CS 100 Project Six – Fall 2018

Project Overview: In this project, you will use linked lists to implement a grade book for CS100. The grade book is just a linked list of students enrolled, and there is a linked list of grades for each student. For that purpose, two node types are defined as below:

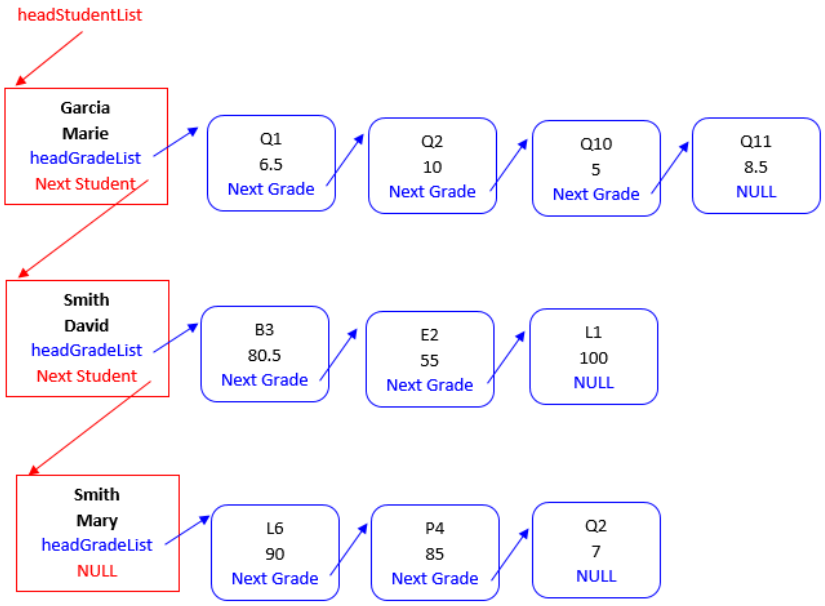
```
typedef struct _student {
    char *lastName;
    char *firstName;
    Grade *headGradeList;
    struct _student *next;
} Student;
```

```
typedef struct _grade {
    char name[4];
    double value;
    struct _grade *next;
} Grade;
```

Each student has a name (with both `lastName` and `firstName`) and a linked list of grades headed by `headGradeList`. Each grade contains a grade name (`name`) and a numeric value (`value`) for the actual grade. For CS100, the following are the valid grade names.

- B1 through B10 for 10 textbook exercises.
- E1 through E8 for 8 exams (tracing and coding are considered as two different exams).
- L1 through L10 for 10 labs.
- P1 through P6 for 6 projects.
- Q1 through Q11 for 11 quizzes.

The example below shows a linked list of three students in red, headed by `headStudentList`. Each student has a linked list of grades in blue. The first student (*Marie Garcia*) has four (4) grades in her grade list and the other two students (`David Smith` and `Mary Smith`) have three (3) grades in their grade lists.



This project consists of three files, **`main.c`**, **`student.h`** and **`student.c`**.

- **`main.c`** – the user interface. It prompts the user for an action and performs that action by calling the corresponding function implemented in **`student.c`**.
- **`student.h`** – definitions of two structures and the signatures of the functions to manipulate the two types of linked lists.
- **`student.c`** – implementation of the functions listed in **`student.h`**.

You can download from Blackboard a working version of **`main.c`** and **`student.h`**, and a skeleton **`student.c`** file. Your job is to complete the nine functions that manipulate linked lists in **`student.c`**.

