60-141 – Introduction to Programming II Winter, 2017
Assignment 5
(Wednesday, March 29, 2017: 23:59:59)

This is a lengthy assignment description. Take time to read it carefully and thoroughly.

Dynamic Linked Lists and File Manipulation

Through this assignment, you are going to be maintaining a file of all students and their registered courses to learn the concepts of structures, pointers to structures, dynamic memory allocation and file manipulation. For this, a structure called "studentInfo" with the following member elements should be defined:

- **StudentID - 9 char long**
- **First Name - 20 char long**
- **Last Name - 25 char long**
- **Number of Courses Attending - integer**
- **Array of CourseInfo - a 10 element array of courseInfo elements**
- **next - Pointer to the next student structure in the list**

where the CourseInfo structure has been defined as follow:

```
struct CourseInfo{
    int courseID;
    char courseName[30];
};
```

Your task is to:

Write a complete, well documented C program that will be able to:

a. Add a new student
b. Delete a student and all information related to that student
c. Search for a student and their information
d. Display a list of current students
e. Save student information to file
f. Load student information from file

Assignment Description:

You are provided with a data file called "studentList.txt" that has the data for a number of students. You are required to read this data from the input file into **a sorted linked list** data structure. The linked list must be sorted based on the StudentID. The sample input file will end with a line that has three stars. An example of an input file with two fictional students is:

studentList.txt
111111111
Lisa
Porter
3
ENEE 114
CMSC 412
ENME 515
333333333
Alex
Simpson
1
CMSC 412
***

For each student, the data file is formatted line by line in the following manner:
<student ID>
<first name>
<last name>
<number of courses they are taking>
<course name> <course id>

After loading the student list, your program should be able to interactively ask the user for input. Your interactive menu should have the following inputs:

1. **Add new student**
2. **Delete a student**
3. **Search for a student**
4. **Display current students**
5. **Save student information to file**
6. **Exit**

Hints:

1- The following points are to be considered:

a. Student ID should be unique and 9 characters of numbers (such as 123456789, 103449977)
b. First name and Last Name should be started with capital letters (upper case).
c. Student information should be sorted based on the student IDs both in the linked list and in the input/output file.

2- Your program should implement at least the following functions:

a) **addStudent ()** : To add a new student and his/her registered courses.
   - Make sure the first letter of the first and last names is upper case (capital letter).
   - The student ID should be unique; you cannot have two students with the same StudentID. So, before adding a student, search for the studentID to be sure that you have not had it previously entered.
   - If the linked list is empty, the new student is the head of the list. If not, search through the linked list to find the appropriate spot within the sorted linked list.

b) **deleteStudent():** To delete a student information using its StudentID.
   - Search the linked list for a student matching the studentID passed in. If it is found, delete it from the linked list.
   - Note that the linked list is sorted based on studentID!

c) **searchStudentID():** To search for a student using studentID
   - You do not have to search all the linked list as it is sorted based on studendIDs.
   - This function can be called from addStudent() and deleteStudent() functions.

d) **searchStudentlName():** To search for a student using his/her last name and display the related information if the student exists.
   - You have to search all the linked list as it is not sorted based on last names
   - Before starting the search, make sure the name supplied by the user has a capital letter for the first letter.

e) **displayStudentInfo():** To display the current student information that exists in the linked list

f) **saveStudentInfo():** To save student information from the sorted linked list to a file (inputStudents.txt)

g) **loadStudentinfo():** To read all the student information from an input file (studentList.txt)
   - This function should be called at the beginning of you program to load all previous student information saved in the input file
   - Student information should be formatted and stored in a sorted linked list.

h) **exit():** To save student information in a file (inputStudents.txt) and exit from the program

Sample Run

A sample run of the program, with its interactive menu is shown below.

