Project #02: Implementing the GradeUtil API

Complete By: Thursday September 12th @ 11:59pm
Assignment: Implement the GradeUtil API
Policy: Individual work only, late work is not accepted
Submission: via GradeScope

Assignment

In project #01 you did the hard work of developing test cases against a set of Grade Analysis utility functions. In this assignment you get to reap the benefit of that hard work and use those test cases to help you implement those functions. If you didn't finish project #01, you should make sure you have read the project #01 handout so you understand the API you are going to implement. To help you get started, we'll provide a few unit tests (but you'll have to write more).

Assignment Details

The details are simple: implement the 11 functions defined in the "gradeutil.h file". A corresponding C++ file, "gradeutil.cpp", is being made available to help you get started --- this is a C++ file of header comments and empty function stubs. The files can be found in the instructor's box page under Projects > Release, with files containing different end of line encodings.

There are no changes in the API from project #01, so refer to the latest version of the gradeutil.h file from the previous assignment and check that there are no differences between your understanding of how the functions are supposed to work and how the functions are described to work in the documentation comments for the API.

Assignment Requirements

The goal is to program in modern C++, not C. So no pointers --- references yes, pointers no. When it comes time to sort in the FindCourses(…) functions, you *must* use the built-in std::sort; do not write your own. Whenever possible, use range-based for loops (aka "foreach"), and consider using std::find_if when searching.

Programming environment: Codio *or* the IDE of your choice

In this assignment you are free to use whatever programming environment you want. Submissions will be

collected using Gradescope, but you are not required to use Codio. We encourage the use of Codio since it will be familiar from project #01, and Codio allows the staff to access your work if you need help. But Codio is not required in this assignment.

If you plan to work in another programming environment:

1. Install Catch testing framework, which can be as simple as download the "catch.hpp" file or if on Linux, executing sudo apt-get install catch
2. In Codio, export your C++ files
    a. Project menu, Export as Zip
    b. Extract main.cpp and test*.cpp files (also extract makefile if needed locally)
3. Download gradeutil.h and gradeutil.cpp from Box
    a. Projects>Project02>Release
4. **If** you are using the makefile downloaded from Codio, you'll need to edit so that it compiles against gradeutil.cpp
    a. Open makefile in editor
    b. Under all: in g++ command change ./util/gradeutil.o to gradeutil.cpp
    c. Under one: in g++ command change ./util/gradeutil.o to gradeutil.cpp
    d. For this assignment, delete / ignore all2, all3, all4, …, all10
    e. File >> Save
5. Build / make all
    a. Most / all of your tests should fail because gradeutil.cpp is just empty function stubs
6. Start implementing the functions in gradeutil.cpp, one by one
7. Change your code in the functions that fail until "make all" passes all tests
8. Submit on gradescope and see how you do on our test cases
9. If Gradescope score < 100 then { add more local test cases; goto step 7; }

Getting started…

Start by implementing the **ParseCourse(…)** function. Recall that our first lecture outlined a C++ program that input from a CSV file, and thus had to parse CSV strings. The lecture notes are available in Box.

When in doubt, use C++, not C. For example, when searching for courses that match the instructor prefix, take advantage of the methods in the string class, documented here. In general, to search for documentation on modern C++, I typically either (a) google search using the std:: prefix, e.g. **std::string** or **std::vector**, or (b) search the **C++ Reference** web site: https://en.cppreference.com/w/ .

If you did not complete project #01, a few sample unit tests are available in Box in the Project02 folder. These are not a complete set, you will need to write more in order to test the entire GradeUtil API. But these will help you get started.

Electronic Submission and Grading

Grading will be based on the correctness of your GradeUtil implementation, i.e. the number of our unit tests that you pass. We are not concerned with efficiency at this point, only correctness.

When you are ready to submit your program for grading, login to Gradescope and upload your "gradeutil.cpp" file; you can upload directly, or in the format of a .zip file exported from Codio (Project menu, Export as Zip). You have unlimited submissions, and Gradescope keeps a complete history of all submissions you have made. By default, Gradescope records the score of your last submission, but if that score is lower, you can click on "Submission history", select an earlier score, and click "Activate" to select it. The activated submission will be the score that gets recorded, and the submission we grade. If you submit on-time and late, we'll grade the last submission (the late one) unless you activate an earlier submission.

Policy

Late work is not accepted. Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them (e.g. your "iClicker teammates"), this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml .