarios: array creation and object creation.

Array Creation

new *type*[*expression*]

Object Creation

new *identifier*(*expression*$_{opt}$)
new *identifier*(*expression-list*)

Memory Deallocation

delete *r-value*
The **delete** operator is used to deallocate heap memory. The *r-value* can be either an l-value or r-value of either an array creation or object creation expression.

Multiplicative Operations

expression multiplicative-operator expression
multiplicative-operator: * | / | %
The multiplicative operators group left-to-right. They operate on numeric types (**int**, **char**, **float**). If both operands are of type **int**, the result is of type **int**. If either operand is of type **float**, then the result is of type **float**. If either operand if of type **char**, then the result is of type **char**.

Additive Operations

expression additive-operator expression
additive-operator: + | −
The additive operators + and group left-to-right. They operate on numeric operands (**int**, **char**, **float**). The same type considerations as for multiplication apply. Overflow of a **char** type during an addition operation results in wraparound.

Relational Operations

expression relational-operator expression
relational-operator: < | > | <= | >=
The relational operators group left-to-right. They operate on numeric operands (**int**, **char**, **float**). The relational operators all yield **true** if the specified relation is true and **false** otherwise.

Equality Operations

expression equality-operator expression
equality-operator: == | !=
The == (equal to) and the != (not equal to) operators are exactly analogous to the relational operators except for their lower precedence.

Logical Operations

expression logical-operator expression
logical-operator: **and** | **or**
Both operands must evaluate to a value of type **bool**. The **and** operator returns **true** if both its operands evaluate to **true**, **false** otherwise. The second expression is not evaluated if the first evaluates to **false**. The **or** operator returns **true** if either of its operands evaluate to **true**, and **false** otherwise. The second operand is not evaluated if the value of the first operand evaluates to **true**.

Assignment Operation

lvalue = expression
primary-lvalue: *identifier* | {**this**|*identifier*} . *expression* | *primary-expression*[*expression*]
The value of the expression replaces that of the object referred to by the lvalue. Both operands must have the same type.

Statements

A statement forms a complete unit of execution. Most statements are expression statements and have the form
expression ;

So that several statements can be used where one is expected, the compound statement is provided:
compound-statement:
$\{statement - list\}$

statement-list:
statement statement-list

Control Flow Statements

The statements inside source files are generally executed from top to bottom, in the order that they appear. Control flow statements, however, break up the flow of execution by employing decision making, looping, and branching, enabling your program to conditionally execute particular blocks of code. This section describes the conditional statements (if-then, if-then-else), looping statements (for, while), and branching statements (break, continue, return) supported by the Dice programming language.

Conditional Statement

The forms of the conditional statement are:
if (*expression*) *statement*
if (*expression*) *statement* (**else if** *statement*)* **else** *statement*

The expression enclosed in balanced parentheses is evaluated and if it is **true**, the first substatement is executed. In the second case, if the expression evaluates to **false** and there is an **else-if** clause, then the substatement in the **else-if** clause is executed. If the expression evaluates to **false** and no **else-if** clause exists, then the substatement in the **else** clause is executed. As usual, the **else** ambiguity is resolved by connecting an else with the last encountered elseless if.

Looping

The while statement has the form
while (*expression*) *statement*
The substatement is executed repeatedly so long as the value of the expression remains non-zero. The test takes place before each execution of the statement.

The **for** statement has the form:
for ($expression_{opt}$; $expression_{opt}$; $expression_{opt}$) *expression*
This statement is equivalent to:

```
while (expression-2) {
        statement
        expression-3 ;
}
```

Thus the first expression specifies initialization for the loop; the second specifies a test, made before each iteration, such that the loop is exited when the expression becomes **false**; the third expression typically specifies an incrementation which is performed after each iteration. Any or all of the expressions may be dropped. A missing expression-2 makes the implied while clause equiva- lent to while(**true**); other missing expressions are simply dropped from the expansion above.

Branching

The statement

break;

causes termination of the outermost enclosing **while** or **for** statement; control passes to the statement following the terminated statement.
The statement

continue;

causes control to pass to the loop-continuation portion of the outermost enclosing **while** or **for** statement; that is to the end of the loop.
A function returns to its caller by means of the return statement, which has one of the forms:

return;
return (*expression*);

In the first case no value is returned. In the second case, the value of the expression is returned to the caller of the function. If a function has no **return** statement, then it returns with no returned value.

File Inclusion

If a .dice file contains a statement of the following form:
include(*expression*);
where the expression is a string literal that specifies the path to another .dice file, then all classes defined in that file are available to be used in definitions of classes in the .dice file in which the include statement appears. Include statements must appear before other types of statements in a .dice file.

Declaration Statements

Instance Field Declaration

A field declaration statement declares an instance field of a class and has the following form:
scope type-specifier identifier ;
scope: **public** | **private**
type-specifier: *type* | **class** *identifier* | **class** *identifier*[] | *type*[]
type: any primitive type in Dice
Note that this is the only legal format of a field declaration statement; assignment statements are not a valid way to declare instance fields in Dice.

Local Variable Declaration

type-specifier identifier ;
type-specifier: *type* | **class** *identifier* | **class** *identifier*[] | *type*[]
type: any primitive type in Dice

Instance Method Declaration

A method declaration statement declares an instance method of a class and has the following form:
scope type name (*formal-list*$_{opt}$) {*statement-list*$_{opt}$}
scope: **public** | **private**
type-specifier: *type* | **class** *identifier* | **class** *identifier*[] — *type*[]
type: Any primitive or non-primitive type in Dice, or **void**. If the *type* is **void**, then the method being declared returns no value.
name: **main** | *identifier*
Only one method per program may be declared with the *name* **main**.
identifier: Any identifier, exluding the following, which are names of built-in functions in Dice:

print	**input**	**malloc**	**open**
close	**read**	**write**	**lseek**
exit	**realloc**	**getchar**	

formal: *type-specifier identifier*
statement: *local-variable-declaration* | *expression-statement*
expression-statement: *assignment-expression-statement* | *function-call-expression-statement*

Constructor Declaration

A constructor declaration statement has the following form:
constructor (*formal-list*$_{opt}$) {*statement-list*$_{opt}$}
formal: *type-specifier identifier*
type-specifier: *type* | **class** *identifier* | **class** *identifier*[] — *type*[]
type: any primitive type in Dice
statement: *local-variable-declaration* | *expression-statement*
expression-statement: *assignment-expression-statement* | *function-call-expression-statement*

Class Declaration

A class declaration statement has one of the following forms:
class *identifier* {*cbody*}
class *identifier* **extends** *identifier* {*cbody*}
identifier: The *identifier* that follows the keyword **extends** must be the name of another class declared in the same program. The *identifier* that follows the keyword **class** must not be identical to the name of any other class declared in the same program.
cbody: {*statement-list*$_{opt}$}
statement: *instance-field-declaration* | *instance-method-declaration* | *constructor-declaration*

Program Structure and Scope

Program structure and scope define what variables are accessible and where. When inside a class, there are many different cases of scope, however those are better defined in chapter 7.

Program Structure

A Dice program may exist either within one source file or spread among multiple files which can be linked at compile-time. An example of such a linked file is the standard library, or *stdlib.dice*. When an include statement is executed at compile time, it will compile all classes in the included file along with the classes in the file on which the compilation was run. Therefore at compilation, one only needs to compile with *dicecmaster.dice*. If an included module defines a class that has the same name as one of the classes defined in the including module, then the compiler throws an error. The compiler does not resolve recursive includes; if *foo.dice* includes *bar.dice* and *bar.dice* includes *foo.dice*, the compiler throws an error.

A program consists of zero or more include statements, followed by one or more class definitions. Each class defined in a module must have a distinct name. Classes cannot have two methods with the same name regardless of the method's signature. Only one class out of all classes may have a main method, defined with *public void main(char[][] args)* which designates the entry point for a program to begin executing code. All Dice files are expected to end with the file extension *.dice* and follow the following syntactic layout.

Scope refers to which variables, methods, and classes are available at any given time in the program. All classes are available to all other classes regardless of their relative position in a program or library. Variable scope falls into two categories: fields (instance variables), which are defined at the top of a class, and local variables, which are defined within a method. Fields and methods can be public or private. If a field or method is public then it is accessible whenever an instance of that class is instantiated. Private fields and methods are only accessible within the same class.

Local variables are variables that are declared inside of a method. Local variables are only accessible within the same method in which they are declared, and they may have the same name as fields within the same class since fields in a class are only accessible by calling the *this* keyword.

Classes

Classes are the constructs whereby a programmer defines their own types. All state changes in a Dice program must happen in the context of changes in state maintained by an object that is an instance of a user-defined class.

Class Declaration

The syntax for declaring a class is in the "Declarations" subsection of the "Statements" section. According to the class declaration syntax, fields, constructor and methods are optional for each class and may appear in any order in the class body.
Methods may not be overloaded: For any method name, only one method per class may be defined with that name.
If no constructors are defined, the compiler defines a default constructor. Unlike methods, they may be overloaded. When the programmer declares an instance of the class, either a user-defined constructor or the default constructor is automatically called. It is a compile-time error to declare two constructors with equivalent signatures in a class.

Inheritance

Dice supports multiple levels of inheritance. The syntax for declaring a class that inherits from another class via the **extends** keyword is in the "Declarations" subsection of the "Statements" section. A class inherits the public fields and methods of all its ancestors. Constructors are not inherited.

Overriding

A class can override any inherited method by defining its own method with the same method signature and a custom body. Two method signatures are considered to be the same if they match on their return type and name and have the same number of formal arguments, with the sequence of types of their formals matching. Constructor declarations are never inherited and therefore are not subject to overriding.

Access Modifiers

Fields and methods must have one of the following access modifiers: **public** | **private**. If a field or method has a public access modifier, then it may be accessed by the method of any class in the program. Private fields and methods are accessible from within the class in which they are declared, but not from any descendant classes.
Unlike fields and methods, access to constructors is not governed by access modifiers. Constructors are accessible from any class.

Referencing instances

When the keyword **this** is used that keyword is effectively replaced with an instance of the containing object at runtime.

Grammar

Below you will find an entire grammar listing for our language. You will see several tokens that were generated directly from our Scanner. The following are the list of tokens and their associated regexes:

```
let alpha = ['a'-'z' 'A'-'Z']
let escape = '\\' ['\\' ''' '"' 'n' 'r' 't']
let escape_char = ''' (escape) '''
let ascii = ([' '-'!' '#'-'[' ']'-'~'])
let digit = ['0'-'9']
let id = alpha (alpha | digit | '_')*
let string = '"' ( (ascii | escape)* as s) '"'
let char = ''' ( ascii | digit ) '''
let float = (digit+) ['.'] digit+
let int = digit+

| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| ';'      { SEMI }
| ','      { COMMA }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| '%'      { MODULO }
| '='      { ASSIGN }
| "=="     { EQ }
| "!="     { NEQ }
| '<'      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "and"    { AND }
| "or"     { OR }
| "not"    { NOT }
| '.'      { DOT }
| '['      { LBRACKET }
| ']'      { RBRACKET }
| '|'           { BAR }
| "if"     { IF }
| "else"   { ELSE }
| "for"    { FOR }
| "while"  { WHILE }
| "return" { RETURN }
| "int"    { INT }
| "float"  { FLOAT }
| "bool"   { BOOL }
| "char"   { CHAR }
| "void"   { VOID }
| "null"   { NULL }
| "true"   { TRUE }
| "false"  { FALSE }
| "class"       { CLASS }
| "constructor" { CONSTRUCTOR }
| "public"      { PUBLIC }
| "private"     { PRIVATE }
| "extends"     { EXTENDS }
| "include"     { INCLUDE }
| "this"        { THIS }
| "break"                 { BREAK }
| "continue"        { CONTINUE }
| "new"                 { NEW }
| "delete"                 { DELETE }

| int as lxm                   { INT_LITERAL(int_of_string lxm) }
| float as lxm                 { FLOAT_LITERAL(float_of_string lxm) }
| char as lxm                  { CHAR_LITERAL( String.get lxm 1 ) }
| escape_char as lxm{ CHAR_LITERAL( String.get (unescape lxm) 1) }
| string                       { STRING_LITERAL(unescape s) }
| id as lxm                    { ID(lxm) }
| eof                          { EOF }

| (* *) {COMMENT*}
```

It should be noted that comments were handled to allow for nested comments. Therefore this cannot be captured strictly using a grammar, and instead is better shown in the scanner.mll documentation at the end of this document. The following grammar is the same as the grammar shown in parser.mly at the end of this document except it does not have the rules it will turn into regarding OCaml code. This is very similar to the syntax for ocamlyacc.

```
program:
        includes cdecls EOF

includes:
                /* nothing */
        |         include_list

include_list:
                include_decl
        |         include_list include_decl

include_decl:
        INCLUDE LPAREN STRING_LITERAL RPAREN SEMI

cdecls:
        cdecl_list

cdecl_list:
```

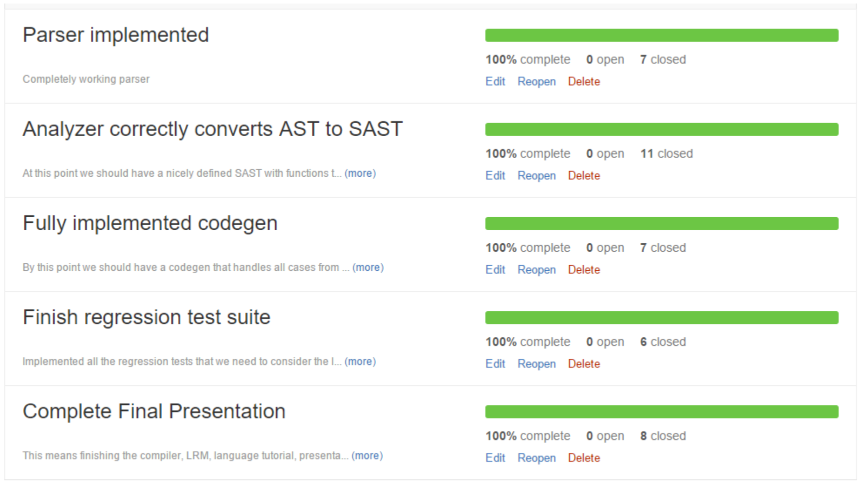
```
                cdecl
        |         cdecl_list cdecl

cdecl:
                CLASS ID LBRACE cbody RBRACE
        |         CLASS ID EXTENDS ID LBRACE cbody RBRACE

cbody:
                /* nothing */
        |         cbody field
        |         cbody constructor
        |         cbody fdecl

constructor:
        CONSTRUCTOR LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE

scope:
                PRIVATE
        |         PUBLIC

field:
        scope datatype ID SEMI

fname:
        ID

fdecl:
        scope datatype fname LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE

formals_opt:
                /* nothing */
        |         formal_list

formal_list:
                formal
        |         formal_list COMMA formal

formal:
        datatype ID

actuals_opt:
                /* nothing */
        |         actuals_list

actuals_list:
                expr
        |         actuals_list COMMA expr

primitive:
```

```
                INT
        |         FLOAT
        |         CHAR
        |         BOOL
        |         VOID

name:
        CLASS ID

type_tag:
                primitive
        |        name

array_type:
        type_tag LBRACKET brackets RBRACKET

datatype:
                type_tag
        |         array_type

brackets:
                /* nothing */
        |         brackets RBRACKET LBRACKET

stmt_list:
                /* nothing */
        |         stmt_list stmt

stmt:
        expr SEMI
|         RETURN expr SEMI
|        RETURN SEMI
|         LBRACE stmt_list RBRACE
|         IF LPAREN expr RPAREN stmt
|         IF LPAREN expr RPAREN stmt ELSE stmt
|         FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
|         WHILE LPAREN expr RPAREN stmt
|        BREAK SEMI
|        CONTINUE SEMI
|   datatype ID SEMI
|         datatype ID ASSIGN expr SEMI

expr_opt:
                /* nothing */
        |         expr

expr:
                literals
        |         expr PLUS   expr
        |         expr MINUS  expr
        |         expr TIMES  expr
        |         expr DIVIDE expr
        |         expr EQ     expr
        |         expr NEQ    expr
        |         expr LT     expr
        |         expr LEQ    expr
        |         expr GT     expr
        |         expr GEQ    expr
        |         expr AND    expr
        |         expr MODULO expr
        |         NOT  expr
        |         expr OR     expr
        |         expr DOT    expr
        |         expr ASSIGN expr
        |         DELETE expr
        |   MINUS expr
        |         ID LPAREN actuals_opt RPAREN
        |         NEW ID LPAREN actuals_opt RPAREN
        |        NEW type_tag bracket_args RBRACKET
        |         expr bracket_args RBRACKET
        |         LPAREN expr RPAREN

bracket_args:
                LBRACKET expr
        |         bracket_args RBRACKET LBRACKET expr

literals:
                INT_LITERAL
        |         FLOAT_LITERAL
        |         TRUE
        |         FALSE
        |         STRING_LITERAL
        |         CHAR_LITERAL
        |         THIS
        |         ID
        |         NULL
        |         BAR array_prim BAR

array_prim:
                expr
        |        array_prim COMMA expr
```

4. Project Plan

Planning Process

Throughout the project we embodied the principles of agile development. At any point in time during our development we had working code on the master branch and every member of the team was brought up to speed with what has been completed and worked on. All goals for the project were put on Github and as they were resolved they were cleared. We created several milestones which captured our goals for completing the parser, scanner, analyzer, codegen, and final report milestones. We also worked closely with Professor Edwards at Columbia University to receive guidance on how best to implement this language. The following milestones were created and cleared over the course of the semester:



Parser implemented
Completely working parser
100% complete 0 open 7 closed
Edit Reopen Delete

Analyzer correctly converts AST to SAST
At this point we should have a nicely defined SAST with functions t... (more)
100% complete 0 open 11 closed
Edit Reopen Delete

Fully implemented codegen
By this point we should have a codegen that handles all cases from ... (more)
100% complete 0 open 7 closed
Edit Reopen Delete

Finish regression test suite
Implemented all the regression tests that we need to consider the l... (more)
100% complete 0 open 6 closed
Edit Reopen Delete

Complete Final Presentation
This means finishing the compiler, LRM, language tutorial, presenta... (more)
100% complete 0 open 8 closed
Edit Reopen Delete

Figure 4.1: Milestoning on Github.

Specification Process

At the beginning of the semester we had originally intended our language to be a distributed software solution that would conveniently allow the developer to distribute tasks to various slave machines that had compiled the tasks to LLVM IR. After discussing this with professor Edwards we then decided to opt for an object oriented programming language that specifically compiled to LLVM IR. This way we as a team could learn more about making compilers and showing the power of LLVM.
Once we decided on the theme of Dice, we met to discuss the features we wanted most in our object oriented language. In our case we wanted arrays, inheritance, objects, and file IO to be some of the key highlights of our language. We then built up the scanner and parser to get a more solid idea as to what the language would look like, and by November 15th we had solidified our plans to implement the aforementioned features.

Development Process

Implementation was very dependent on the course deadlines. We started with the scanner and parser specifically so the language reference manual was better defined. This was completed by October 26th. We then iterated on the analyzer and codegen until it was capable of producing hello world. This was completed on November 15th. The month afterwards was spent implementing inheritance and arrays until they were finally completed on December 18th.

Testing Process

Throughout the development process we had numerous tests. The plan was to always have tests that were non-functional so a feature could then be implemented to get them working. If we encountered an error that we were unsure of how to fix, we added more error messages in our compiler until we could exactly pinpoint where the error was occurring. We also made a rule for our team to handle each and every exception that could occur as a custom error message to be printed out by the compiler.

Team Responsibilities

Team responsibilities were divided up and evenly distributed amongst the four group members. While we could not adhere to a strict division of labor based on group member titles, every member contributed to the codebase.

Team Member	Responsibility
David Watkins	Scanner, Parser, Analyzer, Codegen, Utils, LRM, Final Report, Latex, Code cleanup
Emily Chen	Inheritance in Analyzer, Expression types in Analyzer, LRM
Khaled Atef	Test Suite, Binary and unary expression evaluation in codegen
Phillip Schiffrin	Standard Library, Class map generation

Github Usernames

The following Github usernames correspond to the following group members:

- Emily Chen - six5532one, ec2805
- Khaled Atef - KhaledAtef
- David Watkins - DavidWatkins
- Philip Schiffrin - nethacker11

Project Log

To demonstrate our timeline we captured the number of git commits over time for our project.

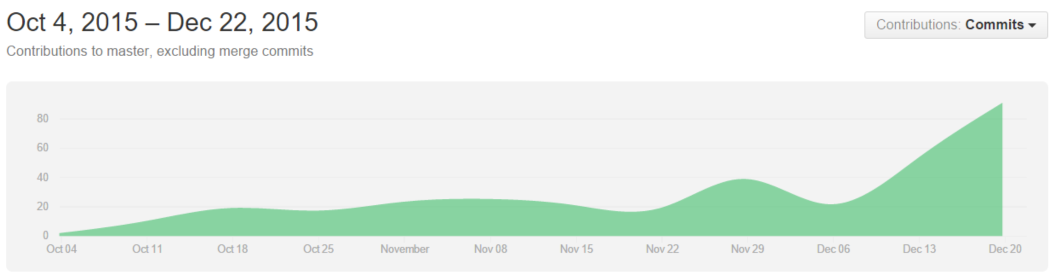


Figure 4.2: Commit timeline on Github.

The timeline shows that we have been diligent at constantly working on the project since the beginning of the semester. All group members have contributed to this project. The following issues are a list of git issues that were cleared as part of our project, as well as the person who closed the issue. We did not have a rule for who closed an issue so sometimes the person who completed the issue may not have been the one to close it.

- #71 Should not be able to access variables outside of scope
- #137 Awesome!
- #134 Subclass assignment [by @six5532one, @DavidWatkins]
- #133 string length tests
- #132 fix delete test, no multiple arrays
- #131 this should raise no exceptions
- #130 Expected stderr: ”exception Exceptions.LocalAssignTypeMismatch(”B”, ”C”)”
- #129 passing in an inherited class for classes
- #128 E-test-privateFieldsAccess.dice
- #127 Create test for cyclical inheritance
- #126 Add error message for assigning parameters
- #125 test-gcd.dice Bug. You cannot assign values to parameters
- #124 test-constructor1.dice is written incorrectly
- #123 Maximum float is limited to 6 digits after the decimal
- #122 char[][] args does not work in main
- #121 Test max/min floats
- #120 Test default constructor
- #119 Test overloading std-lib functions
- #118 Exit not working in runtime
- #117 Test args

- #116 assign ints to floats
- #115 Integer toString generates string twice
- #114 concat adds an extra character to the string
- #113 Test exit
- #111 Errors.log from script output isn't working properly
- #110 add teststdlib .out
- #109 Add Test returning objects
- #108 add tests for empty blocks
- #107 For inheritance of functions we should have an id to determine which function to call
- #106 Includes should check with String_lit not ID
- #105 Odd invalid numbering of blocks bug
- #104 Fix parameters on library functions
- #103 Get Dice exec working so tests can run again.
- #102 "Get the t-shirts made"
- #101 Adapt codegen to changes in analyzer that add inherited fields to sprogram.classes
- #100 Need to test includes
- #99 add test for empty conditionals
- #98 add empty for loop test
- #97 Add nested comments
- #96 test order of fdecl,fields,constructor in classses
- #95 primitive type limit tests
- #94 test constructors
- #93 test private scope function
- #92 Help needed: env.env_class_maps seems correct but exception is raised when I try to access an inherited field
- #91 default constructors
- #90 Need to add an environment variable to point to the includes
- #89 Strings need to be initialized and accessed differently from normal arrays
- #88 This should raise "UndefinedClass: H"
- #87 Use of Delete
- #86 add static scoping test
- #85 Add applicative order test

- #84 Add delete command to free memory
- #82 Add exit call
- #81 return statements in branches aren't recognized
- #80 dice executable doesn't run without any args
- #79 Kappa [by @DavidWatkins]
- #78 Add tests for recursion
- #77 Obj access [by @DavidWatkins]
- #75 Test invalid functions
- #74 Test multiple classes
- #73 Parent cannot have fields of type of its children
- #72 Cannot call return inside of a constructor
- #135 check for overridden methods takes ret type into account [by @six5532one, @DavidWatkins]
- #69 Casting rules questions
- #68 Kappa [by @DavidWatkins]
- #67 Floats print with extra trailing zeros. Kinda ugly.
- #66 Emily [by @six5532one, @DavidWatkins]
- #65 local decl (primitives): stderr should be "DuplicateLocal: myc"
- #64 object creation: this should raise no exception
- #63 object creation: this should raise no exceptions
- #62 Compiler doesn't allow formal to be an object
- #61 object creation: This should throw no exceptions
- #60 Object creation: this should raise "ConstructorNotFound: Foo.constructor.int.bool.char.float"
- #59 object decl without assignment expr: This should throw no exceptions
- #58 This should throw exception "UndefinedClass: Baz"
- #57 incorrect check for duplicate constructors
- #56 Emily [by @six5532one, @DavidWatkins]
- #55 Create arith tests that have signed values
- #54 Parser issue with reading user-defined objects.
- #53 Emily [by @six5532one]
- #52 Decide whether to promote all ints to floats in binops
- #51 Consecutive print statements don't work. Compiler only outputs first print statement.

- #50 Epsilon [by @six5532one]
- #49 Reorganize object accesses for functions
- #46 Kreygasm [by @DavidWatkins]
- #45 Add shakespeare and stephen number to tester
- #44 Create symbol table for cdecls, fdecls, fields
- #39 static analysis checks for variable access
- #38 use ‘new‘ keyword for object and array instantiation
- #37 support addition of chars and ints
- #36 Update LRM: support addition of chars and ints
- #35 Change parser array create type to type tag and not primitive
- #34 Evaluate whether to add new as a keyword to object initialization
- #33 Exceptions, try, catch?
- #32 Implement basic primitive expressions for codegen
- #31 Should we add continuous checking even when an illegal character/parser error occurs like java?
- #30 Add annotation for source code position to AST
- #29 We should evaluate whether we want to move variable declarations to stmts
- #28 Do we need to add an additional layer of abstraction from SAST to Codegen?
- #27 Complete pretty printing abstract syntax tree to Utils
- #26 How does LLVM handle allocating on the heap
- #24 Strings with escape characters are not being displayed properly
- #23 Create OCamlDoc Documentation
- #22 Should we switch the llvm package to ollvm?
- #21 Add file operator functions to Codegen
- #20 Write the File class
- #19 Write the String class
- #18 Write the Math class
- #17 Add support for utilizing line number and character number in Analyzer
- #16 Add class name and function name collission detection
- #15 Add testing for arrays
- #14 Evaluate the type of an expression in Analyzer.get_expr_type
- #13 Add testing for extends

- #12 Add mentioning of unary minus to LRM
- #11 Remove '-' symbol from regex in floats and ints of LRM
- #10 Convert AST.cdecl to SAST.cdecl
- #9 Convert AST.expr to SAST.expr in Analyzer.convert_expr
- #8 Analyzer.process_includes does not check absolute path
- #7 Delta [by @DavidWatkins]
- #6 Delta [by @DavidWatkins]
- #5 Special chars (tabs/newlines/etc) aren't getting tokenized properly
- #4 float limit
- #3 David fix [by @DavidWatkins]
- #2 Merge pull request #1 from DavidWatkins/DavidFix [by @DavidWatkins]
- #1 David fix [by @DavidWatkins]

Git Commit History

Here are all of the commits as performed by the team. Everyone contributed to the project.

```
commit dda28468a3b4742c94d7913c4bc6ff0b0a99bd90
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 23:38:27 2015 -0500

    works now

commit f3b2aa7577bc33ad86e4d0f9e6fdc7c006bd9bdc
Merge: 2a272c5 43b5f73
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 23:36:10 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 2a272c53b0cd524509e328ef9cd6792212db959b
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 23:35:52 2015 -0500

    New tarball

commit 43b5f73cf52535d79b4aee7ab869e4422eedf7da
Merge: 8687402 7cec151
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 23:29:43 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

```

```
commit 86874025d9fa49779ddce79a422fe48f81a0b324
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 23:29:19 2015 -0500

    Updated references

commit 7cec1517ad01b7373fb61978584885e74097b339
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 23:29:07 2015 -0500

    tarball

commit f27267ea354a2cc3a0cf2fb32b68971d6e8e1064
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 23:23:14 2015 -0500

    Updated

commit 71474ab6741c314246a7e5d115573f66ceee279c
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 23:14:10 2015 -0500

    Final version

commit 4ea7bf79afd1e7158cda2693406534979162c0f3
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 23:07:30 2015 -0500

    Fixed LRM

commit 483c33d46296f05db29a66e8ed5f04ca7bc10253
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 23:05:35 2015 -0500

    Fixed some tests

commit 0648fe1c5d2df25c899fa519db295f8509826962
Merge: fdee613 6191e9b
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 22 22:51:41 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit fdee61348f8641ac0c057a679c1790a027622ad8
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 22 22:51:10 2015 -0500

    Fixed tests to match new error messages. This should be the last time I touch these tests. I'm tire

```

```
commit 6191e9b1d1f37d1699dc1f2cc95d0bed8588ff8d
Merge: cc84574 4552c81
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 22:40:32 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit cc845741ba0de805ab5defdba8b9f83b48714b55
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 22:40:08 2015 -0500

    Finished Tutorial

commit 4552c811c8978550a7b910b665d1cfc5e700b3f1
Merge: cdb0e4f 5201688
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Dec 22 22:37:03 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit cdb0e4f2e8ac1d288b4f5490edb4f4049f1aec63
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Dec 22 22:36:22 2015 -0500

    fix type specifiers for non-primitive types

commit 5201688e6ceff8a3d48e7d912297b6ad52c2fc3c
Merge: 5b88cc0 275a5f0
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 22 22:28:00 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 5b88cc01e1bf17b11db1fa57d990612867a8f618
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 22 22:27:50 2015 -0500

    Removed the 812981298129

commit 4bd7f0a61871fca70425d7d7b7cd82ec8d36204e
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 22 22:27:06 2015 -0500

    Added to Tutorial

commit 275a5f012f3a70dc3a2ff2dcb9e804cb54a29fa9
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 22:25:54 2015 -0500

```

```
    Done with Architecture

commit cfd9dee0069b201d9bed13089afb22becf973bca
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Dec 22 22:20:39 2015 -0500

    updated Classes

commit 7943a55f63ac9acb77e4e6eef73060146d319bf2
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 22:10:11 2015 -0500

    Look at me

commit 46e4516edd6d72f4a4f3a451eae9de63d51b48e0
Merge: a97dddc 65f9ed5
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 21:49:57 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit a97dddc5745f0db4baddd426a61ef0aaa45c59d3
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 21:47:01 2015 -0500

    Added new stuf

commit 65f9ed5e72520c2f856ada1ef21bfa84d9261676
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 22 21:41:10 2015 -0500

    updated Architecture.tex for code generation

commit 7452f7b962d3324283af73b5adb8a22fe435072b
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Dec 22 21:27:16 2015 -0500

    add restrictions to class name, method name

commit e1797fbd385bc059ab83150c5813abda31789912
Merge: fbf4ba9 738b055
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 22 21:22:42 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit fbf4ba9a85e6a17419bfdff5c93cdbf592ece31a
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 22 21:22:26 2015 -0500
```

```

    updated Architecture.tex

commit 55ca1d128472a5cb1451784d89be2e174d455ced
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Dec 22 21:10:47 2015 -0500

    mention built-in functions as reserved

commit 4120bbefa038a2a3f7fda5c682e286e7d71327aa
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Dec 22 20:58:29 2015 -0500

    rewrote statements, expressions sections

commit 738b0558ddb9fe894a7611be0f1f9f590f38094a
Merge: 700e197 df6915a
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 20:45:33 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit 700e1979474d977ffb3496c7435f4f9dbace09e2
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 20:39:19 2015 -0500

    Added changes to standard library description

commit df6915a7d7a802e673daca3a6b364060b024035b
Merge: 8ac36b8 dbf27d2
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 22 20:34:58 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 8ac36b8a6d9714d1f096f8c8e990da9bd971afe7
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 22 20:34:51 2015 -0500

    CFuncs.tex added

commit dbf27d2d940c93f0de61a210f98167faefeae014
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 20:28:22 2015 -0500

    Added grammar and small changes to lrm

commit 421588dcb8b30f42134c880143492e4822dbba2e
Author: nethacker11 <philip.schiffrin@gmail.com>
```

```
Date:   Tue Dec 22 20:23:33 2015 -0500

    added Builtin.tex

commit 0ec68907641d9be8a992b3dd7b023ec8e4f48afc
Merge: ef75162 ea3b98f
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 19:46:19 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit ef75162bd113e48a2ba794aa7ce002b613eeae3c
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 19:45:53 2015 -0500

    Added additional code for test plan in final report

commit ea3b98f6be0be4f8a66158b94d8391cd7b719948
Merge: 8524dfd 6378550
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 22 19:45:41 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 8524dfd397ddf421dcf7c7bd948649a825c355f5
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 22 19:45:33 2015 -0500

    updated standard library in Library.tex

commit 63785501516be9117a65f4bc0908396b5496058c
Merge: 48d7e07 035c054
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 19:12:36 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit 48d7e0772b5cfe8e899efecb622041b198199497
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Dec 22 19:12:14 2015 -0500

    Added additional stuff to proposal and tutorial

commit 035c054a00bf0ccc1b2b8d2dc1809f6fbab4dc08
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 22 19:11:12 2015 -0500

    Added Test Plan and Khal lessons learned to Final Report directory

```

```
commit 39a768eca63505299bfacc07eef0322753a4de64
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 22 18:59:25 2015 -0500

    updated Syntax.tex for final report

commit 41e9106396bb0b2e693dd35bfd131151c7c1b641
Author: David Watkins <djw2146@columbia.edu>
Date:   Tue Dec 22 18:28:54 2015 -0500

    ADedd more stuf

commit c58d595f376df552bb65e1fdb33ec05a674eb8cd
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 18:23:32 2015 -0500

    Added Demo_Animals to tex file

commit afa84191ecd40be39d14295c6c1f3fa25e7be6f6
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 18:19:54 2015 -0500

    Fixed hello world demo breaking tests

commit 7e2a1b9e07040cb9929b5dc971a297c83b0a9fe1
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 18:14:31 2015 -0500

    iejsiu

commit ab07735004b3f480677e269e65e9f009e9f10bdb
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 18:12:52 2015 -0500

    ijij

commit b21f0885522047bf0a62afb4da5edb292958ade4
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 18:11:09 2015 -0500

    Maybe this works?

commit 727a2a262837bcf87371f1d5313e065b5a855d36
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Dec 22 16:36:04 2015 -0500

    finished updating expressions

commit 5dd98b2548b8718ebf8342ef3460bfa740a6ffad
```

```
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 15:20:54 2015 -0500

    Fixed another bug

commit d6b49aae775f433bc4c733ef539f9d5b84605c6f
Author: David Watkins <djw2146@columbia.edu>
Date:   Tue Dec 22 15:19:31 2015 -0500

    updated code.texY

commit d92d49c30ebe2da66203d0eb28db41d78b0d9ec5
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 15:17:11 2015 -0500

    Fixed tests

commit cfebb0d5104705df4e358b35879645c6f5190439
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 15:09:13 2015 -0500

    Fixed section title on tests

commit b8e048f0a326c3fc4fccee0a99928aa0564f8233
Merge: e94920a 07ee0b6
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 15:06:33 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit e94920ae5f16643a0f5ae85393d7cae7e8dc58f5
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 15:06:16 2015 -0500

    Added code for adding tests to final report

commit 07ee0b6cc870f6a7c171d159a85f8c142807f6f7
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Tue Dec 22 13:59:47 2015 -0500

    Update README.md

commit a16003fbdec97727c492c857335bc93478a50a70
Author: David Watkins <djw2146@columbia.edu>
Date:   Tue Dec 22 05:15:01 2015 -0500

    Added basis for final project report

commit f3e5fe83dae72565f2950c096c6ff0efecb1b567
```

```
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 22 04:46:38 2015 -0500

    Need to fixed error tests

commit e16dc0448ac4444fe75f7fee46b10825fda2ba6d
Author: David Watkins <djrival7@gmail.com>
Date:   Mon Dec 21 20:14:33 2015 -0500

    Added presentation

commit 0bc2d56336f2bed25b1715a1b5c632a49147eea8
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Dec 21 15:25:43 2015 -0500

    Logo modified

commit d39a5d9feb9ba50426b6caa3c32668ab57c410c5
Author: David Watkins <davidw@tkins.me>
Date:   Mon Dec 21 14:23:01 2015 -0500

    Finished demo code

commit c0ccf162f43b88ef2c732de15acd419250e5db5c
Author: David Watkins <davidw@tkins.me>
Date:   Mon Dec 21 14:21:42 2015 -0500

    Removed unecessary files

commit a03afb187f7b93c8c05874e1357975d3edf69fac
Author: David Watkins <davidw@tkins.me>
Date:   Mon Dec 21 13:55:16 2015 -0500

    Fixed the demo

commit eec6e6f7989d4022ac261cc453bb7646e84e0a69
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Dec 21 07:31:49 2015 -0500

    input/output coordinated

commit ca6abe8eeda764edfa1c2abd2bce730619ee53c9
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Dec 21 07:02:23 2015 -0500

    basics implemented for demo

commit 8d2eda8d25c81a0294b3cc52c285c76314600870
Author: Khaled Atef <kaa2168@columbia.edu>
```

```
Date:   Mon Dec 21 06:23:25 2015 -0500

    modified ascii art for demo

commit 0a3a0c3958e224b1883714e99bd317624dd5514b
Merge: 96d30dd 2437414
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Dec 21 06:18:42 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 96d30ddc28576c7013902f157a5435315967ddd1
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Dec 21 06:18:36 2015 -0500

    file for demo

commit 24374142973e158c61ea3955ac8d963599a2b75d
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Dec 21 05:57:03 2015 -0500

    Othello still broken after many compiler errors

commit b5fbea0a2101e0c18d6bc476f0e7dfc18539c356
Merge: bff1792 502eff9
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Dec 21 02:39:56 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit bff17927857bd562451279c9109ba57f01469829
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Dec 21 02:39:00 2015 -0500

    halfway through translating OthelloGame

commit 502eff9a39c369dcd131c4b36220018c0e16fbc4
Merge: 4da809d 79744e6
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Mon Dec 21 02:35:23 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 4da809d3964870b705e10f8126e77e80c152474f
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Mon Dec 21 02:34:47 2015 -0500

    updated humanplayer, doesn't work

```

```
commit 79744e6e61a16d7e049323d5af621e6be2049bb6
Merge: 1086a20 76df32a
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Dec 21 02:10:20 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 1086a2003fcf4604b4b799b3c3e18cbb05901b48
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Dec 21 02:10:11 2015 -0500

    First round of edits to parserScanner regex rules

commit 76df32ae8b70759eeddb134f57b8e3f6403e2e5f
Merge: 8a75b65 fb0a776
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Dec 21 02:08:18 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 8a75b65ddc464749d36e7998dcd243e8ef47b241
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Dec 21 02:07:45 2015 -0500

    includes classes HumanPlayer, Player, LocationObj

commit fb0a7763290ca205303a36e595792cabc8bda14b
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Mon Dec 21 02:04:24 2015 -0500

    updated demo files

commit a7e0a84173eee4c06f0413a7b8bde8c3a3ee1844
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Mon Dec 21 01:10:57 2015 -0500

    updated demo stuff

commit c5882be1259eee843e06004c347cc1d047c79851
Merge: e91324a 15fe681
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sun Dec 20 23:38:05 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit e91324aef67a7876f967e35b4f4a6ca323af95f7
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sun Dec 20 23:35:45 2015 -0500

```

```
    added toInteger in stdlib

commit 15fe681f3b48135f96cfcf0c191bd6989b76fad9
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 22:19:03 2015 -0500

    125 tests working!

commit 9dc00916011d9c69d13ff247268e615c2b0ac122
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 21:50:00 2015 -0500

    OthelloRunner Basic working

commit 9451871b5f68a79f41c4c463894b0cb6cf802b1f
Merge: e6007de bed598a
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 21:21:19 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit e6007de0f670b43d7ff183860c77b95e0d381b99
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 21:21:04 2015 -0500

    first draft Othello

commit bed598a8d60c21c69228029a024e7a5c3526c77d
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 21:09:58 2015 -0500

    Got object access working

commit d82e1a593479bd9dd04454014feedfa7dab7f0b4
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 20:53:29 2015 -0500

    fileio test output works!

commit f000aa8d545bb8450340105b070501e9c242bcf1
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 20:50:32 2015 -0500

    removed delete test

commit 9a1f7cde27e9c688ec84ad76385e27ffd1e7dcb1
Merge: 4c82a21 41949c7
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 20:45:44 2015 -0500
```

```

    Merge branch ’master’ of https://github.com/DavidWatkins/Dice

commit 41949c76776af134beb6de2a473e3e869403a2d5
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 20:45:29 2015 -0500

    Modified output to match test

commit 4c82a21756ba8abf9aa149d16f9b949e4b3f80c4
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 20:45:17 2015 -0500

    test-fileio now prints and writes itself

commit f86d9cb3250e36ac60bcdc42d65fce9d63bfda90
Merge: 39fea6b 0d28a10
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 20:40:33 2015 -0500

    Merge branch ’master’ of https://github.com/DavidWatkins/Dice

commit 0d28a10d1ae9333877cdadd0f7eb7c99a587d561
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 20:39:55 2015 -0500

    Fixed file io

commit 39fea6ba072e0eb973deadf72c28dc70140432c3
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 20:23:11 2015 -0500

    new tests

commit f989f8fcd03394dd759d65be5fa93406e7300fe8
Merge: ea1fc65 83d8ac3
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Sun Dec 20 19:17:14 2015 -0500

    Merge pull request #135 from DavidWatkins/fix-overrides-check

    check for overridden methods takes ret type into account

commit ea1fc652a4bdde559280c96e38a01cd5ac165783
Merge: 0c7039c 3163d40
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Sun Dec 20 19:16:39 2015 -0500

    Merge pull request #134 from DavidWatkins/subclass_assignment

    Subclass assignment

commit 0c7039c8d05f1a359ce8af67ed3fc0c581770539
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 19:11:32 2015 -0500

    Fixed assignment of obj_access problem

commit 6aeaa4c8a0d3fe6852c80263c918334a0d22dc06
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 18:51:03 2015 -0500

    Fixed stringClassReverse

commit 37ac35175eb27c39665b4bf77ee71d4a566bab4a
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 18:26:16 2015 -0500

    Added array access on obj_access

commit 83d8ac3fa9a130f8667cd6cf82691e8738bc94d4
Author: Emily Chen <emchennyc@gmail.com>
Date:   Sun Dec 20 18:12:23 2015 -0500

    check for overridden methods takes ret type into account

commit 15d429843e5c9a584fa4914936df1ba3783b212f
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 18:05:48 2015 -0500

    Fixed array create initialize

commit f2390b94a80cfff1c217b533cacf61d954bdfac3
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 17:49:22 2015 -0500

    tests...

commit 3163d400ace38ecdc60f41b643a27b9fa60dcd26
Author: Emily Chen <emchennyc@gmail.com>
Date:   Sun Dec 20 17:44:45 2015 -0500

    fixed formatting

commit ab4a07e9e55a5ce2db8f30782faa018b0762a53a
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 17:39:11 2015 -0500

