CS 201: Problem Solving & Programming II
Lab #4

Exercise 1: Class Inheritance and Composition Step-by-Step guide

For this assignment, we're going to create a small system for course information. This will include class objects like Course, Student, and Instructor, and we will load data about each of these from three files - "course.txt", "students.txt", "instructor.txt".

1. Loading in Course Information

Let's begin with just loading in the course information. The course.txt file is a fixed-size containing the following information:

- *Cou*rse Name
- Course ID
- Number of S*tudents*

The course name may have spaces, so will need to be loaded with the **getline** function.

1-A. Load in data from "Course.txt"

First, create our ***main.cpp*** file where you will have **int main()**. Create an input-file-stream object (**ifstream**) for loading in course.txt.
(We're not going to worry about creating the class yet).

Open the course.txt file with our ifstream. We will not need to create a while loop to read this document because it is of a fixed-length and we know the order that the data will be in.

Use the getline function to read each line consecutively, storing the data in three separate variables declared within **int main().**

```
getline( myIfstream, myString, '\n' );
```

Use the version of getline that is a standalone function – don't use the version that is a member of a class.

Once you've used **getline** to load in the course name, id, and number of students, **cout** the data just to verify.

Don't forget to close your ifstream file afterwards!

1-A Goals:

- Open "course.txt" file
- Store each line of "course.txt" in three variables

- Output the stored variables

1-B. Create a class for the Course

Create two files – Course.h and Course.cpp. Create a class named “Course”. The course will eventually store Students and Instructor, but we'll worry about that later.

First, create private members for this data:

- Course ID
- Course Name
- Amount of Students

These can all be strings.

Setup/Setter Functions
Next, create a function (or functions) to change the values of Course ID, Course Name, and Amount of Students member variables. This can be one big “Setup” function with three parameters, or three small “Setter” functions with one parameter each.

Output Function
Create one more function called “Output” - it can be void and have no parameters. The point of this function is simply to output Course ID, Course Name, and Amount of Strings.

Implement these functions so we have a way to pass in an argument and store that value in the member variables.

Updating main()
Now, back in **main()**, we're going to add on to our original code which loaded information from “course.txt”.

Create an instance of Course – you can name it anything. This is our “Course” variable.

We're going to keep those variables we were originally storing the data in – These are ***buffer variables*** and we store the data in these variables while we're still waiting on all of the data to be available.

We're going to use these buffer variables as arguments. When you call functions in the “Course” class to update member variables (ID, Name, Amount of Students), you will pass in these buffer variables.

Now, erase the **cout** for the buffer variables, and replace that with **course.Output();**
Run the program and verify that course.Output() displays the information correctly.

Does it build and run?

2. Loading in Instructor information

The instructor file is also a fixed-length file. First work on loading in the data and outputting it, similar to how we did in 1-A.

After that, we're going to need to create a class for it, but that also requires the parent class.

2-A: Create the “Person” class

Create two new files – Person.h and Person.cpp.

Create a class named “Person”, with two **protected** member variables:

- Name
- Date of Birth

These variables can be strings.

That's all we need to do for the Person class right now. If you want, you can create a Student.h, Student.cpp, Instructor.h, and Instructor.cpp files, or you can store these class declarations within Person.h.

2-B: Create the “Instructor” class

Create a class named “Instructor”. This class needs to inherit from Person.

Create two more members for Instructor, either as **private** or **protected**:

- Office
- Designation

These variables can be strings.

Also create two functions:

- void Setup(…)
- void Output()

The Setup(...) function will take in parameters to store all four of the Instructor members – Name, DOB, Office, and Designation.

The Output() function will be responsible for **cout**-ing all four member variables.

2-C: Storing instructor information in the class

Back in **main()**, we should already be loading information from the “instructor.txt” file and storing those in **buffer variables**.

Now, create an instance of the Instructor class somewhere in main().

Keep your buffer variables, and pass them into the Instructor class' **Setup** function so we are now storing this data within the class.

Afterwards, validate that it worked by calling the Instructor class' **Output()** function.

Is it working?

2-D: Update the Course class to store an Instructor.

We are supposed to be storing the Instructor variable within the Course class.

Go back to Course.h and add an “Instructor” member that is **public** (for now).

Since the instructor is public, we should be able to access it from main() something like:
myCourse.instructor.Setup(buffer1, buffer2, buffer3, buffer4);

Replace the original instance of Instructor created in 2-C. Instead, access the instructor variable directly that is within myCourse.

3. Handling Student information

3-A: Loading Student information

Before we worry about the Student class, let's just load in the student information. student.txt is different from instructor.txt and course.txt in that it contains a lot of data, for different people:

- Student Name
- ID
- DOB
- CGPA

So we are going to want to read four lines at a time – each four lines are for one person.

Start by creating an **ifstream** file to load in student data and load student.txt.

Create a while loop that continues looping until the ifstream hits the End Of File eof().

Within this while loop, we will need to declare four variables for each line that is being loaded from the file. These are our **buffer variables**.

All we're going to do is use the **getline** function with these buffer variables, then immediately **cout**-ing the data. For the time being, we don't care about saving this data.

Make sure that everything is loaded in and then cout-ed.

3-B: Creating a Student class

Now, create two new files: Student.h and Student.cpp. Create a class named Student. Like the Instructor class, this will also inherit from the **Person** class.

Once the Student inherits from Person, add two new member variables to Student:

- ID
- CGPA

Both of these can be strings.

You will also need two functions:

- void Setup(…)
- void Output()

The **Output** function will, again, output the member variables: Student Name, Student DOB, Student ID, and Student CGPA.

The Setup function will receive four parameters which we will use to set the private member variables.

Implement these functions, then head back to main().

3-C: Storing file data in the class

In **main()**, create an **array of Student** objects. This array will be no bigger than 20 items.
Also, create an **index** variable. This is an integer, and should start at 0. It is what we will use to store our Students from the file in order.

Within the while loop where we're loading in student data, keep all of the buffer variables.

Use the Student's **Setup** function and pass the buffer variables in, to be assigned to the Student's private members.

Remember, we have an array of Student items. We will access them one at a time, using the **index** key.

3-D: Store the Student array within the Course

Just like with the Instructor class, we're going to pull the Student array out of main() and into the Course class.

In the Course class, add an array of 20 Students into Course, as a **public** member.

Back in main(), remove the declaration of the Student array. Now, fix all the places that referenced the student array to now use the Course class' member array.

4. Last stuff

There are still some things to do after this, such as making the Instructor and Student members private within the Course class. Try to update the rest yourself!