CSCE 2004 - Homework 5
Due Date - 11/10/2016 at 11:59 PM

0. Overview:

In this assignment, you will develop a program to handle university degree auditing and course management. That is, after taking a set of courses, a student would like to know if she/he has met the graduation requirements for a particular degree (or what the student is missing to meet those requirements). The requirements include required courses, total credit hours, minimum GPA, etc. As a college student, can you see the value of such a tool in the real world?

To build any software system, one must be equipped with the following: a firm grasp of the problem to be solved, the ability to divide the problem into smaller problems, devising related algorithms to solve the smaller problems and combining their solutions, the ability to translate the high level design into a computer program in a particular programming language (in our case C++), and the ability to debug and test the program.

All of our projects in this course are related to this core problem. So before you begin, you may want to review your own degree requirements to make sure you really understand them. If we are not able to understand a problem, then no matter how good we are as programmers or software engineers, we will not be able to solve that problem.

Note that the behaviors of the program developed in project four are identical to that of project three. Project five has all the functionalities of Project four and the additional features of reading the list of courses that a student has taken from a file, writing the current list of courses (stored in various arrays) to a file, and a new menu choice (F or f) of deleting. Project five is an extension of project four and most of the code of project four should stay the same. Here we have another chance to fix bugs of previous project. You should fix the bugs of project four first if any.

1. Problem Statement:

The primary goal of this programming assignment is to give students experience with text file processing, writing and using functions, array data structures, if statements, and loops. A secondary goal is to give students a chance to modify an existing program to make it more useful. To meet these goals, students will be required to improve their previous course and course management program to make the program more modular, flexible, and robust. Students will be required to implement all the functions EXACTLY as specified in the design section.

Enhance features: Three features will be added to our current program (project four). The first feature is to save the current course information to the default file ("courses.txt") when the program terminates. The second feature is to give the user a choice of either entering the course information interactively as being done in Project four or asking the program to read in the course information from a file, which is either the default file or a file specified by the user. The third feature is to include another menu option where the user may delete a course from the

current list of courses. Please run the sample executable (hw5, see instructions on blackboard) provided to understand the expected behavior of your program.

Students are allowed to use any combination of **"built-in" member functions related to files**, **user defined functions, arrays**, **if statements**, and **loops** (**while** or **for**) that they need to complete this assignment. You are **not** required or allowed to look ahead to more advanced C++ features like user defined classes to perform this task. Students must develop and use **all** the functions specified and *may* introduce additional functions if they prefer (you have more freedom as we move forward).

2. Design:

The default file is called "courses.txt". This is a text file and a sample of its content is shown below:

```
2
Programming Foundations 1
Spring 2014
csce2004
A
4
Calculus 1
Fall 2013
math2554
B
4
```

The first line tells us how many courses are in the file (2 in this case). It is followed by the courses. Each course is presented by a group of five lines for course name, time the course was taken, course number, letter grade, and credit hours in that order.

We may assume that the information and format of the file are correct when we open the file for reading. When we write the courses (stored in our arrays) to the default file, we should follow this format as well.

We will encapsulate reading from a file and writing to a file in two functions. For this project, the interface for each function will be provided for you. If you prefer, you may introduce additional functions and design them as you see fit.

Here are the interfaces or headers or prototypes of the functions:

A.
```
bool reading(const char filename[], string name[], string
time[], string number[], char grade[], int hours[], int& n,
int capacity){
```

```
    // filename is course file on disk to be read
    // name is an array of course names
    // time is an array of semester and year a course was taken
    // number is an array of course numbers
    // grade is an array of letter grades
    // hours is an array of credit hours
    // n is the number of courses read upon return
    // capacity indicates the max elements the arrays may hold
    // the function returns true if file open and reading
    // are successful, otherwise it returns false
    // the function reads course information from the file
    // and stores the information to the arrays
    }
B. bool writing(const char filename[], const string name[],
    const string time[], const string number[], const char
    grade[], const int hours[], int n){
    // filename is course file on disk to be written
    // name is an array of course names
    // time is an array of semester and year a course was taken
    // number is an array of course numbers
    // grade is an array of letter grades
    // hours is an array of credit hours
    // n is the number of courses in the arrays
    // the function returns true if file open and writing
    // are successful, otherwise it returns false
    // the function stores course information of the arrays
    // to the file, similar to listing function of project 4
    }
```

For the delete feature, we need to think about how to present the course information to the user so that the user may indicate the course to delete, which may be translated into any index of the array. We may also want to present the course to be deleted for getting a confirmation from the user. Even though many designs are reasonable for this option, we will have **a uniform design** for this project as specified by the executable file hw5.

To encapsulate the delete operation as a function is considered good software design. We give you the opportunities to make your decisions, including whether or not to have such a function and the functionalities, the interfaces of such a function.

3. Implementation:

First correct any error of hw4. Then develop one feature or a function at a time and leave all other code the same and test it thoroughly before we move to the next feature or function. We would suggest that you work first on writing to a file feature and function. Note that this function is almost identical to the listing function of project four with extra code for file processing. Use the existing code to test the implementation of this function.

After that we would suggest that you develop the reading function (and testing it with default file name hard coded in the call) and then the code asking the user to select either the default file or the file that the user will enter.

We would suggest that you work on the new menu choice last.

Since you are starting with your previous program, you already have something that compiles and runs. Since you must add new features and maintain the current behavior of the program, it is important to make these changes **incrementally** one function at a time, writing comments, adding code, compiling, and debugging. This way, you always have a program that "does something" even if it is not complete.

4. Testing:

Test your program to check that it operates correctly for all of the requirements listed above. You may use the given executable file to help you. Also check for the error handling capabilities of the code. Try your program with several input values, and save your testing output in text files for inclusion in your project report.

5. Documentation:

When you have completed your C++ program, write a short report (less than one page long not counting program testing output) describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this report in a separate text file to be submitted electronically. Please use the **Project Documentation Template** provided on Moodle as the basis for your documentation.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic analysis of all programs that are submitted. When you have completed the tasks above go to Blackboard to "upload" your documentation (a single **.pdf**, **.doc**, or **.docx** file), and your C++ program (a single .**cpp** file). Do **NOT** upload an executable version of your program or files in other extensions or format.

The dates on your electronic submission will be used to verify that you met the due date above. **No late projects will be accepted for credit.** Please refer to class syllabus about no late assignment policy as well as suggestions given regarding to submitting a partially working program.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so please hand projects in on time.

7. Food for thought:

In addition to the 5 pieces of data associated with a course, is there other information or any attribute we should add to a course?