COSC2436 GA2: Stack
Due Wednesday, November 10

You need to create a C++ program that can evaluate a list of arithmetic expressions whether they are valid. You also need to compare them if they are similar.

You will get a list of expressions in an input file. Each line will be considered as a single expression. You need to check whether the expression is valid or not. An expression will be invalid if opening and ending parenthesis don't match. If you don't get any invalid expression, then you need to check the similarity among all expressions. Expressions will be similar, if we place values to the variables then all expressions will generate the same result.

Assumptions:

- The input file is a small plain text file; no need to handle binary files.

- Each input file may contain maximum 1000 expressions.

- In an expression, only unit digits (0-9) will be used.

- Variables can be only (a-z, and A-Z), and it is always a single character.

- There will be no space in an expression.

- An input file may contain empty lines between expressions, then you will ignore it.

- Operators: + −.

- Parentheses: (,), {, }, [,].

- The output should be exactly matched with the format.

The input file is a regular text file, where each line is terminated with a '\n' character. Each line will contain an arithmetic expression.

1. Expressions consist of numbers (0-9), lowercase alphabets (a-z), uppercase alphabets (A-Z), '+', '-', '(', ')', '{', '}', '[', and ']'. An operand (numbers, and alphabets) will not appear the second time in a single expression.

2. If an expression is not valid, the output will be "Error at: "+*expression number*. Note, in "Error at: ", there is a space after the colon(:).

3. If Expressions are similar, then the output will be "Yes", otherwise "No".

All records should be processed sequentially from beginning to end. Note that, input and output, all are case sensitive. Please, see examples to be clarified about the format and *expression number*.

Examples
Example 1 of input and output,
Input1.txt a+b-A-1+3+B a+b-1+3+B-
A +(a+b-A-1+3+B) a+b-
(A+1)+3+B
a+b-(A+1)-(-3-B)

a+(b-(A+1)+3+B) a+{(b-[A+1])+3+B} a+b-
A+2+B

Command line:
evalexpr input=input1.txt output=output1.txt

Output1.txt
Yes

Example 2 of input and output,

Input2.txt

    A+B+5+6+a

    A-B+5+6+a

Command line:

    evalexpr input=input2.txt output=output2.txt

Output2.txt

    No

Example 3 of input and output,

Input3.txt a+b+c+d+e   // valid expression

    a-b-c-d-e   // valid

    expression a-(b-c-d)-e   // valid

    expression

    a-{(b-c-d)-e   // Error

                // Empty line will not be counted

    a-{(b-c-d))-e   // Error

                // Empty line will not be counted

    {a-[(b-c-d)]-e}   // valid expression

    a-[{(b-c-d})]-e   // Error a-{(b-[c-d)}-e]   //

    Error

Command line:

    evalexpr input=input3.txt output=output3.txt

Output3.txt

    Error at: 4

    Error at: 5

    Error at: 7

    Error at: 8

The general call to the executable is as follows:

evalexpr input=input1.txt output=output1.txt <u>**Important direction to submit your ga2:**</u>

We expect every student of a group will participate to solve the problem and discuss with each other. ***We will count the lowest grade of a group for every group members***, so make sure everyone is submitting the same copy. The purpose of this GA is to learn the basic functionality of Stack. If someone faces trouble to understand Stack, they should discuss with their friends to get a clear understanding of the Stack. It is encouraged to help your friends, but no copy paste. If you help, it will also help you to get a deep understanding.

Every student of each group needs to submit ***the same copy*** of the solution. GA2 needs to be turned in to our Linux server. Make sure to create a folder under your root directory, name it "***ga2***" (*name must be in lower case and exact*), only copy your .cpp and .h files to this folder, no test case or other files needed. If you use ArgumentManager.h, don't forget to turn it in too. ***GAs will be graded only once***. You will not get the chance of resubmission after grading. So, make sure you are submitting correct solution before the due date.