CSCE 2004 - Homework 4
Due Date - 10/28/2016 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.

Project three allows us to store a list of courses entered and provides the menu that allows the user to carry out a number of tasks. Instead of adding more functionality to the current code base, we will use functions to make our main program shorter and our overall design more modular. We also have another chance to fix bugs (if any) of our previous projects. The functionality of project four is identical to that of project three.

1. Problem Statement:

The primary goal of this programming assignment is to give students experience with 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 modular. 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 modularity: Without functions, we have to have a very long main program and we have to copy and paste similar code, then make small adjustments often. With functions, we may identify major computation elements and separate them from the rest of the program. We may also identify repeated or similar tasks and turn them into reusable functions. Then our program becomes more modular and hopefully easy to maintain and debug. Here are a few major tasks that we can identify in our project three:

1) Compute the GPA for a list of courses

2) Compute the GPA for a particular semester
3) Calculate the total credit hours of courses with a grade of ‘D’ for a list of courses
4) Display a list of courses
5) Read in one course
6) Display a menu and get a user’s menu selection choice

Please note that the above are **not** menu choices; they correspond to tasks that our functions should implement. We will develop functions for each of the tasks above. For example, with the function for task 5), the first part of hw3 (reading courses into arrays) may be implemented by calling this function inside a loop. This function may also be called for menu choice ‘E’ or ‘e’ from hw3. We will factor the common logic into a function that can be used multiple times. The other functions should make the program logic more modular.

Students are allowed to use any combination of **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:

Since the functionality of project four is identical to that of project three, the only design questions are related to the function design. Function design involves the function name to use, the return type, the parameters and their type and ordering, and finally the actions to be performed. The information above is called a **function interface.** 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.
```
double gpa(int n, const char grades[], const int hours[])
{
   // ‘n’ is the number of courses in the array (the size)
   // ‘grades’ is an array of letter grades
   // ‘hours’ is an array of credit hours
   // The function returns the gpa calculated in the function.
   // The basic idea and logic of the body of this function
   // should be the similar to project 3.
}
```

B.
```
double semesterGpa(int n, const string times[], const char
grades[], const int hours[], string semester)
{
   // ‘n’ is the number of courses in the array (the size)
   // ‘times’ is an array of semester and year a course was taken
   // ‘grades’ is an array of letter grades
   // ‘hours’ is an array of credit hours
   // The value of ‘semester’ contains the semester and
```

```
    // year for which we want to compute the gpa.
  }

C. int DRule(int n, char grades[], int hours[])
  {
    // The meanings of the parameters are similar to that in
    // earlier functions.
    // This function returns the sum of the credit hours of all
    // courses where the grade is a ‘D’.
  }

D. void  print(int  n,  string  names[],  string  times[],  string
   numbers[], char grades[], int hours[])
  {
    // ‘names’ is an array of course names
    // ‘numbers’ is an array of course numbers
    // The meanings of all other parameters are similar to
    // those in the earlier functions.
    // The function lists all course information, one piece per
    // line in the order in which the user entered the course
    // information. E.g.: name, time, number, grade, hours.
    // Also, the first line should be the number of courses
  }

E. void getCourse(string& name, string& time, string& number, char&
   grade, int& hours, int n)
  {
    // Get a new course from the user, and put the values into
    // each of the function parameters.
    // Note that since we must pass the value read in this
    // function back to the caller, many parameters are passed
    // by reference, the & after the type.
    // ‘name’ for the course name.
    // ‘time’ for when the course was taken.
    // ‘number’ for course number.
    // ‘grade’ for the letter grade of the course.
    // ‘hours’ for credit hours of the course.
    // ‘n’ for the next course count.
  }

F. char menu()
  {
    // This function does not have any parameters and returns a
    // a character, which is the user’s menu choice.
    // In this function, the menu is shown,
    // then the user’s choice is read and validated.
    // Finally, it returns the validated menu choice (e.g. ‘A’).
  }
```

3. Implementation:

First correct any errors from hw3. Then, develop one function at a time, leaving all other code segments the same. Test your work thoroughly before you move on to the next function. We would suggest that you work first on functions A, C, B, and D in that order. Note that almost all the code of the body of each function is available in hw3. Your main task is to map the parameters (or rename some of the variables in the pasted code) within the function. Use the existing code to test the implementation of each function.

After that, we would suggest that you develop function E of the design and first use it from Menu choice E for testing. When you are satisfied that it works properly, use it in the loop that appears before the user is prompted with the menu.

We would suggest that you work on the menu function (F) last.

Since you are starting with your previous program, you already have something that compiles and runs. Since you must add several functions to this program and maintain the same behavior, 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:

How could we store all the courses (represented by the 5 pieces of data) to a file and read in the information from a file?

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