CSE 307 – Principles of Programming Languages
Spring, 2020
Homework Assignment 05
Extending a Programming Language: Functions

Assigned: 04/23/2020
Due: 05/11/2020, at 11:59 PM
Total Points: 50

For this assignment we will be extending SBML to include function definitions and function calls.

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

SBML extension:

Function Definition:

A function definition begins with the keyword "fun", followed by the name of the function, a left parenthesis, variables representing formal parameters separated by commas, a right parenthesis, an equal sign, a block, and then an expression.

When the function is called, the block is executed first. Then the expression is evaluated and the result of the expression evaluation is returned to the caller.

Function Call:

A function call is an expression. The function name is followed by a left parenthesis, and then argument expressions, followed by a right parenthesis.

The number of arguments passed to the call must match the number of parameters in the function definition.

General Requirements:

An input program might contain multiple function definitions followed by a single main block that gets executed.

You will have to implement a stack to create the scoping for local variables inside recursive function calls.

Call by value will be used to pass arguments to parameters.

Two example programs with functions are given below.

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_hw05_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:

```
fun factorial(n) =
{
  if (n < 1)
  {
    output = 1;
  }
  else
  {
    output = n * factorial(n - 1);
  }
}
output;
{
  print(factorial(3));
}
```

Test and output should look like this:

```
$ python sbml.py example1
6
```

Example2:

```
fun gcd(a, b) =
{
  t = b;
  b = a mod b;
  if (b == 0)
  {
    output = t;
  }
  else
  {
    output = gcd(t, b);
  }
}
output;
{
  print(gcd(32, 18));
}
```

Test and output should look like this:

```
$ python sbml.py example1
2
```