CSCI 220 -- PA 5

Stacks and Postfix Calculator

Feel free to discuss and help each other out but does not imply that you can give away your code or your answers! Make sure to read all instructions before attempting this lab. You cannot work with a lab partner for this assignment.

You must use an appropriate provided template from Canvas or my website (zeus.mtsac.edu/~tvo) and output "Author: Your Name(s)" for all your programs. If you are modifying an existing program, use "Modified by: Your Name(s)".

Exercise 1: You must use either existing C++ **stack** class or Java **Stack** class to solve the "Balancing Symbols" problem. The symbols are (), [], and {}, and each opening symbol must have a corresponding closing symbol as well as in correct order. Ignore operands and arithmetic operators since they are not relevant to our problem. You can assume each token is separated by spaces. Try the 3 examples below:

- { (a + b) * c1 } – valid
- { (a + b) * c1] – invalid
- ((a + b) * c1 } / 15) – invalid

Exercise 2: You must define and implement your own **Stack** class or use the one from the textbook. The **Stack** class supports standard basic stack operations, and you can implement it with an array or a linked list. You should create a class template Stack in C++ or generic class Stack in Java, but an integer stack would work as well. Set up a function or static method that receives a string representing a postfix expression and it returns an integer result. Your function/method uses a stack to evaluate a postfix expression (see an operand -- push; see an operator – pop twice, evaluate, then push result).

- Test your function/method and Stack class with the following postfix expressions by using a test driver:

```
17 2 3 + / 13 -
5 2 3 ^ *
2 3 2 ^ ^
```

- Answers for the postfix expressions above:

```
-10
40
512
```

Question 1: Outline a solution to handle an expression like `{(a+25 )*c1}`. Notice that spaces are now optional.

Question 2: What is the running time for your implementation of postfix evaluation? Justify your answer.

Extra Credit: Set up a function or static method that receives a string representing an infix expression and it returns an equivalent postfix expression. Your function/method uses a stack to convert an infix expression to a postfix expression.

- Test your function/method with the following infix expressions by using the same test driver from exercise 2:

  ```
  17 / ( 2 + 3 ) - 13
  5 * 2 ^ 3
  2 ^ 3 ^ 2
  ```

- Answers for the infix expressions above:

  ```
  17 2 3 + / 13 -
  5 2 3 ^ *
  2 3 2 ^ ^
  ```

Fill out and turn in the PA submission file for this assignment (save as PDF format).