```

```
    Changed function naming collision schema

commit e91e642ad5fbfd8a64bea0b5e2295aaeb3ff4145
Author: Emily Chen <emchennyc@gmail.com>
Date:   Sun Dec 20 17:20:20 2015 -0500

    fixed subclass assignment not to raise exception with reg object creation

commit dba6456b40bf8fc2c032b34984c210c27352a4e2
Author: Emily Chen <emchennyc@gmail.com>
Date:   Sun Dec 20 16:48:52 2015 -0500

    checks subclass assignment

commit 0b512528037bec86727f7e721a08d636759ef845
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 16:45:20 2015 -0500

    more tests and fixes

commit dc3d893e18172bfa7fdb9733fb9990b22f26a3dc
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 16:12:12 2015 -0500

    cyclical inheritance test added

commit 00009886c90714b113bd2e9066df7c0314fe99be
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 15:52:35 2015 -0500

    inheritance object passed in arg test

commit 79585bfacf986d5b013396ecdea2c4ce1f078edd
Merge: ae4bcc4 b5d6640
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 15:41:22 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit ae4bcc4ec6860484529e4431d96531ce245a3823
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 15:40:50 2015 -0500

    Fixed way accessing inherited methods checker thing grammar english pls

commit b5d6640ecfe55fa20bc69d109be8ef38cb2df82a
Merge: 777db46 da9452f
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 15:30:29 2015 -0500
```

```

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 777db465f5de4f9ade562b56254806d86f884f88
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 15:30:18 2015 -0500

    more tests

commit b15dd23dd09a127b4b45eeef83bc8f284c86f3de
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 15:02:14 2015 -0500

    tests =0

commit da9452feecda712b24ae53419fc3858db4f7ffbb
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 15:00:04 2015 -0500

    Fixed empty main problem

commit 7d23e2a16c131048d43fafa146b577ca5f18a8fb
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 14:52:01 2015 -0500

    fixed tests

commit dddd825bf32500fdd232c563c41b77a3e4426c44
Merge: 6b689f2 46d105a
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 14:51:10 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 6b689f2c8446921678637a0d876c4411bbaa360b
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 14:50:51 2015 -0500

    Added casting to subtypes

commit 46d105aef7000673550854485f86d0359b0c8b00
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 14:39:13 2015 -0500

    more tests including cyclical includes

commit 81392df3b88074c974fe897d35ee65b3cfe026d4
Merge: 9ace750 9301a8c
Author: nethacker11 <philip.schiffrin@gmail.com>
```

```
Date:   Sun Dec 20 14:06:38 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 9ace75050be810f9e0e460d47c409e972aaaa990
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sun Dec 20 14:06:23 2015 -0500

    added 2 tests

commit 9301a8c8bebadeb4cf67f4199b1084c9d25107b3
Merge: f9503b9 20c6b6c
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 14:05:44 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit f9503b95b010f8c9516093fe1b9cac3f6e8a7f3c
Merge: df64b34 f17b85f
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 14:05:29 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 20c6b6c16425120bbe1da4d355178c054b384698
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 14:01:26 2015 -0500

    more tests passing

commit df64b347fd6e07abb2d4f0834da862231ff35cba
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 13:49:00 2015 -0500

    Added some broken stuff

commit f17b85fedaf22ff07158c044185229a9d96f4f13
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sun Dec 20 13:46:57 2015 -0500

    added getchar()

commit 034b4a4e8a56c49e0de21385534708706f88f3af
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 12:58:20 2015 -0500

    Functions now have working private scope

commit fef6f2a5139dd5dda3d0d00cb349898d584ac0da
```

```
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 12:32:55 2015 -0500

    main args is now working

commit 47a6d182878aa980a372554b5eb7bd331cf60e7f
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 11:26:54 2015 -0500

    Break and continue now work

commit a9be4f6c34ee4230620875dc92bd7f7489d66c5f
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 10:01:28 2015 -0500

    Added code for checking if break or continue is valid

commit 795773d726798b0b7d698e35293f4ee76c2acdf4
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 09:37:20 2015 -0500

    Added basic private checking, not working for inheritance

commit 2e1c681369eb3397f0de724572cdf413988efbaa
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 08:54:08 2015 -0500

    Added casting at the beginning of overridden function

commit ca425b48bfa72b4f26d4f2be8bc92f69a4cb4fdf
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 08:35:36 2015 -0500

    Added default constructor

commit 98e3f63c3121a86e40c4445ff4bdd7f7dff36893
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 08:06:45 2015 -0500

    Virtual function resolution works

commit 145101c510c43fb8809e5fe2ccdd7de2e8ece722
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 06:56:25 2015 -0500

    Added working vtbl

commit 21f7e5cc757e7f94f3d41e71c95590188119a15b
Author: David Watkins <davidw@tkins.me>
```

```
Date:   Sun Dec 20 05:26:16 2015 -0500

    Cleaned up use of types in SAST

commit 064f098e6ced5aa733a3beabf8edd3dda5173db3
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 05:12:03 2015 -0500

    Added unused integer to all scalls

commit 9ee2d0ef828eff03f3acd0ed117610481d012135
Merge: 2042484 76746fd
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 05:01:45 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 2042484a2a9e8778eb1c4a86c00cb0ba8e5e0625
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 05:01:23 2015 -0500

    Incorporated Emily's changes to Analyzer

commit 76746fdb001845cb72dd757f870fc985b4f2261a
Merge: fa8e2ee c0eedeb
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 03:20:05 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit fa8e2eea360f9b10e068fa1937317cafb003df12
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 03:19:53 2015 -0500

    more tests

commit c0eedebd7866f602cd79bf581ba5030f5a9a53e4
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 20 03:15:15 2015 -0500

    Reformatted some code, fixed exit bug

commit 0a275a096762f01c506384a281c827a0689e8ab5
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 02:20:27 2015 -0500

    modified dice.ml to pass exceptions

commit d61f20801707ee4ac695135909823b3ee4b09073
```

```
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sun Dec 20 00:18:20 2015 -0500

    took out print stmt in stdlib

commit e1bc841aa24a9ef597e94232e04735c26c4276cd
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Dec 20 00:06:16 2015 -0500

    More tests =)

commit 60a80460f04a4ffe25d7bbe319734bab7c8ebc82
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 22:45:15 2015 -0500

    fixed concat in stdlib

commit 7ad7480ee90a8759271b0961507d7f084990a162
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Dec 19 21:25:12 2015 -0500

    Added more tests and modified dice.ml to account for an exception to make the test script work

commit 1eeea68662d793173c0dd4587cd244eb379e3176
Merge: 3529056 50a7529
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 17:20:15 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 3529056aae15850c8e3ce00eb314e0393d5a1ff3
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 17:19:43 2015 -0500

    Added changes to allow for exit

commit 50a7529746b3b7488fb038d75817983dcef56713
Merge: d2b04d3 3fd9fbf
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 17:16:14 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit d2b04d339c239b0000ccfdde4b93ee3bfe13a878
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 17:15:51 2015 -0500

    updated stdlib to include Integer and String has reverse()

```

```
commit 3fd9fbf47a382b0c8bc02e6f13e32c810f7f9807
Merge: 8ac9eed 14e1b19
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Dec 19 16:46:38 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 8ac9eed3f00065424b59350a074749246d411869
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Dec 19 16:46:20 2015 -0500

    more tweaks to tests and script

commit 14e1b190bfb5972a1a0394a23be43f178eef971b
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 16:31:22 2015 -0500

    Fixed codegen for char_lits to i8_t

commit d984aff231ee6eb90e5921994f5d6fd14e044a79
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 16:19:40 2015 -0500

    added test cases and updated stdlib

commit ff79fff82264ba8743377b3515514df0a988d7fc
Merge: b336d0a 602dc41
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 15:57:18 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit b336d0a6333387906a3a63d44541953e1c6a4616
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 15:57:02 2015 -0500

    Added modulo

commit 602dc4179efe1a87778934fb87428ddd5ee72d90
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Dec 19 15:55:55 2015 -0500

    corrected tester script to account for errors from exception tests

commit cedf61d44d4d5a1faf2424eb50cf983df9df22f3
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 15:24:25 2015 -0500

    Fixed function element access
```

```

commit 664bef08cd785fcd8f862874acfce3ced40bc5d2
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 15:16:10 2015 -0500

    added stdlib2 test and updated stdlib

commit f4a81c401d29969e5b341c99f2e68e003318bb2e
Merge: 285aa85 3b7465c
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 15:14:14 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 3b7465cf892745766ce5a4bf08e4fdbdb28468eb
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 15:13:45 2015 -0500

    This time for sure!

commit 285aa8594fe6d3b1fea5e2983e5599cc19bec253
Merge: 3425edc d7ed17e
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 15:10:53 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit d7ed17e991fa0322eac0a63f0b55c84d4e2c1115
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 15:10:09 2015 -0500

    Fixed function param passing bug

commit 3425edceaa05209d8f57da67867dac753d0ea0bc
Merge: 41afbc1 97de937
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 15:02:04 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 41afbc17bf2426ee44dc27bd254b550edbd78245
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 15:02:02 2015 -0500

    updated codegen for lseek

commit 97de93788f1701bcc7e334d8061fe58fca6a5d35
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 15:01:22 2015 -0500
```

```

    Fixed codegen_call for lseek

commit 3d58076d10a7060f85dfb363ec5cbce759038257
Merge: 7413f9b 464fc4c
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 14:57:34 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 464fc4c5d6119034866a6118cce83e59a56b3520
Merge: 87f4d52 7a63abf
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 14:55:19 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 87f4d52f2e6d185d46bc29b8635c6f98d7eb7853
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 14:55:02 2015 -0500

    Added lseek syntax to analyzer

commit 7413f9b0e14a20d00314556df6e6c4890fd243f3
Merge: 3c1c15b 7a63abf
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 14:22:33 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 7a63abffd7edafb87ecf82df2225e7ea2148eeb8
Merge: c482260 afae098
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Dec 19 14:22:02 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit c48226078ee263889c6de75ff8d0f6f572a6a7ee
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Dec 19 14:21:41 2015 -0500

    added stdlib string

commit 3c1c15b99113b0e570fa6625ba4a2a0ee1c917e5
Merge: 480dc4d afae098
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 14:19:44 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice
```

```

commit 480dc4d0c15a9c5cd5bccfb7c8d05aebb423b9e7
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 14:18:18 2015 -0500

    changed stdlib

commit afae098e32e66e69b0349e9809ce6d237f451179
Merge: acbea61 404c6df
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 14:17:52 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit acbea6113ddccfa59ce06c8288a4bfe81b134f6f
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 14:17:31 2015 -0500

    Fixed right associativity of parser

commit 404c6df62cc80b61ceffed8cc666f9591757d5e0
Merge: 782ca3f 3e4e5e6
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Dec 19 14:15:07 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 17c1362a3d24b7edf544491948a259fb816524a4
Merge: c248f39 3e4e5e6
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 14:15:07 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit c248f394794dc1a26b0052db05a4a2abffe5ba89
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Dec 19 14:15:05 2015 -0500

    updated stdlib

commit 782ca3fa5c903b0c87d7402724934c25a3cf3a30
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Dec 19 14:14:49 2015 -0500

    modified tests

commit 3e4e5e6b27248dbe9de6af579040dbc991f2b5be
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 14:13:16 2015 -0500
```

```

    Fixed array access for chars

commit cbcdff6c41b458da3355bf3aecb58a5d3549752e
Merge: 3ca5e39 0c9870c
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 13:54:44 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 3ca5e39a56c7a6c239d38e9c58eabd03304f1526
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 13:54:14 2015 -0500

    Fixed array acces for strings

commit 0c9870c3948b1e193926f363ec551830d8aae9ae
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Dec 19 13:54:04 2015 -0500

    added more tests

commit 91c9bc47dff55afd6269202ad1654145cf55b5da
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 05:07:25 2015 -0500

    Fixed stdlib

commit c603715b9036aa50daa30a423ee6e0b30fd9e8ce
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 04:08:52 2015 -0500

    While loops work

commit 27b53ff8e9131b2e686ed29755d54690936a2131
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 04:02:09 2015 -0500

    Fixed bug with array length

commit 64b72feeb55e71b92c1fd7810e5ccb82ae736f41
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 03:39:57 2015 -0500

    Fixed odd incorrect ordering bug

commit 170e4fd2e2285c0d7f106426651199a48c5b20e6
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 03:34:00 2015 -0500
```

```

    Fixed includes bug, fixed char array assignment of int length

commit 7c8d274ea55d5118e70db8f3d11dd5cff42d36e4
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 19 01:36:29 2015 -0500

    Migrated files and folders to appropriate place for new makefile schema

commit a1ae8ffbc1d1fe84c755abf98a44392680a63c20
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Fri Dec 18 22:57:30 2015 -0500

    updated stdlib and analyzer and codegen for built in functions

commit 1a5244813f0c299c673096a48a09dad022133599
Author: David Watkins <davidw@tkins.me>
Date:   Fri Dec 18 20:01:52 2015 -0500

    Fixed \0, its now \000

commit a2d07124a44c96af5b158996c049fead07644dc5
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Fri Dec 18 20:02:58 2015 -0500

    updated stdlib.dice

commit e9c8d476beb76ebd9a4f4d1a23f5cf722d741744
Author: David Watkins <davidw@tkins.me>
Date:   Fri Dec 18 19:47:00 2015 -0500

    backslash zero yo

commit d6be8f34690274401b8123cf491254274e8030b9
Author: David Watkins <davidw@tkins.me>
Date:   Fri Dec 18 19:33:09 2015 -0500

    works now?

commit 8ad670e00d5b7cf8020581861306cf89ab17b8a6
Merge: aec396d c6af1ee
Author: David Watkins <davidw@tkins.me>
Date:   Fri Dec 18 19:17:00 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit aec396db7c9a6714ce6e5de976596b42c1d03c8e
Author: David Watkins <davidw@tkins.me>
Date:   Fri Dec 18 19:16:41 2015 -0500
```

```

    Works *crosses fingers*

commit c6af1eecd3362b57591589d61493c14707c11479
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Fri Dec 18 19:13:08 2015 -0500

    updated stdlib.dice

commit b0e033a148286f9de9c2cef0b37c799fb5ec36d0
Author: David Watkins <davidw@tkins.me>
Date:   Fri Dec 18 18:43:07 2015 -0500

    So uh, nested comments are a thing

commit 0e91f6aca66d2804747918f460114f356842befd
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Fri Dec 18 17:37:31 2015 -0500

    Exceptions folder created, need to add more tests here

commit 643197852baaf3fff864761ab7376bf32e6bacf0
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Fri Dec 18 17:12:04 2015 -0500

    added stdlib.dice, passes analyzer but not tested

commit b9c354db5a56e4d8e9543a1c00147260283e5d51
Merge: 75cb0da 5ae669c
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Fri Dec 18 03:46:41 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 75cb0daf1e69f062f9cb1e6c66079639ededd3e0
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Fri Dec 18 03:41:15 2015 -0500

    modified test script

commit 5ae669cf25734ab2bdfb6c989bfd933b98bdebb9
Author: David Watkins <davidw@tkins.me>
Date:   Thu Dec 17 19:26:41 2015 -0500

    Works?

commit 1cfe2ae2cf20eb203f45617097d9daa93abf3793
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Thu Dec 17 19:24:59 2015 -0500
```

```

    added write function

commit 013f06fe8fcbf6d8db7dcf2cd32af311d47b7f2c
Merge: 4554586 b0dcfe9
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Thu Dec 17 18:59:31 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 4554586421327badd3daf2acb2c212fb98303a53
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Thu Dec 17 18:59:29 2015 -0500

    added more build in function declarations

commit b0dcfe9708793c4946a123dbf45afea8c305027e
Author: David Watkins <davidw@tkins.me>
Date:   Thu Dec 17 18:58:19 2015 -0500

    Fixed shift/reduce, added linking of c functions

commit 9c7a140e1e036a70bb4af3159d21265a2799bcaf
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Thu Dec 17 18:06:58 2015 -0500

    added c function declarations in codegen.ml under built in functions

commit d04c2b99e7c467914839a1b6429d7284d8c78725
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Thu Dec 17 17:37:41 2015 -0500

    added folder for c library extensions for .bc files to be linked in dice.ml

commit d058e9c00fc86b609da0dce4a906b10718ca3430
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 16 16:55:34 2015 -0500

    Added delete command to free memory

commit 9414ee274b553debcc02a052fff0fd34e46e14e8
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 16 16:29:17 2015 -0500

    Added multi-dimensional c code

commit a08e96f67a96dd181abf6c67b769d371c326fa03
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 16 16:28:52 2015 -0500
```

```

    Array length working, also added multi-dimensional c code

commit 59e4b9b012b92b799cdac22849c667130731163a
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 16 01:41:37 2015 -0500

    Array primitives work

commit 3ab1e0ff494e1bdc57460aa3d930b5f15fe3c0a6
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 15 23:45:42 2015 -0500

    Fixed single dimension arrays

commit f4ccfe7371bdd8c051db4735db872885a0578f42
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 15 22:20:45 2015 -0500

    build_array_malloc in progress

commit 3e27ec7a42f5d91620f8c16390cf53a82f9e858f
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Tue Dec 15 19:20:37 2015 -0500

    changing to single dimensional arrays, compiles but looks like arraycreate is not accessed again

commit 10e87f3b9c82258c06247f972144d63f582dbc4c
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 15 18:44:34 2015 -0500

    Working status

commit c71bfa88710ef0a7c39f98fb4ece382a6dbb877c
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 12 19:04:56 2015 -0500

    ArrayCreate doesn't work, added code for array deref

commit b9ed042660504b766617f397d21c8858756f4f95
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 12 18:57:28 2015 -0500

    Added basic array methods

commit cdc675d5c824d42a7e82749a2472bb1da8726008
Author: David Watkins <davidw@tkins.me>
Date:   Fri Dec 11 15:28:10 2015 -0500

```

```
    Fixed bug where constructors weren't being checked by name

commit 8346a0009480db6799587fa8a1b3ab0178c5ea43
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Thu Dec 10 18:18:39 2015 -0500

    Update README.md

commit b33b3a318fc92dbcdf434c72888f0c2399b3173f
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 8 17:23:53 2015 -0500

    Added help printing to compiler with no arguments

commit ec57d8062f137244729246260d34a7cd47641525
Merge: ae65af0 bb7a89b
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Sun Dec 6 17:21:44 2015 -0500

    Merge pull request #79 from DavidWatkins/Kappa

    Kappa

commit bb7a89b50cd1045cd8c0b711288d7bead3f8af20
Merge: ae65af0 43e4e3b
Author: David Watkins <davidw@tkins.me>
Date:   Sun Dec 6 17:21:21 2015 -0500

    Merge branch 'Kappa' of https://github.com/DavidWatkins/Dice into Kappa

commit ae65af04ea8c138768db9f1e25249d4c564d9882
Merge: 914b15a df7d695
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Sat Dec 5 21:31:11 2015 -0500

    Merge pull request #77 from DavidWatkins/ObjAccess

    Obj access

commit df7d695d9f61cb7709ac5bd24422a23691d969dc
Author: David Watkins <davidw@tkins.me>
Date:   Sat Dec 5 21:29:47 2015 -0500

    Classes are now working, fixed tests to match up with new rules

commit 3547bd54ce8e66a8d984ecac37ef478f43d1d773
Author: David Watkins <davidw@tkins.me>
Date:   Fri Dec 4 15:39:07 2015 -0500

```

```
    Sigh

commit 914b15a3301e9de97ff5b9fcbf57f7731fbd90a0
Merge: bc5da4f b474701
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Fri Dec 4 01:27:57 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit bc5da4f925a2a6995b3d79cff92fff0f87f0384d
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Fri Dec 4 01:27:04 2015 -0500

    added else if tests

commit 43e4e3bf1d4a64e5fa71b3642a21250f37bb7334
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Fri Dec 4 01:14:26 2015 -0500

    unop working

commit 2fedba447dd85d89582b3aad84c0a470db87de7c
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 17:14:26 2015 -0500

    Still WIP

commit a0c3cbf70c80847b0892ef61c9bf34c109ca1f49
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 15:56:52 2015 -0500

    Added sample test script

commit a639719f7a7d885a4008be87ac94cbe5ec170695
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 15:56:03 2015 -0500

    WIP

commit b47470171b10bbe3b8f7bcc9f7f0e52bf73a01e1
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Wed Dec 2 07:33:33 2015 -0500

    Update README.md

commit 15e55374aea051650f8f627205dbcc8160544a75
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 06:48:25 2015 -0500

```

```
    Function parameters are working

commit 0ff181573ba6a1fea1105ecc4b72f8fb269db965
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 06:00:30 2015 -0500

    Added basic function calls to compiler

commit d99e2cc2f5b17ce3826ffe4aa0c6bc39e8297465
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Wed Dec 2 04:26:03 2015 -0500

    unop implemented, but not working. All tests are failing.

commit aaa1368f6872e5c20d669f38614fd431e3b21c65
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 03:48:01 2015 -0500

    Added lazy evaluation and fixed error with function names

commit d0fa8223f546f315afc023d637f240af34329e36
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 03:06:35 2015 -0500

    Changed wording in helper

commit 74059d062fdbbdc1679dac574052c05459751c08
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 03:04:36 2015 -0500

    Added the ability to compile to a file

commit 2078c5fdb94b2cc6d265617c7d12d81e507c7e57
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Wed Dec 2 02:15:27 2015 -0500

    corrected test-bool4.dice

commit c0d5caee65fb50c2aa083309957bd1b20dba1c1c
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 01:58:56 2015 -0500

    Float comparison expressions now evaluate properly

commit c6bb01085947ef3f51cbdc885238c3964039b708
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Wed Dec 2 01:29:37 2015 -0500

    fixed tests and added more for bools
```

```

commit 0d9c3a0dfe3b893c500f75090b5f04c89bb4401c
Merge: 63fdb09 a2300ae
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 01:25:58 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 63fdb093f7254bc4934fdde8b564b1d97463eada
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 01:25:27 2015 -0500

    Added printing string representations of boolean values to codgen

commit a2300aedb2fe11c3fa612e1ae8f7d60537fb3019
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Wed Dec 2 00:36:29 2015 -0500

    Fixed syntax error

commit 861aee2ddb899d888a13e2a42e5a81c0a1528cd4
Merge: e7494e3 b0ab4a8
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Wed Dec 2 00:16:32 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit b0ab4a8e92319f72c3d1bb2376475b424cbf1887
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 00:16:10 2015 -0500

    Reverted change to printing floats

commit e7494e3b6488dc49d28bb3bcad6e77f7ea42d265
Merge: d969ca2 21ac0fa
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Wed Dec 2 00:11:39 2015 -0500

    wMerge branch 'master' of https://github.com/DavidWatkins/Dice

commit d969ca2fc12093e18f946e893328b3cdb788ff43
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Wed Dec 2 00:11:08 2015 -0500

    nested if tests added with boolean tests of logical operators

commit 21ac0fa10db8c24347a0a56ed39cfc1b92e7ae19
Author: David Watkins <davidw@tkins.me>
Date:   Wed Dec 2 00:07:50 2015 -0500
```

```

    Fixed printing of floats

commit 4ca9ff15d8016c5fe78f81a23eb2b5bc19a443de
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 1 23:52:18 2015 -0500

    Added exception for invalid integer operation in codegen

commit 9c25e446d76918a3b11be98a9d0aef72f2345e57
Merge: c45b5f8 72718b2
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Tue Dec 1 23:50:17 2015 -0500

    Merge pull request #68 from DavidWatkins/Kappa

    Kappa

commit 72718b24c9c77818965340c2642b9746452517f9
Merge: 2031096 c45b5f8
Author: David Watkins <davidw@tkins.me>
Date:   Tue Dec 1 23:49:47 2015 -0500

    Merge branch 'master' into Kappa

commit 203109635a92704afcaf6ba8f7686e4bc56ee463
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 1 22:40:47 2015 -0500

    fixed unusued match warnings but matching AST type instead of llvalue. David determined that the Oc

commit c45b5f88281cfa8c5989fbc883bbe97230bac8c2
Merge: 3707602 7630cb1
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Tue Dec 1 21:27:59 2015 -0500

    Merge pull request #66 from DavidWatkins/emily

    Emily

commit 7630cb139714b189697761faeb495d9a6d8055ad
Author: Emily Chen <emchennyc@gmail.com>
Date:   Tue Dec 1 21:26:16 2015 -0500

    raised wrong exception when trying to instantiate undefined class

commit 9d0040a4aa5506d46024e2c870dee099527cb6db
Author: Emily Chen <emchennyc@gmail.com>
Date:   Tue Dec 1 21:13:34 2015 -0500
```

```

    threw wrong exception for UndefinedClass case

commit 63765ae27d235ca0664cb329e72324475f80d6c0
Author: Emily Chen <emchennyc@gmail.com>
Date:   Tue Dec 1 20:26:28 2015 -0500

    object creation flags when actuals don't match any existing constructor

commit 1d3c59c8bf8ce3d1c621697025a9c529f0285a2c
Author: Emily Chen <emchennyc@gmail.com>
Date:   Tue Dec 1 17:26:18 2015 -0500

    types of actuals printed in same order as types of formals

commit 045fc2aa1cd78c1c93f58ce4c4412ccceada0b39
Author: Emily Chen <emchennyc@gmail.com>
Date:   Tue Dec 1 16:52:13 2015 -0500

    can print types of formals and actuals

commit 5e2ea6f7870c893d0e8fa6df422f3be55a240555
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 1 16:06:31 2015 -0500

    Test cases for arith negation added and build_global_stringptr modified for debugging

commit 4913954cb8166998c6aae53a2c1f733c06473890
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 1 08:43:55 2015 -0500

    added cast test (float+int)

commit b4f2afc359fb3ad8dcf1e658d428480c84c183a7
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 1 07:25:26 2015 -0500

    Compilesgit add codegen.ml !

commit 3ea9139620f9b937d7c6892cf1be2209cad34635
Author: Emily Chen <emchennyc@gmail.com>
Date:   Tue Dec 1 03:13:43 2015 -0500

    check_object_creation raises exception if instantiating unknown class

commit d98122c25680d734687c5e95d67832d620830d84
Author: Emily Chen <emchennyc@gmail.com>
Date:   Tue Dec 1 02:41:32 2015 -0500

```

```
    checks object decl to see if the class is available

commit 3bd51afa9cf5b2041543025837ed50d93ffe7d52
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 1 01:48:21 2015 -0500

    fought through several rounds of compilation errors.

commit 4494d69fc57cdac1a944a92ea7660f899a906a9f
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Tue Dec 1 01:22:02 2015 -0500

    Rough draft of handle_binop implemented. Still need to compile it, but pushing to access on VM. I ha

commit 37076028c622a12be9c222ca2331f265c99ac625
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Mon Nov 30 23:49:19 2015 -0500

    Update README.md

commit 34586a39bbe449392a730dcbcf3e85dd2b70941c
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 23:46:51 2015 -0500

    Merged Emily's changes to master

commit a2446010f6af0c06f465581a0b09bde85d4f1a3c
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 23:41:58 2015 -0500

    Fixed pretty printer and loops

commit 9b8880077317b5dafd319becca52528d2fa8a393
Merge: 889c3b7 6f8b207
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Mon Nov 30 23:35:44 2015 -0500

    Merge pull request #56 from DavidWatkins/emily

    Emily

commit 6f8b20749beb30748ed3912f631ad30cbfdf9ab0
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 23:34:52 2015 -0500

    Added primitive variables

commit 1bfb88e3e588de2b2d097d9efa76cee25753129d
Author: Emily Chen <emchennyc@gmail.com>
```

```
Date:   Mon Nov 30 23:27:52 2015 -0500

    remove debugging statements

commit 0fe96a727e2fbd895fdc20e4bc3b4b423fcec17f
Merge: ef16286 889c3b7
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Nov 30 22:41:57 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice into emily

commit ef1628630066d0d7d20112a5def6a221fc38827c
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Nov 30 22:41:07 2015 -0500

    converting local to slocal works for primitive types

commit 16491e2e0c6a513271acd0519f77b97818555344
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Nov 30 22:01:09 2015 -0500

    local var decls are tracked even without assignment expr

commit 2adbb32da2aa5ccc60561594cb402d00b2e9c7bf
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Nov 30 21:48:09 2015 -0500

    local var decl is added to env when statement includes nonempty expr

commit 889c3b715b50b63391a454a75a5d9d7dfbdd2657
Merge: 3064464 3d3154c
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 19:49:10 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 306446425bd29b931a6325b47b3da0cc3b84e04f
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 19:48:49 2015 -0500

    Added pretty printing of sast and ast in JSON

commit 3d3154cea0c622561c7a63946e5e85ec8eb07e8d
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Mon Nov 30 16:18:51 2015 -0500

    Update README.md

commit 64d255b692d1d3f156a210253bb4db2b1bd123ba
```

```
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 30 13:26:06 2015 -0500

    modified test script to perform automatic compilation of Dice Executable at the beginning of each s

commit f4312c13faa500a2601e08b1fbd53568750df70b
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 30 12:14:07 2015 -0500

    corrected syntax error

commit b562f21c0f9e17dc946ddf2d1faa206346607e5d
Merge: 338553e db99c23
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 08:10:05 2015 -0500

    Merge branch 'emily'

commit db99c2314ba0bdf2bba05501e971dd379e1a0bbc
Merge: 9f1d6c7 338553e
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 08:09:54 2015 -0500

    Merge branch 'master' into emily

commit 338553e016ab991c9a5b278eb4e6fccb3e632121
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 03:18:29 2015 -0500

    Added code for building for loops

commit ed35422de1e3530e82a4ecdb99a07c01774707cb
Merge: ac53ca3 7fe5c5c
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 02:35:32 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit ac53ca3084d06312dabbe3058ab51694774c6bc3
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 02:35:07 2015 -0500

    Fixed elseless if problem

commit 7fe5c5ca653e8a5973228953b681f431833a16bd
Merge: 9b5f7d1 50d5298
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 30 02:15:10 2015 -0500

```

```
    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 9b5f7d18f8f895d4979b9a6bc32164262cb9bc31
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 30 02:14:35 2015 -0500

    basic inhertiance test added

commit 50d52984ad083eac3722a630cd027a0714537459
Merge: 75a41c3 1932754
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 01:39:36 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 75a41c3cf0da048bb6b76875f97d428cf84d8e41
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 30 01:39:06 2015 -0500

    Ifs semi-implemented, multi-line programs work now

commit 9f1d6c75a4454794ac10e636bb6dd07291cbd642
Merge: a874f50 1932754
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Nov 30 01:38:40 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice into emily

commit a874f50a43d711086c8038dc92c06014bf11a39c
Author: Emily Chen <emchennyc@gmail.com>
Date:   Mon Nov 30 01:37:41 2015 -0500

    check_binop succeeds when only literal operands; doesn't handle IDs yet

commit 19327541aec867c8b3617ce7a55c2b8c30afc56a
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 30 01:35:32 2015 -0500

    array tests added for single and multidimensional arrays.

commit d1a88f6cc7a112e34869a03c36f0fbb8c95dea73
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Nov 29 23:56:36 2015 -0500

    mroe tests

commit 01c53a3905d6dda85bd7d407ce1024c049401f3f
Merge: 0052631 96c9f21
Author: Emily Chen <emchennyc@gmail.com>
```

```
Date:   Sat Nov 28 16:48:45 2015 -0500

    Merge pull request #50 from DavidWatkins/epsilon

    Epsilon

commit 96c9f21a876921f9fe7b54c7c5520d2684926080
Merge: 0828f97 0052631
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Nov 28 16:43:56 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice into epsilon

commit 0828f97195361fcde1f315f76c0b9b40602fdfa6
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Nov 28 16:43:40 2015 -0500

    current state of LRM, WIP

commit 00526316b382f1fcdbf7e20dd8116d76f3c0af49
Author: David Watkins <davidw@tkins.me>
Date:   Thu Nov 26 03:53:13 2015 -0500

    Added environments as return types for expressions and statements

commit 7bd0f08fd5735207d23ef0282f75571779c17032
Author: David Watkins <davidw@tkins.me>
Date:   Thu Nov 26 03:28:16 2015 -0500

    Added assignment type checking

commit 91f50320126a774977e963b002952cffcdaf8c0b
Author: David Watkins <davidw@tkins.me>
Date:   Thu Nov 26 03:18:53 2015 -0500

    Reorganized analyser unop

commit eb1e72d42ffccfa994b21db18c5ee7594b3086cb
Author: David Watkins <davidw@tkins.me>
Date:   Thu Nov 26 03:09:06 2015 -0500

    Print will now accept variable number of arguments and print integers

commit 2697f7d36eee3267d4d08acd72ee36218cfe885f
Author: David Watkins <davidw@tkins.me>
Date:   Thu Nov 26 02:43:40 2015 -0500

    Added reserved functions to analyzer

```

```
commit f016c356c05016b220b3503f7ef331c0cc6fe9e9
Author: David Watkins <davidw@tkins.me>
Date:   Wed Nov 25 23:14:40 2015 -0500

    Analyzer now uses SExpr instead of expr

commit 405feab53aeb98996d924e1d0c054b2c057893b8
Author: David Watkins <davidw@tkins.me>
Date:   Wed Nov 25 20:46:37 2015 -0500

    Added test ocaml code to produce llvm

commit fb92dc93387bc04a842ce20414642a8e0d6be079
Merge: a70917b d3bfd36
Author: Emily Chen <ec2805@columbia.edu>
Date:   Wed Nov 25 14:19:20 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice into epsilon

commit d3bfd36c6a493a1c4b768bcc86049b5245975fdc
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 23 03:55:35 2015 -0500

    Added a lot

commit 18c53d74b916b57cf79523da4bb5532408f0d623
Merge: e714714 c3635ab
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Sat Nov 21 22:30:11 2015 -0500

    Merge pull request #46 from DavidWatkins/Kreygasm

    Kreygasm

commit c3635ab4859a46182ea3be101ffe08c80567da83
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Nov 21 22:28:57 2015 -0500

    duplicates checked in stringmaps

commit 347bb718b69bba5cedcf8d920e81fb46a9857602
Author: David Watkins <davidw@tkins.me>
Date:   Sat Nov 21 21:58:42 2015 -0500

    fubic

commit 796dad808edb739bb79e140ae8284affc416ba8e
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Sat Nov 21 20:30:52 2015 -0500
```

```

    analyzer broken

commit 83453a96dc22e2b0cb3c0b5fadda6c38cd34f84d
Author: nethacker11 <philip.schiffrin@gmail.com>
Date:   Fri Nov 20 15:34:00 2015 -0500

    updated analyzer for global table

commit a70917bebd8b122fc1456a0de6d647f27e124378
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Nov 17 05:39:53 2015 -0500

    specify wraparound behavior for char overflow during addition operation

commit 5ef7d4040ae2a3f384dc8e49108df9f911da54e8
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Nov 17 05:32:45 2015 -0500

    fixed typos in Type section

commit 519ecd38eb0ff36345e404500a58799ab6e6f22e
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Nov 17 05:32:15 2015 -0500

    fixed typos in Type section

commit 1dbea9cdd7ce99e8afb045d825c10cf7b61da1e6
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Nov 17 05:28:27 2015 -0500

    lrm pdf

commit 218cdd226af54fc7a12aa54174686808b9c0c080
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Nov 17 05:27:01 2015 -0500

    expressions emulate K&R reference

commit a23065b93cfa8ea563b2e5cafe47e4001364329f
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Nov 17 04:14:47 2015 -0500

    remove examples from Types section

commit e71471403e598ff74fff7e1c18b6c26f84db7c4e
Author: Emily Chen <ec2805@columbia.edu>
Date:   Tue Nov 17 01:26:58 2015 -0500

```

```
    update regex for int, float

commit 01369938d06a83b7a411e97ea7f3105355ecb1c7
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 16 21:09:30 2015 -0500

    Added new keyword, fixed pretty printing, allowed varied variable declaration

commit 90ac3e878efdb7d8471a49ac07b2717d568394ec
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 16 05:00:47 2015 -0500

    Hello world demo code

commit 4a37b8b8e8fe6d695354db10b78f4273584ece35
Author: David Watkins <davidw@tkins.me>
Date:   Mon Nov 16 04:55:33 2015 -0500

    Added escape characters to string literals

commit cc898068bdd320886cbf6d6d950edc00a5cb8afe
Merge: be88ee9 d8ed6e5
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Sun Nov 15 15:24:08 2015 -0500

    Merge pull request #7 from DavidWatkins/Delta

    Delta

commit be88ee9635bd7a60d134eac92cd8516dc08ccd06
Author: David Watkins <davidw@tkins.me>
Date:   Sun Nov 15 03:00:33 2015 -0500

    Removed bindings.c

commit b35203029ea05992df4d7356c556d8250379ec3e
Merge: 1a79286 bdfc46f
Author: David Watkins <davidw@tkins.me>
Date:   Sun Nov 15 02:55:59 2015 -0500

    Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into HEAD

commit d8ed6e5ac77c66a1e28e43a91d1c7c90d10d096c
Merge: 819c652 bdfc46f
Author: David Watkins <davidw@tkins.me>
Date:   Sun Nov 15 02:49:47 2015 -0500

    Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta

```

```
commit bdfc46f92ab376ea29d6c672efa1c95cdc547f78
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Nov 15 02:49:23 2015 -0500

    fixed cleaning up of temp files

commit 819c652f8af6058175b61339d75612f798b7f446
Author: David Watkins <davidw@tkins.me>
Date:   Sun Nov 15 02:43:59 2015 -0500

    Added unary minus

commit b79952cc5f486e080c31a7ae00b8977fa6812aa2
Author: David Watkins <davidw@tkins.me>
Date:   Sun Nov 15 02:37:18 2015 -0500

    Removed - from int and float literals

commit b7e306b4eb8dbb8859a79a7242064713f923605d
Author: David Watkins <davidw@tkins.me>
Date:   Sun Nov 15 02:29:54 2015 -0500

    Fixed rule with return

commit 30877b0821d29c59f8e86dbe8a0d4437d63dc6bc
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Nov 15 02:21:01 2015 -0500

    tester corrected to work with lli

commit f037b3b5dc89409d59d55b5be4ed2a816b317be6
Merge: e843f0e ae1d756
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Nov 15 02:19:51 2015 -0500

    Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta

commit e843f0ef31580d6846586a8ca49c2840222276c9
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sun Nov 15 02:19:34 2015 -0500

    Corrected syntax errors in test case code

commit ae1d7560e16dd7eecdbc34a291d8f4e41a97eeeb
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Sun Nov 15 02:10:16 2015 -0500

    Update README.md

```

```
commit 026fd5026bf957515f9b0902aa2d278b48197fe0
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Sun Nov 15 01:55:44 2015 -0500

    Update README.md

commit 5ca0b69612e326f2523c5c7542f1ae5ea02d6f24
Author: David Watkins <davidw@tkins.me>
Date:   Sat Nov 14 20:14:41 2015 -0500

    Small edit to readme

commit 1c2547eef86a517507a6cbec1655f38df7875290
Author: David Watkins <davidw@tkins.me>
Date:   Sat Nov 14 20:12:53 2015 -0500

    Small changes

commit 54d7539d119aa459278dca7b3bcb68c248054948
Author: David Watkins <davidw@tkins.me>
Date:   Sat Nov 14 20:09:16 2015 -0500

    Added to README

commit d88306a8e86159b467a5da701bd315ce8e713d5a
Author: David Watkins <davidw@tkins.me>
Date:   Sat Nov 14 19:57:11 2015 -0500

    Compiler works, run build.sh

commit 32d87fa0fd8a81778ddd2d61936d64f5ef6aebc6
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Nov 14 17:40:47 2015 -0500

    modified test script to use lli

commit 0db46dad8c7f8f97385be5602b4340efb7485c44
Merge: 8aa9ed2 2287265
Author: David Watkins <davidw@tkins.me>
Date:   Sat Nov 14 16:38:18 2015 -0500

    Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta

commit 8aa9ed21512f946604e9824e72d0f32f48460cb9
Author: David Watkins <davidw@tkins.me>
Date:   Sat Nov 14 16:37:59 2015 -0500

    Works!!!!!!

