Sonoma State University
Computer Science Department
CS 460 – Spring 2018 – Watts

Project 3

For this part of the Project you will be generating C++ or Python code for each of the Scheme input files.

Specifications

1. There is a sample version of the CodeGenerator class available in the directory Project3Framework in the course pickup folder.

2. Calls to the Code Generator should be made from your Syntactic Analyzer.

3. For each .ss in put file, your program should generate a .cpp or .py file with the same name. For example, if the input file is P3Test1.ss, the output file should be P3Test1.cpp or P3Test1.py.

4. The Object class (also in the Project3Framework directory) should be included in your C++ files so that you can use the generic type Object in your generated code.

5. The generated code should compile or interpret with no errors and should produce the same output as the original Scheme code. An example is at the end of this document.

6. All of the team's source code should be placed in a folder called Team#P3 (where # is your team's number.) The folder should contain the code required to generate an executable called P3.out (or P3.py).

7. Your folder should also contain a makefile. The first target of your makefile should be P3.out or P3.py.

8. Your folder should also contain a file called README.txt that describes what your project does and what if does not do.

Date due: Thursday, 17 May 2018 at 11:59 pm.

To turn in: Each team should submit a tarred and zipped file called Team#P3.tgz containing their folder Team#P3.

Sample Scheme Input:

```
;; Project 3 Test 3

(define (listop_ex1)
     (cons (car '(a b c)) (cdr '(d e f)))
)

(define (listop_ex2)
     (cons (cadr '(a b c)) (cddr '(d e f)))
)

(define (main)
     (display (listop_ex1)) (newline)
     (display (listop_ex2)) (newline)
)
```

Possible Generated Code:

```
// Autogenerated Scheme to C++ Code
// File: P3-3.cpp
#include <iostream>
#include "Object.h"
using namespace std;

Object listop_ex1 ()
{
  Object __RetVal;
  __RetVal = cons (listop ("car", Object("(a b c )") ),
                               listop ("cdr", Object("(d e f )") ));
  return __RetVal;
}

Object listop_ex2 ()
{
  Object __RetVal;
  __RetVal = cons (listop ("cadr", Object("(a b c )") ),
                               listop ("cddr", Object("(d e f )") ));
  return __RetVal;
}

int main ()
{
  Object __RetVal;
  cout << listop_ex1();
  cout << endl;
  cout << listop_ex2();
  cout << endl;
  return 0;
}
```

Expected output:

```
(a e f)
(b f)
```