What You Need To Do

- Create a directory **`project6`** on your machine. Download **`main.c`**, **`student.h`** and **`student.c`** to that directory.
- To compile this program, use **`gcc -Wall -std=c99 main.c student.c`**
- You can compile and run this program without doing any coding. It does nothing, but you can see how it works.
- You are not allowed to change **`main.c`** and **`student.h`**. You just need to complete **`student.c`**.
- In **`student.c`**, add a header block of comments that includes your name and a brief overview of your task.
- The file **`student.c`** should contain the actual code for the nine functions that are used by the `main` function. There is no implementation (just comments) in each function when downloaded. Your job is to implement these nine functions.
    - **addStudent** – add a student, using the "**add-at-front**" method.
    - **addGrade** – add a grade to a student, using the "**add-at-end**" method.
    - **count** – return the number of students in the linked list.
    - **printStudent** – print the information of a student (the student name and all the grades for that student).
    - **print** – prints all students. For each student, print the name and all the grades for that student.
    - **addStudentOrdered** – add a student in alphabetical order (ordered add of a student). To order two students, you first need to compare their last names. If they have the same last name, you then need to compare their first names. We recommend you write a helper function to compare two student nodes (a new student and an existing student) to demine the order of these two students, similar to `strcmp`.
    - **addGradeOrdered** – add a grade to a student in the specified order (ordered add of a grade). The specified order is B1 through B10, then E1 through E8, then L1 through L10, then P1 through P6, and finally Q1 through Q11. Please note that using `strcmp` alone to compare two grade names will result in an incorrect order. For example, `strcmp` will place Q10 and Q11 before Q2. Again, we recommend you write a helper function to compare two grade nodes (a new grade and an existing grade) to demine the order of these two grade nodes in the grade list.
    - **removeGrade** – remove the specified grade from a student and free the related node.
    - **removeStudent** – remove the specified student and all the grades of that student. You shall also free all the related space.
- Things you must remember when implementing these functions (Please also see the comments of each function in **`student.c`** for details.)
    - When you try to add a student, if a student with that name exists, generate an error message.
    - When you try to add a grade to a student that does not exist, generate an error message.
    - When you try to add a grade to a student, if the student already has such a grade, update the grade.
    - When you try to remove a student that does not exist, generate an error message.
    - When you try to remove a grade from a student, if the student or the grade does not exist, generate an error message.
- We recommend you complete the functions from the easiest. We will grade each of them individually, so you can get partial credit even if you do not complete them all. We recommend the following order when completing the functions. However, you can complete them in any order that you want.
    - **addStudent** and **addGrade** and **printStudent**
    - **count** and **print**
    - **addStudentOrdered** and **addGradeOrdered**
    - **removeGrade** and **removeStudent**

- When testing the program, we will never combine **addStudent** with **addStudentOrdered** (or **addGrade** with **addGradeOrdered**) in the same test case. We will test both ordered lists and unordered lists, but never together.
- Input for this program is entered from standard input. The user types in commands that call the various functions mentioned above. For example, to add a student, the user types **addStudent**, followed by last name and first name.
- You can put all the commands in a file, say **`testData`**. (Do not forget to put `quit` as the last command.) Then you can redirect input from a file using **`./a.out < testData`**.
- A sample execution of the program is shown at the end of the document to build the grade book as illustrated on the first page.

When you are finished and ready to submit your project:

- Make sure your **`project6`** directory has the **student.c** file. When testing, we will use our own **student.h main.c** files.
- Compress your **`project6`** directory into a single (compressed) zip file, **`project6.zip`**.
- Once you have a compressed zip file named **`project6.zip`**, submit that file to Blackboard.

Project 6 is due at 5:00pm on Friday, November 30. Late projects are not accepted.

**A project shall be completed individually, with no sharing of code or solutions.
All submissions will go through MOSS (Measure Of Software Similarity)
for similarity check.
The University of Alabama's Code of Academic Conduct will be rigorously enforced.**

A sample execution of the program

```
Enter a command: addStudentOrdered Garcia Marie

Enter a command: addGradeOrdered Garcia Marie Q2 10

Enter a command: addGradeOrdered Garcia Marie Q1 6.5

Enter a command: addGradeOrdered Garcia Marie Q11 8.5

Enter a command: addGradeOrdered Garcia Marie Q10 5

Enter a command: printStudent Garcia Marie

Student Name: Garcia, Marie
        Q1  : 6.5
        Q2  : 10
        Q10 : 5
        Q11 : 8.5

Enter a command: addStudentOrdered Smith Mary

Enter a command: addStudentOrdered Smith David

Enter a command: count

There are 3 students

Enter a command: addGradeOrdered Smith Mary L6 90

Enter a command: addGradeOrdered Smith Mary Q2 7

Enter a command: addGradeOrdered Smith Mary P4 85

Enter a command: addGradeOrdered Smith David L1 100

Enter a command: addGradeOrdered Smith David B3 80.5

Enter a command: addGradeOrdered Smith David E2 55

Enter a command: print

Student Name: Garcia, Marie
        Q1  : 6.5
        Q2  : 10
        Q10 : 5
        Q11 : 8.5

Student Name: Smith, David
        B3  : 80.5
        E2  : 55
        L1  : 100

Student Name: Smith, Mary
        L6  : 90
        P4  : 85
        Q2  : 7

Enter a command: quit
```