Homework 4 – Due Oct. 19th 23:59, KST

Instructions: Complete the implementation and turn it in before the due date. Any deviations from the instructed deliverable format will result in a deduction of grade. DO NOT COPY OTHER'S WORKS!

In this assignment, you will be implementing the conversion algorithm between infix and postfix expressions using your own stack data structure. There are three main components to this assignment.

- Successful implementation of a stack data structure.
- Converting the given infix expression into a postfix expression.
- Converting the given postfix expression into an infix expression.

For both the infix and the postfix notations, the expression will consist of operands and operators. You only have to consider five arithmetic operators (+-*/^) and length-1 alpha-numeric operands. When converting the notations, your implementation should detect the following types of "syntax" errors in the input string and return an empty string.

- Wrong operator positions. E.g., "a+-b" (infix), "a+b" (postfix)
- Dangling operators: E.g., "a+b-" (infix), "ab+-" (postfix)
- Dangling operands: E.g., "ab+c" (infix), "aab+" (postfix)

The description for converting a postfix to an infix is given below.

> Every time an operand is read, push it onto the stack. Every time an operator is read, pop two elements from the stack and combine them into an infix form using that operator. Push the formed infix onto the stack, and continue until you finish scanning the given postfix expression.

Rubric: Grading will be based on, but not limited to, the following criteria.

- Documentation (40 points): You should provide a header comment that provides a big-O time complexity analysis for each of the two conversion methods. Notice that I'm not providing the variable of complexity: You have to clearly identify with respect to what variable the time complexity will be. In addition to the big-O's, provide a brief explanation of how you arrived at that conclusion.
- Conversion correctness (40 points): Your implementation should behave as specified above. It should be error-free and use appropriate features of Java, such as generics.
- Stack implementation (20 points): Your implementation of the stack should be correct, in terms of the behavior and time complexity.

- Miscellaneous: Do not change the method and class names. Carefully read the comments provided in the given HW4.java file – the instructions in the comments are also part of the official requirements.

Deliverable: A single HW4.java source file. DO NOT provide a zip file. Just submit a single Java file with no package structure.