Homework 5 -- Out: March 28, Monday -- DUE: April 13, Wednesday, 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 "HW5" in the EC327 github classroom using this link: https://classroom.github.com/a/WwhvidXg 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 `student.h`, `student.cpp, course.h, course.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 Class design [25 points]

Take the Student class that you implemented in HW4. Modify it so that all its members (`first`, `last`, `id`, and `gpa`) are **private**. In addition to the methods that you implemented in HW4, implement the following **public** ones:

- A copy constructor that allows to properly copy an instance of `Student.`
- A method `void setFirst(string f)` that takes a string as a parameter and sets it as the value of `first`.
- A method `void setLast(string l)` that takes a string as a parameter and sets it as the value of `last`.
- A method `void setId(unsigned int n)` that takes an unsigned integer as a parameter and sets it as the value of `id`.
- A method `string getFirst()` that returns the value of `first`.
- A method `string getLast()` that returns the value of `last`.
- A method `unsigned int getId()` that returns the value of `id`.

Save your Student class declaration in a file named `student.h`, and your implementation in a file named `student.cpp`.

<u>Files to submit:</u> `student.h`, `student.cpp`

Q2 Class design. **[75 points]**

Design a class named `Course` that handles the enrollment for a college class. Declare the following members and methods in a file `course.h`.

a) Declare the following members:

- A string data field named `name`, which holds the name of the course
- An unsigned integer data field `capacity`, which holds the maximum enrollment for the course
- An unsigned integer data field `numStudents`, which holds the number of students enrolled in the course
- A pointer to Student `students`, which points to an array of Student objects containing the information of all the students in the course

All members should be made **private**.

b) Declare prototypes for the following methods in `course.h`, and implement them in `course.cpp`:

- A no argument constructor, which sets name to "EC000", capacity to 1000, and `numStudents` to 0. It also allocates a `Student` array of `size` capacity on the heap, and makes `students` point to it.
- A constructor `Course(string n, int c)` which sets name to `n` and capacity to `c`. `numStudents` is set to 0, and the constructor allocates a `Student` array of `size` capacity on the heap, and makes student point to it.
- A destructor `~Course()` that properly deletes an instance of class `Course` when invoked.
- A copy constructor that makes a deep copy of a `Course` instance.

- A method `unsigned int getNumStudents()` that returns the number of students in the class.
- A method `unsigned int getCapacity()` that returns the capacity of the class.
- A method `unsigned int getAvailableSeats()` that returns the number of spots that are still available in the class.
- A method `addStudent` that adds a new student to the class. The method should check if there is space for the student in the class, and modify the `numStudents` variable accordingly. If the class is full, print an error message on the screen. The prototype of `addStudent` is as follows:

  ```
        void addStudent(string f, string l, int id, float g)
  ```

  The constructor should then create an instance of class `Student` with `first` set to `f`, `last` set to `l`, `id` set to `id`, and `gpa` set to `g`.
- A method `void dropStudent(string name)` that removes a student from the class. The method takes a string as a parameter and checks for an entry with that name in the `student` array. The string contains a concatenation of first and last name (e.g., "John Doe"), and the function should split this string in first and last name and check for an instance of `Student` that matches those parameters. If a student with that name is found, the student instance is removed from the array, and the array is rearranged so that there are no gaps in it. See the following example, in which instances of the class Student are represented as the concatenation of their first and last names. if the array pointed to by student looks like this:

  ```
        ["John Doe","Jane Doe", "Bruce Wayne"]
  ```

  And the function `dropStudent("Jane Doe")` is called, the array will look like:

  ```
        ["John Doe", "Bruce Wayne"]
  ```

  Note that since the array pointed to by student is private, we do not care about what is contained in the array elements that are beyond the `numStudents-1` index. If no student is found or the class is empty, the method should print an error message.
- A method `getStudents` that print the names of the students in the class, one per line. If the course is empty, the method should return the string "`The course is empty`".

All methods should be made **public**.

You should write a main function to test your `Course` class and the methods that you implemented, but you shouldn't submit this main file.

Files to submit: `course.h`, `course.cpp`