Homework 2 -- Out: February 9, Wednesday -- DUE: February 22, Tuesday, 2:29pm

EC327 Introduction to Software Engineering – Spring 2022

Total: 100 points

Submission: **Failure to follow any of these guidelines **WILL** result in loss of points on the assignment.*

You will use Github Classroom to submit your solution. You can submit as many times as you want until the deadline without incurring in any penalty.

1. Accept the assignment "HW2" in the EC327 github classroom using this link: https://classroom.github.com/a/VSUWnDjV and clone the repository on your own local machine. If you are unfamiliar with git, we will cover how to use it during labs on the week of February 14.
2. Complete the problems in this assignment, saving your solution for each question in a separate file, named Q1.cpp, Q2.cpp, Q3.cpp
3. Stage (add the files)
4. Commit your changes
5. *Push your changes to the repository.*

Late Homework Submissions:

Recall the late submission policy and the **fixed penalty of 20%** for submitting up to three days after the deadline. Also recall that we only grade your latest submission (e.g., if you submit both on time and after the deadline, only your late submission is graded).

** You may use any development environment you wish, as long as it is ANSI C++ compatible. **Make sure your code compiles and runs properly under the Linux environment on the PHO305/307 (or eng-grid) machines before submitting. IF YOUR CODE DOESN'T COMPILE, YOU WON'T GET ANY CREDIT!***

** Comment your code and use a clean, easily understandable coding convention. A coding style guide is at this link, but many others exist online:*
https://google.github.io/styleguide/cppguide.html#Comments (see "Comments")

**** Please run the following command in terminal BEFORE you try compiling your code (without the quotation marks): "module load gcc". If you run "gcc --version" you should see that your compiler version is 5.2.x. This will allow you to use the "--std=c++17" flag when compiling your code with g++.***

Q1. Simple calculator. [35 points]

Write a C++ program (Q1.cpp) that takes simple, one operand arithmetic operations from the command line and calculates the result. The program should accept signed integers as operands, and support the *, /, +, - operators. When provided to the command line, operands and operators should be separated by a space, in the following way:

3 + 5

For the division operator, you should treat it as a floating-point division (with decimals). You should consider the error conditions listed in the sample runs below and terminate the program if such a condition occurs.

Text in < > in the examples demonstrates the user inputs entered via the keyboard.
The actual user inputs should not contain < or > signs.

Sample run 1:
```
Enter an expression to evaluate: <3 + 5>
Result: 8
```

Sample run 2:
```
Enter an expression to evaluate: <3 / 0>
Error! You can't divide a number by 0!
```

```
Sample run 3:
Enter an expression to evaluate: <3 % 0>
Error! '%' is not a supported operator.

Sample run 4:
Enter an expression to evaluate: <3 + b>
Error! You did not provide two numerals as operands.

Sample run 5:
Enter an expression to evaluate: <3 / 5>
Result: 0.6
```

IMPORTANT NOTE: YOUR CODE WILL GO THROUGH AN AUTOMATED GRADER SO MAKE SURE YOUR CODE TAKES THE INPUTS AS IN THE ABOVE EXAMPLE AND ITS OUTPUT FORMAT IS EXACTLY "Result: ...**" AS IN THE ABOVE EXAMPLE.**

Q2. ***Guessing game*** **[30 points]**
Write a C++ program (Q2.cpp) that generates a random positive integer (between 1 and 10) and asks the user to guess it by taking input from the command line. The program should not terminate until the user successfully guesses the correct number. Upon every input from the user, you should validate that the input is a number, and display the proper error message as listed in the sample run. You should also help the user by guiding them towards the generated random number, printing if the guessed number was higher or lower compared to the generated random number.

See the following test cases for examples of how the program should behave.

To generate a random number, you should use the srand() function from the cstdlib library, as explained here:
http://www.cplusplus.com/reference/cstdlib/srand/

```
Sample run 1:
Please guess the number: <4>
Sorry, your guess is incorrect. Your guess was too low.
Please guess the number: <56>
Sorry, your guess is incorrect. Your guess was too high.
Please guess the number: <30>
Sorry, your guess is incorrect. Your guess was too low.
Please guess the number: <36>
Congratulations, your guess is correct!

Sample run 2:
Please guess the number: <T>
Error! You didn't insert a number!
Please guess the number: <4>
Congratulations, your guess is correct!
```

IMPORTANT NOTE: YOUR CODE WILL GO THROUGH AN AUTOGRADER SO MAKE SURE YOUR CODE TAKES THE INPUTS AS THE ABOVE EXAMPLE AND ITS OUTPUTS ARE IN THE SAME FORMAT (SPACES, COMMAS, etc.).

Q3. ***Pyramid*** **[35 points]**

Write a program (Q3.cpp) that asks the user to provide a character, a signed integer between 1 and 20, preferred shape of pyramid (inverted or upright) and prints a “pyramid” on the screen, composed of the chosen character and with the chosen number as height. Upon receiving an invalid height, the program shouldn’t exit but it should print an error and ask the user to input another value (see sample run below). An invalid height is a number higher than 20, zero, a negative number, or a value that is not a number (e.g., ‘a’). You should also check for the shape of pyramid that the user enters, and make sure it is either **U** (upright) or **I** (inverted) (see sample run below). You can disregard the case in which the user inputs a decimal number.

Sample run 1:

```
Enter a character: <#>
Enter a height (between 1 and 100): <5>
Enter pyramid shape (U for upright / I for inverted): <U>
#
##
###
####
#####
```

Sample run 2:

```
Enter a character: <A>
Enter a height (between 1 and 100): <6>
Enter pyramid shape (U for upright / I for inverted): <P>
Error! You specified an invalid pyramid shape!
Enter pyramid shape (U for upright / I for inverted): <I>
AAAAAA
AAAAA
AAAA
AAA
AA
A
```

Sample run 3:

```
Enter a character: <7>
Enter a height (between 1 and 100): <65535>
Error! You specified an invalid height!
Enter a height (between 1 and 100): <4>
Enter pyramid shape (U for upright / I for inverted): <U>
7
77
777
7777
```

IMPORTANT NOTE: YOUR CODE WILL GO THROUGH AN AUTOGRADER SO MAKE SURE YOUR CODE TAKES THE INPUTS AS THE ABOVE EXAMPLE AND ITS OUTPUTS ARE IN THE SAME FORMAT (SPACES, COMMAS, etc.).