Data Structures HW2

Logistics

Due date: **4/18 (Sun) 23:59**

Submission

- Via LMS (no email submission)
- **Three files** (please make sure to upload three files)

1. Report (no specific template, MS word or 한글, in English or in Korean)
2. output.txt (answer file)
3. Zip file of your program (**Compress it** to make a single file.)

Note: No restrictions on programming languages and platforms.

Report

Describe your idea for solving the problem, explain the code in detail, and draw your conclusion. Your report must include the following:

1. Program description (comments on important code lines)
2. How to run (so that I can test your program)

Evaluation Policy (10 pts in total)

Score (10pts) = Report (3pts) + Accuracy (3pts) + Implementation (4pts)

Accuracy: Full Mark x # right answers / # total cases

Implementation

4pts, if you fully implement a stack from scratch.

2pts, if you use an existing stack library.

1pt, if you don't use a stack.

Penalties

1. Unable to build code → Accuracy = 0

2. **Plagiarism → Score = -5 (will affect your overall grade)**
3. Late Submission → Report -= 2
4. Wrong output format and missing files (in case you forget to submit output.txt, or…) → Accuracy /= 2

Problem Description: Evaluating Postfix Expression

Write a program to evaluate given postfix expressions, using a stack. Postfix expressions are provided in the "input.txt" file which is written in the following format:

```
2
382*-8-
8906-
```

The number in the first line (i.e., 2) shows the number of postfix expressions to be processed. From the second line, the postfix expressions are given.

In the above example, the first expression is evaluated as -21. (in infix from, it's 3-8*2-8). But, the second expression cannot be evaluated, since it is an invalid expression. In this case, simply output 0, indicating that it's invalid. Write your results line by line in "output.txt" (**only one answer per line**), as you did in HW1. For the above example, your "output.txt" will be like:

```
-21
0
```

For simplicity, you can use the following facts:

First, all operands are single digits. That is, they can have values from 0 to 9.

Second, operators are restricted to the following threes: +, -, and *. That is, you don't need to consider - operator.

Third, in this task you don't need to convert the postfix expression into the infix or prefix.