CSE 241 Homework Assignment 2

DUE

March 28, 2019, 23:55

Description

• This is an individual assignment. Please do not collaborate.
• If you think that this document does not clearly describe the assignment, ask questions before its too late.
You won't be given a chance to correct any mistakes.
Scope: University Management System
You are expected to develop university management system.
Main Problem: Course Enrollment
Problems:

1. Students need to enrol courses. It is not allowed to enrol too much courses where sum of their credits is more than the credit right of student. It is allowed them to equal.
2. Students mustn't enrol courses whose most hours are overlapped. For example, a course with 3 hours can be overlapped on one hour with another course, Student can enrol these both courses. But if they are overlapped on 2 hours, it cannot enrol one of them.
3. Students can extract their schedule into .txt file.

Models:

1. **Student:** Keeps student information and performs functionality of students. (Class)
   Information: Name, surname, student_no, level
   Functionality:
   - enrollCourse(Course):int -> the student enrolls course, return credit remained
   - disenroll(Course):int -> the student disenrolls course, return credit remained
   - extractSchedule():string ->
   - enterClassroom(Classroom):void -> student enters classroom
   - quitClassroom(Classroom):void -> student quits classroom
2. **Course:** keeps course information. (Struct)
   Information: id, name, code, credit, total hours, lecture dates, field
3. **Classroom:** keeps classroom information. (Struct)
   Information: id, c_no, capacity, student_inroom

Test Scenarios

1. Read txt file for courses and classrooms. Student will enter his information. It selects from course lists. If it is not allowed to enrol course, program display reason and expect new course. When enrolment finish, program will assign classroom to courses and show time table on day of week.
   (input) >-ns Murat Boz 111 4
   (input) >-ns Seda Akyol 112 1
   (input) >-ns Zafer Çalış 113 2
   (input) >-list
   (ouput) > (1) Türkçe (2) SosyalBilgiler (3) BedenEğitimi (4) Müzik
   (input) > -e 111 1
   (output) > DONE! [|| (OR)] (output) > BLOCK! Because of (credits [||] Overlap) [||] (output)>
   ERROR: NO STUDENT (enroll)
   (input) > -de 111 1
   (output) > DONE! [|| (OR)] (output) > BLOCK! Because of (credits [||] Overlap) [||] (output)>
   ERROR: NO STUDENT (disenroll)
   (input) > -o 111
   (output) >Murat Boz: [courses which Murat boz enrolled are listed with their credits] total credits:
   [total credits] credit right: [credit right for 4. Level student] 23
   (input) > -o
   (output)>Murat Boz: [] total credits:0 credit right: 23
   >Seda Akyol: [] total credits:0 credit right: 20
   >Zafer Çalış: [] total credits:0 credit right: 21
   (input) > -o -f 111 file.txt
   (output) > DONE! file.txt
   (input) > -o -f file.txt

(output) > DONE! file.txt
(input)> exit

2. All information needed is assigned in main function. Program will calculate three value;
    a. At most How many courses should be opened according to number of student and capacity of classrooms?
    b. At most How many students can enrol courses according to number of courses and capacity of classrooms?
3. All functions in the program will be executed and tested with 3 different values.

Remarks

• Do not use any elements which is not covered in class.
• Do not submit your code without testing it with several different scenarios.
• Write comments in your code.

Turn in:

• Source code of a complete C++ program. Name of the file should be in this format: <full_name>_<id>.cpp.
• Example: gokhan_kaya_000000.cpp. Please do not use any Turkish special characters.
• You don't need to use an IDE for this assignment. Your code will be compiled and run in a command window.
• Your code will be compiled and tested on a Linux machine(Ubuntu). GCC will be used
• Make sure you don't get compile errors when you issue this command : g++ <full_name>_<id>.cpp.
• A script will be used in order to check the correctness of your results. So, be careful not to violate the expected
output format.
• Provide comments unless you are not interested in partial credit. (If I cannot easily understand your design, you may loose points.)
• You may not get full credit if your implementation contradicts with the statements in this document.

Late Submission

• (0,24] hours: -20%
• (24,48] hours: -40%
• (48,72] hours: -60%
• (72,-) hours: -100%

Grading (Tentative)

• Max Grade : 100.
• Multiple tests(at least 5) will be performed.
All of the followings are possible deductions from Max Grade.
• #define HARD_CODED_VALUES -10.
• No submission: -100. (be consistent in doing this and your overall grade will converge to N/A) (To be specific: if you miss 3 assignments you'll get N/A)
• Compile errors: -100.
• Irrelevant code: -100.
• Major parts are missing: -100.
• Unnecessarily long code: -30.
• Using language elements and libraries which are not allowed: -100.
• Not caring about the structure and efficiency: -30. (avoid using hard-coded values, avoid hard-to-follow expressions, avoid code repetition, avoid unnecessary loops).
• Significant number of compiler warnings: -10.
• Not commented enough: -5. (Comments are in English).
• Source code encoding is not UTF-8 and characters are not properly displayed: -5. (You can use 'Visual Studio Code', 'Sublime Text', 'Atom' etc. . . Check the character encoding of your text editor and set it to UTF-8).
• Missing or wrong output values: Fails the test.
• Output format is wrong: -30.
• Infinite loop: Fails the test.

- Segmentation fault: Fails the test.
- Fails 5 or more random tests: -100.
- Fails the test: deduction up to 20.
- Prints anything extra: -30.
- Unwanted chars and spaces in output.txt: -30.
- Submission includes files other than the expected: -10.
- Submission does not follow the file naming convention: -10.
- Sharing or inheriting code: -200