**CSCD 300 Homework 56**

**Evaluating Infix Expressions (Total 100 points)**

**Rules**

**If you have skipped one or more lectures relevant to this homework assignment without an acceptable excuse, the instructor will refuse to answer ANY questions about this homework. An acceptable excuse is usually documented or related to an emergency. Your instructor has checked the class attendance everyday. Please refer to the syllabus for more information regarding this rule.**

**Turn in:** On EWU Canvas CSCD300🡪Assignments🡪Hw56🡪Submit. Please put all your source code(.java) files into a zip file. Name the zip file with your last name followed by first initial plus hw56.zip. For example, smithjhw56.zip is for John Smith. Please organize your code in a way that the grader can compile all your source files using a command **javac \*.java** and run your program using a command **java Tester** on the command line.

If you forget to include your source code in the zip file, you get a zero credit for this homework. If your code shows a compile-time error, you get a zero credit. If you turned in a corrupted file, or a file that cannot be opened, you get a zero too.

**Problems Description**

Based on the materials we discussed in class, you are required to implement the pseudo code for converting an infix expression into a postfix expression, then evaluating the postfix expression to produce a single result. You have to implement the following functionalities,

1. Your program reads an infix expression represented by a string **S** from the **standard input** (the keyboard).
2. Then your program converts the infix expression into a postfix expression **P** using the algorithm we learned in class.
3. Next, your program evaluates the postfix expression **P** to produce a single result **R**.
4. At last, your program displays the original infix expression **S**, the corresponding postfix expression **P** and the final result **R** on the standard output ( the screen ).
5. Your program is required to handle at least **five** mathematic operations, +, -, \* and /, and ^(exponential). When you perform division, please do integer division. For example,

7 / 2 is 3.

1. Note that in this project, the input infix expression contains **one-digit** operands and produces **one-digit** result. In other words, you do not worry about the infix expressions that involve multiple-digit operands and produce multiple-digit results.
2. Note that you have to implement your own Stack class in this homework. The built-in Java Stack is not allowed in this homework. Please use Java Object type or the generic type E for the data element in your stack implementation.
3. Please catch any potential errors, so that your program will not crash. You do not need to check whether all parentheses in the input infix expression are correctly matched or not. **You can safely assume the format of the infix expression is correct.**
4. Each run of your program will evaluate only one infix expression.

**Sample Run One**

Please enter the infix expression to process: **(((1+2)-(3-4))/(6-5))**

The postfix expression for the input infix is: **1 2 + 3 4 - - 6 5 - /**

The final result after evaluating the postfix is: **4**

**Sample Run Two**

Please enter the infix expression to process: **2 \* 4 - 2 ^ 2 ^ 1**

The postfix expression for the input infix is: **2 4 \* 2 2 1 ^ ^ -**

The final result after evaluating the postfix is: **4**