CS 311 - HW 1 - 100 points

**Stack Implementation**

Write a **stack** application program to evaluate post-fix expressions.

**Steps:**

1. Implement the stack: complete the attached stack.h and stack.cpp
2. Test your stack using the attached stacktest.cpp (Test1).
3. Implement the stack application program to evaluate post-fix expressions: complete the attached client1.cpp
4. Test your application program (Test2). You can double check your results with the attached hw1stackdemo.out. The results must be the same. You may implement your client1.cpp so that your user interface looks nicer, but the results (displayed numbers or errors) must be the same.
5. Copy-paste the tests results to your test1.txt and test2.txt files. These .txt files must also include the commands required to compile your cpp files. We will compile and run your cpp files and also check your .txt files as references.
6. Make a folder containing your files, compress it and submit your zip file to cougar courses. The name of this file must be your first and last name, for example HW1NahidMajd.zip

**Test1:**

You run this test to ensure your implemented stack works correctly. Once you completed the attached stack.h and stack.cpp, compile them with stacktest.cpp as follows

g++ stack.cpp stacktest.cpp

and run it using

./a.out

Then run the following test cases and enter the results to your test1.txt:

1. display the stack
2. push 1,2,3,4
3. display the top element
4. pop
5. display the top element
6. display the stack
7. clear the stack
8. display the stack

Important note: Study stacktest.cpp. For this HW, I gave you the test file. For next HWs, you will be asked to implement the test file.

**Post-fix expression evaluator program**

The user will enter a post-fix expression of the following form as a string. The program evaluates that using a stack and returns the result.

34+ which means 3+4

345+\* which means 3\*(4+5)

722+- which means 7-(2+2)

Note: Operators are +, -, and \*. Single digit numbers only. No blanks.

Note that the input expression could result in an error.

e.g.

3+ Stack underflow (not enough operands)

345 incomplete expression (not enough operations)

345354213512 stack overflow (too many operands). Note: stack size = 10

Your program will display the evaluated result (a number) or an error message describing the occurred error. Use the algorithm given in the stack lecture to complete client1.cpp. Use the attached hw1stackdemo.out to see how the program should work.

* You must use exception handling to handle errors.
* Since each element of the expression string is a character, you will need to convert it to an integer to perform arithmetic operations. Use the stringDemo1.cpp (attached to the lecture) for char to int conversion.
* Run the .out file using the following command:

./hw1stackdemo.out

**Test2**

You run this test to ensure the program you have implemented to evaluate post-fix expressions works correctly. Once you completed the attached client1.cpp, compile it as follows

g++ stack.cpp client1.cpp

and run it using the following command:

./a.out

**Required Test Cases (must test in the following order):**

1. 34+ which means 3+4

2. 345+\* which means 3\*(4+5)

3. 722+- which means 7-(2+2)

4. 34+56++ which means (3+4) + (5+6)

5. 12+34\*45+-+ which means (1+2) + ((3\*4) - (4+5))

6. 1234567891234 expression too long

7. + too few operands

8. 3+ too few operands

9. 3# invalid element

10. 2345+ incomplete expression

Note: Do you think I have listed all possible cases? Always ask yourself if there are other cases you should test to make sure your program is bug free.

**Submission**

Submit a zip file containing the following files.

1. stack.h (30 points) -- class header file
2. stack.cpp (30 points) -- class implementation file
3. client1.cpp (30 points) -- your implemented application
4. test1.txt (5 points) -- results of testing the stack class
5. test2.txt (5 points) -- results of testing the application

Important note1: You will miss up to 10 points if you don’t comment your programs.

Important Note2: Always make sure the files you submit can be compiled on **empress.csusm.edu** with no error. We will compile and test your files on empress.