Homework 3 – Grade Database Management

Due Date –

Project Statement

This assignment is meant to grant you further C experience inside a full Linux environment. You will be designing software that creates a database of students and their grades. The project is developed such that you will implement linked-list functions and develop a software hierarchy mentality while dealing with file input/output, command line input/output, dynamic memory, and pointer implementation.

The program will read in a text file of students and their grades (A, B, C, D). Each student will have their own node in a linked list. That node will contain the students name a list of their grades and their grade average. A user will be able to search for a students' name, via command line, and if that student exists in the database, their grade will be displayed. The user will also be able search a grade and all students with that grade will be displayed. Along with being able to search for a student or grade the user will also be able to add and remove students from the database.

Deliverables

- All code you use to implement the database
- A report with the following
  - o A clear statement on the completeness of your system. Report what testing procedures you performed.
  - o Document design decisions
  - o Provide a usage manual (make it clear what your system DOES not what it was intended to do but you never got working).

Grading

The grade distribution for this project will be as follows

- 15% Coding style
- 20% Report
- 65% Functionality

  The functionality will be broken down into components. Each working component will give you credit. A fully working system will give you full credit for this portion.

Components in the functionality section:

- Linked-List implementation
- File I/O
- Command line I/O
- Dynamic memory
    - Creating
    - Freeing (This will be checked via Valgrind)
- Pointers

Coding style is based on the following guidelines

- All functions need to be commented with:
    - Input parameters
    - Return Value
    - Description of what the function is intended to perform
- Your code should be split into different files based on functionality
    - A sample Makefile will be provided, but you will need a Makefile for your program compilation
- All main code should be documented with its intended functionality.

Database Description

The database will work as follows:

- The program will boot into a splash screen with six options
    - The first option allows the user to print a list of students' first names, last and their associated grades in different tests.
    - The second option allows the user to search for a student to display their grade.
    - The third option allows the user to search for a grade and display all students with that grade in alphabetical order.
    - The fourth option allows the user to enter a student's name and their grade into the database.
    - The fifth option allows the user to delete a student from the database.
    - The sixth option allows the user to exit out of the database. Once this option is entered make sure there are no memory leakage.

- User input is done via command line
    - At the splash screen the user will type 1, 2, 3, 4 ,5 or 6 to select the first, second, third, fourth, fifth or sixth option. If an invalid input is entered the screen must be cleared and the user will be prompted to enter a valid option.

    - If the user selects the first option, a corresponding list of the students' first and last names shall be printed in Alphabetical order according to the last name. A formatted print statement beside each name will generate a table displaying their related grades in column. The final grade average and value will also be displayed. After the full table has been printed, the program will return to the splash screen.

    - If the user selects the second option, they will be taken to a new screen to enter the students name. If the entered student is in the database the grade will be displayed. At this point the user will have the option to search for another student or return to the splash screen. If the user enters a name not in the database, they will be alerted and given the option to search for another name or return to the splash screen.

- If the user selects the third option, they will be taken to a new screen which will prompt the user to enter a grade. If a valid grade (A, B, C, D, or F) is entered the all students with that grade will be displayed in alphabetical order. At this point the user will have the option to search for another grade or return to the splash screen. If no students in the database have the grade search for the user will be prompted and given the option to search for another grade or return to the splash screen. If an invalid grade is entered the user will be prompted and given the option to enter a valid grade or return to the splash screen.

- If the user selects the fourth option, they will be taken to a new screen where they can enter the students name followed by their grades along with existing students' data. After the name and grade is entered the program will return to the splash screen.

- If the user selects the fifth option, they will be taken to a new screen where they will be prompted to enter a students' name to be removed from the database. If the entered name is in the database, that student will be removed from the database. At this point the user will have the option to enter another student to remove from the database or return to the splash screen. If the entered name is not in the database, the user will be prompted and given the option to enter another name or return to the splash screen.

- If the user selects the sixth option, the program will quit leaving no memory leakage.

Implementation

Database

The initial database will be loaded from a text file. The text file will be loaded into the database as a linked list of nodes. Any changes made to the database should be reflected in the linked list (i.e. there should only be one linked list, which will be modified. Don't make a second linked list when adding or deleting a student). A sample input text file will be provided.

Node

A node will be a struct object with the following variables:

- Name: A string containing the students first and last names.
- List_of_Grades: A linked list of unions, each of which contains either quiz information or test information (Percentage of number, Weight)
- Grade: The weighted average of the students' grade.
- Next_student: a pointer which points to the next student in the list.

Important notes

You should approach the design of this assignment using a hierarchical design methodology. The list should be dynamically allocated so it can take any number of students. The program should take any input list (if given) that follows this sample format .

Example:

Linkedlist input.txt

The input text file can be of the following format:

FirstName LastName(First Line)

Quiz1: Percentage of Marks (space) Weight

Quiz2: Percentage of Marks (space) Weight

Midterm: Percentage of Marks (space) Weight

Quiz3: Percentage of Marks (space) Weight

Final: Percentage of Marks (space) Weight

Alternatively, if the input list is not provided, the program should create an empty database and by going to option-4 the user can then enter a student's name and their grade into the database.

***Isn't grading according to the total number necessary. I have put a sample grading criterion below in the table:

Number	Grade
>90	A
80-90	B
70-79	C
Else	D

You should begin with the pieces closest to memory, the smallest blocks, and make their functionality correct before moving up. Create unit-tests to make sure that each portion works before moving further up. Your main loop should not have to focus on anything in memory, but rather database states provided by functions from your hierarchy. I suggest having: a top file which only consists of your main driver which takes file and command line input, and prints the database; a database.h and databse.c file which provides all database specific functions like "Select option", "print grade", etc...; a node.h and node.c which contains the student information, and functions to create, update, and finally free the node. You do not have to have these specific files, and you can have more files, but this is an idea of a structure which may help you. You should, however, have some file organization which makes sense for consistent coding practices. Also, beware of the memory freeing. Leaking memory is not allowed. Try to code in such a way that the criterion is fulfilled.

Lastly, go to your TAs and instructor early with questions regarding the homework. Office hours are a great resource for code specific questions. Piazza is a very good resource for general questions. **DO NOT POST CODE ANYWHERE ONLINE. DO NOT SHARE CODE WITH CLASSMATES**.

If you use any code you get from the web, you must **CITE** it according to citation policy. We will be using cheat check software, if you attempt to plagiarize, you will be caught. It is better to turn in nothing than to turn in plagiarized work.