```

```
commit 228726581d0b46dd87aeeacfe2fc66b276ecd434
Merge: 01e738d 65f6ba6
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Nov 14 16:19:25 2015 -0500

    Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta

commit 01e738dae2e637fd7a2265aa16bc3135172b24e2
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Nov 14 16:11:58 2015 -0500

    testing script and basic test cases

commit 65f6ba6989f389b3db5b95cb4f0eaf0438f10157
Author: David <davidw@tkins.me>
Date:   Sat Nov 14 13:41:36 2015 -0500

    Made changes yo

commit 7dbe37512674db6a5f911bcc336878c45c5c1aca
Author: David <davidw@tkins.me>
Date:   Sat Nov 14 03:38:45 2015 -0500

    I give up for now

commit 04c4053bf43276ac073c4141ce1d2f79ddbc3452
Author: David <davidw@tkins.me>
Date:   Sat Nov 14 03:31:54 2015 -0500

    iWhatever

commit 1bfde45790ff5c37a06fed8a732d95343d6b09fe
Author: David Watkins <djw2146@columbia.edu>
Date:   Fri Nov 13 23:51:52 2015 -0500

    Again WIP

commit 6bc13cfadb5458a77b2d521390683981502d3cd1
Author: David Watkins <djw2146@columbia.edu>
Date:   Fri Nov 13 16:25:05 2015 -0500

    Added new way to make, figuring out layout for code based on tutorial

commit 1a79286feb4a6b21d8ded437dda312143a485f9b
Author: David Watkins <davidw@tkins.me>
Date:   Thu Nov 12 20:37:37 2015 -0400

    Wrong rule for utils

```

```
commit 550cd68be9b44ca6b4bb57dd3ca6539ae4ee04ca
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 11 15:44:05 2015 -0500

    Created base code for compiler and improved processinclude

commit e37f596018482de09bad4e87b85a9b346c33b374
Author: David Watkins <djw2146@columbia.edu>
Date:   Wed Nov 11 02:27:24 2015 -0500

    Added more descriptive error messages to dice files with incorrect syntax

commit 2c93957d1cf2401c8eae282123bcaa397290c194
Author: David Watkins <djw2146@columbia.edu>
Date:   Wed Nov 11 01:14:35 2015 -0500

    Changed primitive arrays to support inclusion of expressions and fixed
    escaped char literals

commit bfe58d3de921dd7428b6fb18d2ba2240336a0164
Merge: 13e70b1 77193cc
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 9 02:28:11 2015 -0500

    Merge branch 'master' of https://github.com/DavidWatkins/Dice

commit 13e70b1a603b87eb5c59dd96a7908a68e2b4286e
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 9 02:27:48 2015 -0500

    testing script implemented for Scanner tokenizer with some basic test cases. More to follow soon

commit 77193cc351212bbe8fd9ff02c73f71c2a958cf91
Author: Emily Chen <ec2805@columbia.edu>
Date:   Mon Nov 9 05:16:18 2015 +0000

    updated roles, re-rendered LRM pdf

commit 833154914d227be8b3d75ff2266cb5b1d016f185
Author: Emily Chen <ec2805@columbia.edu>
Date:   Mon Nov 9 05:13:39 2015 +0000

    updated roles

commit 48b5419853a7dad44ae3ef590ddb48f9f0fc40fb
Author: Emily Chen <ec2805@columbia.edu>
Date:   Mon Nov 9 05:08:07 2015 +0000

    updated LRM pdf
```

```

commit 62673cf7dcffe60a07ba5711e4df02c9d4542589
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Nov 8 02:53:38 2015 +0000

    add description for logical operators and member access operator

commit 650993c85b05f1415aa1fedf4f995358d5e94f6b
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Nov 8 02:24:12 2015 +0000

    add example of inheritance using "extends" kw

commit eb8488828a35dfe92828ff403eb3fa60b734eaf2
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Nov 8 02:05:59 2015 +0000

    updated examples so they don’t declare and initialize in same statement

commit ac91d9956899f8a7246bef381942afcf505f95b6
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Nov 8 01:06:22 2015 +0000

    no class name collisions within module or between modules

commit 662d69a4aa9be43b54f85db25ec23ced249629b6
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Nov 8 00:49:20 2015 +0000

    change .di to .dice; specify that recursive includes are not supported

commit 61afc126141b1ceaa5618ca4af55e1b6229a9f2f
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Nov 8 00:39:14 2015 +0000

    describe Include statement

commit 9d04c148421a6793a71f17543fba74b0179b5a9e
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Nov 7 23:59:35 2015 +0000

    updated array declaration, initialization, access in LRM

commit c73e469da24918a33fbc32fadf46eef0bb34a83d
Merge: 51f4c2e 18a9ca2
Author: David Watkins <djrival7@gmail.com>
Date:   Sat Nov 7 13:55:09 2015 -0500

    Merge branch ’master’ of https://github.com/DavidWatkins/DiceLanguage
```

```

commit 51f4c2e5e49ea49f1b9d7fdc84b116f79da7ef74
Author: David Watkins <djrival7@gmail.com>
Date:   Sat Nov 7 12:16:59 2015 -0500

    Added nested primitive arrays to parser

commit 18a9ca288d017dffec5eb5240f6ed4e6551f4484
Author: Emily Chen <ec2805@columbia.edu>
Date:   Fri Nov 6 23:24:59 2015 +0000

     updated operator precedence in parser

commit 5a8cef865cec749060472a2da08f68e5f66ab603
Merge: 8a3a33b 233b9ae
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Thu Nov 5 01:55:34 2015 -0500

    Merge pull request #3 from DavidWatkins/DavidFix

    David fix

commit 233b9ae4d2ad329217da5f89a02208e65f0f1056
Merge: 2339f25 8a3a33b
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Thu Nov 5 01:51:37 2015 -0500

    Merge pull request #2 from DavidWatkins/master

    Merge pull request #1 from DavidWatkins/DavidFix

commit 2339f25231f1ee437ed6703545d6c7b010ec99a7
Author: David Watkins <djw2146@columbia.edu>
Date:   Thu Nov 5 01:47:31 2015 -0500

    Added AST printing method by using menhir inside ocaml

commit 8a3a33b299306b2322b4c87df2c42f559bb0f612
Merge: 9507d54 d095d57
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Thu Nov 5 00:57:10 2015 -0500

    Merge pull request #1 from DavidWatkins/DavidFix

    David fix

commit d095d57af5fb83805fcbd5d70ff133facad7f64a
Author: David Watkins <djw2146@columbia.edu>
Date:   Thu Nov 5 00:37:34 2015 -0500
```

```

    Pretty printer bug fixed, tokenizer now prints line numbers

commit 7c800f61455e0a09d421afb4d384e1d0a07f5283
Author: David Watkins <djw2146@columbia.edu>
Date:   Wed Nov 4 22:12:26 2015 -0500

    Made changes to front end, pretty print and tokenizer work

commit b2a0bbd31c3484a2a914a1fb01cd8e05ecbc074f
Author: David Watkins <djw2146@columbia.edu>
Date:   Wed Nov 4 20:41:00 2015 -0500

    Fixed makefile?

commit 14f5ace95044a80e3d8d2a49e5e6a645f704a6ae
Author: David Watkins <djw2146@columbia.edu>
Date:   Wed Nov 4 20:35:55 2015 -0500

    This is just a test run of additional useful files, needs to be compiled
    on a unix system

commit a87e9b9a3706cb33fb79f260391b786bf53c6a40
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 15:57:22 2015 -0500

    Fixed parser with class keyword, removed array keyword

commit 41825d291a910847c3aa0d5d67f5c60f3698fcef
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 15:50:46 2015 -0500

    Revert "Fixed operator precedence"

    This reverts commit d132fdb8d21ba69a8d9d1c71c0ab71af5231eac0.

commit d132fdb8d21ba69a8d9d1c71c0ab71af5231eac0
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 15:24:56 2015 -0500

    Fixed operator precedence

commit 0726d908374df6c447d82290e06a83b84d0fdd0a
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 15:23:35 2015 -0500

    Fixed backet_args to refer to general expr list

commit ffec7d32cba2126885009518d79f072feea88628
```

```
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 15:21:37 2015 -0500

    Added datatypes to primitive arrays

commit 096f5a8bd45b44f23a42abaf837ad1f55597bce3
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 15:18:01 2015 -0500

    Fixed bug, apparently no issues wot

commit f30730a1cd041a3df6010c1ee62a59b17842e68f
Author: David Watkins <djw2146@columbia.edu>
Date:   Wed Nov 4 14:43:20 2015 -0500

    Fixed Menhir errors

commit 89a3de8e2e78735910eb385a3b51f4795e115c62
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 13:38:03 2015 -0500

    Removed extraneous files

commit c6d4db34bcca5aa6cc86a4e6f670a858c2f0b6bc
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 13:36:38 2015 -0500

    Cleaned up git directory

commit 4d157027e77211c94f295f34cef0d03a19c7f102
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 13:20:24 2015 -0500

    Found a more elegant solution to array problem

commit d7f28aa1dad4d1e788ab7b2aaab962372dfe1e71
Author: David Watkins <djrival7@gmail.com>
Date:   Wed Nov 4 00:06:05 2015 -0500

    No shift reduce but not ideal

commit d29a030e18489e489f0f8941cfbc70cc85c81a03
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Nov 3 23:45:34 2015 -0500

    Super close, just ambiguity surroundign array access

commit 5473e62f147ec34eaccd4289930c4b2144d7c968
Author: David Watkins <djrival7@gmail.com>
```

```
Date:   Tue Nov 3 18:04:37 2015 -0500

    More stuff

commit dc6ad10dc89c389534ec225082c64da35a512268
Author: David Watkins <djrival7@gmail.com>
Date:   Tue Nov 3 14:10:00 2015 -0500

    Shift/Reduce down to 2, fixed layout of cdecl and cbody

commit 9507d5426cd130cc927941a4620383040c895717
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 2 13:07:36 2015 -0500

    BIBLETHUMP delimiter still in this version. AST modified to remove actions not used

commit 5419ec530d153c46c7ee4a3c12765a73c0fcb0c0
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 2 01:32:27 2015 -0500

    Corrected the array access production to account for multidimensional arrays

commit ce9ed98c205d39871c544a3640d92a6e1c77c31f
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Nov 2 01:29:07 2015 -0500

    Parser compiles w/o any errors, but not tested yet. Multidimensional arrays implemented

commit 8ee5b859b0fea8a91154fe05694cd875c05fbd9e
Author: Philip Schiffrin <philip.schiffrin@gmail.com>
Date:   Thu Oct 29 22:58:33 2015 -0400

    hacked a solution for the last shift/reduce by adding token FUN at beginning of fdecls

commit bd6ecf78c93229b46957e67a50056e975ddcc072
Author: Philip Schiffrin <philip.schiffrin@gmail.com>
Date:   Thu Oct 29 21:29:48 2015 -0400

    fixed all reduce/reduce and most shift/reduce errors in  the parser, lost most of our logic

commit d8cdf93ba6240b597307acb59407165c5b8eeff2
Merge: df1979e 5a7f132
Author: David Watkins <djw2146@columbia.edu>
Date:   Mon Oct 26 20:12:34 2015 -0400

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit df1979e32572f5bb0e3e5e4c95d22ff99ae77c53
Author: David Watkins <djw2146@columbia.edu>
```

```
Date:   Mon Oct 26 20:12:23 2015 -0400

    Fixed language to dice again

commit 5a7f13237ad7c32bc9609653e30b3f71b9c3aaf1
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Mon Oct 26 16:30:45 2015 -0400

    Update README.md

commit f88282699ce5b7e3ea3c672c51e1ac1a69527687
Author: David Watkins <djw2146@columbia.edu>
Date:   Mon Oct 26 02:57:24 2015 -0400

    Fixed with edits

commit 6778395705ceea2284b4f1e13c022ddfd2c45f64
Author: David Watkins <djw2146@columbia.edu>
Date:   Mon Oct 26 02:38:35 2015 -0400

    Fixed intro

commit 3905c2317321473a3e481e2b52902461ba03d5bb
Merge: 764eeb4 f207e7c
Author: David Watkins <djw2146@columbia.edu>
Date:   Mon Oct 26 02:28:08 2015 -0400

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit 764eeb44f8ffa599a2451b037b67078683d6eef9
Author: David Watkins <djw2146@columbia.edu>
Date:   Mon Oct 26 02:27:49 2015 -0400

    Finished final draft of LRM

commit f207e7c0a231b852c21103503bbee8fa6edb483a
Merge: d37bf92 0c85801
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Oct 26 00:34:51 2015 -0400

    Merge branch 'master' of https://github.com/DavidWatkins/JFlat

commit d37bf92df1077fd3f612827f14c34cb7e6095af4
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Mon Oct 26 00:34:34 2015 -0400

    Parser compiles, but produces 457 reduce/reduce errors.

commit 0c858019e2b58ae63d8ec581a185953a558a464f
```

```
Author: David Watkins <djw2146@columbia.edu>
Date:   Sun Oct 25 22:18:37 2015 -0400

    Removed extraneous file

commit fe8bd9d0a8abf0f521a6ff44da6d7d615fa259b1
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Oct 25 04:29:11 2015 +0000

    adding content for statements section; TODO include statements

commit 35b69e9bdf8fb8623f19d7d75790c7197c188c74
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Oct 25 03:30:32 2015 +0000

    remove elseif keyword

commit 71bfd4fc327bcce124455bea1a731351a884ea25
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Oct 25 02:33:32 2015 +0000

    update Statements sections

commit c991aa20fcad8e2a7ddab6f0552c105cd943e752
Author: David Watkins <djrival7@gmail.com>
Date:   Sat Oct 24 22:23:32 2015 -0400

    Whatevs

commit b4a0296539e07421a7436d7530154bbd8208158d
Author: David Watkins <djrival7@gmail.com>
Date:   Sat Oct 24 21:34:36 2015 -0400

    More

commit b6d7bdda712a8bbc77c923861fa2dd1161f383e9
Merge: 00c7c8b 8b447ba
Author: David Watkins <djrival7@gmail.com>
Date:   Sat Oct 24 21:28:34 2015 -0400

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit 00c7c8bff49163684020efa60cccd3ae70e481c4
Author: David Watkins <djrival7@gmail.com>
Date:   Sat Oct 24 21:27:44 2015 -0400

    Stuff

commit 8b447ba1a5c2cbfdb18ec5edf26527a926f28e0c
```

```
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Oct 25 01:01:54 2015 +0000

    updated constructor definition

commit 00225f13ceeb3ac4ba51811a308cf069046b58af
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sun Oct 25 00:21:34 2015 +0000

    method names cannot be same as class name

commit 02cc3c7206ed0626bf93ba93bb7924f9896431ea
Merge: fed5273 1335b3e
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Oct 24 23:16:24 2015 +0000

    Merge branch 'master' of https://github.com/DavidWatkins/JFlat

commit 1335b3e88c32dbbcdfe8e0954bb9fd0b89ce4903
Merge: 45adcd4 d56fbb6
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Oct 24 19:24:12 2015 -0400

    Merge branch 'master' of https://github.com/DavidWatkins/JFlat

commit d56fbb6510545503ea3a1ed8eeb0dd38a20cd42c
Author: Philip Schiffrin <philip.schiffrin@gmail.com>
Date:   Sat Oct 24 19:23:49 2015 -0400

    changed doubles to floats

commit 45adcd4da21201da272a7807a47ebde99fbb1e77
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Oct 24 19:23:04 2015 -0400

    changed double to float

commit be271ea2be45684e4cc07fb2723a75343509bf6a
Author: David Watkins <djrival7@gmail.com>
Date:   Sat Oct 24 19:18:21 2015 -0400

    LOL more stuff

commit fed5273dc9e7adae462996c0a7fadc4bdfb91bdd
Merge: 119f1a5 d56fbb6
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Oct 24 23:15:30 2015 +0000

    Merge branch 'master' of https://github.com/DavidWatkins/JFlat
```

```

commit 119f1a57fef847539f3ab76ba3af1bbbe0ef91b2
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Oct 24 23:13:31 2015 +0000

    update lexical elements to replace double w float

commit a793510d204f60c87c8d44ec5f03c6ea7713c1f0
Author: David Watkins <djrival7@gmail.com>
Date:   Sat Oct 24 18:49:37 2015 -0400

    Whatever

commit 4ef1e1b02c3fbaace4d0c4471e4bd048b5c7e8e1
Author: David Watkins <djw2146@columbia.edu>
Date:   Sat Oct 24 18:20:07 2015 -0400

    Added array and object creation to parser

commit 923b6e6ee2e37705f88441be373b71e478475326
Author: David Watkins <djw2146@columbia.edu>
Date:   Sat Oct 24 17:35:10 2015 -0400

    Added Program def

commit 0873e88184cb95d37ce0e6ceb7a320fda04f3dbe
Author: David Watkins <djw2146@columbia.edu>
Date:   Sat Oct 24 16:52:25 2015 -0400

    Added more info to the parser

commit 049a058293023065afb6f90def5b215c298e5511
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Oct 24 14:36:56 2015 -0400

    Added negative doubles in scanner

commit b10a719cae082c8add0a509266264312d528a7df
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Oct 24 13:42:59 2015 -0400

    Corrected precedence chart by removing modulo reference

commit 32497722d8a7efbba2baaaa8ab80ddf899b8f429
Merge: 1ca3359 388cada
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Oct 24 05:46:04 2015 +0000

    Merge branch 'master' of https://github.com/DavidWatkins/JFlat
```

```

commit 1ca3359bdd54bca5247219fddcd6d7de94dcfaa8
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Oct 24 05:45:12 2015 +0000

    renamed lexical elements LRM

commit 5c18e1c07073971ecd666ee761db3e757693d631
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Oct 24 05:44:06 2015 +0000

    classes LRM

commit 388cada09bd0c3b9d43c052780c708b636c958ba
Merge: 7489892 865accd
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Sat Oct 24 00:00:54 2015 -0400

    Merge branch 'master' of https://github.com/DavidWatkins/JFlat

commit 865accd230bbb198523fac3a74aadf14a0e157ff
Author: Philip Schiffrin <philip.schiffrin@gmail.com>
Date:   Fri Oct 23 23:54:06 2015 -0400

    updated LRM_Phil.txt with structure, scope, and arrays

commit 74898925d7d70a982dd70e3ff2f14b73c7d26f0e
Author: Khaled Atef <kaa2168@columbia.edu>
Date:   Fri Oct 23 23:52:04 2015 -0400

    Expressions/Operators portion of LRM

commit 3b92c848a16d0e7a8f9315d680c13bfbc5f17888
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Oct 24 01:36:09 2015 +0000

    add `this` keyword

commit 1aecbbdfebf15eee2c260c3703274b9dba43777b
Merge: 3650409 42a62d0
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Oct 24 01:12:16 2015 +0000

    Merge branch 'master' of https://github.com/DavidWatkins/JFlat

commit 3650409be97c9ed8cd94802464950aaced1993c5
Author: Emily Chen <ec2805@columbia.edu>
Date:   Sat Oct 24 01:11:34 2015 +0000

```

```
    lexical elements section for LRM

commit 42a62d0bf9d983cd3a0f15d4b8f1cb09bf41598e
Author: David Watkins <djrival7@gmail.com>
Date:   Fri Oct 23 19:14:43 2015 -0400

    More stuff

commit 8e3398b458b45005129ab57c82ff6a13f9abafcb
Author: Philip Schiffrin <philip.schiffrin@gmail.com>
Date:   Fri Oct 23 16:54:05 2015 -0400

    added lrm text for data types

commit fe46f41b37d368df2a0c5b552e7166f47c6b227d
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Fri Oct 23 16:45:12 2015 -0400

    Update README.md

commit ac7ee22f9b3ea22922761752306918e49ce148c8
Merge: 8456847 8aa535a
Author: David Watkins <djrival7@gmail.com>
Date:   Fri Oct 23 16:40:21 2015 -0400

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit 84568474c4b9868e460decac0c99e730fecfc92d
Author: David Watkins <djrival7@gmail.com>
Date:   Fri Oct 23 16:39:44 2015 -0400

    Added stuff

commit 8aa535ad4d802f7205c73c896b93b13dd8180239
Author: David Watkins <davidw@tkins.me>
Date:   Mon Oct 12 11:32:44 2015 -0400

    Added isprime ll

commit 4812638376210accb89e6774c0442a918daa4b2f
Author: David Watkins <davidw@tkins.me>
Date:   Mon Oct 12 11:09:09 2015 -0400

    Code working with llvm helloworld example, but does not return result

commit 4cc504c6d2d3732f2a96bc999ef2a28098a320a1
Author: Emily Chen <ec2805@columbia.edu>
Date:   Mon Oct 12 03:03:44 2015 -0400

```

```
    remove obsolete TODO

commit 491935e26c6e27de4c14fb38f5d911eab53e0127
Author: Emily Chen <ec2805@columbia.edu>
Date:   Mon Oct 12 02:52:38 2015 -0400

    host first sends the number of expected bytes it's sending

commit 546776b3750cfa4dfbe9bdb2388ce16ced7b83b4
Author: Emily Chen <ec2805@columbia.edu>
Date:   Mon Oct 12 02:19:24 2015 -0400

    child proc successfully executes command and writes results to file

commit a129ab12519bfdc1d5805b31ce6a966eeaaef52b
Author: Emily Chen <ec2805@columbia.edu>
Date:   Mon Oct 12 01:45:24 2015 -0400

    worker no longer blocks after host is done sending all file data

commit 3c7dd62f15c82bf9a8e75c4b9e3a8b5b006a8d60
Author: David Watkins <davidw@tkins.me>
Date:   Sun Oct 11 20:02:24 2015 -0400

    Code now compiles and added test.py

commit 89b213ca24cacf788cf48f740a2beaecfa58bb50
Merge: 682c38f bb4d7c3
Author: David Watkins <djrival7@gmail.com>
Date:   Sun Oct 11 19:25:53 2015 -0400

    Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage

commit 682c38fb01a5ccd93c73c1e1e050d3763fcf5e03
Author: David Watkins <djrival7@gmail.com>
Date:   Sun Oct 11 19:25:30 2015 -0400

    Added new code for worker and Makefile

commit bb4d7c386dd1f202183ca3a39fcc2ff184b28fb9
Author: Philip Schiffrin <philip.schiffrin@gmail.com>
Date:   Sun Oct 11 16:37:52 2015 -0400

    added isprime for llvm test - llvm code generated by llvm from isprime.c

commit 6ad9e088cdb74598d64d6ce3d1c55d8064295342
Author: David Watkins <djrival7@gmail.com>
Date:   Fri Oct 9 23:18:24 2015 -0400

```

```
    Added initial C code

commit 87f6d1a0d5c756bce028ee1e0622b51957665906
Author: David Watkins <DavidWatkins@users.noreply.github.com>
Date:   Fri Oct 9 23:17:27 2015 -0400

    Initial commit
```

Software Development Environment

From the beginning of the project we agreed to the following development environment with the following software versions:

- **Ubuntu 15.10** - Very simple to use linux distribution that had the LLVM software and OCaml software easily accessible. Ubuntu was used within Virtualbox to ensure consistency across hardware as well.
- **LLVM-3.7** - The latest version of LLVM and allowed for easy code generation in OCaml using the LLVM module
- **OCaml Packages** - There were some features, such as JSON manipulation, that required additional OCaml packages. Therefore we included the following four OCaml packages in our development process: core, batteries, llvm, and yojson.
- **Slack** - We agreed that the Slack chat messaging platform was the most convenient and efficient way to share code snippets and communicate. It also brought up morale in the group in the form of emojis.
- **Github** - In order to version control our software and maintain a working version at any time, we used Github as our go to source code repository. It made integration with the team simpler and everyone was able to view the repository conveniently in their browser.
- **Latex** - In order to compile the documentation we made sure to all use Latex to ensure high quality material being produced for the project.
- **Vim/Sublime** - We could not create a consensus on which text editor to use, but in the end it did not matter to much which members used which.

Programming Style Guide

We adhered to the following style guide as much as possible:

- No lines greater than 80 characters
- Ensure that pattern matches are on the same indent with respect to each other
- Use tabbed indentation as opposed to spaces. Ensure that the tab width is 4 spaces.

5. Architecture

The Compiler

To give a quick overview of our compiler, we have a total of 8 modules:

- **analyzer.ml** - Semantically checks incoming AST representation to make sure that it includes existing files, adheres to the rules of inheritance, and expressions are properly type-checked
- **codegen.ml** - Converts a semantically checked AST into a working LLVM code by producing LLVM IR
- **dice.ml** - Main module that calls on all the other modules depending on compiler flags passed to it
- **filepath.ml** - Uses system calls to determine the absolute path to any file in the system. Useful for uniquely checking if an include statement refers to the same files
- **parser.mly** - Reads in tokens from the scanner to produce an AST representation of the program
- **processor.ml** - Handles communication between scanner and parser so that error messages regarding invalid input can be handled better
- **scanner.mll** - Reads a source file and tokenizes it to the corresponding token output
- **utils.ml** - Contains several functions for printing out the string representation of various intermediate representations in our language. Most critically used for debugging

and we have 4 interfaces

- **ast.ml** - Representation of program after parser
- **conf.ml** - Contains paths for accessing standard library and bindings
- **exceptions.ml** - All exceptions in the compiler
- **sast.ml** - The semantically checked representation of the language

and we have 2 library files

- **bindings.c** - A c file containing critical functions written in c that are usable in the language. This is compiled to LLVM bitcode and then linked with all source files compiled in our language
- **stdlib.dice** - A file containing user defined classes written in dice that are usable by the user

The Scanner

The Scanner scans through the input file and tokenizes the input, discarding characters which are no longer need such as whitespace.

The Parser

The parser scans the tokens passed to it by the scanner and constructs an abstract syntax tree based on the definitions provided and the input tokens. The top level of the abstract syntax tree is a structure containing all classes and a structure containing all include statements. The Parser produces the following layout:

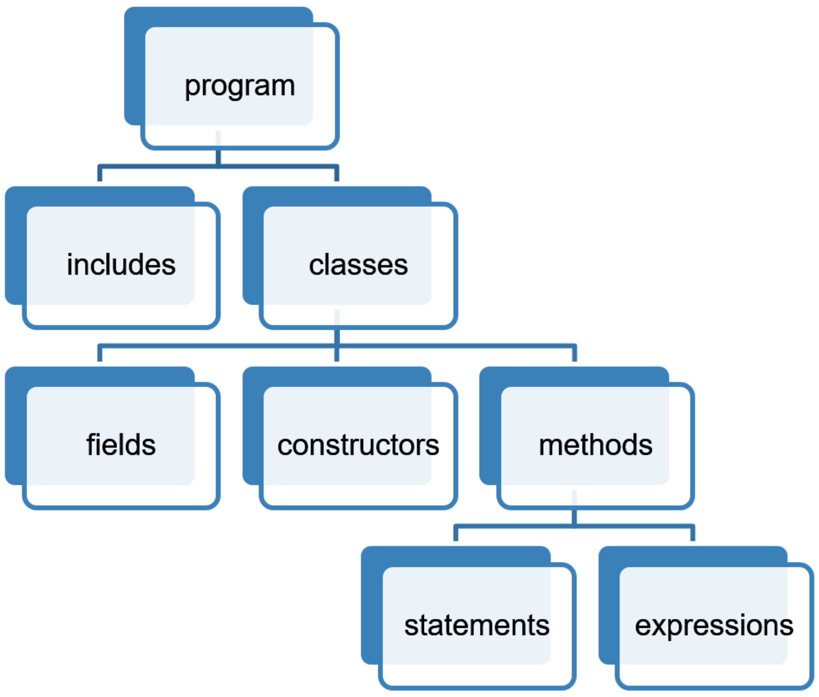


Figure 5.1: AST program representation.

The Semantic Analyzer

The first job of the Analyzer is to run the Scanner and Parser on any files contained in the includes statements of the given abstract syntax tree. The process of building an abstract syntax tree is the same for these files as for the originally compiled file. If any of these new abstract syntax trees contain include statements, the same process is run until there are no more includes. Similarly, each time a new included file's abstract syntax tree is passed to the Analyzer, all classes contained in the class structure of the new abstract syntax tree are appended to the original class list contained in the original class structure which was in the original abstract syntax tree. Once this process is complete, the analyzer is left with a class structure which contains every class defined in every file which was included with the originally compiled file.
Next, the Analyzer performs an inheritance analysis by looking through the class list contained in the class structure and performs an analysis to determine whether any classes are children or parents of other classes. If there are any such relationships, the fields of each parent class are added to the front of its child's fields list, and the methods of each parent class are added to the child's method's list. However, if the child has declared a method or field which shares the same name as the parent's field or method, the child's field or method is not overwritten by the parent. As the inheritance analysis is performed, the list of fields for each class is also assigned a integer key beginning with 0 which will serve as the key to a lookup table which, at runtime, contains pointers to every function for each class.
Once the inheritance analysis is performed, semantic analysis is performed on each statement and expression in each block of code in every method for every class. This semantic analysis consists of making sure that

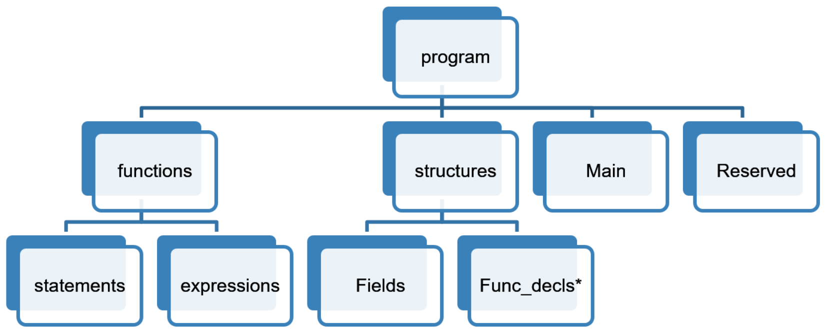


Figure 5.2: SAST representation.

types are consistent in every expression, making sure variables are declared and in the proper scope, and making sure that variables are only declared once. For instance, if an integer x is declared and x is assigned to the return of a method, the analyzer checks that the called method returns the type of x, namely an integer.
As this analysis is performed, the analyzer is simultaneously constructing a semantic abstract syntax tree. The purpose of this new data structure is to provide the code generator with data that is organized more similarly to the LLVM code that it will eventually produce. Thus, instead of classes containing methods and fields, the top level program structure now contains separate sections for methods and fields. This is useful for the code generator because the LLVM code that is produced uses structs to store the fields of a class and functions to store the code within a class's methods. Thus, there is no inherent connection between the functions and the structs in LLVM. However, the analyzer modifies each method so that an instance of the structure containing the fields of the given class is passed in as the first argument to every function for that class. In this way, functions can access each field of a given class by accessing the data inside of the structure.

The Code Generator

The code generator uses the semantic abstract syntax tree passed to it by the analyzer to construct the LLVM IR file which contains the final instructions for the program.

Structs and Inheritance

All structs are given an integer key at the beginning of their definition which will allow them to directly get their own virtual function table. Even if a subclass inherits from a parent class, it will be initialized with a specific key that is unique to the class at the beginning of each struct. For inherited fields they are organized in the order they were inherited, allowing multiple levels of inheritance. However it was too complex of a problem to solve multiple inheritance so we chose not to implement it.

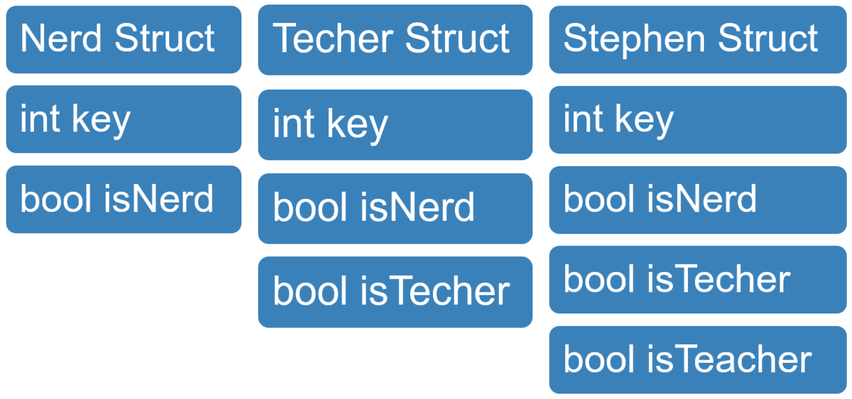
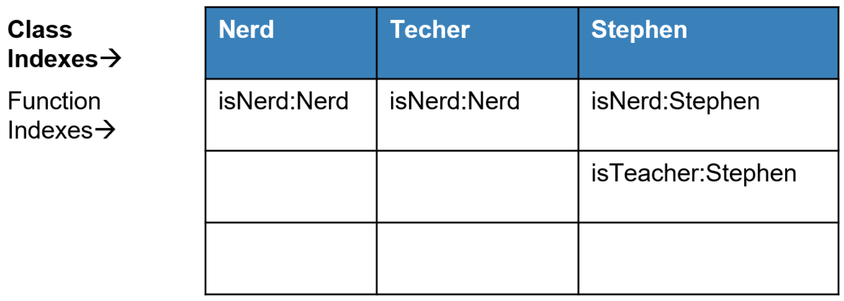


Figure 5.3: Structs example with inheritance.

The Virtual Function Table

At compile time, an intermediate representation of the virtual function table is produced in LLVM IR. It is a function defined as "lookup" that is able to lookup a classes virtual function array by its class index and a function index unique to that function. The function index is generated from the Func_decl list of a struct in the SAST. This way all subclasses have the same index for referring to the same function. Take for example



Class Indexes→	Nerd	Techer	Stephen
Function Indexes→	isNerd:Nerd	isNerd:Nerd	isNerd:Stephen
			isTeacher:Stephen

Figure 5.4: Virtual Function Table Example.

a class Nerd which has a subclass Techer, which itself has a subclass Stephen. Nerd has an isNerd method defined, Techer then inherits that method. Stephen would inherit that method but instead overrides them with its own implementation. But if a Nerd type variable is assigned to a Stephen type variable, the casted struct would still have the corresponding key to the Stephen class, and the function call would receive the correct index of 1 if isNerd were called.

Expressions and Bindings

Once the inheritance code is generated, the code generator iterates through the entire semantic abstract syntax tree and produces the necessary LLVM code for each function, statement, and expression. This code generation is done using the OCaml LLVM library, which uses OCaml functions to produce the desired LLVM code. We then link the resulting LLVM module with a precompiled bindings.bc which allows for the custom C functions we wrote to be incorporated into a user program in LLVM.

The Utilities

Using the utils.ml module we were able to pretty print, print to JSON for AST and SAST, and print out the tokens for any given program. This made debugging the semantic analyzer much easier as we were able to see what went into it and what it produces at any time. The following is an example of what the SAST looks like in JSON.

```
{
"sprogram": {
        "classes": [
        { "scdecl": { "scname": "test", "sfields": [], "sfuncs": [] } }
        ],
        "functions": [
        {
                "sfdecl": {
                        "sfname": "test.constructor",
                        "sreturnType": "class test",
                        "sformals": [],
                        "sbody": [
                        {
                        "slocal": {
                        "datatype": "class test",
                        "name": "this",
                        "val": {
                                "call": {
                                "name": "cast",
                                "params": [
                                {
                                "call": {
                                "name": "malloc",
                                "params": [
                                {
                                        "call": {
                                                "name": "sizeof",
                                                "params": [
                                                {
                                                        "id": {
                                                                "name": "ignore",
                                                                "datatype": "class test"
                                                        }
                                                }
                                                ],
                                                "index": 0,
                                                "datatype": "int"
                                        }
                                }
                                ],
                                "index": 0,
                                "datatype": "char[]"
                                }
                                }
                                ],
                                "index": 0,
                                "datatype": "class test"
                                }
                        }
                        }
                        },
                        {
                        "sexpr": {
                                "expr": {
                                        "assign": {
                                                "lhs": {
                                                        "objaccess": {
                                                                "lhs": {
                                                                        "id": { "name":
↪  "this", "datatype": "class test" }
                                                                },
                                                                "op": ".",
                                                                "rhs": {
                                                                        "id": { "name":
↪  ".key", "datatype": "int" }
                                                                },
                                                                "datatype": "int"
                                                        }
                                                },
                                                "op": "=",
                                                "rhs": { "int_lit": { "val": 0,
↪  "datatype": "int" } },
                                                "datatype": "int"
                                        }
                                },
                                "datatype": "int"
                        }
                        },
                        {
                        "sreturn": {
                                "return": {
                                        "id": { "name": "this", "datatype": "class test"
↪  }
                                },
                                "datatype": "class test"
                        }
                        }
                        ],
                        "func_type": "user"
                }
        }
        ],
        "main": {
                "sfdecl": {
                        "sfname": "main",
                        "sreturnType": "void",
                        "sformals": [
                        { "name": "this", "datatype": "class test" },
                        { "name": "args", "datatype": "char[][]" }
                        ],
                        "sbody": [
                        {
                        "slocal": {
                        "datatype": "class test",
                        "name": "this",
                        "val": {
                                "call": {
                                        "name": "cast",
                                        "params": [
                                        {
                                                "call": {
                                                        "name": "malloc",
                                                        "params": [
                                                        {
                                                                "call": {
                                                                        "name": "sizeof",
                                                                        "params": [
                                                                        {
                                                                                "id": {
                                                                                        "name": "ignore",
                                                                                        "datatype": "class test"
                                                                                }
                                                                        }
                                                                        ],
                                                                        "index": 0,
                                                                        "datatype": "int"
                                                                }
                                                        }
                                                        ],
                                                        "index": 0,
                                                        "datatype": "char[]"
                                                }
                                        }
                                        ],
                                        "index": 0,
                                        "datatype": "class test"
                                }
                        }
                        }
                        },
                        {
                        "sexpr": {
                        "expr": {
                                "assign": {
                                "lhs": {
                                        "objaccess": {
                                                "lhs": {
                                                        "id": { "name": "this",
↪   "datatype": "class test" }
                                                },
                                                "op": ".",
                                                "rhs": { "id": { "name": ".key",
↪   "datatype": "int" } },
                                                "datatype": "int"
                                        }
                                },
                                "op": "=",
                                "rhs": { "int_lit": { "val": 0, "datatype": "int" } },
                                "datatype": "int"
                                }
                        },
                        "datatype": "int"
                        }
                        },
                        {
                        "sexpr": {
                        "expr": {
                                "call": {
                                        "name": "print",
                                        "params": [
                                        {
                                                "string_lit": {
                                                        "val": "Hello, World!",
                                                        "datatype": "char[]"
                                                }
                                        }
                                        ],
                                        "index": 0,
                                        "datatype": "void"
                                }
                        },
                        "datatype": "void"
                        }
                        }
                        ],
                        "func_type": "user"
                }
        },
        "reserved": [
        {
                "sfdecl": {
                        "sfname": "print",
                        "sreturnType": "void",
                        "sformals": [ { "Many": "Any" } ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "malloc",
                        "sreturnType": "char[]",
                        "sformals": [ { "name": "size", "datatype": "int" } ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "cast",
                        "sreturnType": "Any",
                        "sformals": [ { "name": "in", "datatype": "Any" } ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "sizeof",
                        "sreturnType": "int",
                        "sformals": [ { "name": "in", "datatype": "Any" } ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "open",
                        "sreturnType": "int",
                        "sformals": [
                        { "name": "path", "datatype": "char[]" },
                        { "name": "flags", "datatype": "int" }
                        ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "close",
                        "sreturnType": "int",
                        "sformals": [ { "name": "fd", "datatype": "int" } ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "read",
                        "sreturnType": "int",
                        "sformals": [
                        { "name": "fd", "datatype": "int" },
                        { "name": "buf", "datatype": "char[]" },
                        { "name": "nbyte", "datatype": "int" }
                        ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "write",
                        "sreturnType": "int",
                        "sformals": [
                        { "name": "fd", "datatype": "int" },
                        { "name": "buf", "datatype": "char[]" },
                        { "name": "nbyte", "datatype": "int" }
                        ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "lseek",
                        "sreturnType": "int",
                        "sformals": [
                        { "name": "fd", "datatype": "int" },
                        { "name": "offset", "datatype": "int" },
                        { "name": "whence", "datatype": "int" }
                        ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "exit",
                        "sreturnType": "void",
                        "sformals": [ { "name": "status", "datatype": "int" } ],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "getchar",
                        "sreturnType": "int",
                        "sformals": [],
                        "sbody": [],
                        "func_type": "reserved"
                }
        },
        {
                "sfdecl": {
                        "sfname": "input",
                        "sreturnType": "char[]",
                        "sformals": [],
                        "sbody": [],
                        "func_type": "reserved"
                }
        }
        ]
}
}
```

Supplementary Code

The Standard Library

The standard library was written in order to provide the user with a solid foundation on which to start writing interesting programs. To that end we provide for basic file i/o and string and integer manipulation.

String

Provide useful functionality for string manipulation.

Fields

String has no public fields. Private fields include a char array my_string which stores the given string and an int to store the length of the string.

Constructors

String(char[] a) Accepts a char array, such as a string literal or a char array. This string is copied into the my_string field of the object and the private length() method is run to get the length of the input string.

Methods

private int length_internal(char[] input) Returns the length of the given char array.

private char[] copy_iternal(char[] input) Creates a new char array into which it copies the given char array.

public char[] string() Returns the char array contained in the my_string field.

public char getChar(int index) Returns the char contained at the given index in the my_string field.

public int length() Returns the length of the my_string field

public int toInteger() Converts the char array in the my_string field to an integer and returns that int. If the char array contained in the my_string field is not a string representation of an int, the behavior is undefined.

public int toDigit(char digit) Returns the integer corresponding to the character passed in.

public class String copy(class String input) Returns a copy of the current object.

public int indexOf(char input) Returns the index of the input character in the my_string field. Returns -1 if the character is not found in the field.

public class String reverse() Returns a string object with the my_string field containing the reverse of the current my_string char array.

public class String concat(class String temp) Returns a string object with the my_string field containing the concatenation of the current my_string field with the temp's my_string field.

public bool compare(class String input) Returns true if the my_string field of the input String is equal to the my_string field of the current String object.

public bool contains(class String check) Returns true if the my_string field of the input String is contained in the my_string field of the current String object.

public void free() Frees the memory for the my_string field of the current String object.

File

The File class constructor takes two arguments: a char[] that points to an already opened file on which the user wishes to operate and a boolean indicating whether the user wishes to open the file for writing. If the boolean is true the file is opened for reading and writing, and if false the file is opened as read only. The constructor stores the given path in a field and then calls open() on the given path and, if successful, sets the object's file descriptor field to the return of open(). If open() fails, the program exits with error.

Fields

File has no public fields. Private fields are the class String filePath, private bool isWriteEnabled, and the private int fd.

Constructors

File(char[] path, bool isWriteEnabled) Accepts a char array to open a file on, then creates a file object with the file descriptor. isWriteEnabled is a parameter that is used to determine whether the file can be written to or just read from.

Methods

private int openfile(class String path, bool isWriteEnabled) Returns the file descriptor of the opened file if successful, and -1 otherwise.

public char[] readfile(int num) Reads num bytes from the open file and returns the bytes in a char array.

public int writefile(char[] arr, int offset) Writes the contents of the char[] array to the file. If offset is -1 the write starts at the beginning of the file, if 0 it starts at the end of the file, and with any other positive integer it starts writing offset bytes from the beginning of the file.

public void closefile() Closes the open file. On error, the program exits with error.

Integer

The Integer class provides for integers to be converted to char arrays.

Fields

Integer has no public fields. There is one private field my_int which stores the given integer.

Constructors

Integer(int input) Accepts an integer which is stored in the field my_int.

Methods

public int num() Returns the integer stored in the my_int field.

public char toChar(int digit) Returns in teh input digit as a character.

public class String toString() Converts the integer stored in the my_int field into a string using the toChar() method. Returns a string object.

Built-in Functions

These are functions which are mapped from Dice to the C standard library, which is accessed through LLVM IR. The following function names may not be declared by the user since they are reserved. These are the only functions in dice which are not called as the method of an object; instead the user calls them directly with no dot operator.

int print(...)

The print function can take a char array, int, float and boolean. For char arrays, the contents of the array are printed to stdout. For every other type, the type is converted to the proper variable identifier as used in the C standard library printf function, and then the identifier is replaced with the value of the passed in type when the string is printed to standard out. Arguments can be in any order and must be comma separated.

char[] malloc(int size)

Returns a char pointer to an area of allocated memory on the heap of size bytes.

int open(char[] path)

Attempts to open the file located at the path specified and, if successful, returns a file descriptor to the open file. Returns -1 on failure.

int close(int fd)

Closes the open file identified by the integer fd. Returns 0 if successful and -1 on error.

int read(int fd, char[] buf, int num)

Reads num bytes from the open file identified by fd and stores the resulting string in the char array buf. If successful the number of bytes read is returned. Otherwise returns -1.

int write(int fd, char[] buf, int size)

Writes the contents of the char array buf, which contains size bytes, to the open file identified by fd. If successful the number of bytes written is returned. Otherwise returns -1.

int lseek(int fd, int offset, int whence)

The lseek() function repositions the offset of the open file associated with the file descriptor fd to the argument offset according to the directive whence as follows: 0 - the offset is set to offset bytes, 1 - The offset is set to its current location plus offset bytes, 2 - The offset is set to the size of the file plus offset bytes.

void exit(int flag)

Exits the program. Program exits without error is flag is 0 and exits with error if flag is set to any other integer.

int getchar()

Gets a character from stdin. Returns the character cast to an int.

Functions Implemented in C

With LLVM IR dice is able to compile functions written in C to LLVM. The following functions for dice were written in C.

