EECS 560 Lab 4: Postfix calculator using stacks

Due date:
09/27/2020, 11:59 pm -Sunday

Objective
To implement Infix to Post fix calculator using stack with C++ . Use basic operations of stack convert an Infix expression to Postfix and then calculate Postfix result. Please refer to textbook section 3.6.3 for details on Infix Postfix calculation.

Specification of the Infix Postfix calculation using stack:

1. Create a class with name **"myInfixPostfix".**

2. Implement **constructor** with default behavior as **myInfixPostfix (void)** and **destructor** with default behavior.

3. Include **myStack.hpp** in this file to use stack operations. myStack.hpp should be the .hpp file implemented in Lab3.

4. Implement a Infix to Postfix converter function with name "getPostfix" which takes for infix expressions from instance file as input and convert it to postfix expression using stack operations. This method should have an interface of
   std::string getPostfix(std::string str)

5. Implement a Postfix Calculator with function name "PostfixCalculator" which takes the converted postfix expression from above function as input and calculate postfix result using stack operations and mathematic operations. This method should have an interface of
   float PostfixCalculator(std::string exp)

Testing and Grading

We will test your implementation using a tester main function, on a number of instances that are randomly generated. We will release the tester main function, several instances (will be different from the grading instances but will have the same format), and expected output together with the lab instruction via Blackboard. You code will be compiled under Ubuntu 20.04 LTS using g++ version 9.3.0 (default) with C++11 standard.

The command line we are going to use for compiling your code is:

"g++ -std=c++11 main.cpp" (note that main.cpp will include the .hpp files you submit, and your .hpp files needs to be property implemented to compile successfully).

Your final score will be the percentage your program passes the grading instances. Note that if your code does not compile (together with our tester main function), you will receive 0. Therefore, it is very important that you ensure your implementation can be successfully compiled before submission.

Submission and Deadline

Please submit your implementation as two different .hpp files, with file names "myStack_[YourKUID].hpp" (This can same file from Lab3) and "myInfixPostfix_[YourKUID].hpp". For example, if my KU ID is c123z456, my submission will be a single file named "myStack_c124z456.hpp" and "myInfixPostfix_c124z456.hpp". Submissions that do not comply with the naming specification will not be graded. Please submit through Blackboard.