Project #5
CpSc 8270: Language Translation
Computer Science Division, Clemson University
Python Functions, Scope & Decision
Brian Malloy, PhD
November 28, 2017

Due Date:

In order to receive credit for this assignment, your project must be submitted, using the `web handin` command, by 8 AM, Friday, December 1st of 2017. If you are unable to complete the project by the first due date, you may submit the project within three days after the due date with a ten point deduction.

Project Specification:

1. Your solution should be able to translate those constructs from the previous project, including integer and float values, variables, print, assignment, and the expressions specified in the previous project.
2. For this project, your solution should be able to translate Python functions, including scope resolution 1a, return value propogation 1b and 1c, and recursion 1d.
3. In addition, your solution should translate if/else (not elif). You must also translate the six (6) relational operators: <, <=, ==, >, >=, ! =. You are not required to implement and, or, not.
4. In all cases, the oracle for correctness is a Python 2.7 interpreter; your expressions should evaluate to the same value as a Python 2.7 interpreter, but not the same format. So, 5 is the same as 5.0; True is the same as 1, False is the same as 0.
5. In the directory that contains your working interpreter, place a new directory titled cases that contains test cases that adequately test your interpreter.
6. Write a test harness, test.py, and place it in your project folder so that it runs the test cases in cases.
7. Your code should be well organized, formatted, readable, leak free, and exploit object technology.

Light at the end of the tunnel:

In the final project, Project #6, we will translate actual and formal parameters and recursion.

```
def f():
  x = 0
  if x == 0:
    print 99
    x = 17
    if x:
      print 1
    else:
      print 2

f()
print 17
```

(a) Basic Scope

```
def f():
  return 7*2
def g():
  print 15
  x = 12
  def h():
    return x
  print h()
  return 2*x

print f()
print g()
```

(b) Nested Functions

```
def f():
  x = 0
  if x == 0:
    print 99
    x = 17
    if x:
      print 1
      return
    else:
      print 2
    print 101
f()
print 17
```

(c) Return Statement

```
def f():
  x = 0
  def g():
    x+=1
    print x
    if x < 10: g()
    print x
  g()
f()
```

(d) Tail Recursion

Figure 1: Examples of Some Interesting Python Test Cases.