Declarations

char[] input()

The input function reads from stdin with the C standard library getchar() function, storing each character in a malloc'd char array, until a newline character is read. The resulting array is returned.

long[] init_arr(int[] dims, int dimc)

Takes a list of dimensions in the form of ints and initialize a dimc-dimensional array in a one-dimension malloc call. To access element arr[1][2], first dereference a[1], and cast the value to a long*, which is an address to the array at position 1. Then dereference arr[2] and then cast that to a long* and the value is located at that position. This function is implemented in bindings.c, but was never incorporated directly into the language.

6. Test Plan

We embodied a ”Test Driven Development” approach while creating our programming language. This process entailed writing tests for specific features of our language before starting to implement them. Every test should start by failing in an automated script and then the script should be executed after every modification to any portion of the compiler (from scanner to code generation). This way the team members would know if any modifications made resulted in other tests failing that had previously passed.
The majority of the test cases in our suite check the code generation through a comparison of print statement outputs from the code and our expected output. We created a test for every component of our language from basic variable declaration and assignment to class inheritance and method overriding. If it's in our language, there's a test case for it.

Testing Phases

Unit Testing

In the beginning of the testing process, we set out to thoroughly check the scanner and parser; however, the course instructor suggested we focus on the overall output of the project because testing end-to-end flow was his recommendation. To simplify checking of the Abstract Syntax Tree (AST) and the semantically checked AST (SAST), our manager created a pretty printer that would output the trees in a Javascript Object Notation (JSON) format for quick visual confirmation of their structure. In addition to quick visual feedback JSON objects provide, we also considered using an OCaml JSON visualization package known as yojson to render a visual tree of the data. We then compared the results of this output to the expected results based on the input.

Integration Testing

In addition to running the test suite routinely, we streamlined creation of new test cases by allowing any member of the team to create a git issue (labeled with ”Testing”) whenever a test case idea came to mind. Khaled (Test Suite Creator) would then screen all the open testing issues and add/modify the test according to schedule set by the manager.
During the development process, we also realized that in addition to checking proper output from our programs, we should also check if our analyzer was correctly identifying semantically invalid code. For example, if trying to assign a float type number to an integer variable (a feature we do not support), the analyzer should throw the proper exception. We accounted for these cases and placed all the tests in a separate folder with an identifying prefix to easily determine the category of test case.

Automation

Testing was very simple using ./tester.sh. We can verify that a test works individually by running lli on the outputted ll file

Test Suites

We created a total of 121 tests divided into two categories. One checks that the compiler is properly recognizing invalid code. The other checks that the compiler accepts valid code and tests the output program.

7. Lessons Learned

David

Most critically I learned that if you want to make something good, put as much effort as physically possible into it. I was told frequently ”get started early” with respect to this project. After starting early I also learned that working often and with purpose helped not only myself get through the project but also the rest of my team.
As project manager the most critical decision I made was to gain consensus on the development environment that each team member was using. My main takeaway was to make sure that everyone agrees to use the same tools and systems. Having incompatible hardware/software can create unnecessary tension in what is already a stressful situation.
One final note is that I really did not know what to expect from OCaml coming into this class. It seemed very mysterious at first, but after looking through previous examples of compilers from other groups and writing out the Analyzer for my language, I quickly grew to enjoy the language. It certainly was not as daunting as it seemed at first.

Emily

If you’re collaborating with someone to implement a feature where there are design decisions affecting different components of the compiler, then both of you should iterate on your respective parts simultaneously and communicate with each other. In other words, before your teammate has a chance to prototype their part, implement the bare minimum to test whether the overall design works. Also, OCaml turned out to be a good tool for writing a compiler (because of all the tree traversals we did for type-checking and implementing inheritance) so I think learning it was a good investment.

Khaled

Read the lessons learned from previous projects and prioritize (with your group) which of them you will implement. You will not be able to do them all, but if you can agree as a group on which mistakes you can avoid, you’re already ahead. For our group, we determined that we will ACTUALLY start early, which we we did.
Fortunately, we had a very organized and decisive manager that made sure we were all on track throughout the semester. Make sure you nominate a person with same qualities if you don’t want to spend the last week of the semester pulling all-nighters for this project (save that for your other exams).
Track tasks with Github’s issue tracking. Keep this issue tracker open during meetings with the Professor/TAs in order to avoid forgetting discussed to-do items. Ensure the manager of the group delegates through this system.
To spare your team members pain, don’t use the diff command’s output in your test script. Just label the program’s output and your expected output and place them on top of each other for easy reading.

Philip

This project was a good overall lesson in how important it is to plan ahead when constructing a piece of software with a large, complex codebase. Our manager did a great job of making sure that we always had a plan of action when attacking each new problem, which was key in making sure the project came to fruition. Also, watch out for any rogue characters, especially 'h'.

8. Code Listing

Code

analyzer.ml

```
open Sast
open Ast
open Processor
open Utils
open Filepath
open Conf

module StringMap = Map.Make (String)

module StringSet = Set.Make (String)

let struct_indexes:(string, int) Hashtbl.t = Hashtbl.create 10
let predecessors:(string, string list) Hashtbl.t = Hashtbl.create 10

module SS = Set.Make(
        struct
                let compare = Pervasives.compare
                type t = datatype
        end )

type class_map = {
        field_map       : Ast.field StringMap.t;
        func_map        : Ast.func_decl StringMap.t;
        constructor_map : Ast.func_decl StringMap.t;
        reserved_map         : sfunc_decl StringMap.t;
        cdecl                          : Ast.class_decl;
}

type env = {
        env_class_maps: class_map StringMap.t;
        env_name      : string;
        env_cmap           : class_map;
        env_locals    : datatype StringMap.t;
        env_parameters: Ast.formal StringMap.t;
        env_returnType: datatype;
        env_in_for    : bool;
        env_in_while  : bool;
        env_reserved  : sfunc_decl list;
}

let update_env_name env env_name =
{
        env_class_maps = env.env_class_maps;
        env_name       = env_name;
        env_cmap               = env.env_cmap;
        env_locals     = env.env_locals;
        env_parameters = env.env_parameters;
        env_returnType = env.env_returnType;
        env_in_for     = env.env_in_for;
        env_in_while   = env.env_in_while;
        env_reserved   = env.env_reserved;
}

let update_call_stack env in_for in_while =
{
        env_class_maps = env.env_class_maps;
        env_name       = env.env_name;
        env_cmap               = env.env_cmap;
        env_locals     = env.env_locals;
        env_parameters = env.env_parameters;
        env_returnType = env.env_returnType;
        env_in_for     = in_for;
        env_in_while   = in_while;
        env_reserved   = env.env_reserved;
}

let append_code_to_constructor fbody cname ret_type =
        let key = Hashtbl.find struct_indexes cname in
        let init_this = [SLocal(
                ret_type,
                "this",
                SCall(      "cast",
                                [SCall("malloc",
                                        [
                                                SCall("sizeof", [SId("ignore",
↪ ret_type)], Datatype(Int_t), 0)
                                        ],
                                        Arraytype(Char_t, 1), 0)
                                ],
                                ret_type,
                                0
                        )
                );
                SExpr(
                        SAssign(
                                SObjAccess(
                                        SId("this", ret_type),
                                        SId(".key", Datatype(Int_t)),
                                        Datatype(Int_t)
                                ),
                                SInt_Lit(key),
                                Datatype(Int_t)
                        ),
                        Datatype(Int_t)
                )
        ]
        in
        let ret_this =
                [
                        SReturn(
                                SId("this", ret_type),
                                ret_type
                        )
                ]
        in
        (* Need to check for duplicate default constructs *)
        (* Also need to add malloc around other constructors *)
        init_this @ fbody @ ret_this

let default_constructor_body cname =
        let ret_type = Datatype(Objecttype(cname)) in
        let fbody = [] in
        append_code_to_constructor fbody cname ret_type

let default_sc cname =
{
        sfname                          = Ast.FName (cname ^ "." ^ "constructor");
        sreturnType          = Datatype(Objecttype(cname));
        sformals                  = [];
        sbody                           = default_constructor_body cname;
        func_type                 = Sast.User;
        overrides       = false;
        source                          = "NA";
}

let default_c cname =
{
        scope                           = Ast.Public;
        fname                           = Ast.Constructor;
        returnType                = Datatype(ConstructorType);
        formals                 = [];
        body                            = [];
        overrides                 = false;
        root_cname                = None;
}
```

```

let process_includes filename includes classes =
        (* Bring in each include  *)
        let processInclude include_statement =
                let file_in = open_in include_statement in
                let lexbuf = Lexing.from_channel file_in in
                let token_list = Processor.build_token_list lexbuf in
                let program = Processor.parser include_statement token_list in
                ignore(close_in file_in);
                program
        in
        let rec iterate_includes classes m = function
                        [] -> classes
                | (Include h) :: t ->
                        let h = if h = "stdlib" then Conf.stdlib_path else h in
                        (* Check each include against the map *)
                        let realpath = Filepath.realpath h in
                        if StringMap.mem realpath m then
                                iterate_includes (classes) (m) (t)
                        else
                                let result = processInclude realpath in
                                match result with Program(i,c) ->
                                iterate_includes (classes @ c) (StringMap.add realpath 1
↪ m) (i @ t)
        in
        iterate_includes classes (StringMap.add (Filepath.realpath filename) 1
↪ StringMap.empty) includes

let get_name cname fdecl =
        (* We use '.' to separate types so llvm will recognize the function name and it
↪ won't conflict *)
        (* let params = List.fold_left (fun s -> (function Formal(t, _) -> s ^ "." ^
↪ Utils.string_of_datatype t | _ -> "" )) "" fdecl.formals in *)
        let name = Utils.string_of_fname fdecl.fname in
        if name = "main"
                then "main"
                else cname ^ "." ^ name(*  ^ params *)

let get_constructor_name cname fdecl =
        let params = List.fold_left (fun s -> (function Formal(t, _) -> s ^ "." ^
↪ Utils.string_of_datatype t | _ -> "" )) "" fdecl.formals in
        let name = Utils.string_of_fname fdecl.fname in
        cname ^ "." ^ name ^ params

let get_name_without_class fdecl =
        (* We use '.' to separate types so llvm will recognize the function name and it
↪ won't conflict *)
        let params = List.fold_left (fun s -> (function Formal(t, _) -> s ^ "." ^
↪ Utils.string_of_datatype t | _ -> "" )) "" fdecl.formals in
```

```
        let name = Utils.string_of_fname fdecl.fname in
    let ret_type = Utils.string_of_datatype fdecl.returnType in
    ret_type ^ "." ^ name ^ "." ^ params

(* Generate list of all classes to be used for semantic checking *)
let build_class_maps reserved cdecls =
        let reserved_map = List.fold_left (fun m f -> StringMap.add (Utils.string_of_fname f.sfname) f m) StringMap.empty reserved in
        let helper m (cdecl:Ast.class_decl) =
                let fieldfun = (fun m -> (function Field(s, d, n) -> if (StringMap.mem (n) m) then raise(Exceptions.DuplicateField) else (StringMap.add n (Field(s, d, n)) m))) in
                let funcname = get_name cdecl.cname in
                let funcfun m fdecl =
                        if (StringMap.mem (funcname fdecl) m)
                                then raise(Exceptions.DuplicateFunction(funcname fdecl))
                        else if (StringMap.mem (Utils.string_of_fname fdecl.fname) reserved_map)
                                then raise(Exceptions.CannotUseReservedFuncName(Utils.string_of_fname fdecl.fname))
                        else (StringMap.add (funcname fdecl) fdecl m)
                in
                let constructor_name = get_constructor_name cdecl.cname in
                let constructorfun m fdecl =
                        if fdecl.formals = [] then m
                        else if StringMap.mem (constructor_name fdecl) m
                                then raise(Exceptions.DuplicateConstructor)
                                else (StringMap.add (constructor_name fdecl) fdecl m)
                in
                let default_c = default_c cdecl.cname in
                let constructor_map = StringMap.add (get_constructor_name cdecl.cname default_c) default_c StringMap.empty in
                (if (StringMap.mem cdecl.cname m) then raise (Exceptions.DuplicateClassName(cdecl.cname)) else
                        StringMap.add cdecl.cname
                        {       field_map = List.fold_left fieldfun StringMap.empty cdecl.cbody.fields;
                                func_map = List.fold_left funcfun StringMap.empty cdecl.cbody.methods;
                                constructor_map = List.fold_left constructorfun constructor_map cdecl.cbody.constructors;
                                reserved_map = reserved_map;
                                cdecl = cdecl }
                                                                                        m) in
        List.fold_left helper StringMap.empty cdecls

let rec get_all_descendants cname accum =
    if Hashtbl.mem predecessors cname then
        let direct_descendants = Hashtbl.find predecessors cname in
        let add_childs_descendants desc_set direct_descendant =
                get_all_descendants direct_descendant (StringSet.add direct_descendant
↪   desc_set)
    in
    List.fold_left add_childs_descendants accum direct_descendants
    else accum

let inherited potential_predec potential_child =
    match potential_predec, potential_child with
    Datatype(Objecttype(predec_cname)), Datatype(Objecttype(child_cname)) ->
        let descendants = get_all_descendants predec_cname StringSet.empty in
        if (predec_cname = child_cname) || (StringSet.mem child_cname descendants) then
↪   true
            else raise (Exceptions.LocalAssignTypeMismatch(predec_cname, child_cname))
    | _ , _ -> false

let get_equality_binop_type type1 type2 se1 se2 op =
        (* Equality op not supported for float operands. The correct way to test floats
           for equality is to check the difference between the operands in question *)
        if (type1 = Datatype(Float_t) || type2 = Datatype(Float_t)) then raise
↪   (Exceptions.InvalidBinopExpression "Equality operation is not supported for Float
↪   types")
        else
        match type1, type2 with
                Datatype(Char_t), Datatype(Int_t)
        |         Datatype(Int_t), Datatype(Char_t)
        |         Datatype(Objecttype(_)), Datatype(Null_t)
        |         Datatype(Null_t), Datatype(Objecttype(_))
        |         Datatype(Null_t), Arraytype(_, _)
        |         Arraytype(_, _), Datatype(Null_t) -> SBinop(se1, op, se2,
↪   Datatype(Bool_t))
        | _ ->
                if type1 = type2 then SBinop(se1, op, se2, Datatype(Bool_t))
                else raise (Exceptions.InvalidBinopExpression "Equality operator can't
↪   operate on different types, with the exception of Int_t and Char_t")

let get_logical_binop_type se1 se2 op = function
        (Datatype(Bool_t), Datatype(Bool_t)) -> SBinop(se1, op, se2, Datatype(Bool_t))
        | _ -> raise (Exceptions.InvalidBinopExpression "Logical operators only operate
↪   on Bool_t types")

let get_comparison_binop_type type1 type2 se1 se2 op =
        let numerics = SS.of_list [Datatype(Int_t); Datatype(Char_t); Datatype(Float_t)]
        in
                if SS.mem type1 numerics && SS.mem type2 numerics
                        then SBinop(se1, op, se2, Datatype(Bool_t))
                else raise (Exceptions.InvalidBinopExpression "Comparison operators
↪   operate on numeric types only")

```

```

let get_arithmetic_binop_type se1 se2 op = function
            (Datatype(Int_t), Datatype(Float_t))
        |    (Datatype(Float_t), Datatype(Int_t))
        |    (Datatype(Float_t), Datatype(Float_t))    -> SBinop(se1, op, se2, Datatype(Float_t))

        |    (Datatype(Int_t), Datatype(Char_t))
        |    (Datatype(Char_t), Datatype(Int_t))
        |    (Datatype(Char_t), Datatype(Char_t))    -> SBinop(se1, op, se2, Datatype(Char_t))

        |    (Datatype(Int_t), Datatype(Int_t))          -> SBinop(se1, op, se2, Datatype(Int_t))

        | _ -> raise (Exceptions.InvalidBinopExpression "Arithmetic operators don't support these types")

let rec get_ID_type env s =
    try StringMap.find s env.env_locals
    with | Not_found ->
    try let formal = StringMap.find s env.env_parameters in
        (function Formal(t, _) -> t | Many t -> t ) formal
    with | Not_found -> raise (Exceptions.UndefinedID s)

and check_array_primitive env el =
    let rec iter t sel = function
        [] -> sel, t
    |    e :: el ->
        let se, _ = expr_to_sexpr env e in
        let se_t = get_type_from_sexpr se in
        if t = se_t
            then iter t (se :: sel) el
            else
                let t1 = Utils.string_of_datatype t in
                let t2 = Utils.string_of_datatype se_t in
                raise(Exceptions.InvalidArrayPrimitiveConsecutiveTypes(t1, t2))
    in
    let se, _ = expr_to_sexpr env (List.hd el) in
    let el = List.tl el in
    let se_t = get_type_from_sexpr se in
    let sel, t = iter se_t ([se]) el in
    let se_t = match t with
                    Datatype(x) -> Arraytype(x, 1)
                |    Arraytype(x, n) -> Arraytype(x, n+1)
                |    _ as t -> raise(Exceptions.InvalidArrayPrimitiveType(Utils.string_of_datatype t))
    in
```

```
        SArrayPrimitive(sel, se_t)

and check_array_init env d el =
        (* Get dimension size for the array being created *)
        let array_complexity = List.length el in
        let check_elem_type e =
                let sexpr, _ = expr_to_sexpr env e in
                let sexpr_type = get_type_from_sexpr sexpr in
                if sexpr_type = Datatype(Int_t)
                        then sexpr
                        else raise(Exceptions.MustPassIntegerTypeToArrayCreate)
        in
        let convert_d_to_arraytype = function
                Datatype(x) -> Arraytype(x, array_complexity)
        |         _ as t ->
                let error_msg = Utils.string_of_datatype t in
                raise (Exceptions.ArrayInitTypeInvalid(error_msg))
        in
        let sexpr_type = convert_d_to_arraytype d in
        let sel = List.map check_elem_type el in
        SArrayCreate(d, sel, sexpr_type)

and check_array_access env e el =
        (* Get dimensions of array, ex: foo[10][4][2] is dimen=3 *)
        let array_dimensions = List.length el in
        (* Check every e in el is of type Datatype(Int_t). Ensure all indices are ints *)
        let check_elem_type arg =
                let sexpr, _ = expr_to_sexpr env arg in
                let sexpr_type = get_type_from_sexpr sexpr in
                if sexpr_type = Datatype(Int_t)
                        then sexpr
                        else raise(Exceptions.MustPassIntegerTypeToArrayAccess)
        in
        (* converting e to se also checks if the array id has been declared  *)
        let se, _ = expr_to_sexpr env e in
        let se_type = get_type_from_sexpr se in

        (* Check that e has enough dimens as e's in el. Return overall datatype of
 ↪  access*)
        let check_array_dim_vs_params num_params = function
                Arraytype(t, n) ->
                        if num_params < n then
                                Arraytype(t, (n-num_params))
                        else if num_params = n then
                                Datatype(t)
                        else
                                raise
 ↪  (Exceptions.ArrayAccessInvalidParamLength(string_of_int num_params, string_of_int n))
        |         _ as t ->
```

```
                let error_msg = Utils.string_of_datatype t in
                raise (Exceptions.ArrayAccessExpressionNotArray(error_msg))
        in
        let sexpr_type = check_array_dim_vs_params array_dimensions se_type in
        let sel = List.map check_elem_type el in

        SArrayAccess(se, sel, sexpr_type)

and check_obj_access env lhs rhs =
        let check_lhs = function
                This                         -> SId("this",
↪   Datatype(Objecttype(env.env_name)))
        |        Id s                         -> SId(s, get_ID_type env s)
        |         ArrayAccess(e, el)        -> check_array_access env e el
        |         _ as e         -> raise (Exceptions.LHSofRootAccessMustBeIDorFunc
↪   (Utils.string_of_expr e))
        in
        let ptype_name parent_type = match parent_type with
                        Datatype(Objecttype(name))         -> name
                |         _ as d                                                     -> raise
↪   (Exceptions.ObjAccessMustHaveObjectType (Utils.string_of_datatype d))
        in
        let rec check_rhs (env) parent_type (top_level_env) =
                let pt_name = ptype_name parent_type in
                let get_id_type_from_object env (id) cname tlenv =
                        let cmap = StringMap.find cname env.env_class_maps in
                        let match_field f = match f with
                                Field(scope, d, n) ->
                                        (* Have to update this with all parent classes
↪   checks *)
                                        if scope = Ast.Private && tlenv.env_name <>
↪   env.env_name then
                                                raise(Exceptions.CannotAccessPrivateFieldInNonProperScop
↪   env.env_name, tlenv.env_name))
                                        else d
                        in
                        try match_field (StringMap.find id cmap.field_map)
                        with | Not_found -> raise
↪   (Exceptions.UnknownIdentifierForClass(id, cname))
                in
                function
                        (* Check fields in parent *)
                        Id s                                 -> SId(s,
↪   (get_id_type_from_object env s pt_name top_level_env)), env
                        (* Check functions in parent *)
                |         Call(fname, el)        ->
                                let env = update_env_name env pt_name in
                                check_call_type top_level_env true env fname el, env
                        (* Set parent, check if base is field *)
                |       ObjAccess(e1, e2)         ->
                              let old_env = env in
                              let lhs, env = check_rhs env parent_type top_level_env e1
↪ in
                              let lhs_type = get_type_from_sexpr lhs in

                              let pt_name = ptype_name lhs_type in
                              let lhs_env = update_env_name env pt_name in

                              let rhs, env = check_rhs lhs_env lhs_type top_level_env
↪ e2 in
                              let rhs_type = get_type_from_sexpr rhs in
                              SObjAccess(lhs, rhs, rhs_type), old_env
                |       _ as e                          -> raise
↪ (Exceptions.InvalidAccessLHS (Utils.string_of_expr e))
        in
        let arr_lhs, _ = expr_to_sexpr env lhs in
        let arr_lhs_type = get_type_from_sexpr arr_lhs in
        match arr_lhs_type with
                Arraytype(Char_t, 1) -> raise(Exceptions.CannotAccessLengthOfCharArray)
        |        Arraytype(_, _) ->
                        let rhs = match rhs with
                                Id("length") -> SId("length", Datatype(Int_t))
                        |         _ -> raise(Exceptions.CanOnlyAccessLengthOfArray)
                        in
                        SObjAccess(arr_lhs, rhs, Datatype(Int_t))
        | _ ->
                let lhs = check_lhs lhs in
                let lhs_type = get_type_from_sexpr lhs in

                let ptype_name = ptype_name lhs_type in
                let lhs_env = update_env_name env ptype_name in

                let rhs, _ = check_rhs lhs_env lhs_type env rhs in
                let rhs_type = get_type_from_sexpr rhs in
                SObjAccess(lhs, rhs, rhs_type)

and check_call_type top_level_env isObjAccess env fname el =
        let sel, env = exprl_to_sexprl env el in
        (* check that 'env.env_name' is in the list of defined classes *)
        let cmap =
                try StringMap.find env.env_name env.env_class_maps
                with | Not_found -> raise (Exceptions.UndefinedClass env.env_name)
        in

        let handle_param formal param =
                let fty = match formal with Formal(d, _) -> d | _ -> Datatype(Void_t) in
                let pty = get_type_from_sexpr param in
                match fty, pty with
```

```
                      Datatype(Objecttype(f)), Datatype(Objecttype(p)) ->
                              if f <> p then
                              try let descendants = Hashtbl.find predecessors f in
                              let _ = try List.find (fun d -> p = d) descendants
                                              with | Not_found ->
↪   raise(Exceptions.CannotPassNonInheritedClassesInPlaceOfOthers(f, p))
                              in
                              let rt = Datatype(Objecttype(f)) in
                              SCall("cast", [param; SId("ignore", rt)], rt, 0)
                              with | Not_found ->
↪   raise(Exceptions.ClassIsNotExtendedBy(f, p))
                              else param
                  |       _ -> if fty = pty then param else
↪   raise(Exceptions.IncorrectTypePassedToFunction(fname, Utils.string_of_datatype pty))
        in

        let index fdecl fname =
                let cdecl = cmap.cdecl in
                (* Have to update this with all parent classes checks *)
                let _ =
                        if fdecl.scope = Ast.Private && top_level_env.env_name <>
↪   env.env_name then
                        raise(Exceptions.CannotAccessPrivateFunctionInNonProperScope(get_name
↪   env.env_name fdecl, env.env_name, top_level_env.env_name))
                in
                (* Not exactly sure why there needs to be a list.rev *)
                let fns = List.rev cdecl.cbody.methods in
                let rec find x lst =
                    match lst with
                    | [] -> raise (Failure ("Could not find " ^ fname))
                    | fdecl :: t ->
                            let search_name = (get_name env.env_name fdecl) in
                            if x = search_name then 0
                            else if search_name = "main" then find x t
                            else 1 + find x t
                in
                find fname fns
        in

        let handle_params (formals) params =
                match formals, params with
                        [Many(Any)], _ -> params
                |         [], [] -> []
                |         [], _
                |         _, [] -> raise(Exceptions.IncorrectTypePassedToFunction(fname,
↪   Utils.string_of_datatype (Datatype(Void_t))))
                |         _ ->
                let len1 = List.length formals in
                let len2 = List.length params in
                if len1 <> len2 then raise(Exceptions.IncorrectNumberOfArguments(fname,
↪  len1, len2))
                else
                List.map2 handle_param formals sel
        in

        let sfname = env.env_name ^ "." ^ fname in
        try let func = StringMap.find fname cmap.reserved_map in
                let actuals = handle_params func.sformals sel in
                SCall(fname, actuals, func.sreturnType, 0)
        with | Not_found ->
        try let f = StringMap.find sfname cmap.func_map in
                let actuals = handle_params f.formals sel in
                let index = index f sfname in
                SCall(sfname, actuals, f.returnType, index)
        with | Not_found -> raise(Exceptions.FunctionNotFound(env.env_name, sfname)) | _
↪  as ex -> raise ex

and check_object_constructor env s el =
        let sel, env = exprl_to_sexprl env el in
        (* check that 'env.env_name' is in the list of defined classes *)
        let cmap =
                try StringMap.find s env.env_class_maps
                with | Not_found -> raise (Exceptions.UndefinedClass s)
        in
        (* get a list of the types of the actuals to match against defined function
↪  formals *)
        let params = List.fold_left (fun s e -> s ^ "." ^ (Utils.string_of_datatype
↪  (get_type_from_sexpr e))) "" sel in
        let constructor_name = s ^ "." ^ "constructor" ^ params in
        let _ =
                try StringMap.find constructor_name cmap.constructor_map
                with | Not_found -> raise (Exceptions.ConstructorNotFound
↪  constructor_name)
        in
        let ftype = Datatype(Objecttype(s)) in
        (* Add a reference to the class in front of the function call *)
        (* Must properly handle the case where this is a reserved function *)
        SObjectCreate(constructor_name, sel, ftype)

and check_assign env e1 e2 =
        let se1, env = expr_to_sexpr env e1 in
        let se2, env = expr_to_sexpr env e2 in
        let type1 = get_type_from_sexpr se1 in
        let type2 = get_type_from_sexpr se2 in
        match (type1, se2) with
                Datatype(Objecttype(_)), SNull
        |         Arraytype(_, _), SNull -> SAssign(se1, se2, type1)
        |    _ ->
        match type1, type2 with
                Datatype(Char_t), Datatype(Int_t)
        |         Datatype(Int_t), Datatype(Char_t) -> SAssign(se1, se2, type1)
        |         Datatype(Objecttype(d)), Datatype(Objecttype(t)) ->
                if d = t then SAssign(se1, se2, type1)
                else if inherited type1 type2 then
                        SAssign(se1, SCall("cast", [se2; SId("ignore", type1)], type1,
↪   0), type1)
                else raise (Exceptions.AssignmentTypeMismatch(Utils.string_of_datatype
↪   type1, Utils.string_of_datatype type2))
        | _ ->
        if type1 = type2
                then SAssign(se1, se2, type1)
                else raise (Exceptions.AssignmentTypeMismatch(Utils.string_of_datatype
↪   type1, Utils.string_of_datatype type2))

and check_unop env op e =
        let check_num_unop t = function
                        Sub         -> t
                |           _                -> raise(Exceptions.InvalidUnaryOperation)
        in
        let check_bool_unop = function
                        Not         -> Datatype(Bool_t)
                |           _                -> raise(Exceptions.InvalidUnaryOperation)
        in
        let se, env = expr_to_sexpr env e in
        let t = get_type_from_sexpr se in
        match t with
                Datatype(Int_t)
        |        Datatype(Float_t)         -> SUnop(op, se, check_num_unop t op)
        |          Datatype(Bool_t)          -> SUnop(op, se, check_bool_unop op)
        |         _ -> raise(Exceptions.InvalidUnaryOperation)

and check_binop env e1 op e2 =
        let se1, env = expr_to_sexpr env e1 in
        let se2, env = expr_to_sexpr env e2 in
        let type1 = get_type_from_sexpr se1 in
        let type2 = get_type_from_sexpr se2 in
        match op with
        Equal | Neq -> get_equality_binop_type type1 type2 se1 se2 op
        | And | Or -> get_logical_binop_type se1 se2 op (type1, type2)
        | Less | Leq | Greater | Geq -> get_comparison_binop_type type1 type2 se1 se2 op
        | Add | Mult | Sub | Div | Mod -> get_arithmetic_binop_type se1 se2 op (type1,
↪   type2)
        | _ -> raise (Exceptions.InvalidBinopExpression ((Utils.string_of_op op) ^ " is
↪   not a supported binary op"))

and check_delete env e =
        let se, _ = expr_to_sexpr env e in
        let t = get_type_from_sexpr se in
        match t with
                Arraytype(_, _) | Datatype(Objecttype(_)) -> SDelete(se)
        |         _ -> raise(Exceptions.CanOnlyDeleteObjectsOrArrays)

and expr_to_sexpr env = function
                Int_Lit i           -> SInt_Lit(i), env
        |   Boolean_Lit b       -> SBoolean_Lit(b), env
        |   Float_Lit f         -> SFloat_Lit(f), env
        |   String_Lit s        -> SString_Lit(s), env
        |   Char_Lit c          -> SChar_Lit(c), env
        |   This                -> SId("this", Datatype(Objecttype(env.env_name))), env
        |   Id s                -> SId(s, get_ID_type env s), env
        |   Null                -> SNull, env
        |   Noexpr              -> SNoexpr, env

        |   ObjAccess(e1, e2)   -> check_obj_access env e1 e2, env
        |   ObjectCreate(s, el) -> check_object_constructor env s el, env
        |   Call(s, el)         -> check_call_type env false env s el, env

        |   ArrayCreate(d, el)  -> check_array_init env d el, env
        |   ArrayAccess(e, el)  -> check_array_access env e el, env
        |   ArrayPrimitive el   -> check_array_primitive env el, env

        |   Assign(e1, e2)      -> check_assign env e1 e2, env
        |   Unop(op, e)         -> check_unop env op e, env
        |   Binop(e1, op, e2)   -> check_binop env e1 op e2, env
        |         Delete(e)                     -> check_delete env e, env


and get_type_from_sexpr = function
                SInt_Lit(_)                           -> Datatype(Int_t)
        |         SBoolean_Lit(_)                     -> Datatype(Bool_t)
        |         SFloat_Lit(_)                     -> Datatype(Float_t)
        |         SString_Lit(_)                      -> Arraytype(Char_t, 1)
        |         SChar_Lit(_)                      -> Datatype(Char_t)
        |         SId(_, d)                               -> d
        |         SBinop(_, _, _, d)                  -> d
        |         SAssign(_, _, d)                  -> d
        |         SNoexpr                                -> Datatype(Void_t)
        |         SArrayCreate(_, _, d)         -> d
        |         SArrayAccess(_, _, d)          -> d
        |         SObjAccess(_, _, d)               -> d
        |         SCall(_, _, d, _)                 -> d
        |   SObjectCreate(_, _, d)          -> d
        |         SArrayPrimitive(_, d)         -> d
        |          SUnop(_, _, d)                      -> d
        |         SNull                                       -> Datatype(Null_t)
        |         SDelete _                                -> Datatype(Void_t)
```

```

and exprl_to_sexprl env el =
  let env_ref = ref(env) in
  let rec helper = function
          head::tail ->
                let a_head, env = expr_to_sexpr !env_ref head in
                env_ref := env;
                a_head::(helper tail)
        | [] -> []
  in (helper el), !env_ref

let rec local_handler d s e env =
        if StringMap.mem s env.env_locals
                then raise (Exceptions.DuplicateLocal s)
                else
                        let se, env = expr_to_sexpr env e in
                        let t = get_type_from_sexpr se in
                        if t = Datatype(Void_t) || t = Datatype(Null_t) || t = d ||
↪  (inherited d t)
                                then
                                let new_env = {
                                        env_class_maps = env.env_class_maps;
                                        env_name = env.env_name;
                                        env_cmap = env.env_cmap;
                                        env_locals = StringMap.add s d env.env_locals;
                                        env_parameters = env.env_parameters;
                                        env_returnType = env.env_returnType;
                                        env_in_for = env.env_in_for;
                                        env_in_while = env.env_in_while;
                                        env_reserved = env.env_reserved;
                                } in
(* if the user-defined type being declared is not in global classes map, it is an
↪  undefined class *)
                                (match d with
                                        Datatype(Objecttype(x)) ->
                                                (if not (StringMap.mem
↪  (Utils.string_of_object d) env.env_class_maps)
                                                        then raise
↪  (Exceptions.UndefinedClass (Utils.string_of_object d))
                                                        else
                                let local = if inherited d t then SLocal(t, s, se) else
↪  SLocal(d, s, se)
                                in local, new_env)
                                | _ -> SLocal(d, s, se), new_env)
                        else
                                (let type1 = (Utils.string_of_datatype t) in
                                let type2 = (Utils.string_of_datatype d) in
                                let ex = Exceptions.LocalAssignTypeMismatch(type1, type2)
↪  in
```

```
                                raise ex)

let rec check_sblock sl env = match sl with
                [] -> SBlock([SExpr(SNoexpr, Datatype(Void_t))]), env
        |         _  ->
                let sl, _ = convert_stmt_list_to_sstmt_list env sl in
                SBlock(sl), env

and check_expr_stmt e env =
        let se, env = expr_to_sexpr env e in
        let t = get_type_from_sexpr se in
        SExpr(se, t), env

and check_return e env =
        let se, _ = expr_to_sexpr env e in
        let t = get_type_from_sexpr se in
        match t, env.env_returnType with
                Datatype(Null_t), Datatype(Objecttype(_))
        |         Datatype(Null_t), Arraytype(_, _) -> SReturn(se, t), env
        |         _ ->
        if t = env.env_returnType
                then SReturn(se, t), env
                else raise (Exceptions.ReturnTypeMismatch(Utils.string_of_datatype t,
 ↪  Utils.string_of_datatype env.env_returnType))

and check_if e s1 s2 env =
        let se, _ = expr_to_sexpr env e in
        let t = get_type_from_sexpr se in
        let ifbody, _ = parse_stmt env s1 in
        let elsebody, _ = parse_stmt env s2 in
        if t = Datatype(Bool_t)
                then SIf(se, ifbody, elsebody), env
                else raise Exceptions.InvalidIfStatementType

and check_for e1 e2 e3 s env =
        let old_val = env.env_in_for in
        let env = update_call_stack env true env.env_in_while in

        let se1, _ = expr_to_sexpr env e1 in
        let se2, _ = expr_to_sexpr env e2 in
        let se3, _ = expr_to_sexpr env e3 in
        let forbody, _ = parse_stmt env s in
        let conditional = get_type_from_sexpr se2 in
        let sfor =
                if (conditional = Datatype(Bool_t) || conditional = Datatype(Void_t))
                        then SFor(se1, se2, se3, forbody)
                        else raise Exceptions.InvalidForStatementType
        in

        let env = update_call_stack env old_val env.env_in_while in
        sfor, env

and check_while e s env =
        let old_val = env.env_in_while in
        let env = update_call_stack env env.env_in_for true in

        let se, _ = expr_to_sexpr env e in
        let t = get_type_from_sexpr se in
        let sstmt, _ = parse_stmt env s in
        let swhile =
                if (t = Datatype(Bool_t) || t = Datatype(Void_t))
                        then SWhile(se, sstmt)
                        else raise Exceptions.InvalidWhileStatementType
        in

        let env = update_call_stack env env.env_in_for old_val in
        swhile, env

and check_break env =
        if env.env_in_for || env.env_in_while then
                SBreak, env
        else
                raise Exceptions.CannotCallBreakOutsideOfLoop

and check_continue env =
        if env.env_in_for || env.env_in_while then
                SContinue, env
        else
                raise Exceptions.CannotCallContinueOutsideOfLoop

and parse_stmt env = function
                Block sl                                 -> check_sblock sl env
        |         Expr e                                         -> check_expr_stmt e env
        |         Return e                                 -> check_return e env
        |         If(e, s1, s2)                         -> check_if e s1 s2        env
        |         For(e1, e2, e3, e4)         -> check_for e1 e2 e3 e4 env
        |         While(e, s)                                -> check_while e s env
        |          Break                                         -> check_break env (* Need to check if in right context *)
        |   Continue                                 -> check_continue env (* Need to check if in right context *)
        |   Local(d, s, e)                         -> local_handler d s e env

(* Update this function to return an env object *)
and convert_stmt_list_to_sstmt_list env stmt_list =
        let env_ref = ref(env) in
        let rec iter = function
          head::tail ->
```

```
                let a_head, env = parse_stmt !env_ref head in
                env_ref := env;
                a_head::(iter tail)
        | [] -> []
        in
        let sstmt_list = (iter stmt_list), !env_ref in
        sstmt_list

let append_code_to_main fbody cname ret_type =
        let key = Hashtbl.find struct_indexes cname in
        let init_this = [SLocal(
                ret_type,
                "this",
                SCall(        "cast",
                                [SCall("malloc",
                                        [
                                                SCall("sizeof", [SId("ignore",
↪   ret_type)], Datatype(Int_t), 0)
                                        ],
                                        Arraytype(Char_t, 1), 0)
                                ],
                                ret_type, 0
                        )
                );
                SExpr(
                        SAssign(
                                SObjAccess(
                                        SId("this", ret_type),
                                        SId(".key", Datatype(Int_t)),
                                        Datatype(Int_t)
                                ),
                                SInt_Lit(key),
                                Datatype(Int_t)
                        ),
                        Datatype(Int_t)
                )
        ]
        in
        init_this @ fbody

let convert_constructor_to_sfdecl class_maps reserved class_map cname constructor =
        let env = {
                env_class_maps          = class_maps;
                env_name              = cname;
                env_cmap                  = class_map;
                env_locals             = StringMap.empty;
                env_parameters         = List.fold_left (fun m f -> match f with Formal(d,
↪   s) -> (StringMap.add s f m) | _ -> m) StringMap.empty constructor.formals;
                env_returnType         = Datatype(Objecttype(cname));
                env_in_for                = false;
                env_in_while          = false;
                env_reserved          = reserved;
        } in
        let fbody = fst (convert_stmt_list_to_sstmt_list env constructor.body) in
        {
                sfname                  = Ast.FName (get_constructor_name cname
    ↪   constructor);
                sreturnType = Datatype(Objecttype(cname));
                sformals          = constructor.formals;
                sbody                   = append_code_to_constructor fbody cname
    ↪   (Datatype(Objecttype(cname)));
                func_type         = Sast.User;
                overrides          = false;
                source                  = "NA";
        }

let check_fbody fname fbody returnType =
        let len = List.length fbody in
        if len = 0 then () else
        let final_stmt = List.hd (List.rev fbody) in
        match returnType, final_stmt with
                Datatype(Void_t), _ -> ()
        |          _, SReturn(_, _) -> ()
        |          _ -> raise(Exceptions.AllNonVoidFunctionsMustEndWithReturn(fname))

let convert_fdecl_to_sfdecl class_maps reserved class_map cname fdecl =
        let root_cname = match fdecl.root_cname with
        Some(x) -> x
        | None -> cname
    in
    let class_formal =
            if fdecl.overrides then
                    Ast.Formal(Datatype(Objecttype(root_cname)), "this")
            else
                    Ast.Formal(Datatype(Objecttype(cname)), "this")
    in
        let env_param_helper m fname = match fname with
                        Formal(d, s) -> (StringMap.add s fname m)
                |          _ -> m
        in
        let env_params = List.fold_left env_param_helper StringMap.empty (class_formal ::
    ↪   fdecl.formals) in
        let env = {
                env_class_maps          = class_maps;
                env_name              = cname;
                env_cmap                  = class_map;
                env_locals             = StringMap.empty;
                env_parameters         = env_params;
                env_returnType        = fdecl.returnType;
                env_in_for                   = false;
                env_in_while          = false;
                env_reserved          = reserved;
        }
        in
        let fbody = fst (convert_stmt_list_to_sstmt_list env fdecl.body) in
        let fname = (get_name cname fdecl) in
        ignore(check_fbody fname fbody fdecl.returnType);
        let fbody = if fname = "main"
                then (append_code_to_main fbody cname (Datatype(Objecttype(cname))))
                else fbody
        in
        (* We add the class as the first parameter to the function for codegen *)
        {
                sfname                         = Ast.FName (get_name cname fdecl);
                sreturnType         = fdecl.returnType;
                sformals                  = class_formal :: fdecl.formals;
                sbody                              = fbody;
                func_type                 = Sast.User;
                overrides       = fdecl.overrides;
                source                         = cname;
        }

let convert_cdecl_to_sast sfdecls (cdecl:Ast.class_decl) =
        {
                scname = cdecl.cname;
                sfields = cdecl.cbody.fields;
                sfuncs = sfdecls;
        }

(*
 * Given a list of func_decls for the base class and a single func_decl
 * for the child class, replaces func_decls for the base class if any of them
 * have the same method signature
 *)
let replace_fdecl_in_base_methods base_cname base_methods child_fdecl =
        let replace base_fdecl accum =
                let get_root_cname = function
                        None -> Some(base_cname)
                        | Some(x) -> Some(x)
                in
                let modify_child_fdecl =
                        {
                                scope = child_fdecl.scope;
                                fname = child_fdecl.fname;
                                returnType = child_fdecl.returnType;
                                formals = child_fdecl.formals;
                                body = child_fdecl.body;
                                overrides = true;
                                root_cname = get_root_cname base_fdecl.root_cname;
                        }
                in
                if (get_name_without_class base_fdecl) = (get_name_without_class
↪  child_fdecl)
                        then modify_child_fdecl::accum
                        else base_fdecl::accum
        in
        List.fold_right replace base_methods []

let merge_methods base_cname base_methods child_methods =
        let check_overrides child_fdecl accum =
                let base_checked_for_overrides =
                        replace_fdecl_in_base_methods base_cname (fst accum) child_fdecl
                in
                if (fst accum) = base_checked_for_overrides
                        then ((fst accum), child_fdecl::(snd accum))
                        else (base_checked_for_overrides, (snd accum))
        in
        let updated_base_and_child_fdecls =
                List.fold_right check_overrides child_methods (base_methods, [])
        in
        (fst updated_base_and_child_fdecls) @ (snd updated_base_and_child_fdecls)

let merge_cdecls base_cdecl child_cdecl =
(* return a cdecl in which cdecl.cbody.fields contains the fields of
the extended class, concatenated by the fields of the child class *)
        let child_cbody =
                {
                        fields = base_cdecl.cbody.fields @ child_cdecl.cbody.fields;
                         constructors = child_cdecl.cbody.constructors;
                         methods = merge_methods base_cdecl.cname
↪  base_cdecl.cbody.methods child_cdecl.cbody.methods
                }
                in
                {
                        cname = child_cdecl.cname;
                        extends = child_cdecl.extends;
                        cbody = child_cbody
                }

(* returns a list of cdecls that contains inherited fields *)
let inherit_fields_cdecls cdecls inheritance_forest =
        (* iterate through cdecls to make a map for lookup *)
        let cdecl_lookup = List.fold_left (fun a litem -> StringMap.add litem.cname litem
↪  a) StringMap.empty cdecls in
        let add_key key pred maps =
                let elem1 = StringSet.add key (fst maps) in
                let accum acc child = StringSet.add child acc in
                let elem2 = List.fold_left (accum) (snd maps) pred in
                (elem1, elem2)
        in
        let empty_s = StringSet.empty in
        let res = StringMap.fold add_key inheritance_forest (empty_s, empty_s) in
        let roots = StringSet.diff (fst res) (snd res) in
        let rec add_inherited_fields predec desc map_to_update =
                let merge_fields accum descendant =
                        let updated_predec_cdecl = StringMap.find predec accum in
                        let descendant_cdecl_to_update = StringMap.find descendant cdecl_lookup in
                        let merged = merge_cdecls updated_predec_cdecl descendant_cdecl_to_update in
                        let updated = (StringMap.add descendant merged accum) in
                        if (StringMap.mem descendant inheritance_forest) then
                                let descendants_of_descendant = StringMap.find descendant inheritance_forest in
                                add_inherited_fields descendant descendants_of_descendant updated
                        else updated
                in
                List.fold_left merge_fields map_to_update desc
        in
        (* map class name of every class_decl in `cdecls` to its inherited cdecl *)
        let inherited_cdecls =
                let traverse_tree tree_root accum =
                        let tree_root_descendant = StringMap.find tree_root inheritance_forest in
                        let accum_with_tree_root_mapping = StringMap.add tree_root (StringMap.find tree_root cdecl_lookup) accum in
                        add_inherited_fields tree_root tree_root_descendant accum_with_tree_root_mapping
                in
                StringSet.fold traverse_tree roots StringMap.empty
        in
        (* build a list of updated cdecls corresponding to the sequence of cdecls in `cdecls` *)
        let add_inherited_cdecl cdecl accum =
                let inherited_cdecl =
                        try StringMap.find cdecl.cname inherited_cdecls
                        with | Not_found -> cdecl
                in
                inherited_cdecl::accum
        in
        let result = List.fold_right add_inherited_cdecl cdecls [] in
        result

let convert_cdecls_to_sast class_maps reserved (cdecls:Ast.class_decl list) =
        let find_main = (fun f -> match f.sfname with FName n -> n = "main" | _ -> false) in
        let get_main func_list =
                let mains = (List.find_all find_main func_list) in
                if List.length mains < 1 then
                        raise Exceptions.MainNotDefined
                else if List.length mains > 1 then
                        raise Exceptions.MultipleMainsDefined
                else List.hd mains
        in
        let remove_main func_list =
                List.filter (fun f -> not (find_main f)) func_list
        in
        let find_default_constructor cdecl clist =
                let default_cname = cdecl.cname ^ "." ^ "constructor" in
                let find_default_c f =
                        match f.sfname with FName n -> n = default_cname | _ -> false
                in
                try let _ = List.find find_default_c clist in
                        clist
                with | Not_found ->
                        let default_c = default_sc cdecl.cname in
                        default_c :: clist
        in
        let handle_cdecl cdecl =
                let class_map = StringMap.find cdecl.cname class_maps in
                let sconstructor_list = List.fold_left (fun l c -> (convert_constructor_to_sfdecl class_maps reserved class_map cdecl.cname c) :: l) [] cdecl.cbody.constructors in
                let sconstructor_list = find_default_constructor cdecl sconstructor_list in
                let func_list = List.fold_left (fun l f -> (convert_fdecl_to_sfdecl class_maps reserved class_map cdecl.cname f) :: l) [] cdecl.cbody.methods in
                let sfunc_list = remove_main func_list in
                let scdecl = convert_cdecl_to_sast sfunc_list cdecl in
                (scdecl, func_list @ sconstructor_list)
        in
        let iter_cdecls t c =
                let scdecl = handle_cdecl c in
                (fst scdecl :: fst t, snd scdecl @ snd t)
        in
        let scdecl_list, func_list = List.fold_left iter_cdecls ([], []) cdecls in
        let main = get_main func_list in
        let funcs = remove_main func_list in
        (* let funcs = (add_default_constructors cdecls class_maps) @ funcs in *)
        {
                classes                 = scdecl_list;
                functions                 = funcs;
                main                         = main;
```

```
                reserved                  = reserved;
        }

let add_reserved_functions =
        let reserved_stub name return_type formals =
                {
                        sfname                          = FName(name);
                        sreturnType          = return_type;
                        sformals                  = formals;
                        sbody                          = [];
                        func_type                 = Sast.Reserved;
                        overrides                  = false;
                        source                          = "NA";
                }
        in
        let i32_t = Datatype(Int_t) in
        let void_t = Datatype(Void_t) in
        let str_t = Arraytype(Char_t, 1) in
        let mf t n = Formal(t, n) in (* Make formal *)
        let reserved = [
                reserved_stub "print"         (void_t)         ([Many(Any)]);
                reserved_stub "malloc"         (str_t)         ([mf i32_t "size"]);
                reserved_stub "cast"         (Any)                 ([mf Any "in"]);
                reserved_stub "sizeof"         (i32_t)         ([mf Any "in"]);
                reserved_stub "open"         (i32_t)         ([mf str_t "path"; mf i32_t
    ↪ "flags"]);
                reserved_stub "close"         (i32_t)         ([mf i32_t "fd"]);
                reserved_stub "read"         (i32_t)         ([mf i32_t "fd"; mf str_t
    ↪ "buf"; mf i32_t "nbyte"]);
                reserved_stub "write"         (i32_t)         ([mf i32_t "fd"; mf str_t
    ↪ "buf"; mf i32_t "nbyte"]);
                reserved_stub "lseek"         (i32_t)         ([mf i32_t "fd"; mf i32_t
    ↪ "offset"; mf i32_t "whence"]);
                reserved_stub "exit"         (void_t)         ([mf i32_t "status"]);
        reserved_stub "getchar" (i32_t)             ([]);
        reserved_stub "input"         (str_t)         ([]);
        ] in
        reserved

let build_inheritance_forest cdecls cmap =
        let handler a cdecl =
                match cdecl.extends with
                        Parent(s)         ->
                                let new_list = if (StringMap.mem s a) then
                                        cdecl.cname::(StringMap.find s a)
                                else
                                        [cdecl.cname]
                                in
                                Hashtbl.add predecessors s new_list;
                                (StringMap.add s new_list a)
                |         NoParent         -> a
        in
        let forest = List.fold_left handler StringMap.empty cdecls in

        let handler key value =
                if not (StringMap.mem key cmap) then
                        raise (Exceptions.UndefinedClass key)
        in
        ignore(StringMap.iter handler forest);
        forest

let merge_maps m1 m2 =
        StringMap.fold (fun k v a -> StringMap.add k v a) m1 m2

let update_class_maps map_type cmap_val cname cmap_to_update =
        let update m map_type =
                if map_type = "field_map" then
                        {
                                field_map = cmap_val;
                                func_map = m.func_map;
                                constructor_map = m.constructor_map;
                                reserved_map = m.reserved_map;
                                cdecl = m.cdecl;
                        }
                else m
        in
        let updated = StringMap.find cname cmap_to_update in
        let updated = update updated map_type in
        let updated = StringMap.add cname updated cmap_to_update in
        updated

let inherit_fields class_maps predecessors =
        (* Get basic inheritance map *)
        let add_key key pred map = StringMap.add key pred map in
        let cmaps_inherit = StringMap.fold add_key class_maps StringMap.empty in
        (* Perform accumulation of child classes *)
        let add_key key pred maps =
                let elem1 = StringSet.add key (fst maps) in
                let accum acc child = StringSet.add child acc in
                let elem2 = List.fold_left (accum) (snd maps) pred in
                (elem1, elem2)
        in
        let empty_s = StringSet.empty in
        let res = StringMap.fold add_key predecessors (empty_s, empty_s) in
        let roots = StringSet.diff (fst res) (snd res) in
        (*in let _ = print_set_members roots*)
        let rec add_inherited_fields predec desc cmap_to_update =
                let cmap_inherit accum descendant =
                        let predec_field_map = (StringMap.find predec accum).field_map in
                        let desc_field_map = (StringMap.find descendant accum).field_map
↪   in
                        let merged = merge_maps predec_field_map desc_field_map in
                        let updated = update_class_maps "field_map" merged descendant
↪   accum in
                        if (StringMap.mem descendant predecessors) then
                                let descendants_of_descendant = StringMap.find descendant
↪   predecessors in
                                add_inherited_fields descendant descendants_of_descendant
↪   updated
                        else updated
                in
                List.fold_left cmap_inherit cmap_to_update desc
                (* end of add_inherited_fields *)
        in
        let result = StringSet.fold (fun x a -> add_inherited_fields x (StringMap.find x
↪   predecessors) a) roots cmaps_inherit
        (*in let _ = print_map result*)
        in result

(* TODO Check that this actually works *)
let check_cyclical_inheritance cdecls predecessors =
        let handle_predecessor cdecl parent predecessor =
                if cdecl.cname = predecessor then
                        raise(Exceptions.CyclicalDependencyBetween(cdecl.cname, parent))
        in
        let handle_cdecl cdecl =
                if StringMap.mem cdecl.cname predecessors
                        then
                                let pred_list = StringMap.find cdecl.cname predecessors
↪   in
                                List.iter (handle_predecessor cdecl (List.hd pred_list))
↪   pred_list
                        else ()
        in
        List.iter handle_cdecl cdecls

let build_func_map_inherited_lookup cdecls_inherited =
        let build_func_map cdecl =
                let add_func m fdecl = StringMap.add (get_name cdecl.cname fdecl) fdecl m
↪   in
                List.fold_left add_func StringMap.empty cdecl.cbody.methods
        in
        let add_class_func_map m cdecl = StringMap.add cdecl.cname (build_func_map cdecl)
↪   m in
        List.fold_left add_class_func_map StringMap.empty cdecls_inherited

let add_inherited_methods cmaps cdecls func_maps_inherited =
        let find_cdecl cname =
                try List.find (fun cdecl -> cdecl.cname = cname) cdecls
                with | Not_found -> raise Not_found
        in
        let update_with_inherited_methods cname cmap =
                let fmap = StringMap.find cname func_maps_inherited in
                let cdecl = find_cdecl cname in
                {
                        field_map = cmap.field_map;
                        func_map = fmap;
                        constructor_map = cmap.constructor_map;
                        reserved_map = cmap.reserved_map;
                        cdecl = cdecl;
                }
        in
        let add_updated_cmap cname cmap accum = StringMap.add cname
↪ (update_with_inherited_methods cname cmap) accum in
        StringMap.fold add_updated_cmap cmaps StringMap.empty

let handle_inheritance cdecls class_maps =
        let predecessors = build_inheritance_forest cdecls class_maps in
        let cdecls_inherited = inherit_fields_cdecls cdecls predecessors in
        let func_maps_inherited = build_func_map_inherited_lookup cdecls_inherited in
        ignore(check_cyclical_inheritance cdecls predecessors);
        let cmaps_with_inherited_fields = inherit_fields class_maps predecessors in
        let cmaps_inherited = add_inherited_methods cmaps_with_inherited_fields
↪ cdecls_inherited func_maps_inherited in
        cmaps_inherited, cdecls_inherited

let generate_struct_indexes cdecls =
        let cdecl_handler index cdecl =
                Hashtbl.add struct_indexes cdecl.cname index
        in
        List.iteri cdecl_handler cdecls

(* Main method for analyzer *)
let analyze filename program = match program with
        Program(includes, classes) ->
        (* Include code from external files *)
        let cdecls = process_includes filename includes classes in
        ignore(generate_struct_indexes cdecls);

        (* Add built-in functions *)
        let reserved = add_reserved_functions in
        (* Generate the class_maps for look up in checking functions *)
        let class_maps = build_class_maps reserved cdecls in
        let class_maps, cdecls = handle_inheritance cdecls class_maps in
        let sast = convert_cdecls_to_sast class_maps reserved cdecls in
        sast
```

