Homework 2

CS 173: Intermediate Computer Science — Spring 2014
Instructor: Thomas Bressoud — Due: 2014-02-07, classtime

- The point of this project is to make concrete your understanding of ADTs by implementing a Stack ADT and an RPN calculator ADT in the context of this specific application.
- A Grading Guide will be published separately, but you are expected to use good functional abstraction and pre and post conditions specified in docstrings along with good inline comments documenting your solution.

In this project, you will create an object-oriented Reverse Polish Notation (RPN) calculator. RPN, also called postfix notation, is a *parenthesis-free* notation in which a binary operator follows the two operands. For example, the infix expression

$$3 + 5$$

is written as

$$3\ 5\ +$$

in RPN.
RPN is of interest because RPN expressions can be evaluated easily from left to right using a stack. When you encounter an operand, push it on the stack. When you encounter a binary operator, pop two numbers off the stack, apply the operator, and push the result back onto the stack. At the end of the expression, the result is on the top of the stack. For example, consider the following longer RPN expression:

$$3\ 5\ +\ 7\ *\ 8\ 11\ *\ -$$

First, we push 3 and 5 onto the stack. To apply the addition operator, we pop 3 and 5 off the stack, and push the result 8. Then we push 7. To apply the multiplication operator, we pop 8 and 7, and push the result, 56. Then we push 8 and 11, pop them off when we get to the multiplication operator, and push 88. Finally, we pop 56 and 88, subtract them, and push the final result, -32.

The first step in this project is to implement an ADT Stack as a Python class. Your class should have 6 methods: a constructor, top, pop, push, and isEmpty and __str__. Your class and each of your methods should have a docstring describing the class or method and containing appropriate precondition and postconditions, and your methods should test preconditions and raise appropriate exceptions. Build a unit test and test your class and methods thoroughly before moving on to the next step. Your underlying representation for this stack class can be a Python list. As implementor of the ADT, you can decide whether the index 0 element of the list is the "top of stack" (TOS) or the last element in the list is the TOS.

Next, implement a class called RPNCalculator that contains a Stack object as part of the information it maintains and has 8 methods: a constructor, evaluate (evaluate a string using the other methods), push (push a number), pop (remove the top of the stack and return the top), add, subtract, multiply, and divide. The last four methods follow the steps outlined above. Your class should check for different types of errors. You should build a unit test for this class as well. The string for the evaluate method should contain space separated numbers and operators.

Finally, in a file called main.py write a main() function that creates an instance of your calculator and allows someone to interactively use it:

```
$ python3 main.py

RPN> 2 2 +
4
RPN> 8 4 /
2
RPN> 3 5 + 7 * 8 11 * -
-32
RPN> 2 + 2
INVALID
RPN> quit
```

So in the above sequence, the main() function will first create the RPNCalculator object and then, repeatedly, print the 'RPN> ' prompt, retrieve the string entered by the user, evaluate the string using the RPNCalculator, and retrieve (pop) the top value and print it out to the user. If an exception is raised by the RPNCalculator, the main should detect (catch) it and output the string 'INVALID' to the user. If the string 'quit' is input by the user, the program should terminate.
Please submit your homework (Stack.py, test_Stack.py, RPNCalculator.py, test_RPNCalculator.py, and RPNmain.py) in submitbox by the due date.