CS 014: Introduction to Data Structures and Algorithms
Fall 2017
Lab 2

Objectives:
In this lab, you will use the stack data structure to convert an infix expression to postfix.

Deliverables:
- (10%) You have to attend the lab on Monday.
- (50%) The final code.
- (20%) A brief report that describes the data structures and algorithms that you used in your program.
- (20%) There will be a relevant question during the lab on Monday 10/16 that you will be asked to do during the lab.

Groups:
- This lab should be done individually.

Due date:
- The deliverables are due on Tuesday 10/17 by 11:59 PM Pacific Time. However, you are highly encouraged to deliver it during the lab on Monday 10/16 to save your time.

Important Note:
- You will need to carry out this lab at home to be able to perform the task that will be given to you during the lab on Monday 10/16. The official due date is on Tuesday just in case you have questions for your TA during the lab.

Problem definition:
Most humans prefer to read and write mathematical expressions in infix notation where the operator is written between the two operations, e.g. 2+3. On the other hand, computers deal better with postfix expressions where the operator is written *after* the two operands, e.g. 23+. In order to bridge this gap, you are required to write a function that uses the stack data structure to convert an infix expression into postfix. For example, if the input is "2+3", the output is "23+". If the input is "A+B*C", the output is "ABC*+". For more details, check section 3.6 in the textbook.

Detailed requirements:
- The input is provided as a single C++ string.
- The output is also a single C++ string.
- The operators that you need to consider are
  - +
  - –
  - *
  - /
  - (
  -)
- The operands are either single-digit numbers (0-9) or single-letter characters (a-z, A-Z).
- Similar to C++, the parentheses () have the highest priority, then multiplication and division, and finally addition and subtraction.
- You do NOT need to evaluate the expression; you only need to return the postfix expression.
- You are allowed to use the stack data structure provided by STL.

Steps:

1. Create a workspace in Cloud9 and initialize it with the code that is attached to this lab. You can do this by pasting the URL ‘https://github.com/aseldawy/CS014Lab2’ in the field ‘Clone from Git or Mercurial URL.’ Instead, you can download the source code from iLearn and upload it to an empty workspace in Cloud9.
2. To compile the code, type “`make`” in the shell. In Cloud9, you can access the shell which is usually at the bottom of the screen.
3. To run the code, type “`./infix2postfix`” in the shell.
4. The code uses Google Test library to test five basic cases for infix/postfix conversion. Initially, all the cases fail. Your job is to implement the infix2postfix function. If implemented correctly, all the tests should pass.
5. At delivery, we might use additional test cases other than the five listed in the source code. Try to make your source code robust by making additional tests.