UNIVERSITY OF PORTSMOUTH
FACULTY OF TECHNOLOGY

Academic Year 2019/20

P21387 Introduction to Programming and Algorithms

Coursework 2

Deadline For Submission:	Friday 8th May 2020 Coursework may be submitted up until 11:00am on the day of submission.
Submission Instructions	Submission made to CW2 link on Moodle page.
Instructions for completing the assessment:	Please upload your full source code and pseudo as a single **text file** (.txt). Your code must be ready to compile so please test it before submission. Your code must be written in the C language only.
Examiners:	Dr Josh Robertson

This coursework contributes 40% to the ENG421 unit mark.

Course Assignment:
Write a C program that simulates a Student Report Card system. The user should be able to add a new student record, edit, search or delete records from a file. Additionally the program should generate overall grades based on marks for each of your modules.

You must implement the following key features:

1. Generate a student marks record – write a **function,** which allows a user to add a record containing a student's marks. You must use an appropriate **data structure** for storing the record the student's record.

   The record must include the student's first name, family name, UP number and marks for each artefact for the ENG421 module.

   An example record would then be:

   JOSH ROBERTSON 999999 96 76 82

   If you add all your other modules to the program and complete the counterpart in part 2, you will be awarded an extra five marks. If you do not know the breakdown of each of your modules assignments use the Module Web Search site (posted on ENG421 Moodle page).

   At this point, the record must NOT save to a file (see section 4)

2. Read all students marks record – write a **function**, which **reads and then prints to console** the mark report for all students from a **text file**.

   If there is no file in the directory, your program should ask the user if they wish to create a file. If there is a file but no information present (i.e. empty), request the user to save all students marks (see section 4 below.)

   When reading the marks from a file, the student's name, UP number and mark for all subjects should be printed in a **tabulated form**.

   The function should then calculate and print the overall grade based on the marks entered. For example, if the record from part 1 above was saved on a text file, read in 96 (CW1) worth 30% of the grade, 76 (CW2) is 40% of the grade and 82 (exam) is worth 30% of the grade which equals 83.8%.

   If you have expanded to include all modules from part 1, the average mark for each module, based on the weighting of the artefacts, and the overall mark for the year should be calculated and printed.

   These results should not be stored on the record file.

3. Read a specific student record – as above but should allow the user to search for a particular student based on their UP number and then print the record.

4. Save all students marks – write a **function** that saves and writes all current records in the program to a **text** file printing student's name, UP number and all marks and calculated grades as mentioned above. If the file does not exist, then the file should be created.

Additionally the program should do the following:

5. Modify student mark record – allow the user to edit the marks record for a particular student. The user will need to enter the UP number to identify a particular student and then asked what module and what assignment to modify the mark for.

6. Delete a student mark record – allow the user to delete the mark record for a particular student. Requires the user to enter the UP number.

After completing any of the above functionality, an option should appear allowing the user to go back to the main menu.

You make appropriate use of error checking, input validation, memory allocation, pointers and using the most optimal methods you can for a higher degree of marks. Finally, the program must be user-friendly by providing clear instructions and provide appropriate feedback to the user.

The coursework must contain pseudo code at the top the solution as comments contained within the source code. **A penalty of five marks will be deducted if this is not done correctly.**

The submission should be submitted as a readable text file e.g. .doc or txt file**. The solution including pseudo code will be run in Codeblocks. The sourcecode must be written in C.**

Comments are also required to explain the implementation.

Your solutions may be written in different IDEs (of your choice) BUT must be able to compile (i.e. successfully build) with **CodeBlocks and in the C language. Make sure you test your solution on Codeblocks before submitting your work!**

Marking Criteria can be seen below. There are 100 marks available.

Marks (for Pseudocode and inputs)	**Program Output & Features**
20-30	● Pseudo code clearly explains key source code features ● Program handles all input errors and displays informative error messages ● Program correctly handles the required input and provides the required output
10-20	● Pseudo code explains some code features ● Program handles some input errors and displays error messages ● Correctly handles required input and provides required output
6-10	● Program partially handles required input and provides required output for some test cases
0-5	● Program shows minimal effort to handle the required input or provide required output ● No pseudo code is submitted

Marks (for source code file)	**Source code organisation and efficiency**
30-70	● Consistent use of efficient techniques to improve the source code maintenance or performance ● Efficient structures are used to organise source code ● Logical flow of the program is clearly visible within the source code and comments ● All requirements of the build are met
20-24	● Some efficient techniques have been applied ● Some useful structures are used to organise source code ● Logical flow of the program can be observed from the source code and source comments
12-16	● Some basic structure is used to organise source code ● Some simple logical flow is observed
4-8	● Lack of structure for the source code, or ● Difficult to follow the logical flow of the program