1. Add new student
2. Delete a student
3. Search for a student
4. Display current students
5. Save student information to file
6. Exit
> 1
Adding new student:
Student ID : 222222222
First name : john
Last name : Rezaei
Number of courses: 2
Course ID: 412
Course Name: CMSC
Course ID: 123
Course Name: MATH

1. Add new student
2. Delete a student
3. Search for a student
4. Display current students
5. Save student information to file
6. Exit
> 4
Student 1:
111111111
Lisa
Porter
3
ENEE 114
CMSC 412
ENME 515

Student 2:
222222222
John
Rezaei
2
CMSC 412
MATH 123

Student 3:
333333333
Alex
Simpson
1
CMSC 412

1. Add new student
2. Delete a student
3. Search for a student
4. Display current students
5. Save student information to file
6. Exit
> 3
What is the last name of student? porter
111111111
Lisa
Porter
3
ENEE 114
CMSC 412
ENME 515

1. Add new student
2. Delete a student
3. Search for a student
4. Display current students
5. Save student information to file
6. Exit
> 2
Student ID: 111111111
Student information deleted.

1. Add new student
2. Delete a student
3. Search for a student
4. Display current students
5. Save student information to file
6. Exit
> 4
Student 1:
222222222
John
Rezaei
2
CMSC 412
MATH 123

Student 2:
333333333
Alex
Simpson
1
CMSC 412

1. Add new student
2. Delete a student
3. Search for a student
4. Display current students
5. Save student information to file

6. Exit
> 6
Save student information to file before leaving? y
Student List saved successfully.
Bye!

Requirements:

- Write and document a complete C program that is capable of satisfying the requirements of this assignment problem.
- UNDOCUMENTED OR POORLY DOCUMENTED code will automatically lose 50% marks.
- PLAGIARIZED work will not be graded and receive a mark of ZERO and reported according to the Senate bylaws.
- The question can use I/O redirection if appropriate. Please review the textbook for an example on using I/O redirection from flat files.
- TO SUBMIT: No later than the submission deadline, your assignment should be uploaded in Blackboard. Late submissions are not accepted and will receive a mark of ZERO.
- Upload your work as an attachment, include both the source file (assign5.c) and the script file (assign5.txt) - see below how to create the script file.

To create a script file (one that logs your compilation steps and your output in a text file):
1. script assign5.txt
2. cat assign5.c
3. cat input.txt
4. cc assign5.c
5. a.out
6. ls -l
7. exit (DO NOT FORGET THIS STEP!!)

The example script execution presumes that the input was specified in the file input.txt and that all I/O is handled within the program.

If you are compiling your code under Cygwin shell, then you need to change line 5 to:

5. ./a.exe < input.txt

In line 4, students may prefer to use the gcc compiler to ensure improved warning and error diagnostic reports from compiling the program. You must have access to gcc in order to make this change, however.

NOTE: Submissions that are not received correctly by the deadline will automatically receive a ZERO mark. **Late assignment submissions are not accepted!**

NOTES:
1. Your assignment must be RECEIVED by the due date and time. Late assignment submissions are NOT accepted. Keep your script file, and all your code unmodified as proof of its completion in case it is not received.
2. It is your responsibility to get an early start on the assignment, research and ask questions ahead of time from the due date.
3. Marks will be deducted for unclear code (improper spacing and alignment, hard to read programs and missing outputs).
4. Make sure you turn in a complete script file that clearly shows: your code, your compilation

process, a listing of the directory showing your source file(s) and the a.out with the date/time stamps, and the output. DO NOT SUBMIT a.out FILES!

5. **PLAGIARISM:** CHEATING IS NOT TOLERATED – PLAGIARISM IS CHEATING! You must submit your own work. Students who are suspected of copying someone else's work will be reported to the department's chair and the Dean of Science and be dealt with in accordance with the University policies. You should not share your code with others. Codes that are similar to each other will BOTH be reported as potential evidence of copying. It is imperative that you write your own code.

6. Authorized/limited help on this assignment may be provided directly from your Lecture or Lab instructors and Teaching Assistants.