EECS 560 Lab 4: Index and Postfix Calculation using Stacks

Objective

Understand the infix and postfix calculation using stack.

Specifications

1. Only addition ("+"), subtraction ("-"), multiplication ("*"), and division ("/") are assumed to be valid operators from the infix input.
2. Only parenthesis ("(" and ")") are assumed from the infix input.
3. Both integers and floating numbers are assumed to be valid inputs.
4. Both positive and negative numbers are assumed to be valid inputs.
5. The input infix string is balanced and valid (fully complied with customary mathematical notation).
6. Implement a class named `MyInfixCalculator`. The class is expected to use the `MyStack` and `MyVector` classes you have implemented in Lab 3.
7. Implement `void tokenize(const std::string& s, MyVector<std::string>& tokens)` to decompose the infix string into a set of tokens.
8. Implement `void infixToPostfix(MyVector<std::string>& infix_tokens, MyVector<std::string>& postfix_tokens)` to convert a set of infix tokens to a set of postfix tokens.
9. Implement `double calPostfix(const MyVector<std::string>& postfix_tokens) const` to calculate result the result.
10. Implement a wrapper function `double calculate(const std::string& s)` to directly compute the result from a given infix input.

Hint: You can make use of the functions that have already been implemented in the template header file to facilitate your implementation.

Testing and Grading

We will test your implementation using the tester main function posted online. The posted input and output examples should be used for a testing purpose, while we will use another set of inputs for grading. Your code will be compiled under Ubuntu 20.04 LTS using g++ version 9.3.0 (default) with C++11 standard.

Your final score will be the determined by the success percentage of your program when fed with many random inputs. 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 three .h files, with names "MyStack_[YourKUID].h" (the same as your Lab 03 submission), "MyVector_[YourKUID].h" (the same as your Lab 01 submission), and "MyInfixCalculator_[YourKUID].h". For example, if my KU ID is c123z456, my submission will be three files named "MyStack_c124z456.h", "MyVector_c124z456.h", and "MyInfixCalculator_c124z456.h". Submissions that do not comply with the naming specification will not be graded. All submission will go through Blackboard. The deadline is Friday Mar 24th, 2023, 11:59PM.