To develop a Student's Registration System

Note: Students must demonstrate their assignment during their respective lab sessions. This assignment is worth 20marks. A Missed demonstration will result in 100% grade deduction. You must talk to me or your lab instructor in advance in case of any unprecedented situation which may result in late submission or late demonstration.

Problem Statement:

In this assignment, we will look at a way to develop a students' registration system using C for some college. The goal of this program is to allow the administrators to register a student in offered courses, and also allows them to drop a course for a student.

This assignment is intended to get you to practice using arrays, pointers, functions, header files, user inputs, and formatting output among other things.

Background Information

In this assignment we will have two arrays, one to store the student IDs, and the second is to store the course codes.

In addition, there will be a registration table, represented by two-dimensional array that will store the courses each student is registered in. This will be presented as a simple yes/no value stored in the registration table for each student using their index in the students' array, and the course index in the courses array.

For example, if the system has three students with IDs {12345, 34567, 56789}, and have two courses with codes {"CST8234", "CST8288"}. Then the administrator can register a student with ID "34567" in a course with code "CST8234" by recording "Yes" in the registration table for index 1 (representing the student index) and 0 (representing the course index).

The following illustrates the memory representation for of each of the arrays:

Students		
Address	Index	Value
0x303300	0	12345
0x303304	1	34567
0x303308	2	56789

Courses		
Address	Index	Value
0x308800	0	CST8234
0x308809	1	CST8288

Registration Table			
Index	Student Index	Course Index	Value 0 = no / 1 = yes
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1
4	2	0	1
5	2	1	0

Consequently, from the above table we can see that the student with index 0 (ID: 12345) is registered in course with index 1 (Code: CST8288) only.

Requirements:

Write a program that achieves the following requirements:

1) The program should prompt the user to enter:
   - The number of students the user wishes to register.
   - The student ID for each student. Student IDs are 5-digit integers, i.e. 45234. **NOTE:** More or less should throw an error message and returns the control. Leading and trailing zeroes must be considered.

2) Students should be stored in their own array.

3) The program then should prompt the user to enter:
   - The number of courses offered.
   - The course code of each of these courses. Course codes are 7-digit alphanumeric strings, i.e. CST8234. **NOTE:** Your code should not allow any data other than characters on the first three leading spaces and only numbers are allowed on the last 4 spaces of the CC; otherwise, your code must throw an error message and returns the control.

4) Courses should be stored in their own array.

5) The program then should prompt the user to choose one of three actions:
   - Register a student into a course.
   - Drop a student from a course
   - Display Registration table.

6) If the user chooses to register a student or drop a student, then the program should prompt for the student ID first then the course code second and perform the correct action as follows:
   - Validate the student ID as entered by the user. Display an appropriate error message if the student ID is not found and re-prompt the user for a valid student ID.
   - Validate the course code as entered by the user. Display an appropriate error message if the course code is not found and re-prompt the user for a valid course code.
   - Registering a valid student will update the registration table by adding “1” (i.e. true) to the element with the [student index][course index] element.
   - Dropping an existing course will update the registration table by adding “0” (i.e. false) to the element with the [student index][course index] element.

   NOTE: Student cannot drop a CC in which he/she’s not presently registered.

7) If the user chooses to display the registration table, the program should print all entries in the table.

8) Additional requirements:
   - Add another action to the menu that will allow the user to quit the program by choosing the quit action.

- The program would loop until the user quits the program. The program will loop only starting from point 5 above, so the user doesn't need to enter the students or the courses information anymore.
- Validate the menu action as entered by the user. Display an appropriate error message if the menu action is not found and re-prompt for a valid menu action.

9) **Make sure you use functions**, design your program to separate functionality into its own functions. Using only main function will make you lose points.
   - The .c files should contain functions that represent sensible groupings of functionality
   - You must define .h files as appropriate

10) Give your functions and variables a descriptive name. For example, students[], not x[].

11) Write a program that will implement ALL the requirements, explicit and implicit, listed above.

12) Each function must have a header comments that explain what it does, and describe/explain its inputs (if any) and return value (if any)

13) **You must use a "makefile"**, with the CC_FLAGS set to "-g –ansi –pedantic –Wall –w")

14) For Level A+ (Optional), implement the following data structures using dynamic memory allocation:
   - studentsArray (1-d array)
   - coursesArray (1-d array)
   - registrationTable (2-d array)

   All memory must be dynamically allocated using malloc() or calloc(). Remember to free( ) your memory properly before the program gracefully terminates (i.e. EXIT_SUCCSS and not a crash).

   For all other Levels, you're permitted to compile with the C'99, the C standard from 1999. Change your GCC_OPTIONS to:

   GCC_OPTIONS='-g -ansi -pedantic -std=c99 -Wall'

Submission Instructions

- Code (.c file) -> Compressed Zip folder
- Screenshots (terminal: a) gcc -ansi -pedantic -Wall and b) input(s)/output(s)) Compressed Zipfolder)
- Demonstration is mandatory. A missed demonstration will result in 100% deduction.
- Late submission: 10% per day up to 5 days. After that, the deliverable will worth '0'

Marking Scheme

Refer to the attached Rubric on BRS as well.

Documentation Requirements

Document your solution by adding a header comment to the start of your C source file that has the main( ) function:

```
/*
* Title: Assignment #1 - Student Registration System
* Course: CST8234 C Language
* @Author: <<< firstname lastname (studentID) >>>
* Lab Section: 011
* Professor: Surbhi Bahri
* Due date: MM/DD/YY
* Submission date: MM/DD/YY
*/*
*
*
*
* Demo malloc(), memset(), calloc() and free() in C, Ansi-style

* Status:
* Requirement #1: {complete xor incomplete}
* Requirement #2: {complete xor incomplete}
* Requirement #3: {complete xor incomplete}
* Requirement #4: {complete xor incomplete}
* Requirement #5: {complete xor incomplete}
* Requirement #6: {complete xor incomplete}
* Requirement #7: {complete xor incomplete}
* Requirement #8: {complete xor incomplete}
* Requirement #9: {complete xor incomplete}
*
* Write implementation comments above each function you've created in your
assessment like-
/*****************************************************************************
*
Function Name: main
Purpose: abjkndx,asm c.a
Function in parameters: void
Function out parameter: EXIT_SUCCESS
Version: 1
Author: XYZ
*****************************************************************************
/
*/
```