Homework 3 -- Out: February 23, Wednesday -- DUE: March 14, Monday, 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 "HW3" in the EC327 github classroom using this link: https://classroom.github.com/a/0KtK_aDy and clone the repository on your own local machine.
2. Complete the problems in this assignment, saving your solution for each question in a separate file, named Q1.cpp, Q2.cpp, Q3.cpp, Q4.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++.***

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.).

Q1. *Loops, functions.* [25 points]

Implement a function to calculate the greatest common divisor (gcd) of two positive integers. The function will need to have the following prototype:

```
int gcd(int x, int y);
```

Given two positive integers, it must return their gcd. Implement your lcm function in a file named Q1.cpp. Include the gcd function only in this file (plus any #includes that the function might need, the appropriate namespace declaration etc.). The gcd function should not check for invalid values (e.g., negative numbers as input). Instead, we provide two files (Q1.h and Q1main.cpp) to test your code, but you are not supposed to submit those for grading. **Submit only Q1.cpp**.

Q2. Strings. [25 points]

Implement a function replace with the following prototype:

void replace(string &s, char c1, char c2);

The function parses the string passed as an argument and substitutes any occurrence of character c1 with character c2. For example, replace("hello",'l','X') will modify the string to be "heXXo". Implement the function in a file named Q2.cpp. We provide two files (Q2.h and Q2main.cpp) to test your code, but you are not supposed to submit those for grading. **Submit only Q2.cpp**.

Q3. File I/O. [25 points]

You are given a file `grades.txt` in the format <student ID, class, grade> as follows:

123456 EC327 91
493218 EC330 67
904531 EC327 82

Implement a function calculateAverage with the following prototype:

float calculateAverage(string class, string filename);

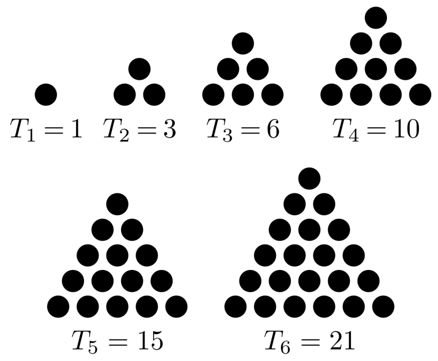
The function reads the file passed in the filename parameter and calculates the average grade for the class passed in the class parameter. For example, by calling calculateAverage("EC327","grades.txt") the function would parse the file above and return 86.5 (as a float). Implement the function in a file named Q3.cpp. We provide two files (Q3.h and Q3main.cpp) to test your code, but you are not supposed to submit those for grading. **Submit only Q3.cpp**.

Q4 *Recursion.* [25 points]

Write a recursive function to calculate triangular numbers. The function must follow this prototype:

int triangular(int num);

Given a positive number "num", it must return the correct triangular number (as an integer). For example, triangular(6) will return 21, according to the following scheme:



Implement your triangular function in a file named Q4.cpp. Include only the triangular function in this file (plus any #includes that the function might need, the appropriate namespace declaration etc.). We provide two files (Q4.h and Q4main.cpp) to test your code, but you are not supposed to submit those for grading (i.e., we will use our own main implementation). **Submit only Q4.cpp**.