Final Project

In [1]:
```
// [THIS IS READ-ONLY]
@file:DependsOn("/antlr-4.11.1-complete.jar")
@file:DependsOn("./target")
```

In [2]:
```
// [THIS IS READ-ONLY]
import org.antlr.v4.runtime.*
import backend.*
```

In [3]:
```
// [THIS IS READ-ONLY]
fun execute(source:String) {
    val errorlistener = object: BaseErrorListener() {
        override fun syntaxError(recognizer: Recognizer<*,*>,
                offendingSymbol: Any?,
                line: Int,
                pos: Int,
                msg: String,
                e: RecognitionException?) {
            throw Exception("${e} at line:${line}, char:${pos}")
        }
    }
    val input = CharStreams.fromString(source)
    val lexer = PLLexer(input).apply {
        removeErrorListeners()
        addErrorListener(errorlistener)
    }
    val tokens = CommonTokenStream(lexer)
    val parser = PLParser(tokens).apply {
        removeErrorListeners()
        addErrorListener(errorlistener)
    }

    try {
        val result = parser.program()
        result.expr.eval(Runtime())
    } catch(e:Exception) {
        println("Error: ${e}")
    }
}
```

String arithmetics

In [4]:
```
// [THIS IS READ-ONLY]
val program1 = """
x = "Hello";
y = "World";

print(x ++ " " ++ y);
"""
```

In [5]:
```
// [YOUR WORK HERE]
// @workUnit
// execute the program

execute(program1)
```

```
"Hello World"
```

Mixed arithmetics

In [6]:
```
// [THIS IS READ-ONLY]
val program2 = """
x = "woof ";
y = "Dog goes " ++ (x * 2);

print(y);
"""
```

In [7]:
```
// [YOUR WORK HERE]
// @workUnit

execute(program2)
```

```
"Dog goes woof woof "
```

Loops

In [8]:
```
// [THIS IS READ-ONLY]
val program3 = """
sum = 0
for(i in 10..20) {
  sum = sum + i;
}

print(sum)
"""
```

In [9]:
```
// [YOUR WORK HERE]
// @workUnit

execute(program3)
```

```
sum = 0
165
```

Function

In [10]:
```
// [THIS IS READ-ONLY]
val program4 = """
function greeting(name, message) {
  x = "Hi,";
  x = x ++ " my name is " ++ name ++ ".";
  print(x);
  print(message);
}

greeting("Albert", "How are you?");
"""
```

In [11]:
```
// [YOUR WORK HERE]
// @workUnit

execute(program4)
```

```
"Hi, my name is Albert."
"How are you?"
```

Recursion

In [12]:
```
// [THIS IS READ-ONLY]
val program5 = """
function factorial(n) {
  if(n < 2) {
    1;
  } else {
    n * factorial(n-1);
  }
}

print(factorial(10));
"""
```

In [13]:
```
// [YOUR WORK HERE]
// @workUnit

execute(program5)
```

```
3628800
```

Optional Variable Typing

In [18]:
```
val program6 = """
    Int x = 5;
    String y = "hello";
    x = "string"; // This should throw a type mismatch error
    print(x);
    print(y);
"""
execute(program6)
```

```
Error: java.lang.RuntimeException: Type mismatch: expected Int, found StringData
```

Foreach Loop, Array Indexing

In [19]:
```
val program7 = """
    z = 1;
    alpha = 4334.555;
    g = {2,3,4,5,"hello world",alpha};
    foreach(item in g){
         if(g[3] > 2) {
            print(g[(5-4)+3]);
          } else {
            print("Something went wrong!");
          }
    }
"""
```

In [20]:
```
execute(program7)
```

```
"hello world"
"hello world"
"hello world"
"hello world"
"hello world"
"hello world"
```

Dictionary

In [22]:
```
val program8 = """
myDict = {"one": 1, "two": 2, "three": 3};
    print(myDict["two"]);
    myDict.put("hello","world");
    print(myDict);
    myDict.remove("hello");
    print(myDict);

    print("The keys are: ");
        print(myDict.keys());

    print("The vals are: ");
        print(myDict.values());
"""
```

In [23]:
```
execute(program8)
```

```
2
{"one": 1, "two": 2, "three": 3, "hello": "world"}
{"one": 1, "two": 2, "three": 3}
"The keys are: "
["one", "two", "three"]
"The vals are: "
[1, 2, 3]
```

Size

In [26]:
```
val program9 = """

x = 5;
y = 120;
z = 3.14;
a = 4244.44242;
g = {x,y,z,a};
print(g.size());
"""
```

In [27]:
```
execute(program9)
```

```
4
```

Function(0 parameters) & Logical Expression

In [28]:
```
// [THIS IS READ-ONLY]
val program10 = """
function a() {
    4;
}
z = !(7 == a()) && (7 == a());
y = !(7 == a()) || (7 == a());
print("this should be false");
print(z);
print("this should be true");
print(y);
"""
```

In [29]:
```
execute(program10)
```

```
"this should be false"
false
"this should be true"
true
```

In [ ]: