CSE 307 – Principles of Programming Languages
Spring, 2020
Homework Assignment 04
Extending a Programming Language: Statements

Assigned: 04/09/2020
Due: 04/24/2020, at 11:59 PM
Total Points: 50

For this assignment we will be extending SBML to include variables, assignment, conditionals, loops, and output. For now, SBML will have single global, static scope for all variables.

The code written for this assignment will be reused in future assignments. Please keep that in mind when working. Keep your code clean, neat, and modular so that it will be easy to update and extend in future assignments.

We will use continue to use the parser generator PLY – Python Lex and Yacc – to specify and implement the SBML language.

SBML extension:

Variables and Assignment:

All variable names begin with an ASCII character, which may be followed by zero or more ASCII characters, digits, or underscores.

A regular expression to match this definition of variable names is: "[a-zA-Z][a-zA-Z0-9_]*"

Expressions from the previous homework (HW 03) must be extended in the following two ways:

1. Support for assignment to variables must be added. For example, "x = 1;". This will include assignment to indexed list variables. For example, if we have performed the assignment "array = [1, 2, 3];", and then perform the assignment "array[2] = 5", then the list stored in "array" should now contain [1, 2, 5].

2. Support must be added for variables used in expressions. For example, if x was assigned 1, then "print(x);" will print 1. Similarly, we should evaluate indexed variables if they occur in a place where their value is needed. For example, if array was

assigned the list [1, 2, 5], then "print(array[0] + array[1] + array[2]);" should print 8.

Evaluating a variable for its value should have the following behavior. If the variable has had a value assigned to it, then the value should be returned. Otherwise, a "Semantic Error" should be reported and your program should stop.

When an indexed list variable is used in an expression, then both the list and the index are evaluated to their value, and then the indexed list expression is evaluated. If the variable is not a list (or a string), or the index is not an integer, then a Semantic Error should be reported. If the index is outside the bounds of the list, then a Semantic Error should be reported.

New Statement Types:

1. Block: A block statement consists of zero or more statements enclosed in curly-braces, "{…}". When the block executes, each of the statements is executed in sequential order.

2. Assignment: an assignment statement consists of an expression, an equals sign, an expression, and a semicolon, "exp1 = exp2;". When the assignment statement executes the left-hand side expression is assigned the value evaluated for the right-hand side expression.

3. Print: a print statement consists of the "print" keyword, a left parenthesis, an expression, a right parenthesis, and then a semicolon. When the statement executes, the expression is evaluated for its value. The output displayed should be the same as that produced by Python for that value.

For example,

```
>>> print('a')

a

>>> print(['a'])

['a']

>>> print(1 < 2)

True
```

4. Conditional Statements:

A. If Statements: Consist of a keyword "if", a left parenthesis, an expression, a right parenthesis, and a block statement as the body of the If statement.

When the If statement executes, if the expression evaluates to True, then the block statement composing the body is executed.

B. If-Else Statements: Consist of a keyword "if", a left parenthesis, an expression, a right parenthesis, a block statement as the body of the If clause, the keyword "else", and a bloc statement as the body of the Else clause.

When the IF-Else statement executes, if the expression is True, then execute the block statement that is the body of the If clause. Otherwise, execute the block statement that is the body of the Else clause.

5. Loop Statements:

A. While Loops: A While statement consists of the keyword "while", a left parenthesis, an expression, a right parenthesis, and a block statement that is the body of the While statement.

Executing the while statement begins by evaluating the condition expression for its value. If the expression evaluates to False, then the While statement terminates. Otherwise, execute the block of statements that compose the body of the While statement, and then repeat the execution of the While statement from the evaluation of the condition expression.

Input Programs:

An SBML input program for your interpreter consists of a single, outermost Block statement. Executing the program consists of executing this block statement.

Your program will be called with a single command-line argument. This argument will be a filename for a file containing an SBML input program.

If the program contains any Syntax Errors, then your program should print out "SYNTAX ERROR" and exit. Otherwise, try to execute the input program. If the execution causes a Semantic

Error, then your interpreter program should print "SEMANTIC ERROR", and then exit.

Execution of the program may cause Print statements to execute, and the output from Print statements should be sent to Standard Output. **No other output should be produced by your program.**

Suggestions:

1. Create a global map/dictionary to store the values of variables (a very simple Symbol table).

2. The AST node used to represent variables should include an evaluation method that looks up the current value assigned to the variable in the global mapping and return that value.

3. The AST nodes representing statements should have a method to resolve the execution of those statements. Inside the method, you will evaluate any necessary expression, perform branching or looping as required, and produce any side-effects (like sending values to the display in Print statements).

Submission Instructions:

1. Please name your program sbml.py
2. Please include your name and student id as comments at the top of your program file.
3. Please collect and submit your program file as a compressed zip file.
4. The title of the compressed file should be:
   cse307_hw04_LastNameFirstName.zip
5. This is an individual assignment. Any collaboration on writing your programs will be treated as a violation of academic integrity.

Example SBML Input Programs:

Example1:

```
{
  number = 33;
  isPrime = 1;
  i = 2;
  while(isPrime == 1 andalso number > i){
    if (number mod i == 0) {
      isPrime = 0;
    }
    i = i + 1;
  }
  if(isPrime == 1){
    print("isPrime is true");
  } else {
    print("isPrime is false");
  }
}
```

Test and output should look like this:

```
$ python sbml.py example1
```

isPrime is False

Example2:

```
{
  data = [ [ 100, 42 ], [ 100, 50 ], [ 123, 456 ], [ 300, 9000 ] ];
  result = [ 0, 0, 0, 0 ];
  i = 0;
  while (i < 4){
    a = data[i][0];
    b = data[i][1];
    if (a > 0){
      while (b > 0){
        if (a > b){
          a = a - b;
        } else {
          b = b - a;
        }
       }
    }
    result[i] = a;
    i = i + 1;
  }
  print(result);
}
```

Test and output should look like this:

```
$ python sbml.py Example
[2, 50, 3, 300]
```