ast.ml

```
type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater | Geq | And | Not |
↪  Or | Mod
type scope = Private | Public
type primitive = Int_t | Float_t | Void_t | Bool_t | Char_t | Objecttype of string |
↪  ConstructorType | Null_t
type datatype = Arraytype of primitive * int | Datatype of primitive | Any

type extends = NoParent | Parent of string
type fname = Constructor | FName of string
type formal = Formal of datatype * string | Many of datatype

type expr =
                Int_Lit of int
        |         Boolean_Lit of bool
        |         Float_Lit of float
        |         String_Lit of string
        |         Char_Lit of char
        |         This
        |         Id of string
        |         Binop of expr * op * expr
        |         Assign of expr * expr
        |         Noexpr
        |         ArrayCreate of datatype * expr list
        |         ArrayAccess of expr * expr list
        |         ObjAccess of expr * expr
        |         Call of string * expr list
        |   ObjectCreate of string * expr list
        |         ArrayPrimitive of expr list
        |          Unop of op * expr
        |         Null
        |         Delete of expr

type stmt =
                Block of stmt list
        |         Expr of expr
        |         Return of expr
        |         If of expr * stmt * stmt
        |         For of expr * expr * expr * stmt
        |         While of expr * stmt
        |          Break
        |   Continue
        |   Local of datatype * string * expr

type field = Field of scope * datatype * string
type include_stmt = Include of string

type func_decl = {
        scope : scope;
        fname : fname;
        returnType : datatype;
        formals : formal list;
        body : stmt list;
        overrides : bool;
        root_cname : string option;
}

type cbody = {
        fields : field list;
        constructors : func_decl list;
        methods : func_decl list;
}

type class_decl = {
        cname : string;
        extends : extends;
        cbody: cbody;
}

type program = Program of include_stmt list * class_decl list
```

bindings.c

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

#define INIT_SIZE 100

struct s {
        int x;
        int y;
};

char* input() {
        int initial_size = INIT_SIZE;
        char* str = malloc(initial_size);
        int index = 0;
        char tmp = '0';
        while((tmp = getchar() )!= '\n') {
                if(index >= initial_size - 1) {
                        str = realloc(str, initial_size *= 2);
                }
                str[index++] = tmp;
        }
        str[index] = '\0';
        return str;
}

void rec_init(long* arr, int curr_offset, int* static_offsets, int* indexes, int* dims,
 ↪  int dimc, int dim_curr) {

        //Assign length
        arr[curr_offset] = dims[dim_curr];

        if(dim_curr + 1 >= dimc)
                return;

        //Determine the static offset and the dynamic offset
        int static_offset = static_offsets[dim_curr];
        int dynamic_offset = 0;
        for(int i = 0; i < dim_curr; i++) {
                int tmp = indexes[i];
                for(int j = i + 1; j <= dim_curr; j++) {
                        tmp *= dims[j];
                }
                dynamic_offset += tmp;
        }

        //Iterate through position and iniitalize subarrays
        //Set local indexes to pointers to the subarrays
	for(int i = 0; i < dims[dim_curr]; i++) {
		int offset = (static_offset + (dynamic_offset + i) * (dims[dim_curr + 1]
 ↪  + 1));

		long* sub = arr + offset;
		arr[curr_offset + 1 + i] = (long) sub;

		indexes[dim_curr] = i;
		rec_init(arr, offset, static_offsets, indexes, dims, dimc, dim_curr + 1);
	}
}

long* init_arr(int* dims, int dimc) {

	int static_offsets[dimc];
	int total = 0;
	for(int i = 0; i < dimc; i++) {
		static_offsets[i] = 1;
		for(int j = 0; j < i; j++) {
			static_offsets[i] *= dims[j];
		}
		static_offsets[i] *= dims[i] + 1;
		static_offsets[i] += total;
		total = static_offsets[i];
	}

	int indexes[dimc];
	for(int i = 0; i < dimc; i++) {
		indexes[i] = 0;
	}

	//Get total length of array
	int length = 0;
	for(int i = 0; i < dimc; i++) {
		int tmp = 1;
		for(int j = i - 1; j >= 0; j--) {
			tmp *= dims[j];
		}
		tmp *= dims[i] + 1;
		length += tmp;
	}

	//Malloc array
	long* arr = malloc(length);

	//Set all values to 0 initially
	for(int i = 0 ; i < length; i++) {
		arr[i] = 0;
	}
```

```

        //Initialize the entire array
        rec_init(arr, 0, static_offsets, indexes, dims, dimc, 0);

        return arr;
}

// int main() {

//         //Array creation
//         int dims[5] = {2, 3, 4, 5, 6};
//         int dimc = 5;

//         long* arr = init_arr(dims, dimc);

//         //Get total length of array
//         int length = 0;
//         for(int i = 0; i < dimc; i++) {
//                 int tmp = 1;
//                 for(int j = i - 1; j >= 0; j--) {
//                         tmp *= dims[j];
//                 }
//                 tmp *= dims[i] + 1;
//                 length += tmp;
//         }

//         for(int i = 0; i < length; i++) {
//                 printf("val: %ld | addr: %ld\n", arr[i], (long) arr + i);
//         }
//         printf("\n");
// }
```

codegen.ml

```
(* ===------------------------------------------------------------------------===
 * Code Generation
 *===------------------------------------------------------------------------===*)

open Llvm
open Ast
open Sast
open Analyzer
open Exceptions
open Batteries
open Hashtbl
open Conf

open Llvm.MemoryBuffer
open Llvm_bitreader

let context = global_context ()
let the_module = create_module context "Dice Codegen"
let builder = builder context
let named_values:(string, llvalue) Hashtbl.t = Hashtbl.create 50
let named_params:(string, llvalue) Hashtbl.t = Hashtbl.create 50
let struct_types:(string, lltype) Hashtbl.t = Hashtbl.create 10
let struct_field_indexes:(string, int) Hashtbl.t = Hashtbl.create 50

let i32_t = i32_type context;;
let i8_t = i8_type context;;
let f_t = double_type context;;
let i1_t = i1_type context;;
let str_t = pointer_type i8_t;;
let i64_t = i64_type context;;
let void_t = void_type context;;

let str_type = Arraytype(Char_t, 1)

let (br_block) = ref (block_of_value (const_int i32_t 0))
let (cont_block) = ref (block_of_value (const_int i32_t 0))
let is_loop = ref false

let debug = fun s ->
        print_endline ("`````````````````````````````````````"^s);
        dump_module the_module;
        print_endline ("`````````````````````````````````````"^s);
        ()

let rec get_ptr_type datatype = match datatype with
                Arraytype(t, 0) -> get_type (Datatype(t))
        |        Arraytype(t, 1) -> pointer_type (get_type (Datatype(t)))
```

```
        |       Arraytype(t, i) -> pointer_type (get_ptr_type (Arraytype(t, (i-1))))
        |        _ -> raise(Exceptions.InvalidStructType "Array Pointer Type")

and find_struct name =
        try Hashtbl.find struct_types name
        with | Not_found -> raise(Exceptions.InvalidStructType name)

and get_type (datatype:Ast.datatype) = match datatype with
                Datatype(Int_t) -> i32_t
        |         Datatype(Float_t) -> f_t
        |         Datatype(Bool_t) -> i1_t
        |         Datatype(Char_t) -> i8_t
        |         Datatype(Void_t) -> void_t
        |         Datatype(Null_t) -> i32_t
        |         Datatype(Objecttype(name)) -> pointer_type(find_struct name)
        |         Arraytype(t, i) -> get_ptr_type (Arraytype(t, (i)))
        |         d -> raise(Exceptions.InvalidStructType (Utils.string_of_datatype d))

(* cast will return an llvalue of the desired type *)
(* The commented out casts are unsupported actions in Dice *)
let cast lhs rhs lhsType rhsType llbuilder =
        match (lhsType, rhsType) with
                (* int to,__ ) ( using const_sitofp for signed ints *)
                (Datatype(Int_t), Datatype(Int_t))                                ->
↪ (lhs, rhs), Datatype(Int_t)
        |         (Datatype(Int_t), Datatype(Char_t))                               ->
↪ (build_uitofp lhs i8_t "tmp" llbuilder, rhs), Datatype(Char_t)
        (* |            (Datatype(Int_t),
↪ Datatype(Bool_t))                                -> (lhs, const_zext rhs i32_t) *)
        |    (Datatype(Int_t), Datatype(Float_t))                        -> (build_sitofp
↪ lhs f_t "tmp" llbuilder, rhs), Datatype(Float_t)

                (* char to,__)  ( using uitofp since char isn't signed *)
        |    (Datatype(Char_t), Datatype(Int_t))                         -> (lhs,
↪ build_uitofp rhs i8_t "tmp" llbuilder), Datatype(Char_t)
        |    (Datatype(Char_t), Datatype(Char_t))                        -> (lhs, rhs),
↪ Datatype(Char_t)
        (* |          (Datatype(Char_t), Datatype(Bool_t))                         -> (lhs,
↪ const_zext rhs i8_t) *)
        (* |          (Datatype(Char_t), Datatype(Float_t))                         ->
↪ (const_uitofp lhs f_t, rhs) *)

                (* bool to,__)  ( zext fills the empty bits with zeros, zero extension *)
        (* |            (Datatype(Bool_t), Datatype(Int_t))                          ->
↪ (const_zext lhs i32_t, rhs) *)
        (* |          (Datatype(Bool_t), Datatype(Char_t))                          ->
↪ (const_zext lhs i8_t, rhs) *)
        |             (Datatype(Bool_t), Datatype(Bool_t))                         -> (lhs,
↪ rhs), Datatype(Bool_t)
```

```
        (* |            (Datatype(Bool_t), Datatype(Float_t))                    ->
↪  (const_uitofp lhs f_t, rhs) *)

                (* float to,__) ( using fptosi for signed ints *)
        |   (Datatype(Float_t), Datatype(Int_t))                       -> (lhs,
↪  build_sitofp rhs f_t "tmp" llbuilder), Datatype(Float_t)
        (* |          (Datatype(Float_t), Datatype(Char_t))                    ->
↪  (lhs, const_uitofp rhs f_t) *)
        (* |            (Datatype(Float_t), Datatype(Bool_t))                    ->
↪  (lhs, const_uitofp rhs f_t) *)
        |   (Datatype(Float_t), Datatype(Float_t))                       -> (lhs, rhs),
↪  Datatype(Float_t)

        | Datatype(Objecttype(d)), Datatype(Null_t)                      -> (lhs, rhs),
↪  lhsType
        | Datatype(Null_t), Datatype(Objecttype(d))               -> (rhs, lhs),
↪  rhsType
        | Datatype(Objecttype(d)), t                                            ->
↪  raise(Exceptions.CanOnlyCompareObjectsWithNull(d, (Utils.string_of_datatype t)))

        | Arraytype(d, s), Datatype(Null_t)                                   ->
↪  (lhs, rhs), lhsType
        | Datatype(Null_t), Arraytype(d, s)                               -> (rhs,
↪  lhs), rhsType
        | Arraytype(d, _), t
↪                                                                ->
↪  raise(Exceptions.CanOnlyCompareArraysWithNull(Utils.string_of_primitive d,
↪  (Utils.string_of_datatype t)))

        |         _
↪                                                                                      ->
↪  raise (Exceptions.CannotCastTypeException(Utils.string_of_datatype lhsType,
↪  Utils.string_of_datatype rhsType))

let rec handle_binop e1 op e2 d llbuilder =
        (* Get the types of e1 and e2 *)
        let type1 = Analyzer.get_type_from_sexpr e1 in
        let type2 = Analyzer.get_type_from_sexpr e2 in

        (* Generate llvalues from e1 and e2 *)

        let e1 = codegen_sexpr llbuilder e1 in
        let e2 = codegen_sexpr llbuilder e2 in

        let float_ops op e1 e2 =
        match op with
                Add                  -> build_fadd e1 e2 "flt_addtmp" llbuilder
        |         Sub                  -> build_fsub e1 e2 "flt_subtmp" llbuilder
        |         Mult                  -> build_fmul e1 e2 "flt_multmp" llbuilder
        |          Div                   -> build_fdiv e1 e2 "flt_divtmp" llbuilder
        |          Mod                   -> build_frem e1 e2 "flt_sremtmp" llbuilder
        |          Equal                   -> build_fcmp Fcmp.Oeq e1 e2 "flt_eqtmp"
↪ llbuilder
        |          Neq                   -> build_fcmp Fcmp.One e1 e2 "flt_neqtmp" llbuilder
        |          Less                   -> build_fcmp Fcmp.Ult e1 e2 "flt_lesstmp"
↪ llbuilder
        |          Leq                   -> build_fcmp Fcmp.Ole e1 e2 "flt_leqtmp" llbuilder
        |          Greater                 -> build_fcmp Fcmp.Ogt e1 e2 "flt_sgttmp"
↪ llbuilder
        |          Geq                   -> build_fcmp Fcmp.Oge e1 e2 "flt_sgetmp" llbuilder
        |          _                           -> raise Exceptions.FloatOpNotSupported

        in

        (* chars are considered ints, so they will use int_ops as well*)
        let int_ops op e1 e2 =
        match op with
                Add                    -> build_add e1 e2 "addtmp" llbuilder
        |          Sub                   -> build_sub e1 e2 "subtmp" llbuilder
        |          Mult                   -> build_mul e1 e2 "multmp" llbuilder
        |          Div                   -> build_sdiv e1 e2 "divtmp" llbuilder
        |          Mod                   -> build_srem e1 e2 "sremtmp" llbuilder
        |          Equal                   -> build_icmp Icmp.Eq e1 e2 "eqtmp" llbuilder
        |          Neq                   -> build_icmp Icmp.Ne e1 e2 "neqtmp" llbuilder
        |          Less                   -> build_icmp Icmp.Slt e1 e2 "lesstmp" llbuilder
        |          Leq                   -> build_icmp Icmp.Sle e1 e2 "leqtmp" llbuilder
        |          Greater                 -> build_icmp Icmp.Sgt e1 e2 "sgttmp" llbuilder
        |          Geq                   -> build_icmp Icmp.Sge e1 e2 "sgetmp" llbuilder
        |          And                   -> build_and e1 e2 "andtmp" llbuilder
        |          Or                            -> build_or  e1 e2 "ortmp" llbuilder
        |          _                           -> raise Exceptions.IntOpNotSupported
        in

        let obj_ops op e1 e2 =
                match op with
                        Equal -> build_is_null e1 "tmp" llbuilder
                |          Neq -> build_is_not_null e1 "tmp" llbuilder
                |          _          -> raise
↪ (Exceptions.ObjOpNotSupported(Utils.string_of_op op))
        in

        let (e1, e2), d = cast e1 e2 type1 type2 llbuilder in

        let type_handler d = match d with
                        Datatype(Float_t)    -> float_ops op e1 e2
                |          Datatype(Int_t)
                |     Datatype(Bool_t)
                |          Datatype(Char_t)          -> int_ops op e1 e2
                |         Datatype(Objecttype(_))
                |         Arraytype(_, _) -> obj_ops op e1 e2
                |   _ -> raise Exceptions.InvalidBinopEvaluationType
        in

        type_handler d

and handle_unop op e d llbuilder =
        (* Get the type of e *)
        let eType = Analyzer.get_type_from_sexpr e in
        (* Get llvalue  *)
        let e = codegen_sexpr llbuilder e in

        let unops op eType e = match (op, eType) with
                (Sub, Datatype(Int_t))                 ->  build_neg e "int_unoptmp"
↪ llbuilder
        |   (Sub, Datatype(Float_t))          ->          build_fneg e "flt_unoptmp"
↪ llbuilder
        |   (Not, Datatype(Bool_t))          ->  build_not e "bool_unoptmp" llbuilder
        |    _           -> raise Exceptions.UnopNotSupported        in

        let unop_type_handler d = match d with
                                Datatype(Float_t)
                        |        Datatype(Int_t)
                        |   Datatype(Bool_t)        -> unops op eType e
                        |   _ -> raise Exceptions.InvalidUnopEvaluationType
                in

                unop_type_handler d

and func_lookup fname =
        match (lookup_function fname the_module) with
                        None         -> raise (Exceptions.LLVMFunctionNotFound fname)
                |          Some f         -> f

and codegen_print el llbuilder =
        let printf = func_lookup "printf" in
        let tmp_count = ref 0 in
        let incr_tmp = fun x -> incr tmp_count in

        let map_expr_to_printfexpr expr =
                let exprType = Analyzer.get_type_from_sexpr expr in
                match exprType with
                Datatype(Bool_t) ->
                        incr_tmp ();
                        let tmp_var = "tmp" ^ (string_of_int !tmp_count) in
                        let trueStr = SString_Lit("true") in
                        let falseStr = SString_Lit("false") in
                        let id = SId(tmp_var, str_type) in
```

```
            ignore(codegen_stmt llbuilder (SLocal(str_type, tmp_var,
↪ SNoexpr)));
            ignore(codegen_stmt llbuilder (SIf(expr,
                                            SExpr(SAssign(i
↪ trueStr, str_type), str_type),
                                            SExpr(SAssign(i
↪ falseStr, str_type), str_type)
                                        )));
            codegen_sexpr llbuilder id
        | _ -> codegen_sexpr llbuilder expr
    in

    let params = List.map map_expr_to_printfexpr el in
    let param_types = List.map (Analyzer.get_type_from_sexpr) el in

    let map_param_to_string = function
            Arraytype(Char_t, 1)         -> "%s"
        |        Datatype(Int_t)            -> "%d"
        |        Datatype(Float_t)           -> "%f"
        |        Datatype(Bool_t)           -> "%s"
        |        Datatype(Char_t)           -> "%c"
        |        _                                  -> raise
↪ (Exceptions.InvalidTypePassedToPrintf)
    in
    let const_str = List.fold_left (fun s t -> s ^ map_param_to_string t) ""
↪ param_types in
    let s = codegen_sexpr llbuilder (SString_Lit(const_str)) in
    let zero = const_int i32_t 0 in
    let s = build_in_bounds_gep s [| zero |] "tmp" llbuilder in
    build_call printf (Array.of_list (s :: params)) "tmp" llbuilder

and codegen_func_call fname el d llbuilder =
    let f = func_lookup fname in
    let params = List.map (codegen_sexpr llbuilder) el in
    match d with
        Datatype(Void_t) -> build_call f (Array.of_list params) "" llbuilder
    |        _ ->                  build_call f (Array.of_list
↪ params) "tmp" llbuilder

and codegen_sizeof el llbuilder =
    let type_of = Analyzer.get_type_from_sexpr (List.hd el) in
    let type_of = get_type type_of in
    let size_of = size_of type_of in
    build_bitcast size_of i32_t "tmp" llbuilder

and codegen_cast el d llbuilder =
    let cast_malloc_to_objtype lhs currType newType llbuilder = match newType with
        Datatype(Objecttype(x)) ->
            let obj_type = get_type (Datatype(Objecttype(x))) in
            build_pointercast lhs obj_type "tmp" llbuilder
        |        _ as t -> raise
↪ (Exceptions.CannotCastTypeException(Utils.string_of_datatype currType,
↪ Utils.string_of_datatype t))
    in
    let expr = List.hd el in
    let t = Analyzer.get_type_from_sexpr expr in
    let lhs = match expr with
    |        Sast.SId(id, d) -> codegen_id false false id d llbuilder
    |         SObjAccess(e1, e2, d) -> codegen_obj_access false e1 e2 d llbuilder
    |        SArrayAccess(se, sel, d) -> codegen_array_access true se sel d
↪ llbuilder
    | _ -> codegen_sexpr llbuilder expr
    in
    cast_malloc_to_objtype lhs t d llbuilder

and codegen_call llbuilder d el = function
        "print"         -> codegen_print el llbuilder
    |        "sizeof"        -> codegen_sizeof el llbuilder
    |        "cast"                -> codegen_cast el d llbuilder
    |        "malloc"         -> codegen_func_call "malloc" el d llbuilder
    |        "open"                -> codegen_func_call "open" el d llbuilder
    |        "write"               -> codegen_func_call "write" el d llbuilder
    |        "close"               -> codegen_func_call "close" el d llbuilder
    |        "read"                -> codegen_func_call "read" el d llbuilder
    |        "lseek"         -> codegen_func_call "lseek" el d llbuilder
    |        "exit"                -> codegen_func_call "exit" el d llbuilder
    |        "input"         -> codegen_func_call "input" el d llbuilder
  |   "getchar"    -> codegen_func_call "getchar" el d llbuilder
    |        _ as fname         -> raise
↪ (Exceptions.UnableToCallFunctionWithoutParent fname)(* codegen_func_call fname el
↪ llbuilder *)

and codegen_id isDeref checkParam id d llbuilder =
    if isDeref then
        try Hashtbl.find named_params id
        with | Not_found ->
        try let _val = Hashtbl.find named_values id in
            build_load _val id llbuilder
        with | Not_found -> raise (Exceptions.UnknownVariable id)
    else
        try Hashtbl.find named_values id
        with | Not_found ->
            try
                let _val = Hashtbl.find named_params id in
                if checkParam then raise (Exceptions.CannotAssignParam
↪ id)
                else _val
        with | Not_found -> raise (Exceptions.UnknownVariable id)
```

```

and codegen_assign lhs rhs d llbuilder =
        let rhsType = Analyzer.get_type_from_sexpr rhs in
        (* Special case '=' because we don't want to emit the LHS as an
        * expression. *)
        let lhs, isObjAccess = match lhs with
        |         Sast.SId(id, d) -> codegen_id false false id d llbuilder, false
        |          SObjAccess(e1, e2, d) -> codegen_obj_access false e1 e2 d llbuilder, true
        |         SArrayAccess(se, sel, d) -> codegen_array_access true se sel d llbuilder, true
        | _ -> raise Exceptions.AssignLHSMustBeAssignable
        in
        (* Codegen the rhs. *)
        let rhs = match rhs with
        |         Sast.SId(id, d) -> codegen_id false false id d llbuilder
        |          SObjAccess(e1, e2, d) -> codegen_obj_access true e1 e2 d llbuilder
        | _ -> codegen_sexpr llbuilder rhs
        in
        let rhs = match d with
                        Datatype(Objecttype(_))        ->
                                if isObjAccess then rhs
                                else build_load rhs "tmp" llbuilder
                |         Datatype(Null_t) -> const_null (get_type d)
                | _ -> rhs
        in
        let rhs = match d, rhsType with
                Datatype(Char_t), Datatype(Int_t) -> build_uitofp rhs i8_t "tmp" llbuilder
        |         Datatype(Int_t), Datatype(Char_t) -> build_uitofp rhs i32_t "tmp" llbuilder
        |         _ -> rhs
        in
        (* Lookup the name. *)
        ignore(build_store rhs lhs llbuilder);
        rhs

and deref ptr t llbuilder =
        build_gep ptr (Array.of_list [ptr]) "tmp" llbuilder

and codegen_obj_access isAssign lhs rhs d llbuilder =
        let codegen_func_call param_ty fptr parent_expr el d llbuilder =
                let match_sexpr se = match se with
                        SId(id, d) -> let isDeref = match d with
                                Datatype(Objecttype(_)) -> false
                        |         _ -> true
                        in codegen_id isDeref false id d llbuilder
                |         se -> codegen_sexpr llbuilder se
                in
```

```
      let parent_expr = build_pointercast parent_expr param_ty "tmp" llbuilder in
      let params = List.map match_sexpr el in
      match d with
          Datatype(Void_t) -> build_call fptr (Array.of_list (parent_expr :: params)) "" llbuilder
      |    _ -> build_call fptr (Array.of_list (parent_expr :: params)) "tmp" llbuilder
  in
  let check_lhs = function
      SId(s, d)                  -> codegen_id false false s d llbuilder
  |    SArrayAccess(e, el, d)    -> codegen_array_access false e el d llbuilder
  |    se    -> raise (Exceptions.LHSofRootAccessMustBeIDorFunc (Utils.string_of_sexpr se))
  in
  (* Needs to be changed *)
  let rec check_rhs isLHS parent_expr parent_type =
      let parent_str = Utils.string_of_object parent_type in
      function
          (* Check fields in parent *)
          SId(field, d) ->
              let search_term = (parent_str ^ "." ^ field) in
              let field_index = Hashtbl.find struct_field_indexes search_term in
              let _val = build_struct_gep parent_expr field_index field llbuilder in
              let _val = match d with
                  Datatype(Objecttype(_)) ->
                  if not isAssign then _val
                  else build_load _val field llbuilder
              | _ ->
              if not isAssign then
                  _val
              else
                  build_load _val field llbuilder
              in
              _val

      |    SArrayAccess(e, el, d) ->

          let ce = check_rhs false parent_expr parent_type e in
          let index = codegen_sexpr llbuilder (List.hd el) in
          let index = match d with
              Datatype(Char_t) -> index
          |    _ -> build_add index (const_int i32_t 1) "tmp" llbuilder
          in
```

```
                let _val = build_gep ce [| index |] "tmp" llbuilder in
                if isLHS && isAssign
                        then _val
                        else build_load _val "tmp" llbuilder

                    (* Check functions in parent *)
            |         SCall(fname, el, d, index)         ->
                            let index = const_int i32_t index in
                            let c_index = build_struct_gep parent_expr 0 "cindex" llbuilder in
                            let c_index = build_load c_index "cindex" llbuilder in
                            let lookup = func_lookup "lookup" in
                            let fptr = build_call lookup [| c_index; index |] "fptr" llbuilder in
                            let fptr2 = func_lookup fname in
                            let f_ty = type_of fptr2 in
                            let param1 = param fptr2 0 in
                            let param_ty = type_of param1 in
                            let fptr = build_pointercast fptr f_ty fname llbuilder in
                            let ret = codegen_func_call param_ty fptr parent_expr el d llbuilder in
                            let ret = ret
                                    (* if not isLHS && not isAssign then
                                            build_load ret "tmp" llbuilder
                                    else
                                            ret *)
                            in
                            ret
                    (* Set parent, check if base is field *)
            |         SObjAccess(e1, e2, d)         ->
                            let e1_type = Analyzer.get_type_from_sexpr e1 in
                            let e1 = check_rhs true parent_expr parent_type e1 in
                            let e2 = check_rhs true e1 e1_type e2 in
                            e2
            |         _ as e -> raise (Exceptions.InvalidAccessLHS (Utils.string_of_sexpr e))
    in
    let lhs_type = Analyzer.get_type_from_sexpr lhs in
    match lhs_type with
            Arraytype(_, _) ->
                    let lhs = codegen_sexpr llbuilder lhs in
                    let _ = match rhs with
                            SId("length", _) -> "length"
                    |         _ -> raise(Exceptions.CanOnlyAccessLengthOfArray)
                    in
                    let _val = build_gep lhs [| (const_int i32_t 0) |] "tmp" llbuilder in
                    build_load _val "tmp" llbuilder
    |         _ ->
```

```
                let lhs = check_lhs lhs in
                let rhs = check_rhs true lhs lhs_type rhs in
                rhs

and codegen_obj_create fname el d llbuilder =
        let f = func_lookup fname in
        let params = List.map (codegen_sexpr llbuilder) el in
        let obj = build_call f (Array.of_list params) "tmp" llbuilder in
        obj

and codegen_string_lit s llbuilder =
        if s = "true" then build_global_stringptr "true" "tmp" llbuilder
        else if s = "false" then build_global_stringptr "false" "tmp" llbuilder
        else build_global_stringptr s "tmp" llbuilder

and codegen_array_access isAssign e el d llbuilder =
        let index = codegen_sexpr llbuilder (List.hd el) in
        let index = match d with
                Datatype(Char_t) -> index
        |         _ -> build_add index (const_int i32_t 1) "tmp" llbuilder
        in
    let arr = codegen_sexpr llbuilder e in
    let _val = build_gep arr [| index |] "tmp" llbuilder in
    if isAssign
            then _val
            else build_load _val "tmp" llbuilder

and initialise_array arr arr_len init_val start_pos llbuilder =
        let new_block label =
                let f = block_parent (insertion_block llbuilder) in
                append_block (global_context ()) label f
        in
  let bbcurr = insertion_block llbuilder in
  let bbcond = new_block "array.cond" in
  let bbbody = new_block "array.init" in
  let bbdone = new_block "array.done" in
  ignore (build_br bbcond llbuilder);
  position_at_end bbcond llbuilder;

  (* Counter into the length of the array *)
  let counter = build_phi [const_int i32_t start_pos, bbcurr] "counter" llbuilder in
  add_incoming ((build_add counter (const_int i32_t 1) "tmp" llbuilder), bbbody) counter;
  let cmp = build_icmp Icmp.Slt counter arr_len "tmp" llbuilder in
  ignore (build_cond_br cmp bbbody bbdone llbuilder);
  position_at_end bbbody llbuilder;

  (* Assign array position to init_val *)
  let arr_ptr = build_gep arr [| counter |] "tmp" llbuilder in
  ignore (build_store init_val arr_ptr llbuilder);
  ignore (build_br bbcond llbuilder);
  position_at_end bbdone llbuilder

and codegen_array_create llbuilder t expr_type el =
        if(List.length el > 1) then raise(Exceptions.ArrayLargerThan1Unsupported)
        else
        match expr_type with
                Arraytype(Char_t, 1) ->
                let e = List.hd el in
                let size = (codegen_sexpr llbuilder e) in
                let t = get_type t in
                let arr = build_array_malloc t size "tmp" llbuilder in
                let arr = build_pointercast arr (pointer_type t) "tmp" llbuilder in
                (* initialise_array arr size (const_int i32_t 0) 0 llbuilder; *)
                arr
        |         _ ->
                let e = List.hd el in
                let t = get_type t in

                (* This will not work for arrays of objects *)
                let size = (codegen_sexpr llbuilder e) in
                let size_t = build_intcast (size_of t) i32_t "tmp" llbuilder in
                let size = build_mul size_t size "tmp" llbuilder in
                let size_real = build_add size (const_int i32_t 1) "arr_size" llbuilder
↪   in

            let arr = build_array_malloc t size_real "tmp" llbuilder in
                let arr = build_pointercast arr (pointer_type t) "tmp" llbuilder in

                let arr_len_ptr = build_pointercast arr (pointer_type i32_t) "tmp"
↪   llbuilder in

