Task:

Write a C program that conforms to the requirements listed below. In this exercise, you will create an application to generate report cards from student records.

Requirements:

* Part 1: Create a Student Record using a function. This record is comprised of multiple parts, so you will need a function to set up the record:
  + The Student Record is represented by a struct containing these fields:
    - StudentID: type int
    - LastName: string of type char [21] – Field to be filled by a NULL terminated string of a maximum of 20 characters and the NULL
    - FirstName: string of type char[21] – Field to be filled by a NULL terminated string of a maximum of 20 characters and the NULL
    - Course records array containing 5 structs (CourseRecords) with these fields:
      * CourseName: string of type char [21] – Field to be filled by a NULL terminated string of a maximum of 20 characters and the NULL
      * Mark: type float
    - AverageMark: type float
  + Your program must use a function to create three Student Records. This function should accept a pointer to a StudentRecord and any other appropriate parameters.
    - Options: Create the record using static data in the program
    - Or, create the record by prompting the user for each field. Assume the user will enter reasonable data to keep validation to a minimum
  + Display each record as it is created.
* Part 2:
  + Develop a function to calculate the AverageMark for each student and set the appropriate result in each StudentRecord.
* Part 3:
  + Develop a function to print a report card for each StudentRecord. This function should accept a pointer to a StudentRecord. Format the report and show:
    - StudentID
    - FirstName
    - LastName
    - Each CourseName and associated Mark
    - AverageMark
* General requirements:
  + Include clear comments
  + Maintain a standard layout/format for the code. Be consistent with spacing or tabbing, use the layout to make nested operation visually clear.
  + Provide clear visual feedback to the user
  + Hint: The console display of the student record and the printed report card can use the same function if the function is designed to provided well-formatted output.

Evaluation:

This assignment is worth 29 marks. Please see the marking rubric below.

Assignment Notes:

The assignment must be demonstrated to the instructor on or before the due date during class.

If your assignment is late please submit to the Brightspace drop box to confirm submission. This timestamp will be used for evaluating any late penalty the assignment may incur.

| Criteria | Below Standard | Developing | Acceptable | | Professional | Marks |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 |
| Struct | Struct(s) not used | Struct(s) used but contains more than 2 errors | Struct(s) used and almost matches specification but has an error | | nested structs correct in every way | \_\_\_\_x2 |
| Pointers | pointers not attempted or not used as parameters to functions | pointers as parameters used but more than 2 errors | pointers work mostly as expected but still one error | | pointers implemented and work completely to specification | \_\_\_\_x3 |
| Strings | strings not used | Character table(s) used but code contains more than 2 errors | character table(s) used but contains an error | | character table(s) implemented completely to specification | \_\_\_\_x2\_ |
| Aesthetics of Output | incorrect or non-existent use of whitespace in output  output is confusing and hard to follow | most output is clear, but poorly presented | good use of whitespace  output is clear and well presented | | excellent use of whitespace  output is clear and attractively presented | \_\_\_\_\_\_ |
| Readability | source code does not match specification | source code is exceptionally well organized and easy to follow |  | |  | \_\_\_\_\_\_ |
| Program Architecture | Presented as a single source file | files and functions have headers  Code is self-documenting |  | |  | \_\_\_\_\_\_ |
| Comments | comments do not match specification | not over/under commented  comments are meaningful and easily understood |  | |  | \_\_\_\_\_\_\_\_ |
| NamingConvention | no standard naming convention followed | industry standard naming convention used throughout the program |  | |  | \_\_\_\_\_\_ |
| Consistency | no consistency in formatting or layout of source code | source code formatting never deviated from expected layout |  | |  | \_\_\_\_\_\_ |
| Total | | **29** |

Name: Emon Majumder ID: W0411567