COSC2430 HW5: Stack & Queue Problem

1. Introduction

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.

2. Program and input specification

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.

<u>Assumptions</u>:

- 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 main C++ program will become the executable to be tested by the TAs. The Result file should be written to another text file (output file), provided with the Command line. Notice the input and output files are specified in the command line, not inside the C++ code. Notice also the quotes in the program call, to avoid Unix/Windows gets confused.

3. Input and Output

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*.

4. Program Execution:

The general call to the executable is as follows:

```
evalexpr “input=input1.txt;output=output1.txt”
```

You can call the example with another command line type,

```
evalexpr input=input1.txt output=output1.txt
```

5. 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
```

6. Requirements

- Homework is an individual. Your homework will be automatically screened for code plagiarism against code from the other students and code from external sources. If you copy/download source code from the Internet or a book, it is better for you to acknowledge it in your comments, instead of the TAs detecting it. Code that is detected to be copied from another student (for instance, renaming variables, changing for and while loops, changing indentation, etc) will result in ”Fail” in the course and being reported to UH upper administration.

- timeout is set to 2s.

7. Hand over your homework

- Homework 5 needs to be handed over to our Linux server, follow the link here http://www2.cs.uh.edu/~rizk/homework.html.

- Make sure to create a folder under your root directory, name it hw5 (name need to be lower case), only copy your code to this folder, no test case or other files needed. If you use ArgumentManager.h, don’t forget to hand over it too.