                (* Store length at this position *)
                ignore(build_store size_real arr_len_ptr llbuilder);
                initialise_array arr_len_ptr size_real (const_int i32_t 0) 0 llbuilder;
                arr

and codegen_array_prim d el llbuilder =
    let t = d in
    let size = (const_int i32_t ((List.length el))) in
    let size_real = (const_int i32_t ((List.length el) + 1)) in
        let t = get_type t in
    let arr = build_array_malloc t size_real "tmp" llbuilder in
        let arr = build_pointercast arr t "tmp" llbuilder in
        let size_casted = build_bitcast size t "tmp" llbuilder in
        ignore(if d = Arraytype(Char_t, 1) then ignore(build_store size_casted arr
↪   llbuilder);); (* Store length at this position *)
        (* initialise_array arr size_real (const_int i32_t 0) 1 llbuilder; *)
    let llvalues = List.map (codegen_sexpr llbuilder) el in
    List.iteri (fun i llval ->
                            let arr_ptr = build_gep arr [| (const_int i32_t (i+1)) |] "tmp" llbuilder in
                            ignore(build_store llval arr_ptr llbuilder);  ) llvalues;
    arr

and codegen_delete e llbuilder =
        let ce = match e with
                SId(id, d) -> codegen_id false false id d llbuilder
        |         _ -> codegen_sexpr llbuilder e
        in
        build_free ce llbuilder

and codegen_sexpr llbuilder = function
                SInt_Lit(i)                             -> const_int i32_t i
        |   SBoolean_Lit(b)                         -> if b then const_int i1_t 1 else const_int i1_t 0
        |   SFloat_Lit(f)                           -> const_float f_t f
        |   SString_Lit(s)                          -> codegen_string_lit s llbuilder
        |   SChar_Lit(c)                            -> const_int i8_t (Char.code c)
        |   SId(id, d)                          -> codegen_id true false id d llbuilder
        |   SBinop(e1, op, e2, d)               -> handle_binop e1 op e2 d llbuilder
        |   SAssign(e1, e2, d)                  -> codegen_assign e1 e2 d llbuilder
        |   SNoexpr                           -> build_add (const_int i32_t 0) (const_int i32_t 0) "nop" llbuilder
        |   SArrayCreate(t, el, d)              -> codegen_array_create llbuilder t d el
        |   SArrayAccess(e, el, d)              -> codegen_array_access false e el d llbuilder
        |   SObjAccess(e1, e2, d)               -> codegen_obj_access true e1 e2 d llbuilder
        |   SCall(fname, el, d, _)                -> codegen_call llbuilder d el fname
        |   SObjectCreate(id, el, d)            -> codegen_obj_create id el d llbuilder
        |   SArrayPrimitive(el, d)              -> codegen_array_prim d el llbuilder
        |   SUnop(op, e, d)                     -> handle_unop op e d llbuilder
        |   SNull                                   -> const_null i32_t
        |         SDelete e                                             -> codegen_delete e llbuilder

and codegen_if_stmt exp then_ (else_:Sast.sstmt) llbuilder =
        let cond_val = codegen_sexpr llbuilder exp in

        (* Grab the first block so that we might later add the conditional branch
         * to it at the end of the function. *)
        let start_bb = insertion_block llbuilder in
        let the_function = block_parent start_bb in

        let then_bb = append_block context "then" the_function in

        (* Emit 'then' value. *)
        position_at_end then_bb llbuilder;
        let _(* then_val *) = codegen_stmt llbuilder then_ in

        (* Codegen of 'then' can change the current block, update then_bb for the
         * phi. We create a new name because one is used for the phi node, and the
         * other is used for the conditional branch. *)
        let new_then_bb = insertion_block llbuilder in

        (* Emit 'else' value. *)
        let else_bb = append_block context "else" the_function in
        position_at_end else_bb llbuilder;
        let _ (* else_val *) = codegen_stmt llbuilder else_ in

        (* Codegen of 'else' can change the current block, update else_bb for the
         * phi. *)
        let new_else_bb = insertion_block llbuilder in


        let merge_bb = append_block context "ifcont" the_function in
        position_at_end merge_bb llbuilder;
        (* let then_bb_val = value_of_block new_then_bb in *)
        let else_bb_val = value_of_block new_else_bb in
        (* let incoming = [(then_bb_val, new_then_bb); (else_bb_val, new_else_bb)] in *)
        (* let phi = build_phi incoming "iftmp" llbuilder in *)

        (* Return to the start block to add the conditional branch. *)
        position_at_end start_bb llbuilder;
        ignore (build_cond_br cond_val then_bb else_bb llbuilder);

        (* Set a unconditional branch at the end of the 'then' block and the
         * 'else' block to the 'merge' block. *)
        position_at_end new_then_bb llbuilder; ignore (build_br merge_bb llbuilder);
        position_at_end new_else_bb llbuilder; ignore (build_br merge_bb llbuilder);

        (* Finally, set the builder to the end of the merge block. *)
        position_at_end merge_bb llbuilder;

        else_bb_val (* phi *)

and codegen_for init_ cond_ inc_ body_ llbuilder =
        let old_val = !is_loop in
        is_loop := true;

        let the_function = block_parent (insertion_block llbuilder) in

        (* Emit the start code first, without 'variable' in scope. *)
        let _ = codegen_sexpr llbuilder init_ in

        (* Make the new basic block for the loop header, inserting after current
        * block. *)
        let loop_bb = append_block context "loop" the_function in
        (* Insert maintenance block *)
        let inc_bb = append_block context "inc" the_function in
        (* Insert condition block *)
        let cond_bb = append_block context "cond" the_function in
        (* Create the "after loop" block and insert it. *)
        let after_bb = append_block context "afterloop" the_function in

        let _ = if not old_val then
                cont_block := inc_bb;
                br_block := after_bb;
        in

        (* Insert an explicit fall through from the current block to the
        * loop_bb. *)
        ignore (build_br cond_bb llbuilder);

        (* Start insertion in loop_bb. *)
        position_at_end loop_bb llbuilder;

        (* Emit the body of the loop.  This, like any other expr, can change the
        * current BB.  Note that we ignore the value computed by the body, but
        * don't allow an error *)
        ignore (codegen_stmt llbuilder body_);

        let bb = insertion_block llbuilder in
        move_block_after bb inc_bb;
        move_block_after inc_bb cond_bb;
        move_block_after cond_bb after_bb;
        ignore(build_br inc_bb llbuilder);

        (* Start insertion in loop_bb. *)
        position_at_end inc_bb llbuilder;
        (* Emit the step value. *)
        let _ = codegen_sexpr llbuilder inc_ in
        ignore(build_br cond_bb llbuilder);

        position_at_end cond_bb llbuilder;

        let cond_val = codegen_sexpr llbuilder cond_ in
        ignore (build_cond_br cond_val loop_bb after_bb llbuilder);

        (* Any new code will be inserted in after_bb. *)
        position_at_end after_bb llbuilder;

        is_loop := old_val;

        (* for expr always returns 0.0. *)
        const_null f_t

and codegen_while cond_ body_ llbuilder =
        let null_sexpr = SInt_Lit(0) in
        codegen_for null_sexpr cond_ null_sexpr body_ llbuilder

and codegen_alloca datatype var_name expr llbuilder =
        let t = match datatype with
                        Datatype(Objecttype(name)) -> find_struct name
                |           _ -> get_type datatype
        in
        let alloca = build_alloca t var_name llbuilder in
        Hashtbl.add named_values var_name alloca;
        let lhs = SId(var_name, datatype) in
        match expr with
                SNoexpr -> alloca
        |           _ -> codegen_assign lhs expr datatype llbuilder

and codegen_ret d expr llbuilder =
        match expr with
                SId(name, d) ->
                        (match d with
                        | Datatype(Objecttype(_)) -> build_ret (codegen_id false false
↪   name d llbuilder) llbuilder
                        | _ -> build_ret (codegen_id true true name d llbuilder)
↪   llbuilder)
                | SObjAccess(e1, e2, d) -> build_ret (codegen_obj_access true e1 e2 d
↪   llbuilder) llbuilder
                | SNoexpr -> build_ret_void llbuilder
                | _ -> build_ret (codegen_sexpr llbuilder expr) llbuilder

and codegen_break llbuilder =
        let block = fun () -> !br_block in
        build_br (block ()) llbuilder

and codegen_continue llbuilder =
        let block = fun () -> !cont_block in
        build_br (block ()) llbuilder

and codegen_stmt llbuilder = function
                SBlock sl                               -> List.hd(List.map
↪   (codegen_stmt llbuilder) sl)
        |   SExpr(e, d)                         -> codegen_sexpr llbuilder e
        |   SReturn(e, d)                           -> codegen_ret d e llbuilder
        |   SIf (e, s1, s2)                      -> codegen_if_stmt e s1 s2 llbuilder
        |   SFor (e1, e2, e3, s)                 -> codegen_for e1 e2 e3 s llbuilder
        |   SWhile (e, s)                           -> codegen_while e s llbuilder
        |   SBreak                                  -> codegen_break llbuilder
        |   SContinue                               -> codegen_continue llbuilder
        |   SLocal(d, s, e)                         -> codegen_alloca d s e llbuilder

let codegen_funcstub sfdecl =
        let fname = (Utils.string_of_fname sfdecl.sfname) in
        let is_var_arg = ref false in
        let params = List.rev (List.fold_left (fun l -> (function Formal(t, _) ->
    ↪ get_type t :: l | _ -> is_var_arg := true; l )) [] sfdecl.sformals) in
        let fty = if !is_var_arg
                        then var_arg_function_type (get_type sfdecl.sreturnType)
    ↪ (Array.of_list params)
                        else function_type (get_type sfdecl.sreturnType) (Array.of_list
    ↪ params)
        in
        define_function fname fty the_module

let init_params f formals =
        let formals = Array.of_list (formals) in
        Array.iteri (fun i a ->
        let n = formals.(i) in
        let n = Utils.string_of_formal_name n in
        set_value_name n a;
        Hashtbl.add named_params n a;
    ) (params f)

let codegen_func sfdecl =
        Hashtbl.clear named_values;
        Hashtbl.clear named_params;
        let fname = (Utils.string_of_fname sfdecl.sfname) in
        let f = func_lookup fname in
        let llbuilder = builder_at_end context (entry_block f) in
        let _ = init_params f sfdecl.sformals in
        let _ = if sfdecl.overrides then
                let this_param = Hashtbl.find named_params "this" in
                let source = Datatype(Objecttype(sfdecl.source)) in
                let casted_param = build_pointercast this_param (get_type source)
    ↪ "casted" llbuilder in
                Hashtbl.replace named_params "this" casted_param;
        in
        let _ = codegen_stmt llbuilder (SBlock (sfdecl.sbody)) in
        if sfdecl.sreturnType = Datatype(Void_t)
                then ignore(build_ret_void llbuilder);
        ()

let codegen_vtbl scdecls =
        let rt = pointer_type i64_t in
        let void_pt = pointer_type i64_t in
        let void_ppt = pointer_type void_pt in

        let f = func_lookup "lookup" in
        let llbuilder = builder_at_end context (entry_block f) in

        let len = List.length scdecls in
        let total_len = ref 0 in
        let scdecl_llvm_arr = build_array_alloca void_ppt (const_int i32_t len) "tmp" llbuilder in

        let handle_scdecl scdecl =
                let index = Hashtbl.find Analyzer.struct_indexes scdecl.scname in
                let len = List.length scdecl.sfuncs in
                let sfdecl_llvm_arr = build_array_alloca void_pt (const_int i32_t len) "tmp" llbuilder in

                let handle_fdecl i sfdecl =
                        let fptr = func_lookup (Utils.string_of_fname sfdecl.sfname) in
                        let fptr = build_pointercast fptr void_pt "tmp" llbuilder in

                        let ep = build_gep sfdecl_llvm_arr [| (const_int i32_t i) |] "tmp" llbuilder in
                        ignore(build_store fptr ep llbuilder);
                in
                List.iteri handle_fdecl scdecl.sfuncs;
                total_len := !total_len + len;

                let ep = build_gep scdecl_llvm_arr [| (const_int i32_t index) |] "tmp" llbuilder in
                ignore(build_store sfdecl_llvm_arr ep llbuilder);
        in
        List.iter handle_scdecl scdecls;

        let c_index = param f 0 in
        let f_index = param f 1 in
        set_value_name "c_index" c_index;
        set_value_name "f_index" f_index;

        if !total_len == 0 then
                build_ret (const_null rt) llbuilder
        else
                let vtbl = build_gep scdecl_llvm_arr [| c_index |] "tmp" llbuilder in
                let vtbl = build_load vtbl "tmp" llbuilder in
                let fptr = build_gep vtbl [| f_index |] "tmp" llbuilder in
                let fptr = build_load fptr "tmp" llbuilder in

                build_ret fptr llbuilder

let codegen_library_functions () =
        (* C Std lib functions *)
        let printf_ty = var_arg_function_type i32_t [| pointer_type i8_t |] in
        let _ = declare_function "printf" printf_ty the_module in
        let malloc_ty = function_type (str_t) [| i32_t |] in
        let _ = declare_function "malloc" malloc_ty the_module in
    let open_ty = function_type i32_t [| (pointer_type i8_t); i32_t |] in
    let _ = declare_function "open" open_ty the_module in
    let close_ty = function_type i32_t [| i32_t |] in
    let _ = declare_function "close" close_ty the_module in
    let read_ty = function_type i32_t [| i32_t; pointer_type i8_t; i32_t |] in
    let _ = declare_function "read" read_ty the_module in
    let write_ty = function_type i32_t [| i32_t; pointer_type i8_t; i32_t |] in
    let _ = declare_function "write" write_ty the_module in
    let lseek_ty = function_type i32_t [| i32_t; i32_t; i32_t |] in
    let _ = declare_function "lseek" lseek_ty the_module in
    let exit_ty = function_type void_t [| i32_t |] in
    let _ = declare_function "exit" exit_ty the_module in
        let realloc_ty = function_type str_t [| str_t; i32_t |] in
        let _ = declare_function "realloc" realloc_ty the_module in
    let getchar_ty = function_type (i32_t) [| |] in
    let _ = declare_function "getchar" getchar_ty the_module in

        (* Dice defined functions *)
        let fty = function_type (pointer_type i64_t) [| i32_t; i32_t |] in
        let _ = define_function "lookup" fty the_module in
    let rec_init_ty = function_type void_t [| (pointer_type i64_t); i32_t; (pointer_type
↪   i32_t); (pointer_type i32_t); (pointer_type i32_t); i32_t; i32_t |] in
    let _ = declare_function "rec_init" rec_init_ty the_module in
    let init_arr_ty = function_type (pointer_type i64_t) [| (pointer_type i32_t); i32_t
↪   |] in
    let _ = declare_function "init_arr" init_arr_ty the_module in
    let input_ty = function_type str_t [||] in
    let _ = declare_function "input" input_ty the_module in
    ()

let codegen_struct_stub s =
        let struct_t = named_struct_type context s.scname in
        Hashtbl.add struct_types s.scname struct_t

let codegen_struct s =
                let struct_t = Hashtbl.find struct_types s.scname in
        let type_list = List.map (function Field(_, d, _) -> get_type d) s.sfields in
        let name_list = List.map (function Field(_, _, s) -> s) s.sfields in

        (* Add key field to all structs *)
        let type_list = i32_t :: type_list in
        let name_list = ".key" :: name_list in

        let type_array = (Array.of_list type_list) in
        List.iteri (fun i f ->
        let n = s.scname ^ "." ^ f in
        Hashtbl.add struct_field_indexes n i;
    ) name_list;
        struct_set_body struct_t type_array true

let init_args argv args argc llbuilder =
        let new_block label =
                let f = block_parent (insertion_block llbuilder) in
                append_block (global_context ()) label f
        in
        let bbcurr = insertion_block llbuilder in
        let bbcond = new_block "args.cond" in
        let bbbody = new_block "args.init" in
        let bbdone = new_block "args.done" in
        ignore (build_br bbcond llbuilder);
        position_at_end bbcond llbuilder;

        (* Counter into the length of the array *)
        let counter = build_phi [const_int i32_t 0, bbcurr] "counter" llbuilder in
        add_incoming ((build_add counter (const_int i32_t 1) "tmp" llbuilder), bbbody)
 ↪  counter;
        let cmp = build_icmp Icmp.Slt counter argc "tmp" llbuilder in
        ignore (build_cond_br cmp bbbody bbdone llbuilder);
        position_at_end bbbody llbuilder;

        (* Assign array position to init_val *)
        let arr_ptr = build_gep args [| counter |] "tmp" llbuilder in
        let argv_val = build_gep argv [| counter |] "tmp" llbuilder in
        let argv_val = build_load argv_val "tmp" llbuilder in
        ignore (build_store argv_val arr_ptr llbuilder);
        ignore (build_br bbcond llbuilder);
        position_at_end bbdone llbuilder

let construct_args argc argv llbuilder =
        let str_pt = pointer_type str_t in
        let size_real = build_add argc (const_int i32_t 1) "arr_size" llbuilder in

        let arr = build_array_malloc str_pt size_real "args" llbuilder in
        let arr = build_pointercast arr str_pt "args" llbuilder in
        let arr_len_ptr = build_pointercast arr (pointer_type i32_t) "argc_len" llbuilder
 ↪  in
        let arr_1 = build_gep arr [| const_int i32_t 1 |] "arr_1" llbuilder in

        (* Store length at this position *)
        ignore(build_store argc arr_len_ptr llbuilder);
        ignore(init_args argv arr_1 argc llbuilder);
        arr

let codegen_main main =
        Hashtbl.clear named_values;
        Hashtbl.clear named_params;
        let fty = function_type i32_t [| i32_t; pointer_type str_t |] in
        let f = define_function "main" fty the_module in
        let llbuilder = builder_at_end context (entry_block f) in

        let argc = param f 0 in
        let argv = param f 1 in
        set_value_name "argc" argc;
        set_value_name "argv" argv;
        let args = construct_args argc argv llbuilder in
        Hashtbl.add named_params "args" args;

        let _ = codegen_stmt llbuilder (SBlock (main.sbody)) in
        build_ret (const_int i32_t 0) llbuilder

let linker filename =
        let llctx = Llvm.global_context () in
        let llmem = Llvm.MemoryBuffer.of_file filename in
        let llm = Llvm_bitreader.parse_bitcode llctx llmem in
        ignore(Llvm_linker.link_modules the_module llm)

let codegen_sprogram sprogram =
        let _ = codegen_library_functions () in
        let _ = List.map (fun s -> codegen_struct_stub s) sprogram.classes in
        let _ = List.map (fun s -> codegen_struct s) sprogram.classes in
        let _ = List.map (fun f -> codegen_funcstub f) sprogram.functions in
        let _ = List.map (fun f -> codegen_func f) sprogram.functions in
        let _ = codegen_main sprogram.main in
        let _ = codegen_vtbl sprogram.classes in
        let _ = linker Conf.bindings_path in
        the_module

(* Need to handle assignment of two different types *)
(* Need to handle private/public access *)
```

conf.ml

```
let bindings_path = "_includes/bindings.bc"
let stdlib_path = "_includes/stdlib.dice"
```

dice.ml

```
open Llvm
open Llvm_analysis
open Analyzer
open Utils
open Ast
open Yojson
open Exceptions
open Filepath

type action = Tokens | TokenEndl | PrettyPrint | Ast | Sast | Compile | CompileToFile |
↪  Help

let get_action = function
                "-tendl"         -> TokenEndl
        |         "-t"                 -> Tokens
        |         "-p"                 -> PrettyPrint
        |         "-ast"                 -> Ast
        |         "-sast"         -> Sast
        |         "-h"                 -> Help
        |         "-c"                 -> Compile
        |         "-f"                 -> CompileToFile
        |          _ as s                 -> raise (Exceptions.InvalidCompilerArgument s)

let check_single_argument = function
                "-h"         -> Help, ""
        |         "-tendl"
        |         "-t"
        |         "-p"
        |         "-ast"
        |         "-sast"
        |         "-c"
        |         "-f"         -> raise (Exceptions.NoFileArgument)
        |          _ as s         -> CompileToFile, s

let dice_name filename =
        let basename = Filename.basename filename in
        let filename = Filename.chop_extension basename in
        filename ^ ".ll"

let help_string = (
          "Usage: dice [optional-option] <source file>\n" ^
                "optional-option:\n" ^
                "\t-h: Print help text\n" ^
                "\t-tendl: Prints tokens with newlines intact\n" ^
                "\t-t: Prints token stream\n" ^
                "\t-p: Pretty prints Ast as a program\n" ^
                "\t-ast: Prints abstract syntax tree as json\n" ^
        "\t-sast: Prints semantically checked syntax tree as json\n" ^
        "\t-c: Compiles source\n" ^
        "\t-f: Compiles source to file (<filename>.<ext> -> <filename>.ll)\n" ^
        "Option defaults to \"-f\"\n"
    )

let _ =
    ignore(Printexc.record_backtrace true);
    try
        let action, filename =
            if Array.length Sys.argv = 1 then
                Help, ""
             else if Array.length Sys.argv = 2 then
                 check_single_argument (Sys.argv.(1))
            else if Array.length Sys.argv = 3 then
                get_action Sys.argv.(1), Sys.argv.(2)
            else raise (Exceptions.InvalidNumberCompilerArguments
↪ (Array.length Sys.argv))
        in
        (* Added fun () -> <x> so that each is evaluated only when requested *)
        let filename          = Filepath.realpath filename in
        let file_in          = fun () -> open_in filename in
         let lexbuf                 = fun () ->        Lexing.from_channel
↪ (file_in ()) in
         let token_list          = fun () -> Processor.build_token_list (lexbuf
↪ ()) in
         let program          = fun () -> Processor.parser filename (token_list
↪ ()) in
         let sprogram          = fun () -> Analyzer.analyze filename (program ())
↪ in
         let llm                    = fun () -> Codegen.codegen_sprogram (sprogram
↪ ()) in
     (* let _ = Llvm_analysis.assert_valid_module llm in *)
     match action with
             Help                          -> print_string help_string
        |        Tokens                         -> print_string
↪ (Utils.token_list_to_string (token_list ()))
        |        TokenEndl                   -> print_string
↪ (Utils.token_list_to_string_endl (token_list ()))
        |        Ast                          -> print_string (pretty_to_string
↪ (Utils.print_tree (program ())))
        |        Sast                          -> print_string (pretty_to_string
↪ (Utils.map_sprogram_to_json (sprogram ())))
        |        PrettyPrint          -> print_string (Utils.string_of_program
↪ (program ()))
        |        Compile                     -> dump_module (llm ())
        |        CompileToFile          -> print_module (dice_name filename) (llm
↪ ())
    with
```

```
              Exceptions.IllegalCharacter(filename, c, ln) ->
                      print_string
                      (
                              "In \"" ^ filename ^ "\", Illegal Character, '" ^
                              Char.escaped c ^ "', line " ^ string_of_int ln ^ "\n"
                      )
      |         Exceptions.UnmatchedQuotation(ln)         -> print_endline("Unmatched
↪ Quotation, line " ^ string_of_int ln)
      |         Exceptions.IllegalToken(tok)                 -> print_endline("Illegal
↪ token " ^ tok)
      |         Exceptions.MissingEOF                                 ->
↪ print_endline("Missing EOF")
      |         Parsing.Parse_error ->
                      print_string
                      (
                              "File \"" ^ !Processor.filename ^ "\", " ^
                              "line " ^ string_of_int !Processor.line_number ^ ", " ^
                              "character " ^ string_of_int !Processor.char_num ^ ", " ^
                              "Syntax Error, token " ^ Utils.string_of_token
↪ !Processor.last_token ^ "\n"
                      )

      |          Exceptions.InvalidNumberCompilerArguments i -> print_endline ("Invalid
↪ argument passed " ^ (string_of_int i)); print_string help_string
      |         Exceptions.InvalidCompilerArgument s                 -> print_endline
↪ ("Invalid argument passed " ^ s); print_string help_string
      |         Exceptions.NoFileArgument                                         ->
↪ print_string ("Must include file argument\n" ^ help_string)

      |         Exceptions.IncorrectNumberOfArgumentsException
↪                         -> print_endline("Incorrect number of arguments passed to
↪ function")
      |         Exceptions.ConstructorNotFound(cname)
↪                                         -> print_endline("Constructor" ^ cname ^ "
↪ not found")
      |         Exceptions.DuplicateClassName(cname)
↪                                         -> print_endline("Class " ^ cname ^ " not
↪ found")
      |         Exceptions.DuplicateField
↪                                                                 ->
↪ print_endline("Duplicate field defined")
      |         Exceptions.DuplicateFunction(fname)
↪                                         -> print_endline("Duplicate function defined
↪ " ^ fname)
      |         Exceptions.DuplicateConstructor
↪                                                 -> print_endline("Duplicate
↪ constructor found")
```

```
    |         Exceptions.DuplicateLocal(lname)                                      -> print_endline("Duplicate local variable defined " ^ lname)
    |         Exceptions.UndefinedClass(cname)                                      -> print_endline("Undefined class " ^ cname)
    |         Exceptions.UnknownIdentifier(id)                                      -> print_endline("Unkown identifier " ^ id)
    |         Exceptions.InvalidBinopExpression(binop)                      -> print_endline("Invalid binary expression " ^ binop)
    |         Exceptions.InvalidIfStatementType                                     -> print_endline("Invalid type passed to if statement, must be bool")
    |         Exceptions.InvalidForStatementType                                    -> print_endline("Invalid type passed to for loop, must be bool")
    |         Exceptions.ReturnTypeMismatch(t1, t2)                                 -> print_endline("Incorrect return type " ^ t1 ^ " expected " ^ t2)
    |         Exceptions.MainNotDefined                                                       -> print_endline("Main not found in program")
    |
          Exceptions.MultipleMainsDefined                                                                   -> print_endline("Multiple mains defined, can only define 1")
    |         Exceptions.InvalidWhileStatementType                          -> print_endline("Invalid type passed to while loop, must be bool")
    |         Exceptions.LocalAssignTypeMismatch(t1, t2)                    -> print_endline("Invalid assignment of " ^ t1 ^ " to " ^ t2)
    |         Exceptions.InvalidUnaryOperation                                      -> print_endline("Invalid unary operator")
    |         Exceptions.AssignmentTypeMismatch(t1, t2)                     -> print_endline("Invalid assignment of " ^ t1 ^ " to " ^ t2)
    |         Exceptions.FunctionNotFound(fname, scope)                     -> print_endline("function " ^ fname ^ " not found in scope " ^ scope)
    |         Exceptions.UndefinedID(id)                                                      -> print_endline("Undefined id " ^ id)
    |         Exceptions.InvalidAccessLHS(t)                                            -> print_endline("Invalid LHS expression of dot operator with " ^ t)
```

```
        |       Exceptions.LHSofRootAccessMustBeIDorFunc(lhs)                    -> print_endline("Dot operator expects ID, not " ^ lhs)
        |       Exceptions.ObjAccessMustHaveObjectType(t)                        -> print_endline("Can only dereference objects, not " ^ t)
        |       Exceptions.UnknownIdentifierForClass(c, id)                      -> print_endline("Unknown id " ^ id ^ " for class " ^ c)
        |       Exceptions.CannotUseReservedFuncName(f)                          -> print_endline("Cannot use name " ^ f ^ " because it is reserved")
        |       Exceptions.InvalidArrayPrimitiveConsecutiveTypes(t1,t2)          -> print_endline("Array primitive types must be equal, not " ^ t1 ^ " " " ^ t2)
        |       Exceptions.InvalidArrayPrimitiveType(t)                          -> print_endline("Array primitive type invalid, " ^ t)
        |       Exceptions.MustPassIntegerTypeToArrayCreate                      -> print_endline("Only integer types can be passed to an array initializer")
        |       Exceptions.ArrayInitTypeInvalid(t)                               -> print_endline("Only integer types can be passed to an array initializer, not " ^ t)
        |       Exceptions.MustPassIntegerTypeToArrayAccess                      -> print_endline("Only integer types can be passed to an array access")
        |       Exceptions.ArrayAccessInvalidParamLength(o,a)                    -> print_endline("Only arrays can have access to length, not " ^ o ^ " " " ^ a)
        |       Exceptions.ArrayAccessExpressionNotArray(a)                      -> print_endline("This expression is not an array " ^ a)
        |       Exceptions.CanOnlyAccessLengthOfArray                            -> print_endline("Can only access the length of an array")
        |       Exceptions.CanOnlyDeleteObjectsOrArrays                          -> print_endline("Can only delete objects or arrays")
        |       Exceptions.CannotAccessLengthOfCharArray                         -> print_endline("Cannot access the length of a char array")
        |       Exceptions.AllNonVoidFunctionsMustEndWithReturn(f)               -> print_endline("Non-void function " ^ f ^ " does not end in return")
        |       Exceptions.CyclicalDependencyBetween(c1, c2)                     -> print_endline("Class " ^ c1 ^ " and " ^ c2 ^ " have a cylical dependence")
        |       Exceptions.CannotAccessPrivateFieldInNonProperScope(f, cp, cc) -> print_endline("Cannot access private field " ^ f ^ " in scope " ^ cp ^ " from object " ^ cc)
        |       Exceptions.CannotCallBreakOutsideOfLoop                          -> print_endline("Cannot call break outside of loop")
        |       Exceptions.CannotCallContinueOutsideOfLoop                       -> print_endline("Cannot call continue outside of loop")
        |         Exceptions.CannotAccessPrivateFunctionInNonProperScope(f, cp, cc) ->
↪   print_endline("Cannot access private function " ^ f ^ " in scope " ^ cp ^ " from
↪   object " ^ cc)
        |         Exceptions.CannotPassNonInheritedClassesInPlaceOfOthers(c1, c2)
↪           -> print_endline("Cannot pass non-inherited classe" ^ c1 ^ " to parameter " ^
↪   c2)
        |         Exceptions.IncorrectTypePassedToFunction(id, t)
↪                                        -> print_endline("Canot pass type " ^ t ^ "
↪   to " ^ id)
        |         Exceptions.IncorrectNumberOfArguments(f, a1, a2) ->
↪   print_endline("Cannot pass " ^ string_of_int a1 ^ " args when expecting " ^
↪   string_of_int a2 ^ " in " ^f)
        |         Exceptions.ClassIsNotExtendedBy(c1, c2)                         ->
↪   print_endline("Class " ^ c1 ^ " not extended by " ^ c2)

        |         Exceptions.InvalidTypePassedToPrintf                                ->
↪   print_endline("Invalid type passed to print")
        |
↪           Exceptions.InvalidBinaryOperator                                  ->
↪   print_endline("Invalid binary operator")
        |         Exceptions.UnknownVariable(id)
↪                                                -> print_endline("Unknown variable "
↪   ^ id)
        |         Exceptions.AssignLHSMustBeAssignable                                 ->
↪   print_endline("Assignment lhs must be assignable")
        |         Exceptions.CannotCastTypeException(t1, t2)                          ->
↪   print_endline("Cannot cast " ^ t1 ^ " to " ^ t2)
        |         Exceptions.InvalidBinopEvaluationType
↪                                   -> print_endline("Invalid binary expression
↪   evaluation type")
        |         Exceptions.FloatOpNotSupported
↪                                                -> print_endline("Float operation not
↪   supported")
        |         Exceptions.IntOpNotSupported
↪                                                -> print_endline("Integer operation
↪   not supported")
        |         Exceptions.LLVMFunctionNotFound(f)                                  ->
↪   print_endline("LLVM function " ^ f ^ " not found")
        |         Exceptions.InvalidStructType(t)
↪                                        -> print_endline("Invalid structure type " ^
↪   t)
        |         Exceptions.UnableToCallFunctionWithoutParent(f)          ->
↪   print_endline("Unable to call function " ^ f ^ " without parent")
        |         Exceptions.CannotAssignParam(p)
↪                                        -> print_endline("Cannot assign to param " ^
↪   p)
        |         Exceptions.InvalidUnopEvaluationType                                 ->
↪   print_endline("Invalid unary expression evaluation type")
```

```
        |         Exceptions.UnopNotSupported                                  -> print_endline("Unary operator not supported")
        |         Exceptions.ArrayLargerThan1Unsupported                       -> print_endline("Array dimensions greater than 1 not supported")
        |         Exceptions.CanOnlyCompareObjectsWithNull(e1, e2)         -> print_endline("Can only compare objects with null " ^ e1 ^ " " ^ e2)
        |         Exceptions.ObjOpNotSupported(op)                             -> print_endline("Object operator not supported " ^ op)
        |         Exceptions.CanOnlyCompareArraysWithNull(e1, e2)          -> print_endline("Can only compare arrays with null " ^ e1 ^ " " ^ e2)
```

exceptions.ml

```
(* Dice Exceptions *)
exception InvalidNumberCompilerArguments of int
exception InvalidCompilerArgument of string
exception NoFileArgument

(* Processor Exceptions *)
exception MissingEOF

(* Scanner Exceptions *)
exception IllegalCharacter of string * char * int
exception UnmatchedQuotation of int
exception IllegalToken of string

(* Analyzer Exceptions *)
exception IncorrectNumberOfArgumentsException
exception ConstructorNotFound of string
exception DuplicateClassName of string
exception DuplicateField
exception DuplicateFunction of string
exception DuplicateConstructor
exception DuplicateLocal of string
exception UndefinedClass of string
exception UnknownIdentifier of string
exception InvalidBinopExpression of string
exception InvalidIfStatementType
exception InvalidForStatementType
exception ReturnTypeMismatch of string * string
exception MainNotDefined
exception MultipleMainsDefined
exception InvalidWhileStatementType
exception LocalAssignTypeMismatch of string * string
exception InvalidUnaryOperation
exception AssignmentTypeMismatch of string * string
exception FunctionNotFound of string * string
exception UndefinedID of string
exception InvalidAccessLHS of string
exception LHSofRootAccessMustBeIDorFunc of string
exception ObjAccessMustHaveObjectType of string
exception UnknownIdentifierForClass of string * string
exception CannotUseReservedFuncName of string
exception InvalidArrayPrimitiveConsecutiveTypes of string * string
exception InvalidArrayPrimitiveType of string
exception MustPassIntegerTypeToArrayCreate
exception ArrayInitTypeInvalid of string
exception MustPassIntegerTypeToArrayAccess
exception ArrayAccessInvalidParamLength of string * string
exception ArrayAccessExpressionNotArray of string
exception CanOnlyAccessLengthOfArray
exception CanOnlyDeleteObjectsOrArrays
exception CannotAccessLengthOfCharArray
exception AllNonVoidFunctionsMustEndWithReturn of string
exception CyclicalDependencyBetween of string * string
exception CannotAccessPrivateFieldInNonProperScope of string * string * string
exception CannotCallBreakOutsideOfLoop
exception CannotCallContinueOutsideOfLoop
exception CannotAccessPrivateFunctionInNonProperScope of string * string * string
exception CannotPassNonInheritedClassesInPlaceOfOthers of string * string
exception IncorrectTypePassedToFunction of string * string
exception IncorrectNumberOfArguments of string * int * int
exception ClassIsNotExtendedBy of string * string

(* Codegen Exceptions *)
exception InvalidTypePassedToPrintf
exception InvalidBinaryOperator
exception UnknownVariable of string
exception AssignLHSMustBeAssignable
exception CannotCastTypeException of string * string
exception InvalidBinopEvaluationType
exception FloatOpNotSupported
exception IntOpNotSupported
exception LLVMFunctionNotFound of string
exception InvalidStructType of string
exception UnableToCallFunctionWithoutParent of string
exception CannotAssignParam of string
exception InvalidUnopEvaluationType
exception UnopNotSupported
exception ArrayLargerThan1Unsupported
exception CanOnlyCompareObjectsWithNull of string * string
exception ObjOpNotSupported of string
exception CanOnlyCompareArraysWithNull of string * string
```

filepath.ml

```
open Filename
open Unix

exception Safe_exception of (string * string list ref)

let raise_safe fmt =
  let do_raise msg = raise @@ Safe_exception (msg, ref []) in
  Printf.ksprintf do_raise fmt

let reraise_with_context ex fmt =
  let do_raise context =
    let () = match ex with
    | Safe_exception (_, old_contexts) -> old_contexts := context :: !old_contexts
    | _ -> Printf.eprintf "warning: Attempt to add note '%s' to non-Safe_exception!"
↪   context
    in
    raise ex
  in Printf.ksprintf do_raise fmt

module StringMap = struct
  include Map.Make(String)
  let find_nf = find
  let find_safe key map = try find key map with Not_found -> raise_safe "BUG: Key '%s'
↪   not found in StringMap!" key
  let find key map = try Some (find key map) with Not_found -> None
  let map_bindings fn map = fold (fun key value acc -> fn key value :: acc) map []
end

type path_component =
  | Filename of string  (* foo/ *)
  | ParentDir           (* ../ *)
  | CurrentDir          (* ./ *)
  | EmptyComponent      (* / *)

type filepath = string


let on_windows = Filename.dir_sep <> "/"

let path_is_absolute path = not (Filename.is_relative path)

let string_tail s i =
  let len = String.length s in
  if i > len then failwith ("String '" ^ s ^ "' too short to split at " ^ (string_of_int
↪   i))
  else String.sub s i (len - i)

```

```
let split_path_str path =
  let l = String.length path in
  let is_sep c = (c = '/' || (on_windows && c = '\\')) in

  (* Skip any leading slashes and return the rest *)
  let rec find_rest i =
    if i < l then (
      if is_sep path.[i] then find_rest (i + 1)
      else string_tail path i
    ) else (
      ""
    ) in

  let rec find_slash i =
    if i < l then (
      if is_sep path.[i] then (String.sub path 0 i, find_rest (i + 1))
      else find_slash (i + 1)
    ) else (
      (path, "")
    )
  in
  find_slash 0

let split_first path =
  if path = "" then
    (CurrentDir, "")
  else (
    let (first, rest) = split_path_str path in
    let parsed =
      if first = Filename.parent_dir_name then ParentDir
      else if first = Filename.current_dir_name then CurrentDir
      else if first = "" then EmptyComponent
      else Filename first in
    (parsed, rest)
  )

let normpath path : filepath =
  let rec explode path =
    match split_first path with
    | CurrentDir, "" -> []
    | CurrentDir, rest -> explode rest
    | first, "" -> [first]
    | first, rest -> first :: explode rest in

  let rec remove_parents = function
    | checked, [] -> checked
    | (Filename _name :: checked), (ParentDir :: rest) -> remove_parents (checked, rest)
    | checked, (first :: rest) -> remove_parents ((first :: checked), rest) in

  let to_string = function
    | Filename name -> name
    | ParentDir -> Filename.parent_dir_name
    | EmptyComponent -> ""
    | CurrentDir -> assert false in
  String.concat Filename.dir_sep @@ List.rev_map to_string @@ remove_parents ([], explode
↪  path)


let abspath path =
  let (+/) = Filename.concat in
  normpath (
    if path_is_absolute path then path
    else (Sys.getcwd ()) +/ path
  )

let realpath path =
  let (+/) = Filename.concat in   (* Faster version, since we know the path is relative
↪  *)

  (* Based on Python's version *)
  let rec join_realpath path rest seen =
    (* Printf.printf "join_realpath <%s> + <%s>\n" path rest; *)
    (* [path] is already a realpath (no symlinks). [rest] is the bit to join to it. *)
    match split_first rest with
    | Filename name, rest -> (
      (* path + name/rest *)
      let newpath = path +/ name in
      let link = try Some (Unix.readlink newpath) with Unix.Unix_error _ -> None in
      match link with
      | Some target ->
          (* path + symlink/rest *)
          begin match StringMap.find newpath seen with
          | Some (Some cached_path) -> join_realpath cached_path rest seen
          | Some None -> (normpath (newpath +/ rest), false)    (* Loop; give up *)
          | None ->
              (* path + symlink/rest -> realpath(path + target) + rest *)
              match join_realpath path target (StringMap.add newpath None seen) with
              | path, false ->
                  (normpath (path +/ rest), false)   (* Loop; give up *)
              | path, true -> join_realpath path rest (StringMap.add newpath (Some path)
↪  seen)
          end
      | None ->
          (* path + name/rest -> path/name + rest (name is not a symlink) *)
          join_realpath newpath rest seen
    )
    | CurrentDir, "" ->
        (path, true)
    | CurrentDir, rest ->
      (* path + ./rest *)
      join_realpath path rest seen
    | ParentDir, rest ->
      (* path + ../rest *)
      if String.length path > 0 then (
        let name = Filename.basename path in
        let path = Filename.dirname path in
        if name = Filename.parent_dir_name then
          join_realpath (path +/ name +/ name) rest seen    (* path/.. +  ../rest ->
↪ path/../.. + rest *)
        else
          join_realpath path rest seen                      (* path/name + ../rest ->
↪ path + rest *)
      ) else (
        join_realpath Filename.parent_dir_name rest seen    (* "" + ../rest -> .. + rest
↪ *)
      )
    | EmptyComponent, rest ->
        (* [rest] is absolute; discard [path] and start again *)
        join_realpath Filename.dir_sep rest seen
  in

  try
    if on_windows then
      abspath path
    else (
      fst @@ join_realpath (Sys.getcwd ()) path StringMap.empty
    )
  with Safe_exception _ as ex -> reraise_with_context ex "... in realpath(%s)" path
```

parser.mly

```
%{  open Ast  %}

%token CLASS EXTENDS CONSTRUCTOR INCLUDE DOT THIS PRIVATE PUBLIC
%token INT FLOAT BOOL CHAR VOID NULL TRUE FALSE
%token SEMI LPAREN RPAREN LBRACE RBRACE LBRACKET RBRACKET COMMA
%token AND NOT OR PLUS MINUS TIMES DIVIDE ASSIGN MODULO
%token EQ NEQ LT LEQ GT GEQ BAR
%token RETURN IF ELSE FOR WHILE BREAK CONTINUE NEW DELETE
%token <int> INT_LITERAL
%token <float> FLOAT_LITERAL
%token <string> STRING_LITERAL
%token <string> ID
%token <char> CHAR_LITERAL
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left AND OR
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE MODULO
%right NOT
%right DELETE
%right RBRACKET
%left LBRACKET
%right DOT

%start program
%type <Ast.program> program

%%

program:
                includes cdecls EOF { Program($1, $2) }

/******************
        INCLUDE
******************/

includes:
                /* nothing */ { [] }
          |         include_list  { List.rev $1 }

include_list:
            include_decl             { [$1] }
          |         include_list include_decl { $2::$1 }

include_decl:
        INCLUDE LPAREN STRING_LITERAL RPAREN SEMI { Include($3) }


/******************
 CLASSES
******************/
cdecls:
    cdecl_list    { List.rev $1 }

cdecl_list:
    cdecl             { [$1] }
  | cdecl_list cdecl  { $2::$1 }

cdecl:
                CLASS ID LBRACE cbody RBRACE { {
                        cname = $2;
                        extends = NoParent;
                        cbody = $4
                } }
        |         CLASS ID EXTENDS ID LBRACE cbody RBRACE { {
                        cname = $2;
                        extends = Parent($4);
                        cbody = $6
                } }

cbody:
                /* nothing */ { {
                        fields = [];
                        constructors = [];
                        methods = [];
                } }
         |         cbody field { {
                        fields = $2 :: $1.fields;
                        constructors = $1.constructors;
                        methods = $1.methods;
                } }
         |         cbody constructor { {
                        fields = $1.fields;
                        constructors = $2 :: $1.constructors;
                        methods = $1.methods;
                } }
         |         cbody fdecl { {
                        fields = $1.fields;
                        constructors = $1.constructors;
                        methods = $2 :: $1.methods;
                } }
```

