Project 1 – Grade Report (Part A)

For this project, you will be creating an application that produces a **grade report** for one or more students. The project has several parts; you need to fully implement each part before working on the next one.

Project name: You will be working on the project given to you, and later you will re-name the folder that contains the project according to instructions that will be given to you.

Max number of students to work on the project: Four (4) students. Your group needs to stay the same throughout the project.

Team name: You need to find a name for your team (even if you are a one-programmer team). We will be setting the teams in class.

The registrar's office at your college would like to have an application that prepares grade reports as soon as the students' grades are recorded. The application has a menu that allows the user to request specific reports:

1: Print all students
2: Print student information
3: Search student by last name
4: Print students by course
5: Print students on hold
6: Print students to file
7: To exit

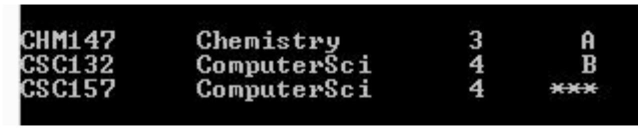
Grades are shown on the grade report together with the grade-point average (GPA). Whether the information is printed on screen or on a file, the report cannot print grades for those students who have not paid the tuition. For these students, the grade report will produce a message indicating that the grades have been held for non-payment of the tuition, showing the billing amount that is due.

The data that needs to be analyzed by your application is stored in a text file that will be available later on, when you implement the rest of the application.

For this section of the project, you will be implementing **two (2) classes**: **Course** and **Person**. The project already contains some implementation, and you will need to complete the rest according to the following specifications.

Course Class	
Member variables (already declared)	• **courseName**: Chemistry, ComputerSci, etc. • **courseNo**: CHM147, CSC201, etc. • **courseGrade**: A-D, F • **courseCredits**: These are the units for the course.

Default constructor	Initializes the course credits to 0 and the course grade to an asterisk (*).
Function **setCourseInfo**	**Parameters (in this order):** a course name, a course number, a grade, and the credits. Re-sets all member variables to the given values passed by the parameters.

Function **printCourseInfo**	**Parameter:** A **Boolean** value indicating whether the grade can be printed (if the tuitions was not paid, then the grade will not be printed).
	
	The function prints the grades in this format:
	
	
	`CHM147      Chemistry       3      A`
	`CSC132      ComputerSci     4      B`
	`CSC157      ComputerSci     4    ***`
	
	DO pay attention to indentation (use **setw** to format the output).
	
	If the parameter is **false**, then instead of the grade print ***.

Function **getCourseCredits**	Returns the course credits.
Function **getCourseNumber**	Returns the course number.
Function **getCourseName**	Returns the course name.
Function **getCourseGrade**	Returns the course grade.
Destructor	(empty)

Person Class (You will need to create the **Person.h** and **Person.cpp** file in the appropriate **folders** of your solution.)	
Member variables	A **string** storing the last name and a **string** storing the first name.
Default constructor	Although there is no need to initialize a string, because the **STL string class** already has a constructor that initializes strings to empty strings, you need to set the member variables to **N/A**.
Function **setName**	**Parameters (in this order):** first and last name of the person. Changes the value of all member variables to the given values passed by the parameters.

Function **getLastName**	Returns the last name of the person.
Function **getName**	**Parameters:** A string that stores the first name and a string that stores the last name. It re-sets the parameters to the values retrieved from the member variables. (**Note** that the function does not return any value, but it transmits the new value through the parameters. How does it do that?)
Function **printName**	Prints last name and first name in this format: `Smith, Jane`
Destructor	(empty)

The **Main.cpp** file does **not** contain any testing cases. It is your job to create temporary testing cases to test your program.

A Few Rules:

- Add the **const** modifier to functions when needed.
- **Pass by reference** if needed (and use the **const** modifier if needed).
- Pay attention when **naming your functions**; you will be testing your functions with other files that need to match all names given.
- Pay attention to the way the **output** should be formatted.
- Do **NOT** remove any of the code already given.
- When implementing your code, write the functions in the order specified by the comments and the instructions given.