Data Structures and Algorithms

Manal ElZant-Karray

Project : Manage a Subject’s Exam Using SLL project

(to be done by three or four students group)

We want to use a singly linked list to manage a subject's exam, i.e. the list of students who participates to the test. For each student, we'll store his name, his student ID number and his exam grade.

A. Structures and types :

Define the **student** structure with the following fields:

- The **name**, of string type,

- student card **id-num** of type *long*,

- the **grade** which is a *real* type,

- A pointer to the **next** student.

N.B: If you want to facilitate the code, you can define the type *T_Student* as a pointer to the structure **struct student**.

B. Functions to implement :

1. Write a function to create a student for the exam. The prototype is :

   struct student* createstudent(void);

2. Write a function to display the list of students.

   void displaylist(struct student*);

3. Given that there are two exam’s rooms, the data entry is done in two stages. Write a function to add data on the number of students entered by the supervisor:

   struct student* addstudent(struct student *,char [],long ,float);

4. Write a function that returns the number of students that participate to the exam. The prototype of the function is as follows:

   int studentcount(struct student*);

5. Write a function that searches for a student with his card id-number and displays all the data concerning him. The prototype of this function is :

   int findstudent(struct student*, long);

6. Knowing that the last student entered does not satisfy the exam conditions. You are asked to remove him from the list. Write the function that deletes a student from the end of the list. The prototype of this delete function is as follows:

   void deletelaststudent(struct student*);

7. Write a function to sort the list of students according to the students' grades. The prototype function is as follows:

   void sortlist(struct student*);

8. Write a list that calculates the exam average and displays a general interpretation depending on whether the average is: greater than 65, between 50 and 65, or less than 50. The prototype function is :

float averagexam (struct student *);

9. Write a function to free a list. The prototype of the function is:

void freelist(struct student*);

<u>To have more performance you should add two more functions</u>:**

10. Write a function that splits the list of students into two sub-lists, high-level and low-level lists, according to their exam grades. We'll assume that a grade of 65 or higher is assigned to high-level students. The prototype of this function is as follows:

    void splitlist(struct student*,struct student, struct student**);**

11. Write a function that merges the two sub-lists. The prototype of this function is :

    struct student* mergelists(struct student*, struct student*);

C. Interface :

Write the previous functions to write a program that manage the Student's exam. The program provides a menu that contains the following features:

1. Enter students,
2. Add new students,
3. Search for a student,
4. Display student list,
5. Sort the list of students by grade,
6. Split the list into two sub-lists according to grades (<65 and >=65),
    a. Sort sub-lists by grade,
    b. Merge the two sub-lists by grade order.
7. Calculate exam average
8. Free lists
9. Exit

<u>P.S:</u> ** The split/Sort part (point 6) is related to both implemented functions: ***splitlist*()** **and *mergelists()*.** It is a *separate part* so if you don't manage to do it, this will not be a problem to achieve the other parts of your program. But it is a so important part as it gives more performance to your program and it is a direct application using the Merge sort algorithm to your linked list.

D. General guidelines:

The MINIMUM organization of the project is as follows:
- Header file dsaproject.h, containing the declaration of the basic structures/functions,
- Source file dsaproject.c, containing the definition of each function,
- Source file main.c, containing the main program.

Your report (of no more than two pages) will contain:

- a list of the additional structures or/and functions you have chosen to implement and the reasons for these choices.

- a brief description of the complexity of each of the implemented functions.

Your report and your three files will be submitted as an assignment on Moodle in the space provided for this purpose (only one assignment per team!).

<u>A presentation of your program and some Q/A about the project will be done by every team at the last exercise sessions!</u>