```


/*****************
 CONSTRUCTORS
*****************/

constructor:
        CONSTRUCTOR LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE {
                {
                        scope = Public;
                        fname = Constructor;
                        returnType = Datatype(ConstructorType);
                        formals = $3;
                        body = List.rev $6;
                        overrides = false;
            root_cname = None;
                }
        }

/*****************
 FIELDS
*****************/

scope:
                PRIVATE { Private }
        |         PUBLIC  { Public }

/* public UserObj name; */
field:
                scope datatype ID SEMI { Field($1, $2, $3) }

/*****************
 METHODS
*****************/

fname:
        ID { $1 }

fdecl:
        scope datatype fname LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
        {
                {
                        scope = $1;
                        fname = FName($3);
                        returnType = $2;
                        formals = $5;
                        body = List.rev $8;
                        overrides = false;
            root_cname = None;
```

```
                }
        }

/*****************
 FORMALS/PARAMETERS & VARIABLES & ACTUALS
*****************/

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

formal_list:
                formal                   { [$1] }
        |         formal_list COMMA formal { $3 :: $1 }

formal:
        datatype ID { Formal($1, $2) }

actuals_opt:
                /* nothing */ { [] }
        |         actuals_list  { List.rev $1 }

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


/**************
        DATATYPES
**************/
primitive:
                INT                 { Int_t }
        |         FLOAT                 { Float_t }
        |         CHAR                 { Char_t }
        |         BOOL                  { Bool_t }
        |         VOID            { Void_t }

name:
        CLASS ID { Objecttype($2) }

type_tag:
                primitive { $1 }
        |        name          { $1 }

array_type:
        type_tag LBRACKET brackets RBRACKET { Arraytype($1, $3) }

datatype:
                type_tag   { Datatype($1) }
        |        array_type { $1 }

brackets:
                /* nothing */                          { 1 }
        |        brackets RBRACKET LBRACKET { $1 + 1 }

/******************
 EXPRESSIONS
******************/

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

stmt:
                expr SEMI { Expr($1) }
        |        RETURN expr SEMI { Return($2) }
        |       RETURN SEMI                  { Return(Noexpr) }
        |        LBRACE stmt_list RBRACE { Block(List.rev $2) }
        |        IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5,
↪   Block([Expr(Noexpr)])) }
        |        IF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7) }
        |        FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
                { For($3, $5, $7, $9) }
        |        WHILE LPAREN expr RPAREN stmt         { While($3, $5) }
        |       BREAK SEMI                                              { Break }
        |       CONTINUE SEMI                                       { Continue }
        |   datatype ID SEMI                                { Local($1, $2, Noexpr) }
        |        datatype ID ASSIGN expr SEMI         { Local($1, $2, $4) }

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

expr:
                literals                                                 { $1 }
        |        expr PLUS   expr                                   { Binop($1,
↪   Add,   $3) }
        |        expr MINUS  expr                                   { Binop($1,
↪   Sub,   $3) }
        |        expr TIMES  expr                                   { Binop($1,
↪   Mult,  $3) }
        |        expr DIVIDE expr                                   { Binop($1,
↪   Div,   $3) }
        |        expr 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 MODULO expr                                        { Binop($1,
↪   Mod,    $3)}
        |           NOT  expr                                                        {
↪   Unop (Not,    $2) }
        |           expr OR     expr                                        { Binop($1,
↪   Or,     $3) }
        |           expr DOT    expr                                        {
↪   ObjAccess($1, $3) }
        |           expr ASSIGN expr                                        { Assign($1,
↪   $3) }
        |           DELETE expr                                                { 
↪   Delete($2) }
        |     MINUS expr                                                      { Unop
↪   (Sub, $2) }
        |           ID LPAREN actuals_opt RPAREN                 { Call($1, $3) }
        |           NEW ID LPAREN actuals_opt RPAREN         { ObjectCreate($2, $4) }
        |          NEW type_tag bracket_args RBRACKET          { ArrayCreate(Datatype($2),
↪   List.rev $3) }
        |           expr bracket_args RBRACKET                          { ArrayAccess($1,
↪   List.rev $2) }
        |           LPAREN expr RPAREN                                        { $2 }

bracket_args:
                  LBRACKET expr                                                { [$2] }
        |           bracket_args RBRACKET LBRACKET expr { $4 :: $1 }

literals:
          INT_LITERAL                     { Int_Lit($1) }
        | FLOAT_LITERAL                   { Float_Lit($1) }
        | TRUE                                          { Boolean_Lit(true) }
        | FALSE                                          { Boolean_Lit(false) }
        | STRING_LITERAL                  { String_Lit($1) }
        | CHAR_LITERAL                       { Char_Lit($1) }
        | THIS                                           { This }
        | ID                                           { Id($1) }
        | NULL                                    { Null }
        | BAR array_prim BAR          { ArrayPrimitive($2) }

/* ARRAY LITERALS */
```

```

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

processor.ml

```
open Parser

type token_attr = {
  lineno: int;
  cnum: int;
}

let line_number = ref 1
let last_token = ref EOF
let char_num = ref 1
let filename = ref ""

let build_token_list lexbuf =
        Scanner.filename := !filename;
  let rec helper prev_cnum prev_lineno lexbuf token_list =
    let token = Scanner.token lexbuf in
    let lineno = !Scanner.lineno in
    let cnum = (Lexing.lexeme_start_p lexbuf).Lexing.pos_cnum in
    let prev_cnum = if lineno > prev_lineno then cnum else prev_cnum in
    let cnum = cnum - prev_cnum in
    match token with
        EOF as eof -> (eof, { lineno = lineno; cnum = cnum } )::token_list
    |   t          -> (t, { lineno = lineno; cnum = cnum } )::(helper prev_cnum lineno
↪   lexbuf token_list)
  in helper 0 0 lexbuf []

let parser filen token_list =
  let token_list = ref(token_list) in
  let tokenizer _ =
    match !token_list with
      | (head, curr) :: tail ->
          filename := filen;
          line_number := curr.lineno;
          char_num    := curr.cnum;
          last_token := head;
          token_list := tail;
          head
      | [] -> raise (Exceptions.MissingEOF)
  in
  let program = Parser.program tokenizer (Lexing.from_string "") in
  program
```

sast.ml

```
open Ast

type sexpr =
                SInt_Lit of int
        |         SBoolean_Lit of bool
        |         SFloat_Lit of float
        |         SString_Lit of string
        |         SChar_Lit of char
        |         SId of string * datatype
        |         SBinop of sexpr * op * sexpr * datatype
        |         SAssign of sexpr * sexpr * datatype
        |         SNoexpr
        |         SArrayCreate of datatype * sexpr list * datatype
        |         SArrayAccess of sexpr * sexpr list * datatype
        |         SObjAccess of sexpr * sexpr * datatype
        |         SCall of string * sexpr list * datatype * int
        |   SObjectCreate of string * sexpr list * datatype
        |         SArrayPrimitive of sexpr list * datatype
        |          SUnop of op * sexpr * datatype
        |         SNull
        |         SDelete of sexpr

type sstmt =
                SBlock of sstmt list
        |         SExpr of sexpr * datatype
        |         SReturn of sexpr  * datatype
        |         SIf of sexpr * sstmt * sstmt
        |         SFor of sexpr * sexpr * sexpr * sstmt
        |         SWhile of sexpr * sstmt
        |          SBreak
        |   SContinue
        |   SLocal of datatype * string * sexpr

type func_type = User | Reserved

type sfunc_decl = {
        sfname : fname;
        sreturnType : datatype;
        sformals : formal list;
        sbody : sstmt list;
        func_type : func_type;
        source : string;
        overrides : bool;
}

type sclass_decl = {
        scname : string;
        sfields : field list;
        sfuncs: sfunc_decl list;
}

(* Class Declarations | All method declarations | Main entry method *)
type sprogram =  {
        classes : sclass_decl list;
        functions : sfunc_decl list;
        main : sfunc_decl;
        reserved : sfunc_decl list;
}
```

scanner.mll

```
{
        open Parser
    let lineno = ref 1
    let depth = ref 0
    let filename = ref ""

    let unescape s =
            Scanf.sscanf ("\"" ^ s ^ "\"") "%S%!" (fun x -> x)
}

let alpha = [’a’-’z’ ’A’-’Z’]
let escape = ’\\’ [’\\’ ’’’ ’"’ ’n’ ’r’ ’t’]
let escape_char = ’’’ (escape) ’’’
let ascii = ([’ ’-’!’ ’#’-’[’ ’]’-’~’])
let digit = [’0’-’9’]
let id = alpha (alpha | digit | ’_’)*
let string = ’"’ ( (ascii | escape)* as s) ’"’
let char = ’’’ ( ascii | digit ) ’’’
let float = (digit+) [’.’] digit+
let int = digit+
let whitespace = [’ ’ ’\t’ ’\r’]
let return = ’\n’

rule token = parse
  whitespace { token lexbuf }
| return          { incr lineno; token lexbuf}
| "(*"      { incr depth; comment lexbuf }

| ’(’      { LPAREN }
| ’)’      { RPAREN }
| ’{’      { LBRACE }
| ’}’      { RBRACE }
| ’;’      { SEMI }
| ’,’      { COMMA }

(* Operators *)
| ’+’      { PLUS }
| ’-’      { MINUS }
| ’*’      { TIMES }
| ’/’      { DIVIDE }
| ’%’      { MODULO }
| ’=’      { ASSIGN }
| "=="     { EQ }
| "!="     { NEQ }
| ’<’      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "and"    { AND }
| "or"     { OR }
| "not"    { NOT }
| '.'      { DOT }
| '['      { LBRACKET }
| ']'      { RBRACKET }
| '|'           { BAR }

(* Branch Control *)
| "if"     { IF }
| "else"   { ELSE }
| "for"    { FOR }
| "while"  { WHILE }
| "return" { RETURN }

(* Data Types *)
| "int"    { INT }
| "float"  { FLOAT }
| "bool"   { BOOL }
| "char"   { CHAR }
| "void"   { VOID }
| "null"   { NULL }
| "true"   { TRUE }
| "false"  { FALSE }

(* Classes *)
| "class"       { CLASS }
| "constructor" { CONSTRUCTOR }
| "public"      { PUBLIC }
| "private"     { PRIVATE }
| "extends"     { EXTENDS }
| "include"     { INCLUDE }
| "this"        { THIS }
| "break"                 { BREAK }
| "continue"        { CONTINUE }
| "new"                 { NEW }
| "delete"                 { DELETE }

| int as lxm                    { INT_LITERAL(int_of_string lxm) }
| float as lxm                  { FLOAT_LITERAL(float_of_string lxm) }
| char as lxm                   { CHAR_LITERAL( String.get lxm 1 ) }
| escape_char as lxm{ CHAR_LITERAL( String.get (unescape lxm) 1) }
| string                        { STRING_LITERAL(unescape s) }
| id as lxm                     { ID(lxm) }
| eof                           { EOF }

| '"'                          { raise (Exceptions.UnmatchedQuotation(!lineno)) }
| _ as illegal  { raise (Exceptions.IllegalCharacter(!filename, illegal, !lineno)) }
```

```

and comment = parse
            return         { incr lineno; comment lexbuf }
          |        "*)"         { decr depth; if !depth > 0 then comment lexbuf else
↪  token lexbuf }
        |   "(*"         { incr depth; comment lexbuf }
        |        _            { comment lexbuf }
```

stdlib.dice

```
class Integer {

    private int my_int;

    constructor(int input) {
        this.my_int = input;
    }

    public int num() {
        return this.my_int;
    }


    public char toChar(int digit) {

        if (digit == 0) {
            return '0';
        } else if (digit == 1) {
            return '1';
        } else if (digit == 2) {
            return '2';
        } else if (digit == 3) {
            return '3';
        } else if (digit == 4) {
            return '4';
        } else if (digit == 5) {
            return '5';
        } else if (digit == 6) {
            return '6';
        } else if (digit == 7) {
            return '7';
        } else if (digit == 8) {
            return '8';
        } else if (digit == 9) {
            return '9';
        }

        return 'z';
    }





    public class String toString() {

        (* integer cannot be greater than 10 digits in 32 bit *)
        int temp = this.my_int;
        int i = 0;
        char[] str = new char[9];

        int digit = temp % 10;
        str[i] = this.toChar(digit);
        i = i + 1;
        temp = temp / 10;
        while (temp > 0) {

            digit = temp % 10;
            str[i] = this.toChar(digit);
            temp = temp / 10;
            i = i + 1;
        }

        str[i] = 0;
        class String newString = new String(str);
        class String a = newString.reverse();
        return newString.reverse();
    }
}



class String {

   private char[] my_string;
   private int length;

   constructor(char[] input) {

       this.my_string = this.copy_internal(input);

       this.length = this.length();
   }

   (* PRIVATE CLASSES -----------------------------------------   *)

   private int length_internal(char[] input) {
        int length = 0;

        while(input[length] != 0) {
          length = length + 1;
        }

        return length;
    }

   private char[] copy_internal(char[] input) {

        char[] newString = new char[this.length_internal(input) + 1];

        int i = 0;
        for (; input[i] != 0; i = i + 1) {
            newString[i] = input[i];
        }

        newString[i] = 0;
        return newString;
   }

   (* PUBLIC CLASSES ----------------------------------------    *)

   public char[] string() {
        return this.my_string;
    }

   public char getChar(int index) {

        return this.my_string[index];
   }

   public int length() {

       int length = 0;

       while(this.my_string[length] != 0){
        length = length + 1;
      }

       return length;
   }

    public int toInteger() {

        char[] temp = this.string();
        int ndigit = 0;
        int i;
        int j;
        for (i = 0; i < this.length; i = i + 1) {

            int exp = 1;
            int xdigit = this.toDigit(temp[i]);
            for (j = 0; j < (this.length-i-1); j = j + 1) {
                exp = exp * 10;
            }
            xdigit = xdigit * exp;
```

```
            ndigit = ndigit + xdigit;
        }

        return ndigit;
    }

    public int toDigit(char digit) {

        if (digit == '0') {
            return 0;
        } else if (digit == '1') {
            return 1;
        } else if (digit == '2') {
            return 2;
        } else if (digit == '3') {
            return 3;
        } else if (digit == '4') {
            return 4;
        } else if (digit == '5') {
            return 5;
        } else if (digit == '6') {
            return 6;
        } else if (digit == '7') {
            return 7;
        } else if (digit == '8') {
            return 8;
        } else if (digit == '9') {
            return 9;
        }

        return -1;
    }


   public class String copy(class String input) {

        char[] newArray = this.copy_internal(input.string());
        class String newString = new String(newArray);
        return newString;
    }

   public int indexOf(char x) {

        int i = 0;
        for (; this.getChar(i) != x and this.getChar(i) != 0; i = i + 1) {
            }

        (* If the char was not found, return -1 *)
        if (i == this.length()) {
            return -1;
        }

        return i;
    }

  public class String reverse() {

      class String newString;

      char[] temp = new char[this.length + 1];
      int i = this.length;
      for (; i > 0; i = i - 1) {

          temp[this.length - i] = this.getChar(i-1);
      }
      temp[this.length] = 0;
      newString = new String(temp);
      return newString;
  }

  public class String concat(class String temp) {

       char[] temparray = new char[this.length() + temp.length() + 1];

       (* Copy over the current string into a new char array *)
       int i = 0;
       for (; this.getChar(i) != 0; i = i + 1) {
           temparray[i] = this.getChar(i);
       }

       (* Append the new string *)
       int j = 0;
       for (; temp.getChar(j) != 0; j = j + 1) {
           temparray[i+j] = temp.getChar(j);
       }

       temparray[this.length() + temp.length()] = 0;
       class String newString = new String(temparray);
       return newString;
  }

  public bool compare(class String check) {

      if (check.length != this.length) {
          return false;
      }

      int i = 0;
```

```

        for (; i < check.length(); i = i + 1) {

            if (check.getChar(i) != this.getChar(i)) {
                return false;
            }
        }

        return true;
    }

    public bool contains(class String check) {


        if (this.length < check.length) {
            return false;
        } else if (this.compare(check)) {
            return true;
        } else {

            int diff = this.length - check.length + 1;
            int i;
            int j;
            for ( i = 0; i < diff; i = i + 1)

                for ( j = 0; j < check.length; j = j + 1) {

                    if (this.getChar(i+j) != check.getChar(j)) {
                        break;
                    }

                    if (j == check.length - 1) {
                        return true;
                    }
                }
            }
        return false;
    }

    public void free() {

        delete(this.my_string);
    }

}



class File {

    private class String filePath;
    private bool isWriteEnabled;
    private int fd;

    constructor(char[] path, bool isWriteEnabled) {

        this.filePath = new String(path);
        this.isWriteEnabled = isWriteEnabled;
        class String a = this.filePath;
        this.fd = this.openfile(a, this.isWriteEnabled);
        if (this.fd < 0) {
            print("open failed");
            exit(1);
        }
    }

   (* PRIVATE CLASSES -----------------------------------------   *)

    private int openfile(class String path, bool isWriteEnabled) {

        if (isWriteEnabled) {
            (* 2 is the value for O_RDWR *)
            return open(path.string(), 2);
        }

        (* 0 is the value for O_RDONLY *)
        return open(path.string(), 0);
    }

   (* PUBLIC CLASSES -----------------------------------------   *)

    public void closefile() {

        if (close(this.fd) < 0) {
            print("close failed");
        }
    }

    public char[] readfile(int bytes) {

        char[] buf = new char[bytes];

        int ret = read(this.fd, buf, bytes);

        if (ret < 0) {
            print("read failed");
        }

        return buf;
    }

    public int writefile(char[] buf, int offset) {

        class String temp = new String(buf);
        int err;
        (* seek to desired offset from beginning of file *)
        if (offset > 0) {
            err = lseek(this.fd, offset, 0);
        } else if (offset == -1) {
            err = lseek(this.fd, 0, 0);
        } else {
        (* Seek to the end of the file by default *)
        err = lseek(this.fd, 0, 2);
        }

        if (err < 0) {
            print("seek failed");
        }

        err = write(this.fd, temp.string(), temp.length());
        if (err < 0) {
            print("write failed");
        }
        return err;
    }

}
```

utils.ml

```
(* Pretty Printer *)
open Ast
open Sast
open Parser
open Processor
open Yojson

let save file string =
         let channel = open_out file in
         output_string channel string;
         close_out channel

let replace input output =
        Str.global_replace (Str.regexp_string input) output

(* Print data types *)

let string_of_scope = function
                Public         -> "public"
        |         Private -> "private"

let string_of_primitive = function
                Int_t                                        -> "int"
        |         Float_t                                  -> "float"
        |         Void_t                                         -> "void"
        |         Bool_t                                          -> "bool"
        |         Char_t                                          -> "char"
        |         Objecttype(s)                         -> "class " ^ s
        |         ConstructorType                        -> "constructor"
        |          Null_t                                          -> "null"

let string_of_object = function
                Datatype(Objecttype(s))        -> s
        |         _ -> ""

let rec print_brackets = function
                1 -> "[]"
        |         a -> "[]" ^ print_brackets (a - 1)

let string_of_datatype = function
                Arraytype(p, i)        -> (string_of_primitive p) ^ (print_brackets i)
        |         Datatype(p)                 -> (string_of_primitive p)
        |          Any                          -> "Any"

(* Print expressions *)

let string_of_op = function
```

```
                Add                         -> "+"
         |         Sub                         -> "-"
         |         Mult                 -> "*"
         |         Div                         -> "/"
         |         Equal                 -> "=="
         |         Neq                         -> "!="
         |         Less                 -> "<"
         |         Leq                         -> "<="
         |         Greater                 -> ">"
         |         Geq                         -> ">="
         |         And                         -> "and"
         |         Not                         -> "not"
         |         Or                         -> "or"
         |         Mod                 -> "%"

let rec string_of_bracket_expr = function
                []                                  -> ""
        |          head :: tail          -> "[" ^ (string_of_expr head) ^ "]" ^
↪   (string_of_bracket_expr tail)
and string_of_array_primitive = function
                []                                  -> ""
        |    [last]                         -> (string_of_expr last)
        |          head :: tail          -> (string_of_expr head) ^ ", " ^
↪   (string_of_array_primitive tail)
and string_of_expr = function
                Int_Lit(i)                                    -> string_of_int i
        |        Boolean_Lit(b)                          -> if b then "true" else "false"
        |        Float_Lit(f)                          -> string_of_float f
        |        String_Lit(s)                          -> "\"" ^ (String.escaped s) ^ "\""
        |        Char_Lit(c)                                 -> Char.escaped c
        |        This                                         -> "this"
        |        Id(s)                                         -> s
        |        Binop(e1, o, e2)                   -> (string_of_expr e1) ^ " " ^
↪   (string_of_op o) ^ " " ^ (string_of_expr e2)
        |        Assign(e1, e2)                         -> (string_of_expr e1) ^ " = " ^
↪   (string_of_expr e2)
        |        Noexpr                                         -> ""
        |        ObjAccess(e1, e2)                  -> (string_of_expr e1) ^ "." ^
↪   (string_of_expr e2)
        |        Call(f, el)                                 -> f ^ "(" ^ String.concat ",
↪   " (List.map string_of_expr el) ^ ")"
        |        ArrayPrimitive(el)                  -> "|" ^ (string_of_array_primitive
↪   el) ^ "|"
        |          Unop(op, e)                                  -> (string_of_op op) ^ "("
↪   ^ string_of_expr e ^ ")"
        |        Null                                         -> "null"
        |   ArrayCreate(d, el)          -> "new " ^ string_of_datatype d ^
↪   string_of_bracket_expr el
```

```
        |   ArrayAccess(e, el)          -> (string_of_expr e) ^ (string_of_bracket_expr
↪  el)
        |   ObjectCreate(s, el)         -> "new " ^ s ^ "(" ^ String.concat ", "
↪  (List.map string_of_expr el) ^ ")"
        |         Delete(e)                                   -> "delete (" ^
↪  (string_of_expr e) ^ ")"
;;

let rec string_of_bracket_sexpr = function
                []                                  -> ""
        |         head :: tail          -> "[" ^ (string_of_sexpr head) ^ "]" ^
↪  (string_of_bracket_sexpr tail)
and string_of_sarray_primitive = function
                []                                  -> ""
        |   [last]                         -> (string_of_sexpr last)
        |         head :: tail          -> (string_of_sexpr head) ^ ", " ^
↪  (string_of_sarray_primitive tail)
and string_of_sexpr = function
                SInt_Lit(i)                                         -> string_of_int i
        |         SBoolean_Lit(b)                                -> if b then "true" else
↪  "false"
        |         SFloat_Lit(f)                                -> string_of_float f
        |         SString_Lit(s)                                -> "\"" ^ (String.escaped
↪  s) ^ "\""
        |         SChar_Lit(c)                                -> Char.escaped c
        |         SId(s, _)                                         -> s
        |         SBinop(e1, o, e2, _)                -> (string_of_sexpr e1) ^ " " ^
↪  (string_of_op o) ^ " " ^ (string_of_sexpr e2)
        |         SAssign(e1, e2, _)                        -> (string_of_sexpr e1) ^ " =
↪  " ^ (string_of_sexpr e2)
        |         SNoexpr                                             -> ""
        |         SObjAccess(e1, e2, _)                -> (string_of_sexpr e1) ^ "." ^
↪  (string_of_sexpr e2)
        |         SCall(f, el, _, _)                        -> f ^ "(" ^ String.concat ",
↪  " (List.map string_of_sexpr el) ^ ")"
        |         SArrayPrimitive(el, _)                -> "|" ^
↪  (string_of_sarray_primitive el) ^ "|"
        |           SUnop(op, e, _)                             -> (string_of_op op) ^
↪  "(" ^ string_of_sexpr e ^ ")"
        |         SNull                                               -> "null"
        |   SArrayCreate(d, el, _)          -> "new " ^ string_of_datatype d ^
↪  string_of_bracket_sexpr el
        |   SArrayAccess(e, el, _)          -> (string_of_sexpr e) ^
↪  (string_of_bracket_sexpr el)
        |   SObjectCreate(s, el, _)         -> "new " ^ s ^ "(" ^ String.concat ", "
↪  (List.map string_of_sexpr el) ^ ")"
        |         SDelete(e)                                        -> "delete (" ^
↪  (string_of_sexpr e) ^ ")"
;;
```

```

let string_of_local_expr = function
                Noexpr -> ""
        |          e            -> " = " ^ string_of_expr e

(* Print statements *)

let rec string_of_stmt indent =
        let indent_string = String.make indent '\t' in
        let get_stmt_string = function

                        Block(stmts)                         ->
                                indent_string ^ "{\n" ^
                                        String.concat "" (List.map (string_of_stmt
↪ (indent+1)) stmts) ^
                                indent_string ^ "}\n"

                |         Expr(expr)                            ->
                                indent_string ^ string_of_expr expr ^ ";\n";

                |         Return(expr)                      ->
                                indent_string ^ "return " ^ string_of_expr expr ^ ";\n";

                |         If(e, s, Block([Expr(Noexpr)]))         ->
                                indent_string ^ "if (" ^ string_of_expr e ^ ")\n" ^
                                        (string_of_stmt (indent+1) s)

                |         If(e, s1, s2)                       ->
                                indent_string ^ "if (" ^ string_of_expr e ^ ")\n" ^
                                        string_of_stmt (indent+1) s1 ^
                                indent_string ^ "else\n" ^
                                        string_of_stmt (indent+1) s2

                |         For(e1, e2, e3, s)                 ->
                                indent_string ^ "for (" ^ string_of_expr e1  ^ " ; " ^
↪ string_of_expr e2 ^ " ; " ^ string_of_expr e3  ^ ")\n" ^
                                        string_of_stmt (indent) s

                |         While(e, s)                         ->
                                indent_string ^ "while (" ^ string_of_expr e ^ ")\n" ^
                                        string_of_stmt (indent) s

                |          Break                                      -> indent_string
↪ ^ "break;\n"
                |          Continue                              -> indent_string ^
↪ "continue;\n"
                |   Local(d, s, e)                         -> indent_string ^
↪ string_of_datatype d ^ " " ^ s ^ string_of_local_expr e ^ ";\n"
        in get_stmt_string
```

```

let string_of_local_sexpr = function
                SNoexpr         -> ""
        |          e                            -> " = " ^ string_of_sexpr e

let rec string_of_sstmt indent =
        let indent_string = String.make indent '\t' in
        let get_stmt_string = function

                        SBlock(stmts)                          ->
                                indent_string ^ "{\n" ^
                                        String.concat "" (List.map (string_of_sstmt
 ↪  (indent+1)) stmts) ^
                                indent_string ^ "}\n"

                |         SExpr(expr, _)                                 ->
                                indent_string ^ string_of_sexpr expr ^ ";\n";

                |         SReturn(expr, _)                          ->
                                indent_string ^ "return " ^ string_of_sexpr expr ^ ";\n";

                |         SIf(e, s, SBlock([SExpr(SNoexpr, _)]))          ->
                                indent_string ^ "if (" ^ string_of_sexpr e ^ ")\n" ^
                                        (string_of_sstmt (indent+1) s)

                |         SIf(e, s1, s2)                          ->
                                indent_string ^ "if (" ^ string_of_sexpr e ^ ")\n" ^
                                        string_of_sstmt (indent+1) s1 ^
                                indent_string ^ "else\n" ^
                                        string_of_sstmt (indent+1) s2

                |         SFor(e1, e2, e3, s)                    ->
                                indent_string ^ "for (" ^ string_of_sexpr e1  ^ " ; " ^
 ↪  string_of_sexpr e2 ^ " ; " ^ string_of_sexpr e3  ^ ")\n" ^
                                        string_of_sstmt (indent) s

                |         SWhile(e, s)                           ->
                                indent_string ^ "while (" ^ string_of_sexpr e ^ ")\n" ^
                                        string_of_sstmt (indent) s

                |          SBreak                                        -> indent_string
 ↪  ^ "break;\n"
                |          SContinue                                -> indent_string ^
 ↪  "continue;\n"
                |   SLocal(d, s, e)                        -> indent_string ^
 ↪  string_of_datatype d ^ " " ^ s ^ string_of_local_sexpr e ^ ";\n"
        in get_stmt_string

(* Print Function *)
```

```ocaml

let string_of_fname = function
                Constructor -> "constructor"
        |        FName(s)        -> s

let string_of_formal = function
                Formal(d, s) -> (string_of_datatype d) ^ " " ^ s
        |          _                          -> ""

let string_of_formal_name = function
                Formal(_, s) -> s
        |         _ -> ""

let string_of_func_decl fdecl =
        "" ^ (string_of_scope fdecl.scope) ^ " " ^ (string_of_datatype fdecl.returnType)
↪  ^ " " ^ (string_of_fname fdecl.fname) ^ " " ^
        (* Formals *)
        "(" ^ String.concat "," (List.map string_of_formal fdecl.formals) ^ ") {\n" ^
                (* body *)
                String.concat "" (List.map (string_of_stmt 2) fdecl.body) ^
        "\t}\n\n"

(* Class Printing *)

let string_of_extends = function
                NoParent        -> ""
        |         Parent(s)        -> "extends " ^ s ^ " "
let string_of_field = function
        Field(s, d, id) -> (string_of_scope s) ^ " " ^ (string_of_datatype d) ^ " " ^ id
↪  ^ ";\n"

let string_of_cbody cbody =
        String.concat "" (List.map (fun s -> "\t" ^ s) (List.map string_of_field
↪  cbody.fields)) ^
        String.concat "" (List.map (fun s -> "\t" ^ s) (List.map string_of_func_decl
↪  cbody.constructors)) ^
        String.concat "" (List.map (fun s -> "\t" ^ s) (List.map string_of_func_decl
↪  cbody.methods))

let string_of_class_decl cdecl =
        "class " ^ cdecl.cname ^ " " ^ (string_of_extends cdecl.extends) ^ "{\n" ^
        (string_of_cbody cdecl.cbody) ^
        "}\n"

(* Include Printing *)

let rec string_of_include = function
        Include(s) -> "include(" ^ s ^ ");\n"

```

```
(* Print whole program *)

let string_of_program = function
        Program(includes, cdecls) ->
                String.concat "" (List.map string_of_include includes) ^ "\n" ^
                String.concat "\n" (List.map string_of_class_decl cdecls)

(* Print AST tree representation *)

let includes_tree includes =
        `List (List.map (function Include s -> `String s) includes)

let map_fields_to_json fields =
        `List (List.map (function Field(scope, datatype, s) ->
                `Assoc [
                        ("name", `String s);
                        ("scope", `String (string_of_scope scope));
                        ("datatype", `String (string_of_datatype datatype));
                ]) fields)

let map_formals_to_json formals =
        `List (List.map (function Formal(d, s) -> `Assoc [

↪  `String s);

↪  `String (string_of_datatype d));
                                                                                                              ]
                                                                         | Many d -> `Assoc [("Many",
↪  `String (string_of_datatype d));]
                ) formals)

let rec map_expr_to_json = function
                Int_Lit(i)                                  -> `Assoc [("int_lit", `Int i)]
        |       Boolean_Lit(b)                           -> `Assoc [("bool_lit", `Bool b)]
        |       Float_Lit(f)                           -> `Assoc [("float_lit", `Float f)]
        |       String_Lit(s)                           -> `Assoc [("string_lit", `String
↪  s)]
        |       Char_Lit(c)                                   -> `Assoc [("char_lit",
↪  `String (Char.escaped c))]
        |       This                                           -> `String "this"
        |       Id(s)                                           -> `Assoc [("id", `String
↪  s)]
        |       Binop(e1, o, e2)                   -> `Assoc [("binop", `Assoc [("lhs",
↪  map_expr_to_json e1); ("op", `String (string_of_op o)); ("rhs", map_expr_to_json
↪  e2)])]
        |       Assign(e1, e2)                         -> `Assoc [("assign", `Assoc
↪  [("lhs", map_expr_to_json e1); ("op", `String "="); ("rhs", map_expr_to_json e2)])]
        |       Noexpr                                           -> `String "noexpr"
        |       ObjAccess(e1, e2)                -> `Assoc [("objaccess", `Assoc [("lhs", map_expr_to_json e1); ("op", `String "."); ("rhs", map_expr_to_json e2)])]
        |       Call(f, el)                                -> `Assoc [("call", `Assoc ([("name", `String f); ("params", `List (List.map map_expr_to_json el)); ]) )]
        |       ArrayPrimitive(el)                     -> `Assoc [("arrayprimitive", `List(List.map map_expr_to_json el))]
        |         Unop(op, e)                                -> `Assoc [("Unop", `Assoc [("op", `String (string_of_op op)); ("operand", map_expr_to_json e)])]
        |       Null                                         -> `String "null"
        |   ArrayCreate(d, el)          -> `Assoc [("arraycreate", `Assoc [("datatype", `String (string_of_datatype d)); ("args", `List (List.map map_expr_to_json el))])]
        |   ArrayAccess(e, el)          -> `Assoc [("arrayaccess", `Assoc [("array", map_expr_to_json e); ("args", `List (List.map map_expr_to_json el))])]
        |   ObjectCreate(s, el)         -> `Assoc [("objectcreate", `Assoc [("type", `String s); ("args", `List (List.map map_expr_to_json el))])]
        |        Delete(e)                                   -> `Assoc [("delete", `Assoc [("expr", map_expr_to_json e)])]

let rec map_stmt_to_json = function
                Block(stmts)                      -> `Assoc [("block", `List (List.map (map_stmt_to_json) stmts))]
        |        Expr(expr)                                 -> `Assoc [("expr", map_expr_to_json expr)]
        |        Return(expr)                         -> `Assoc [("return", map_expr_to_json expr)]
        |        If(e, s1, s2)                          -> `Assoc [("if", `Assoc [("cond", map_expr_to_json e); ("ifbody", map_stmt_to_json s1)]); ("else", map_stmt_to_json s2)]
        |        For(e1, e2, e3, s)                   -> `Assoc [("for", `Assoc [("init", map_expr_to_json e1); ("cond", map_expr_to_json e2); ("inc", map_expr_to_json e3); ("body", map_stmt_to_json s)])]
        |        While(e, s)                           -> `Assoc [("while", `Assoc [("cond", map_expr_to_json e); ("body", map_stmt_to_json s)])]
        |         Break                                         -> `String "break"
        |         Continue                                 -> `String "continue"
        |   Local(d, s, e)                         -> `Assoc [("local", `Assoc [("datatype", `String (string_of_datatype d)); ("name", `String s); ("val", map_expr_to_json e)])]

let map_methods_to_json methods =
        `List (List.map (fun (fdecl:Ast.func_decl) ->
                `Assoc [
                        ("name", `String (string_of_fname fdecl.fname));
                        ("scope", `String (string_of_scope fdecl.scope));
                        ("returnType", `String (string_of_datatype fdecl.returnType));
                        ("formals", map_formals_to_json fdecl.formals);
                        ("body", `List (List.map (map_stmt_to_json) fdecl.body));
                ]) methods)
```

```

let cdecls_tree cdecls =
        let map_cdecl_to_json cdecl =
                `Assoc [
                        ("cname", `String cdecl.cname);
                        ("extends", `String (string_of_extends cdecl.extends));
                        ("fields", map_fields_to_json cdecl.cbody.fields);
                        ("methods", map_methods_to_json cdecl.cbody.methods);
                        ("constructors", map_methods_to_json cdecl.cbody.constructors)
                ]
        in
        `List (List.map (map_cdecl_to_json) cdecls)

let print_tree = function
        Program(includes, cdecls) ->
                `Assoc [("program",
                        `Assoc([
                                ("includes", includes_tree includes);
                                ("classes", cdecls_tree cdecls)
                        ])
                )]

(* Print SAST tree representation *)

let rec map_sexpr_to_json =
        let datatype d = [("datatype", `String (string_of_datatype d))] in
        function
                SInt_Lit(i)                     -> `Assoc [("int_lit", `Assoc ([("val", `Int
↪    i)] @ (datatype (Datatype(Int_t)))))]
        |   SBoolean_Lit(b)                 -> `Assoc [("bool_lit", `Assoc ([("val", `Bool
↪    b)] @ (datatype (Datatype(Bool_t)))))]
        |   SFloat_Lit(f)                   -> `Assoc [("float_lit", `Assoc ([("val", `Float
↪    f)]  @ (datatype (Datatype(Float_t)))))]
        |   SString_Lit(s)                  -> `Assoc [("string_lit", `Assoc ([("val",
↪    `String s)] @ (datatype (Arraytype(Char_t, 1)))))]
        |   SChar_Lit(c)                    -> `Assoc [("char_lit", `Assoc ([("val", `String
↪    (Char.escaped c))] @ (datatype (Datatype(Char_t)))))]
        |   SId(s, d)                  -> `Assoc [("id", `Assoc ([("name", `String s)] @
↪    (datatype d)))]
        |   SBinop(e1, o, e2, d)       -> `Assoc [("binop", `Assoc ([("lhs",
↪    map_sexpr_to_json e1); ("op", `String (string_of_op o)); ("rhs", map_sexpr_to_json
↪    e2)] @ (datatype d)))]
        |   SAssign(e1, e2, d)         -> `Assoc [("assign", `Assoc ([("lhs",
↪    map_sexpr_to_json e1); ("op", `String "="); ("rhs", map_sexpr_to_json e2)] @
↪    (datatype d)))]
        |   SNoexpr                         -> `Assoc [("noexpr", `Assoc (datatype
↪    (Datatype(Void_t))))]
```

```
        |   SArrayCreate(t, el, d)  -> `Assoc [("arraycreate", `Assoc ([("datatype", `String (string_of_datatype d)); ("args", `List (List.map map_sexpr_to_json el))] @ (datatype d)))]
        |   SArrayAccess(e, el, d)  -> `Assoc [("arrayaccess", `Assoc ([("array", map_sexpr_to_json e); ("args", `List (List.map map_sexpr_to_json el))] @ (datatype d)))]
        |   SObjAccess(e1, e2, d)   -> `Assoc [("objaccess", `Assoc ([("lhs", map_sexpr_to_json e1); ("op", `String "."); ("rhs", map_sexpr_to_json e2)] @ (datatype d)))]
        |   SCall(fname, el, d, i)  -> `Assoc [("call", `Assoc ([("name", `String fname); ("params", `List (List.map map_sexpr_to_json el)); ("index", `Int i) ] @ (datatype d)) )]
        |   SObjectCreate(s, el, d) -> `Assoc [("objectcreate", `Assoc ([("type", `String s); ("args", `List (List.map map_sexpr_to_json el))] @ (datatype d)))]
        |   SArrayPrimitive(el, d)  -> `Assoc [("arrayprimitive", `Assoc ([("expressions", `List(List.map map_sexpr_to_json el))] @ (datatype d)))]
        |   SUnop(op, e, d)         -> `Assoc [("Unop", `Assoc ([("op", `String (string_of_op op)); ("operand", map_sexpr_to_json e)] @ (datatype d)))]
        |   SNull                     -> `Assoc [("null", `Assoc (datatype (Datatype(Void_t))))]
        |         SDelete(e)                                  -> `Assoc [("delete", `Assoc ([("expr", map_sexpr_to_json e)] @ (datatype (Datatype(Void_t)))))]

let rec map_sstmt_to_json =
        let datatype d = [("datatype", `String (string_of_datatype d))] in
        function
                SBlock sl                                -> `Assoc [("sblock", `List (List.map (map_sstmt_to_json) sl))]
        |   SExpr(e, d)                              -> `Assoc [("sexpr", `Assoc ([("expr", map_sexpr_to_json e)] @ (datatype d)))]
        |   SReturn(e, d)                                -> `Assoc [("sreturn", `Assoc ([("return", map_sexpr_to_json e)] @ (datatype d)))]
        |   SIf (e, s1, s2)                           -> `Assoc [("sif", `Assoc [("cond", map_sexpr_to_json e); ("ifbody", map_sstmt_to_json s1)]); ("selse", map_sstmt_to_json s2)]
        |   SFor (e1, e2, e3, s)                      -> `Assoc [("sfor", `Assoc [("init", map_sexpr_to_json e1); ("cond", map_sexpr_to_json e2); ("inc", map_sexpr_to_json e3); ("body", map_sstmt_to_json s)])]
        |   SWhile (e, s)                                -> `Assoc [("swhile", `Assoc [("cond", map_sexpr_to_json e); ("body", map_sstmt_to_json s)])]
        |   SBreak                                       -> `String "sbreak"
        |   SContinue                                    -> `String "scontinue"
        |   SLocal(d, s, e)                              -> `Assoc [("slocal", `Assoc [("datatype", `String (string_of_datatype d)); ("name", `String s); ("val", map_sexpr_to_json e)])]

let string_of_func_type = function
        User -> "user" | Reserved -> "reserved"
```

```
let map_sfdecl_to_json sfdecl =
        `Assoc[("sfdecl", `Assoc[
                ("sfname", `String (string_of_fname sfdecl.sfname));
                ("sreturnType", `String (string_of_datatype sfdecl.sreturnType));
                ("sformals", map_formals_to_json sfdecl.sformals);
                ("sbody", `List (List.map (map_sstmt_to_json) sfdecl.sbody));
                ("func_type", `String(string_of_func_type sfdecl.func_type));
        ])]

let map_sfdecls_to_json sfdecls =
        `List(List.map map_sfdecl_to_json sfdecls)

let map_scdecls_to_json scdecls =
        `List(List.map (fun scdecl ->
                                        `Assoc [("scdecl",
                                                `Assoc[
                                                        ("scname", `String
    ↪   scdecl.scname);
                                                        ("sfields",
    ↪   map_fields_to_json scdecl.sfields);
                                                        ("sfuncs",
    ↪   map_sfdecls_to_json scdecl.sfuncs);
                                                ])
                                        ])
                scdecls)

let map_sprogram_to_json sprogram =
        `Assoc [("sprogram", `Assoc [
                ("classes", map_scdecls_to_json sprogram.classes);
                ("functions", map_sfdecls_to_json sprogram.functions);
                ("main", map_sfdecl_to_json sprogram.main);
                ("reserved", map_sfdecls_to_json sprogram.reserved);
        ])]

(* Print tokens *)

let string_of_token = function
                LPAREN                          -> "LPAREN"
        |         RPAREN                          -> "RPAREN"
        |         LBRACE                          -> "LBRACE"
        |         RBRACE                          -> "RBRACE"
        |         SEMI                          -> "SEMI"
        |         COMMA                          -> "COMMA"
        |         PLUS                          -> "PLUS"
        |         MINUS                          -> "MINUS"
        |         TIMES                          -> "TIMES"
        |         DIVIDE                          -> "DIVIDE"
        |         ASSIGN                          -> "ASSIGN"
        |         EQ                                  -> "EQ"
        |         NEQ                                        -> "NEQ"
        |         LT                                        -> "LT"
        |         LEQ                                        -> "LEQ"
        |         GT                                        -> "GT"
        |         GEQ                                        -> "GEQ"
        |         AND                                        -> "AND"
        |         OR                                        -> "OR"
        |         NOT                                        -> "NOT"
        |         DOT                                        -> "DOT"
        |         LBRACKET                        -> "LBRACKET"
        |         RBRACKET                        -> "RBRACKET"
        |         BAR                                        -> "BAR"
        |         IF                                        -> "IF"
        |         ELSE                                -> "ELSE"
        |         FOR                                        -> "FOR"
        |         WHILE                                -> "WHILE"
        |         RETURN                                -> "RETURN"
        |         INT                                        -> "INT"
        |         FLOAT                                -> "FLOAT"
        |         BOOL                                -> "BOOL"
        |         CHAR                                -> "CHAR"
        |         VOID                                -> "VOID"
        |         NULL                                -> "NULL"
        |         TRUE                                -> "TRUE"
        |         FALSE                                -> "FALSE"
        |         CLASS                                -> "CLASS"
        |         CONSTRUCTOR                        -> "CONSTRUCTOR"
        |         PUBLIC                                -> "PUBLIC"
        |         PRIVATE                                -> "PRIVATE"
        |         EXTENDS                                -> "EXTENDS"
        |         INCLUDE                                -> "INCLUDE"
        |         THIS                                -> "THIS"
        |         BREAK                                -> "BREAK"
        |         CONTINUE                        -> "CONTINUE"
        |   NEW                                 -> "NEW"
        |         INT_LITERAL(i)                -> "INT_LITERAL(" ^ string_of_int i ^ ")"
        |         FLOAT_LITERAL(f)        -> "FLOAT_LITERAL(" ^ string_of_float f ^ ")"
        |         CHAR_LITERAL(c)                -> "CHAR_LITERAL(" ^ Char.escaped c ^
↪   ")"
        |         STRING_LITERAL(s)        -> "STRING_LITERAL(" ^ s ^ ")"
        |         ID(s)                                -> "ID(" ^ s ^ ")"
        |         DELETE                                 -> "DELETE"
        |         MODULO                                 -> "MODULO"
        |          EOF                                        -> "EOF"

let string_of_token_no_id = function
                LPAREN                                -> "LPAREN"
        |         RPAREN                                -> "RPAREN"
        |         LBRACE                                -> "LBRACE"
        |       RBRACE                          -> "RBRACE"
        |       SEMI                          -> "SEMI"
        |       COMMA                          -> "COMMA"
        |       PLUS                          -> "PLUS"
        |       MINUS                          -> "MINUS"
        |       TIMES                          -> "TIMES"
        |       DIVIDE                          -> "DIVIDE"
        |       ASSIGN                          -> "ASSIGN"
        |       EQ                                  -> "EQ"
        |       NEQ                                  -> "NEQ"
        |       LT                                  -> "LT"
        |       LEQ                                  -> "LEQ"
        |       GT                                  -> "GT"
        |       GEQ                                  -> "GEQ"
        |       AND                                  -> "AND"
        |       OR                                  -> "OR"
        |       NOT                                  -> "NOT"
        |       DOT                                  -> "DOT"
        |       LBRACKET                    -> "LBRACKET"
        |       RBRACKET                    -> "RBRACKET"
        |       BAR                                  -> "BAR"
        |       IF                                  -> "IF"
        |       ELSE                          -> "ELSE"
        |       FOR                                  -> "FOR"
        |       WHILE                          -> "WHILE"
        |       RETURN                          -> "RETURN"
        |       INT                                  -> "INT"
        |       FLOAT                          -> "FLOAT"
        |       BOOL                          -> "BOOL"
        |       CHAR                          -> "CHAR"
        |       VOID                          -> "VOID"
        |       NULL                          -> "NULL"
        |       TRUE                          -> "TRUE"
        |       FALSE                          -> "FALSE"
        |       CLASS                          -> "CLASS"
        |       CONSTRUCTOR                     -> "CONSTRUCTOR"
        |       PUBLIC                          -> "PUBLIC"
        |       PRIVATE                          -> "PRIVATE"
        |       EXTENDS                          -> "EXTENDS"
        |       INCLUDE                          -> "INCLUDE"
        |       THIS                          -> "THIS"
        |       BREAK                          -> "BREAK"
        |       CONTINUE                    -> "CONTINUE"
        |   NEW                                 -> "NEW"
        |       INT_LITERAL(i)                  -> "INT_LITERAL"
        |       FLOAT_LITERAL(f)          -> "FLOAT_LITERAL"
        |       CHAR_LITERAL(c)                -> "CHAR_LITERAL"
        |       STRING_LITERAL(s)          -> "STRING_LITERAL"
        |       ID(s)                          -> "ID"
        |          DELETE                                 -> "DELETE"
        |          MODULO                                 -> "MODULO"
        |           EOF                                        -> "EOF"

let token_list_to_string_endl token_list =
  let rec helper last_line_number = function
                (token, curr)::tail ->
                let line = curr.lineno in
                (if line != last_line_number then "\n" ^ string_of_int line ^ ". " else "
↪  ") ^
                string_of_token token ^ helper line tail
        |          [] -> "\n"
  in helper 0 token_list

let token_list_to_string token_list =
  let rec helper = function
                (token, line)::tail ->
                string_of_token_no_id token ^ " " ^ helper tail
        |          [] -> "\n"
  in helper token_list
```

