Data Structures
Assignment #1

Dates:
Released: Thursday Feb 20, 6pm.
Submission deadline: **Saturday Feb 29, 11:55pm**

This homework assignment aims at practicing the concepts of C++ programming language by suggesting a data structure to store the information of an academic department. For simplification, we consider that a department contains students and courses and some other functions to manipulate this information. Students and courses are stores in two dynamic arrays.

You need to write three classes CDept, CStudent and CCourse. You should separate the class declaration (.h) from the class definition (.cpp). Write a main() function where you should test all the functionalities of these classes. We expect you to provide a clean and error free source code that can be compiled and run without any compiler and run time error.

Constants:
The following constants should be used to define the above mentioned classes. These constants should be defined in a separate file "constants.h".

```
#ifndef constants_h
#define constants_h
const int MAXNBST = 10 ; // max number of students in the Dept.
const int MAXNBCR = 10 ; // max number of courses offered in Dept.
const int NBEXAMS = 5 ;  // max number of exams in a course
const int MAXSTCR = 7 ;  // max nbre Students in a course
const int MAXCRST = 4 ;  // max nbre of course to be taken by a student
#endif
```

Class CDept:
A department is to be defined with the following attributes and member functions:

- Name, char *
- students: an array of CStudent of size MAXNBST
- nbStudents, int: the number of students currently registered in the department
- courses: an array of CCourses of size MAXNBCR
- nbCourses, int: the number of courses currently offered by the department
- The constructor of CDept should allocate arrays students and courses, then fill them by calling the following functions (in this order):
  createStudents() ;
  createCourses() ;
  enrollStudentInCourses() ;
  createDefaultGrades() ;
  All the above four functions should be declared private and should be called only by the constructor during the construction of the department.
- The code of the functions CDept::createStudents() and CDept::createCourses() is provided.

- The function enrollStudentsInCourses() uses the default set of students and courses created above and enroll each student in few courses. One way to do that is: Traverse the array of students, for each student generate a random 0<=nbc< nbCourses in the department, then call the function enroll() to enroll the current student in nbc courses chosen randomly.
- enroll(): receive a pointer to a student (ps), and a pointer to a course (pc). If neither the student no the course is full, then this function should call CStudent::enroll() to insert pc in the array of courses of ps. Similarly, this function should also call CCourse::enroll() to insert ps in the array of students enrolled in the course pc.
- displayStudents(): display all the students registered in the department
- displayCourses(): display all the courses offered by the department
- addStudent(): add a new Student to the list of students registered in the department. Make sure that this doesn't exceed the maximum number of students, MAXNBST
- addCourse(): add a new Course to the list of courses offered. Make sure that this doesn't exceed the maximum number of students, MAXNBCR
- enterStudentGrades(): receive a pointer to a student, ps, a pointer to a course, pc, and an array of grades, gr. It enters the content of gr in the row corresponding to course pc in the attribute grades of ps.
- Add the functions that accomplish the following search operations:
    1. *Find the courses taken by a particular student,*
    2. *Find the best student in a particular course,*
    3. *Find the student who took at least three courses and has the highest average grade in the department,*

Class CStudent:

A student is defined with the following attributes and member functions:

- name, char *
- ID, int
- maxCourses, int: the maximum number of courses the student can take
- courses: that is an array of pointers to the courses in which the student has registered
- nbCourses: the number of courses currently taken.
- grades: a 2D array to store the grades obtained by the student in the exams of each of his courses. Knowing that courses should have the same number of exams (NBEXAMS) defined as a const of type int. e.g. if a student is registered in maxCourses= 7, then the attribute grades should be allocated dynamically as an array of 7 rows and NBEXAMS columns.
- enroll(): receive a pointer to a course, adds this pointer to the array of courses, and increases the attribute nbCourses.
- setCourseGrades(): receive a course index (0<=ci<NbCourses) and an array of scores to be used to set the grades of course ci.
- setExamGrade(): receive a course index (ci), an exam index (0<=ei<NBEXAM), and a score to be stored as the grade of exam ei in the course ci.
- displayCourses(): traverse the array of courses and displays the information of each course (code, name, capacity, and number of students enrolled) by calling the method CCourse::displayInfo().

- displayInfo(): display student information e.g. name and ID.
- calcAverages(): calculate the average grade for each of the courses taken by the student, returns the averages as an array.
- You may add getters and setters as needed.
- You may also add other useful functions as you see fit eg. isFull() to check if the student reached the maximum number of courses a student can take, MAXCRST. isEnrolledIn(CCourse *pc) to check if the student is enrolled in a particular course.

Class CCourse:

A course is to be defined with the following attributes:

- name, char *
- code, int
- capacity, int: the maximum number of students who can enroll in the course, this is set to be equal to MAXSTCR.
- enrolled: that is an array of pointers to the students registered in this course. This array should be allocated to contain capacity pointers to CStudents.
- nbEnrolled: the number of students who are currently registered in this course.
- enroll(): receive a pointer to a student, if the course is not full, the function adds this pointer to the array of students enrolled, and increases the attribute nbEnrolled.
- displayStudents(): traverse the array of enrolled students and calls CStudent::displayInfo() to displays the information of each student.
- displayInfo(): display course information. e.g. name, code, capacity, number of students enrolled.
- calcAverages(): calculate the average grade for each of the students enrolled in the course, returns the averages as an array
- getStudent(): receive an index ind, verify that 0<=ind<NbEnrolled, the returns the student located at this index, or NULL.
- findBestStudent(): find the student with the highest average score among all the students registered in the course. The caller should get best average score and the index where the best student is located in the array of enrolled students.
- You may add getters and setters as needed,
- You may also add other useful functions as you see fit eg. isFull() to check if the course reached the maximum number of students enrolled, MAXSTCR.

Tasks:

1. Download the assignment code from NYU classes
2. Compile and run the program as is before doing any editing of the code

3. Incrementally, define the class CStudent and test it. You should have a function TestStudent() in the main.cpp and call this function from the main(). In testStudent() create instances of CStudent and test all the functionalities of this class.

4. Incrementally, define the class CCourse and test it. You should have a function TestCourse() in the main.cpp and call this function from the main(). In testCourse() create instances of CCourse and test all the functionalities of this class.

5. Incrementally, define the class CDept, task 6 and 7 are dedicated to test and use this class.

6. To test the class CDept, insert the below code in main(), then compile and run your program

```
// Create one dept with some default courses and students
 CDept dept ;
 dept.displayStudents() ;
 dept.displayCourses() ;
```

7. And the below code in the main(), complete the missing parts, then run the program.

```
 // Display the courses taken by a student with index 2
    //---- do it here ----

 // Find the best student in a particular course
 // Display the student info and his/her Total average score
    //---- do it here ----

 // Create one dept with some default courses and students
    //---- do it here ----

 // Find the courses taken by a particular student
    //---- do it here ----

 // Create a new course and it to the department offering
    //---- do it here ----

 // Register 2 students in a newly created course
    //---- do it here ----

 // Display the updated list of all courses offered
    //---- do it here ----

 // Register 2 students in a newly created course
    //---- do it here ----

 // Enter the grades of these 2 students in the new course
    //---- do it here ----
```

Grading:

- Tasks 1 and 2 (5 pts): Source code clarity and compilation
- Task 3 (30 pts): implementation of class CStudent and the TestStudent() function
- Task 4 (25 pts): implementation of class CCourse and the TestCourse() function
- Tasks 5 (20 pts): implementation of class CDept
- Tasks 6 and 7 (20 pts): main() function and tests of the functionalities of the class CDept.