Algorithms and Data Structures 2

Laboratory Sheet 5

Finish last week's lab sheet if you have not already done so. This week's lab sheet gives practice in creating arrays of structures and taking a first look at a linked list.

1. Write a program to create a structure to define a student record in which a student has a name, course code, year of study (1 - 4) and an array of 6 continuous assessment results. Define an array of 5 student records that is local to the main function. (note the program *arrayExams*.c has a similar structure to this program). Include the following functions in your program :
    a. void inputStudents (struct student *studentArray)
        Input students into the array

    b. void displayAllStudent(struct student *studentArray)
        Display all student details

    c. int numberFailed(struct student *studentArray)
        Calculate and display the number of assessments failed by each student. Then calculate the total number of assessments failed by all students and return this value and display it in the main method.

    d. float displayAverage (int studentIndex, struct student *studentArray);
        Calculate and return the average of all continuous assessment results for a specific student indicated by an index, which is then displayed in main(). Before this function is called, the user should input which student they wish to process by inputting the name and searching for the index position of that student. This position is input as a parameter into this function.

2. Open the linked list file in this week's set of programs. Remember this program has exactly the same logic as linked lists from last semester. Make sure you can understand the code (and of course, please ask questions if you don't) that is currently in the program before you make changes to the code.

Create new two functions to add to this programs

a. void addNodesToEnd() – the current addNodes() function adds nodes to the start of the list. This new function should add nodes to the end of the list. Call this function instead of addNodes() in main() to see if it is working.

b. bool duplicate() – this function searches through the list and returns true/false to indicate if any duplicate values were found. This method should be called after the viewAllNodes() in main() as part of an IF statement where an appropriate message is displaying informing the user of the presence of duplicates in the list or not.