COSC2430 Hw2: Evaluate Arithmetic Expressions using stack

1. Introduction

You will create a C++ program that can evaluate arithmetic expressions with integer numbers ranging between -9223372036854775808 and 9223372036854775807. These numbers are long long integers (dependent on size of 64bit CPU register).

2. Input and Output

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

1. Operators are parentheses and basic operators including ( )+ － ＊ and /. The program should display the input expression and the results, separated with =.
2. You need to check whether the expression is valid or not. If one expression is not valid, output “error”, then continue to process next expression.

All records should be processed sequentially from begin to end. No very big numbers that exceed the capacity of long long integer will be given.

3. Program and input specification

The main C++ program will become the executable to be tested by the TAs. The result file should be write 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 get confused.

Assumptions:

- The file is a small plain text file (say < 10000 expressions); no need to handle binary files.
- Only integer numbers as input (no decimals!). Input numbers may have leading zeroes. Output number will be written without leading zeroes.
- Operators: + − * /.
- Parentheses: ( ) .
- Keep in mind a single + can be a sign instead of an operator.
- You can assume the input are only integers and the output is only an integer. Therefore, you can truncate the decimal part when evaluating division.
- do not break an arithmetic expression into multiple lines as it will mess testing.

The general call to the executable is as follows:

```
calculate “input=input21.txt;output=output21.txt”
```

Call example with another command line type.

```
calculate input=input21.txt output=output21.txt
```

Example 1 of input and output

Input21.txt

```
0*00000000000000000+0000000000000001
(1+2)*(1000+2000)
(+1+2)*(1000+2000)
((1+2)*(1000+2000))*(1+10000)
(1000000000000000-1)
99999999999999999/99999999999999999
(-100000+10000)*8
((-1))-(-1)
```

Command line:

```
calculate input=input21.txt output=output21.txt
```

output21.txt

```
0*00000000000000000+0000000000000001=1
(1+2)*(1000+2000)=9000
(+1+2)*(1000+2000)=9000
((1+2)*(1000+2000))*(1+10000)=90009000
(1000000000000000-1)=999999999999999
99999999999999999/99999999999999999=1
(-100000+10000)*8=-720000
((-1))-(-1)=0
```

Example 2 of input and output

Input22.txt

```
(3+5*2))
1000*100+20000
-(2+3+)(-3
123+123*-123
```

Command line:

```
calculate input=input22.txt output=output22.txt
```

output22.txt

```
error
1000*100+20000=120000
error
error
```

Example 3 of input and output

Input23.txt

```
-(-(-1))+3
(-1*(3+5))
(3+(5*2)-(6/2))+1
100/99+100/39
```

Command line:

```
calculate input=input23.txt output=output23.txt
```

output23.txt

```
-(-(-1))+3=2
(-1*(3+5))=-8
(3+(5*2)-(6/2))+1=11
100/99+100/39=3
```

4. Requirements

- Homework is 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.
- Remove leading zeroes from the result (e.g. 1, instead of 00001; 0 instead of 00000).
- timeout is set to 10s.

5. Hand over your homework

Homework 2 need 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 hw2 (name need to be lower case), only copy your code to this folder, no testcase or other files needed. If you use ArgumentManager.h, don’t forget to hand over it too.

ps. This document may has typos, if you think something illogical, please email TAs or Teacher for confirmation.