_tags

```
<filepath.*> or <**/*.native> or <**/*.byte>: package(unix)
```

Demo

Demo_Animals.dice

```
include("stdlib");

class Animal{
        public int weight;
        constructor(){
                this.weight = 0;
        }

        constructor(int w){
                this.weight = w;
        }

        public void move(){
                print("Animals move in many ways");
        }
}

class Bird extends Animal {
        public int maxFlyingHeight;

        constructor(){
                this.weight = 0;
                this.maxFlyingHeight = 0;
        }

        constructor(int w, int h){
                this.weight = w;
                this.maxFlyingHeight = h;
        }

        public void move(){
                print("Birds fly!");
        }

}

class Dog extends Animal {
        public int speed;

        constructor(){
                this.weight = 0;
                this.speed = 0;
        }

        constructor(int w, int s){
                this.weight = w;
                this.speed = s;
        }

        public void move(){
                print("Dogs run!");
        }
}

class Stephen extends Animal {
        private bool isDone;

        constructor() {
                this.isDone = true;
        }

        constructor(bool isDone) {
                this.isDone = isDone;
        }

        public void move() {
                if(not this.isDone) {
                        print("I am a techer!");
                } else {
                        print("Also my favorite number is 42");
                }
                this.isDone = true;
        }

}

class Snake extends Animal {
        public int slitherSpeed;

        constructor(){
                this.weight = 0;
                this.slitherSpeed = 0;
        }

        constructor(int w, int s){
                this.weight = w;
                this.slitherSpeed = s;
        }

        public void move(){
                print("Snakes slither!");
        }
}

```

```
class Marnie extends Dog {
        public int cuteness;

        constructor(){
                this.weight = 0;
                this.speed = 0;
        }

        constructor(int w, int s){
                this.weight = w;
                this.speed = s;
        }

        constructor(int w, int s, int c){
                this.weight = w;
                this.speed = s;
                this.cuteness = c;
        }

        public void move(){
                class File a = new File("Demo/marnie1.txt", true);
        char[] buf = a.readfile(4500);
        a.closefile();
        print(buf);
        print("\n");
        }
}

class test {
        private bool isDone;
        public void main(char[][] args) {
                this.logo();
                this.isDone = false;

                bool keepGoing = true;
                while(keepGoing){
                        this.animalsToChoose();
                        char[] buf = input();
                        print("\n");

                        int choice = this.getInt(buf[0]);

                        if(choice==5)
                                break;
                        else
                                this.printMovement(choice);

                        print("\n");
                }

        class Marnie a = new Marnie();
        a.move();
        }

        public int getInt(char num){
                if(num=='1')
                        return 1;
                else if(num=='2')
                        return 2;
                else if(num=='3')
                        return 3;
                else if(num=='4')
                        return 4;
                else if(num=='5')
                        return 5;

                return 0;

        }

        public void printMovement(int choice){

                class Animal b = new Bird();
                class Animal d = new Dog();
                class Animal s = new Snake();
                class Animal stephen = new Stephen(this.isDone);

                if(choice == 1)
                        b.move();
                else if(choice == 2)
                        d.move();
                else if(choice == 3)
                        s.move();
                else if(choice == 4) {
                        stephen.move();
                        this.isDone = true;
                }
                else
                        print("Animal not selected!\n");

                print("\n");
        }

        public void animalsToChoose(){
                print("1-Bird\n2-Dog\n3-Snake\n4-Stephen\n5-Exit\nPlease choose an animal
↪  or exit(by selecting a number):");

        }

        public void logo(){
                class File a = new File("Demo/logo.txt", true);
        char[] buf = a.readfile(4500);
        a.closefile();
        print(buf);

        int i;
                for(i=0;i<3;i=i+1){
                print("\n");
                }

                print("Welcome to the animal farm!\n\n");
        }
}
```

logo.txt

```
 .----------------.  .----------------.  .----------------.  .----------------.
| .--------------. || .--------------. || .--------------. || .--------------. |
| |  ________    | || |     _____    | || |     ______   | || |  _________   | |
| | |_   ___ ‘.  | || |    |_   _|   | || |   .’ ___  |  | || | |_   ___  |  | |
| |   | |   ‘. \ | || |      | |     | || |  / .’   \_|  | || |   | |_  \_|  | |
| |   | |    | | | || |      | |     | || |  | |         | || |   |  _|  _   | |
| |  _| |___.’ / | || |     _| |_    | || |  \ ‘.___.’\  | || |  _| |___/ |  | |
| | |________.’  | || |    |_____|   | || |   ‘._____.’  | || | |_________|  | |
| |              | || |              | || |              | || |              | |
| ’--------------’ || ’--------------’ || ’--------------’ || ’--------------’ |
 ’----------------’  ’----------------’  ’----------------’  ’----------------’
 ..................................~~~~~~~~~~~...................................
...............................~~~~~.......~~~~~................................
............................~~~~~.............~~~~~.............................
.........................~~~~~:.................,~~~~~..........................
......................:~~~~:.....~~~~~~~~~~........,~~~~~.......................
...................,~~~~~.......~~~:.........~~.......:~~~~~....................
.................~~~~~..........~~~~.......:~~~..........~~~~~..................
..............~~~~~..............~~~~~~~~~~~~~..............~~~~~...............
...........~~~~~...............................................~~~~~............
........~~~~~:......~~~~~:..........,~~~~~,..........,~~~~~,.....,~~~~~.........
......~~~~~.....~~~~~~~~~~:......~~~~~~~~~~.......~~~~~~~~~~........:~~~~.......
.....~~~.......~~~..........~~..~~~..........~~..~~~..........~~.......~~~......
....~~~~~,.....~~~~~,....:~~~~..~~~~~.....~~~~~..~~~~:.....~~~~~......~~~~~.....
....~~.~~~~~.....~~~~~~~~~~~,.....~~~~~~~~~~~......~~~~~~~~~~~.....~~~~~.~~.....
...~~~....~~~~~.................................................~~~~~....~~~....
...~~~.......~~~~~.................~~~~~~~~..................~~~~~,......~~~....
...~~~.........:~~~~~...........,~~~~~~~~~~...............~~~~~:.........~~~....
...~~~............:~~~~:........~~,..........~~........:~~~~~............~~~....
...~~~...............~~~~~,.....~~~~~~~~~~~~~~~.....,~~~~~......~~~~~....~~~....
...~~~..................~~~~~.....:~~~~~~~~~~.....~~~~~.......:~~~.......~~~....
...~~~.....................~~~~~................~~~~.........:~~.........~~~....
...~~~........................~~~~~.........~:...............~~:....~~,..~~~....
...~~~..........................,~~~~.....~~~~..............:~~.....~~...~~~....
...~~~.............................:~.....::................~~~....~~~...~~~....
...~~~......................................................:~~..~~~~....~~~....
...~~~.......................................................~~~~~~:.....~~~....
...~~~................................~~~......................,,........~~~....
...~~~............~~~~~:..............~~~...............~~~~~............~~~....
...~~~...........~~~.~~~..............~~~.............~~~~.:.............~~~....
...~~~...........~~,..................~~~............~~~.................~~~....
...~~~...........~~~..................~~~...........:~~.....~~...........~~~....
...~~~...........~~~.....~~...........~~~...........~~~....~~~...........~~~....
...~~~............~~~....~~...........~~~...........~~,...:~~............~~~....
...~~~.............~~~...~~...........~~~...........~~~..~~~,............~~~....
...~~~..............~~~~~~~...........~~~............~~~~~~..............~~~....
...~~~................:~~.............~~~.............~~~................~~~....
```

```
....~~................................~~~......,~~~~.....................~~.....
....~~~...............................~~~.....~~~~:~....................~~~.....
.....~~~..............................~~~....~~~.......................~~~......
......~~~~~...........................~~~...~~~....:~,..............:~~~~.......
........~~~~~:........................~~~...~~.....~~:...........,~~~~~.........
...........~~~~~......................~~~..:~~....~~~..........~~~~~............
..............~~~~~...................~~~..,~~...~~~........~~~~~...............
.................~~~~~................~~~...~~~~~~~......~~~~~..................
...................,~~~~~.............~~~....~~~~.....:~~~~~....................
......................:~~~~:..........~~~..........,~~~~~.......................
.........................~~~~~:.......~~~.......,~~~~~..........................
............................~~~~~.....~~~.....~~~~~.............................
...............................~~~~~..~~~..~~~~~................................
..................................~~~~~~~~~~~...................................
......................................:~:.......................................
```

marnie1.txt

```




                             7  777777777 7
                               7777777777777
                    7        7 777777777I77 7
                   777777777777I77++?I?7777I7
                7 77I77?II77777777?????II?I777777777 77 7
                777I77??II??????I???IIII?I?77777777I7777
               77II7I??IIIIII??I???II????IIII7777777777 77
              7I 777IIIIIIIIIIIIIII??I??IIIIII7IIII777777I7  7
             77777I777IIIIIIIII??I?????II?I?IIIII7777777777I77
             7I77777III??I?III???+??II????I?I?I777IIIIIII+777777
              II 77I??????I77?I??++?I?IIIII??IIIIIIIIIII?I+I77777
          7 77I777IIIIII???IIIIIIIIIIIIIIIIIIIIIIIIIIIII???I7777
            7I 7I7I7III?IIIII????III????IIIIII7IIIIIII???+++?77I7
            7? 7??I?++??77?+~~~:::~+++++???I?II?IIIIIIIII?==+?7I777
           77II77?+===?7I+~~~:,.,,,,:~=+++??+???+++??I7II?+==+77I7
          777I?77?+??77??++=+,:~~~=,::==+++++==~=~=+?????I??=+?7?7
          7 777II77777II??+?+:,:,,:,:=========~~~~==+?II=??++=+77I
          77777777777IIIIIII?:~=~~=~===:~~~=~~:~~===+I?I=+==~=?77I 7
          777777777777I7777IIIII??++??+++====:,,:~==++??I+++~==+I I777
          7I7777 77777777II?I???+?+~~~+???+?=,,,,,:++++++====~=++77I77
          777777777I77II???++++??=::,:~:~+??+,,=,,,:=+++?==~==+++77I77
          7I77777II7I???+==?I??I?=::,,,.,:???=,,,,,:~=+?+=~=+=++77+77
          7I7777III?????===?++??++~:,,,,,=+??I~,,,:::====+=++??+7+77
          77+777II???+=:~:+~=~~~=~,:.,:,:~~+=+?+~+++~~~??++???+=7?7
          77I?77??+=~:,,,,,:.,,,,,,,,,=:~=?=~=?+?II???+?++?+??++?7+7
          7 77?+77+=~::~::~::.,...,,~:,:~~~~~~~~=+?II??++++?++??77I
             77I=77++===~~~==.?:..:~~:.,=,,:,~~~==??????+~=~+++77I7
               77+77??++++++?~,:::,===~:::,.,,::=+=?++???+77I?77+7
               77I777=?I??????+,:~+=++=~~~~,,,,:~==+++???77777I=I7
                77?= 7I???+???++~:,.?+==~~~,:,~:===+++++7 +?III777
                7777?I77+++=??+=~~~:====~~=~::~~==++++?77+77
                77 77?? 7=++??+=~~::,==~~~~=~~~~==++777 +I77
                    77I=777~=?=~~~~::=~~~~~~++??7777+=+I777
                     777?=77777?~~~~~~=====77~~~==??I77
                       777?+==?7777:::~~~~77IIII777777
                        7 7777I??==I7    7I=I7
                         7 7777777I??????????777
                                7   7777


```

marnie2.txt

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++++I++?+++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++I++++++?+?I++???II??I????+++++++++++++++++++++++++
++++++++++++++++++++++++++++?+I?+++??IIII7IIIIII7I7II7+++?++++++++++++++++++++++
++++++++++++++++++++++I+++++I+??++??7IIIIIIIIIII7I7I77??I??+++++++++++++++++++++
+++++++++++++++++++++++??+++I??I???+IIIIII?IIIIII7I777IIIIII?I?+++++++++++++++++
+++++++++++++++++++++++?+??++I?II??III?I????IIIIIIII777I7IIII?++++++++++++++++++
+++++++++++++++++++++++++?+I??I?I?II?+??I???IIIIIIIII77IIIIIIII??+++++++++++++++
+++++++++++++++++++++??+++???I?III????+??I???IIIII7IIIIIIIIIIIIII?++++++++++++++
++++++++++++++++++++++?????????????????+??I??IIII?IIIIIIII77IIIIIII+++++++++++++
++++++++++++++++++?IIIIII?++????+++??????????IIII?II??IIIIIIIIIIIIII?+++++++++++
+++++++++++++++++++IIII?++?++??+====++????????III???III7IIIIII7III7I?+++++++++++
+++++++++++++++++++?III??+??+??+==~=~~=+?+?????II??III???IIIII????I77+++++++++++
++++++++++++++++++?+=~~=~~=?I+=+=:,,,,,=+++????????III777IIIIIIIIIII7?++++++++++
++++++++++++++++++++=~::++?+~=:,,~~:,:=~+??+????????????IIIIIII?I???III?++++++++
++++++++++++++?I??+=I+=?I~~?:,~=,,,,,,~~??++==+?+++====++?I????+++????+??+++++++
+++++++++++++????II??II7IIIII??+~+~~+I==+=~~~+++??++=+===???++?????????+?+++++++
++++++++++?????+???IIIIIIIIIIII?IIIIIIIII?==~??+++==:~~=+=?II~:~~=+++??+?+++++++
++++++++++?I????I77IIIIIIIIIIIIIII+=:~::,+?==?++?~~~:~??+I+=+?+==++?+???++++++++
++++++?III????I777IIIII????IIIIII++~,.~:::,?~~+=,,,,,:~??=??++?+==++?+I????+??++
+++++???II???I7IIII?IIII??IIIIII?+=~,,:,.:,=++==,,,,,,,?+?+++~~~=~+?+I+I???????+
+++???III????IIIII?+=++IIIIIIIII?=+::,,,,,,:~+++:,I,:,,:I+?+?===~==+==?I????????
+???IIIIII??IIII?++++++I??I??++?~=+=,::=,+,~+++:,,:,,,,:~+++===+:+==::+I????????
++?????III??III+???==+~+:+~.:,=,,~~~~=:~.=::~~~=,,,,,,,~===+=:=++==:::=I????????
???????III????+???++=~:::,,,,.I,,:~,=:~:?++:=~+++.,,,,:~~~~~~=~:=~:::,~?????????
?????IIIII????II+++=~=:::::~:,?,,:++=:=:?I=+~~==++~:::,::~::=:~=::~::=:+I???????
???I77IIII???+???=~~~~=~~==??:=,,,=?+~:+++=+=~===++~:::::~,:~::~::~=:~=+I??????I
??7777III????++??+=~~~~+?IIII?~~:?,===~,~I+~=~=+==+=~~::::~:~~:~:~~=:==+?IIIIIII
?I7777IIII??++?+?+=~~=??IIIIII?~:,,77:~::~:~+~+~=+~=+=~::==,=~~~::~=~=~=I??IIIII
?7777IIIIII?+?+++?+++++I???IIIII=::,=~~~~~:,:=+~=++=~=~=~==~::~:::~=:+~~?IIIIIII
?777IIIIIII???+=+I?===?I?????I??I?~==~~~~:~:,::====+=~=~=~=~~~~~~:~+:+~~+IIIIIII
?77IIIIIIII??+++=+?+????????II?+I?+~=~~:~~~::::~==+=~==~~~=:=~~:~:==~+=+~?IIIIII
I7IIIIIIIII??+++==~?+?????+III?++=?=+~~:~~~~:~::~++==~=~====~~~::~=::::~:+IIIIII
7IIIIIIIIII??++++===?I??==+I?II?+++++~~~~~~:::~~~========~=+:=~:~~~,,::=+I+IIIII
7IIIIIIIIIII??++++===I?=~~=++++?++++++~~~~~~~:~==~====+++=====~~:~~:~+IIIIIIIIII
7IIIIIIIIIIII?+++++=+?++==~====++++===:::~~~~~~~=~============:~~=~?IIIIIIIIIIII
7IIIIIIIIIIII???+??+=~=++==~~+?I?+++==~~~~~~~=~=+===++++===??IIIIIIIIIIIIIIIIIII
7IIIIIIIIIIII????+?++~=+?+?=~~==++++++==~=~~====~=~=~=+=?IIIIIIIIIIIIIIIIIIIIIII
7IIIIIIIIIIIII?++I?++=+?===~~+=~I???+++=+=+===~=+?+?I?IIIIIIIIIIIIIIIIIIIIIIIIII
IIIIIIIIIIIIII?+?I?+++III?==?+~=::+?+==+++~===++???IIIIIIIIIIIIIIIIIIIIIIIIIIIII
IIIIIIIIII?III????I=?I++==++~=~==?=+~==?=~=~~=+??IIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
III7IIIIIII?III?+I?=I???I+=+=====~=~::::~~~++++???IIIIIIIIIIIIIIIIIIIIIIIIIIIIII
?III7IIIIII?III??+7?I7?+++++==~==~~~~~~~=~~=?++?????IIIIIIIIIIIIIIIIIIIIIIIIIIII
??IIIIIIIIII?III?+==I?7I+++I+==+=====~=~=+==+++I?????????????I??IIIIIIIIIIIIIIII
??IIIIIIIIIII?II??==?II+??I?+++??++++=+===+?++?????????????????????I?IIIIIIIIIII
```

```
???II7IIIIIIII?I???~?I???+??+=+???+++=+++??=++?II????????????????????????IIIIIII
?+?I?IIIIIIII???I???=?III??II?+???I+++++?+?=++?III?????????????????????????IIIII
?+++?I???IIIII????I?III?III+??+=+++++++++?+=++??IIII???????????????????????????I
??++?I??+????++??I?II????+??+++=+++++++++?=+++????III???????????????????????????
```

Othello.dice

```
include("stdlib");


class Player {


    public class LocationObj placeTile(bool retry)   {
        return new LocationObj();
    }

    public void setResult(class LocationObj move) {
    }
}

(*********************************************************************)

class HumanPlayer extends Player {
    private class Board board;
    public int myPieceType;

    constructor()   {
        this.board = new Board();
        this.myPieceType = 2;
        class Board b = this.board;
        b.initializeBoard();
    }

    public class LocationObj placeTile(bool retry)   {
        if (this.myPieceType == 2)
            this.myPieceType = 1;
        if (retry){
            print("Last move was invalid. Retry.\n"); }
        print("It's your turn\n");
        class Board b = this.board;
        b.printBoard();

        print("Please enter your move\n");
        class LocationObj move = this.getLocationObjChoice();
        int temp = this.myPieceType;
        b.setPlayerMove(move, temp);
        return move;
    }

   public void setResult(class LocationObj move) {
        int temp = this.myPieceType;
        if (temp == 1) {
            bool one = (move.getHorizontal() == 3);
            bool two = (move.getHorizontal() == 4);
            bool three = (move.getVertical() == 3);
            bool four = (move.getVertical() == 4);
            bool five = ((one or two ) and (three or four));
             if(not five){
                this.myPieceType = 0;
                }
        }

        int opponentPieceType;
        int temp2 = this.myPieceType;
        if (temp2 == 0){
            opponentPieceType = 1; }
        else {
            opponentPieceType = 0;}

        class Board b = this.board;
        b.setPlayerMove(move, opponentPieceType);
    }

    private class LocationObj getLocationObjChoice(){
        char[] userInput;
        class String uInput;
        class Board b = new Board();
        class LocationObj move = null;
        int temp = this.myPieceType;
        while (not (b.isValid(move, temp))) {
            print("You are " , this.myPieceType , ". What is the x location of your next
↪   move?");
            userInput = input();
            uInput = new String(userInput);
            int x = uInput.toInteger();
            print("You are " , this.myPieceType , ". What is the y location of your next
↪   move?");
            userInput = input();
            uInput = new String(userInput);
            int y = uInput.toInteger();
            move = new LocationObj(x - 1, y - 1);
            bool one = b.isValid(move,temp);
            if (not one){
                print("invalid move, try again.\n"); }
        }
        return move;
    }
}



(*********************************************************************)
```

```


class ComputerPlayer extends Player {

    private class Board board;
    public int myType;

    constructor(){
        this.board = new Board();
        class Board board = this.board;
        board.initializeBoard();
        this.myType = 2;
    }

    public class LocationObj placeTile(bool retry) {
        class Board board = this.board;
        if(this.myType == 2){
            this.myType = 1;
        }


        class LocationObj move = this.getBestMove(this.myType);
        int temp = this.myType;
        board.setPlayerMove(move, temp);
        return move;
    }

    public void setResult(class LocationObj move) {
        class Board board = this.board;
        if(this.myType == 2 and
                (((move.getHorizontal() == 3 or move.getHorizontal() == 4) == false) and
                        (move.getVertical() == 3 or move.getVertical() == 4))){
            this.myType = 0;
        }

        int opponent;
        if(this.myType == 1)
            opponent = 0;
        else
            opponent = 1;

        board.setPlayerMove(move, opponent);
    }

    private int[] createPointArray(){
        int[] points = new int[64];
        int i;
        int j;
        for(i = 0; i < 8; i = i + 1)
        {
            for(j = 0; j < 8; j = j + 1)
            {
                points[j*8+i] = 1;
            }
        }
(*
        //[4*8+2*8+3*8+2*8+2*8+3*8+2*8+4]
        //[2*8+3*8+1*8+1*8+1*8+1*8+3*8+2]
        //[3*8+1*8+2*8+1*8+1*8+2*8+1*8+3]
        //[2*8+1*8+1*8+0*8+0*8+1*8+1*8+2]
        //[2*8+1*8+1*8+0*8+0*8+1*8+1*8+2]
        //[3*8+1*8+2*8+1*8+1*8+2*8+1*8+3]
        //[2*8+3*8+1*8+1*8+1*8+1*8+3*8+2]
        //[4*8+2*8+3*8+2*8+2*8+3*8+2*8+4]
*)
        points[(0*8)+0] = points[(7*8)+0] = points[(0*8)+7] = points[(7*8)+7] = 4;

        points[2] = points[2*8] = points[2*8+7] = points[7*8+2] = 3;
        points[1*8+1] = points[6*8+6] = points[1*8+6] = points[6*8+1] = 3;
        points[5] = points[5*8+0] = points[5*8+7] = points[7*8+5] = 3;

        points[0*8+1] = points[0*8+3] = points[0*8+4] = points[0*8+6] = 2;
        points[1*8+0] = points[0*8+7] = 2;
        points[2*8+2] = points[2*8+5] = 2;
        points[3*8+0] = points[3*8+7] = 2;
        points[4*8+0] = points[4*8+7] = 2;
        points[5*8+2] = points[5*8+5] = 2;
        points[6*8+0] = points[6*8+7] = 2;
        points[7*8+1] = points[7*8+3] = points[7*8+4] = points[7*8+6] = 2;

        points[4*8+4] = points[4*8+5] = points[5*8+4] = points[5*8+5] = 0;

        return points;
    }

    private class LocationObj getBestMove(int turn){
        class LocationObj best = null;
        int currentValue = -2147483647;
        int[] pointArray = createPointArray();
        class Board board = this.board;
        int i;
        int j;
        for(i = 0; i < 8; i = i + 1) {
            for(j = 0; j < 8; j = j + 1){
                class LocationObj l = new LocationObj(i,j);
                if(board.isValid(l, turn) and pointArray[i*8+j] > currentValue){
                    currentValue = pointArray[i*8+j];
                    best = l;
                }
            }
        }
        return best;
    }

}



(**
 * Implementation of Location that has horizontal and verticale coordinates
 *
 * @author David Watkins
 * @UNI djw2146
 *)
(**
 * Implementation of Location that has horizontal and verticale coordinates
 *
 * @author David Watkins
 * @UNI djw2146
 *)



class LocationObj {

public void main(char[][] args) {}
 (* ================================== *)

    private int horizontal;
    private int vertical;

    (**
     * Creates a LocationObj with horizontal and vertical coordinates
     *
     * @param horizontal x coordinate
     * @param vertical y coordinate
     *)
    constructor(int horizontal, int vertical){
        this.horizontal = horizontal;
        this.vertical = vertical;
    }

    (*
     * @Return Horizontal coordinate
     *)
    (* @Override *) (* ================================== *)
    public int getHorizontal() {
        return this.horizontal;
    }

    (*
     * @Return Vertical coordinate
     *)
    (* @Override *) (* ================================== *)
    public int getVertical() {
        return this.vertical;
    }
}


(**
 * Maintains and operates on the board.
 * Has methods for checking if particular moves are valid and initializing the board.
 *
 * @author David Watkins
 * UNI djw2146
 *)



class Board {
    private int[] board;

    (**
     * Initializes a new board with size Game.SIZE x Game.SIZE
     *)

    constructor(){
        this.board = new int[64];
        this.initializeBoard();
    }

    (**
     * Initializes the board to have the center most four pieces in the correct formation
     *
     * @param type The type of player, as the user of the Board could differ
     *)
    public void initializeBoard(){
        int i;
        int j;
        int[] board = this.board;
        for (i = 0 ; i < 8 ; i = i + 1) {
            for (j = 0 ; j < 8 ; j = j + 1) {
                board[i+j]=2;
            }
        }

        board[(3*8)+4] = 1;
        board[(4*8)+3] = 1;
        board[(3*8)+3] = 0;
        board[(4*8)+4] = 0;
    }

    (**
     * Prints out a formatted version of the board to the console.
     *)
    public void printBoard() {
        print("-----------------------------------------------------------------");
        int i;
        int j;
        int[] board = this.board;

        for(i = 0; i < 7; i=i+1){
        (* Prints out each line individually *)
            for( j = 0; j < 7; j = j + 1){

            (*//Prints out each section of a line of the board*)
                if(board[(j*8+i)] == 0) {  (*//SELF player*)
                    print("|(WHITE)");
              }  else if(board[(8*j)+i] == 1) { (*//OPPONENT player*)
                    print("|(BLACK)");
              }  else { (*//No piece in location*)
                    print("|(", (j+1), ",  ", (i+1), ")");
              }
            }
            print("|"); (*//Finishes the line*)
            print("\n");
        }
        print("-----------------------------------------------------------------");
        print("\n");
    }

    (**
     * Checks all possible indices of the board to determine whether or not a winner has
↪   been determined
     * Counts the number of each piece and returns the type of winner
     *
     * @return The winner if there is a winner, null if no winner is determined yet
     *)
    public int thereIsWinner(){
        int[] temp = this.totalCount();

        if(temp[0] + temp[1] == 64){ (*//If the total number of pieces equals the entire
↪   board *)
            return this.whoHasMore();
        } else { (*//If no winner yet *)
            return -1;
        }
        return 9;
    }

    (**
     * A method for determining which player has more pieces on the board.
     *
     * @return Which PIece has more positins in the board
     *)
    public int whoHasMore(){
        int[] temp = this.totalCount();

        if(temp[0] > temp[1]) {
            return  0;
      }  else if(temp[1] > temp[0]) {
            return 1;
      }  else { (* //If there is a tie *)
            return 2;
      }
        (* default return *)
        return 9;
    }

    (**
     * Private helper metod for determining the current count of a particular type of
↪   player
     *
     * @return A size two vector containing whiteCount is [0] and blackCount in [1]
     *)
    private int[] totalCount(){
        int whiteCount = 0;
        int blackCount = 0;
        int i;
        int j;
        int[] board = this.board;
        for(i = 0; i < 8; i = i + 1) {
            for(j = 0; j < 8; j = j + 1) {
                if(board[(i*8)+j] == 0) {
                    whiteCount = whiteCount + 1;
               } else if(board[(i*8)+j] == 1) {
                    blackCount = blackCount + 1;
               }
            }
        }
        int[] temp = new int[2];
        temp[0] = blackCount;
        temp[1] = whiteCount;
        return temp;
    }

    (**
     * Will check to see if the particular direction is a valid move
     * If bool updateBoard is true, will also update any pieces appropriately modified by
↪   a particular player move.
     * Is only privately used by isValid and setPlayerMove to make sure a move is valid
     * Returns 0 if the direction is invalid
     *
     * @param move The player's new move
     * @param incx The coefficient of x
     * @param incy The coefficient of y
     * @param player The current player playing the new move
     * @param updateBoard Whether or not to update pieces
     * @return Whether the move is valid or not
     *)


    private bool checkDir(class LocationObj move, int incx, int incy, int player , bool
↪   updateBoard)  {

        int[] board = this.board;
        int opponent; (*//The opposite color of player*)
        int x = move.getHorizontal();
        int y = move.getVertical();

        (*//Current player's move *)
        if (player == 1) {
            opponent=0;
       } else {
            opponent=1;
        }
        (* /Modify the position by one *)
        int dist = 0;
        x = x + incx;
        y = y + incy;

        (*//While x and y are in bounds and the current position is an opponent piece *)
        while ((x < 8) and (x >= 0) and (y < 8) and (y >= 0) and (board[(x*8)+y] ==
↪   opponent)) {
            x = x + incx;
            y = y + incy;
            dist = dist + 1;
        }

        (* //If x and y are still in bounds and the final position is a player piece,
↪   will *)
```

```
        if ((dist != 0) and (x < 8) and (x >= 0) and (y < 8) and (y >= 0) and
↪   (board[(x*8)+y]==player)) {
            if (updateBoard) { (* //Will update the board if true *)
                int j;
                for (j = 1 ; j <= dist ; j = j+1) {
                   x = x - incx; (*//Decrease x by one*)
                   y = y - incy; (*//Decrease y by one*)
                   class LocationObj l = new LocationObj(x,y);
                   this.setLoc(l,player); (*//Update location to player piece *)
                }
           return true; (* //The current distance from the initial position *)
            } else {
               return false; (*//Not a valid direction *)
           }
        }
        return false; (* default return *)
    }

    (**
     * Will set the location defined by move to the type player
     *
     * @param move The position to be modified
     * @param player The player's type
     *)

    private void setLoc(class LocationObj move, int player){

        this.board[(move.getHorizontal()*8)+move.getVertical()] = player;
    }

    (**
     * Sets the location of a particular player (c)'s move. Returns false
     * if move is invalid.
     *
     * @param move Location object containing horizontal and vertical
     * coordinates
     * @param c The type of player currently placing tile
     * @return False if invalid move
     *)


    public bool setPlayerMove(class LocationObj move, int player) {
        bool valid = false;
        int yinc = 0;
        int xinc = 0;
        int[] board = this.board;

        (* //If move is null or move space is taken up *)
        if(move == null or board[(move.getHorizontal()*8)+move.getVertical()] != 2) {
            return false;
        }
        int i;
        for(i = 0; i < 8; i = i + 1){ (*//For the length of potential neighbors*)
            (*//Linear Directions *)
           if(i == 0){xinc = 1; yinc = 0;} (*//E *)
           else if(i == 1){xinc = -1; yinc = 0;} (* //W *)
           else if(i == 2){xinc = 0; yinc = 1;} (* //S *)
           else if(i == 3){xinc = 0; yinc = -1;} (* //N *)

           (*//Diagonals*)
           else if(i == 4){xinc = 1; yinc = 1;}(*//SE*)
           else if(i == 5){xinc = -1; yinc = 1;}(*//SW*)
           else if(i == 6){xinc = 1; yinc = -1;}(*//NE*)
           else if(i == 7){xinc = -1; yinc = -1;}(*//NW*)

           (*//Change all potential old markers*)
           if(this.checkDir(move, xinc, yinc, player, true)) {
               valid = true;
            }
        }

        if (valid) { (*//Valid move*)
            this.setLoc(move, player);
            return true;
        }
        return false; (*//Invalid move*)
    }

    (**
     * Checks all possible directions to determine whether or not a move is valid
     * Uses the private method checkDir to make sure that the move is valid
     *
     * @param move The current move to be checked
     * @param kind The current type of player's move
     * @return Whether or not the move was valid
     *)
    public bool isValid(class LocationObj move, int kind) {
        int yinc = 0;
        int xinc = 0;
        int[] board = this.board;
        (*//If move is null, within the boundaries of the array, or move space is taken
↪   up *)
        if(move == null or
                ((move.getHorizontal() > 0 and move.getHorizontal()< 8
                and move.getVertical() > 0 and move.getVertical() < 8) == false) or

                this.board[(move.getHorizontal()*8)+move.getVertical()] != 2) {
            return false;
        }
        int i;
        for(i = 0; i < 8; i = i +1){ (*//For the length of potential neighbors*)
            (*//Linear Directions*)
           if(i == 0){xinc = 1; yinc = 0;}(*//E*)
           else if(i == 1){xinc = -1; yinc = 0;}(*//W*)
           else if(i == 2){xinc = 0; yinc = 1;}(*//S*)
           else if(i == 3){xinc = 0; yinc = -1;}(*//N*)

           (*//Diagonals*)
           else if(i == 4){xinc = 1; yinc = 1;}(*//SE*)
           else if(i == 5){xinc = -1; yinc = 1;}(*//SW*)
           else if(i == 6){xinc = 1; yinc = -1;}(*//NE*)
           else if(i == 7){xinc = -1; yinc = -1;}(*//NW*)

           (*//Move is valid*)
           if (this.checkDir(move, xinc, yinc, kind, false)) {
               return true;
            }
        }
        (*//Move was invalid for all directions*)
        return false;
    }

    (**
     * Determines whether or not there is a valid move available for a particular player
     *
     * @param player The current player's color
     * @return Whether or not the particular player can play
     *)
    public bool userMoveAvailable(int player)  {
        int i;
        int j;
        int[] board = this.board;
        for (i = 0 ; i < 8 ; i=i+1) {
            for (j = 0 ; j < 8 ; j=j+1) {
                (*//If the potential position is not occupied and is a valid move*)
                class LocationObj l = new LocationObj(j,i);
                if ((board[(j*8)+i] == 2) and this.isValid(l,player)) {
                    return true;
                }
            }
        }
        return false;
    }

}
```

```
(**
 * OthelloGame implementation of Game that appropriately defines initialize and
↪  playGame()
 * Utilizes a Board object for maintaining and managing the board and the pieces on it
 *
 *
 * @author David Watkins
 * UNI djw2146
 *)

 (*********************************************************************)

class OthelloGame {

    private class Player p1;
    private class Player p2;
    private class Board board;

    (*
     * Initializes the boards and the players to begin a new game.
     * Also gives p1 and p2 the initial four positions
     * @see Game#initialize(Player, Player)
     *)
    public void initialize(class Player pl1, class Player pl2) {
        this.p1 = pl1;
        this.p2 = pl2;
        this.board = new Board();
        class Board b = this.board;
        b.initializeBoard();

        class LocationObj move1 = new LocationObj(3, 3);
        class LocationObj move2 = new LocationObj(4, 4); (*//White moves *)

        class LocationObj move3 = new LocationObj(4, 3);
        class LocationObj move4 = new LocationObj(3, 4); (*//Black moves*)



        (*//Both p1 and p2 need to be initialized on initial state
        //p1 initialization*)

        class Player p1 = this.p1;
        p1.setResult(move1);
        p1.setResult(move2);
        p1.setResult(move3);
        p1.setResult(move4);

        (*//p2 initialization*)
        class Player p2 = this.p2;
        p2.setResult(move1);
        p2.setResult(move2);
        p2.setResult(move3);
        p2.setResult(move4);


    }



    (*
     * Main playGame method for the Othello game
     * Continuously calls the various placeTile methods until a winner is found
     * Returns the winner of the game in type Player
     *
     * @see Game#playGame()
     *)
   (* @Override *)


    public class Player playGame() {
        int turn = 1; (*//Black always goes first *)
        (* ================================== *)
        int moveSkip = 0;
        class Board b = this.board;
        while( (b.thereIsWinner() == -1 and moveSkip != 2) ){ (* //No winner yet and two
↪   moves weren't skipped *)
            bool retry = false;
            class LocationObj move = null;

            if (not (b.userMoveAvailable(turn))) { (*//No valid moves available for
↪   user*)
                (* //Switch turn *)
                (* ================================== *)
                if (turn == 1) {turn = 0;}
                else {turn = 1;}

                moveSkip = moveSkip + 1;
            }

            else if (turn == 1) { (*//p1 turn *)
                while(not (b.isValid(move, turn))) { (*//Get p1 move*)
                    class Player temp = this.p1;
                    move = temp.placeTile(retry);
                    retry = true;
                }

                b.setPlayerMove(move, turn);

                class Player temp2 = new Player();
              (*)  temp2.setResult(move); *)
              (*============================================*)
                this.p1 = temp2;

                turn = 0;

                moveSkip = 0;
            }

            else{    (*//p2 turn *)
                class Player temp3 = new Player();
                while (not b.isValid(move, turn)) { (*//get p2 move *)
                    temp3 = this.p2;
                    move = temp3.placeTile(retry);

                    retry = true;
                }

                b.setPlayerMove(move, turn);
                class Player temp4 = new Player();
                temp4 = this.p1;
               (*) temp4.setResult(move); *)
               (*============================================*)
                this.p1 = temp4;
                turn = 1;
                moveSkip = 0;
            }


            b.setPlayerMove(move, turn);
            retry = false;
        }

        (*//The winner of the game *)
        int winner = b.whoHasMore();

        (* //Return winner *)
        if(winner == 1)
            return this.p1;
        else if(winner == 0)
            return this.p2;
        else (*//Tie *)
            return null;

        class Player toReturn = new Player();
        return toReturn; (* Default return, should never get called *)
    }
```

```
}
```

References

[1] http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html *The GNU C Reference Manual.*. N.p., n.d. Web. 26 Oct. 2015.

[2] https://docs.oracle.com/javase/specs/jls/se8/html/index.html *The Java Language Specification.* . N.p., n.d. Web. 26 Oct. 2015.

[3] Edwards, Stephen. *"Programming Language and Translators."* Lecture.

[4] "Control Flow Statements." *The Java Tutorials Learning the Java Language Language Basics* N.p., n.d. Web. 26 Oct. 2015.

[5] https://en.wikipedia.org/wiki/Dice_%28programming_language%29 *The Dice Wikipedia Page.*. N.p., n.d. Web. 22 Dec. 2015.