Systems Software
Lab 3: Creating libraries, Makefile, and Dynamic Memory Allocation

Lab Objectives

In this activity, students should demonstrate the following abilities:

1. Write C code that can be reused in the form of a library
2. Write makefile to build C programs with multiple source files
3. Use dynamic memory allocation C functions to create dynamic data structures

Lab Assignment

In this lab, you will design, implement and test a program that stores and manipulates student records in a linked list. Follow the steps below to complete the program.

1. Use a structure **`student`** for the student information (ID, name, and GPA). Use the structure header and implementation files shown in lecture 7. Modify the function **`compare_students`** to return **0** if the two student arguments have identical ids. Compile the implementation of the student structure into an object file named **`student.o`**.

2. Use a structure **`node`** for the linked list elements (student record and pointer to the next element). Use the structure header and implementation files shown in lecture 7 and complete the implementation of the missing functions (**`insert_last`**, **`delete_last`**, and **`delete_item`**). Mpdify the function search() by adding a statement to print the student record when it is found. Compile the implementation of the linked list structure into an object file named **`linked_list.o`**.

3. Package **`student.o`** and **`linked_list.o`** into a shared library that you name **`libLL.so`**.

4. Write a C program, **`student_database.c`**, that uses the linked list data structure to store the information of 500 students from the file **`students.txt`**. Your program should create an empty linked list by declaring a pointer **`list_head`** to type **`node`** and initializing it to **`NULL`**. Open the file **`students.txt`** for reading and read the information of each student, create a student structure with the information read from the file, and insert the structure into the linked list. Once all the student records have been inserted in the linked list, allow the user to perform one of the following operations:

    a. *Add a new student record to the list*. The user must provide an id, name, and gpa for the new student. The new student is inserted at the head of the list.

b. *Find an existing student record with a given id*. If the student is found, print the student information.

c. *Delete an existing student record with a given id*. If the student is found, print the student information and prompt the user for a confirmation to delete the record and call the method **`delete_item`** if he/she confirms the deletion.

5. Write a **`makefile`** to represent the file dependencies of the project and to build all the required object , library, and executable files.

6. Test your program for different operations. Use the sample run below for testing.

```
>>./student_database
File students.txt opened successfully.
Linked list built from file students.txt  successfully.

Select an operation:
1: Add a new student
2: Find an existing student
3: Delete an existing student
4: Quit
1
Enter the student information (id, gpa, and name):
99999
3.25
Some_Student
Student record inserted successfully.

Select an operation:
1: Add a new student
2: Find an existing student
3: Delete an existing student
4: Quit
2
Enter the student id: 66246
Student record found.
66246   FirstName475_LastName475    0.88

Select an operation:
1: Add a new student
2: Find an existing student
3: Delete an existing student
4: Quit
3
Enter the student id: 94435
Student record found.

94435   FirstName490_LastName490    2.68
```

```
Are you sure you want to delete student record? (Enter y or n):
y
Student record deleted successfully.

Select an operation:
1: Add a new student
2: Find an existing student
3: Delete an existing student
4: Quit
4
>>
```

Organize the C project in a folder named **Lab3** with four sub folders: **src** for source files (**.c**), **include** for header files (**.h**), **lib** for library and object files (**.so** and **.o**), and **bin** for executable file and the data file **students.txt**. Place the **makefile** in the main folder (**Lab3**). Make sure you use the right paths in the makefile to access the different pre-requisite files. Compress your main folder **Lab3** and submit the zipped file on courseSite. Do not forget to show your working program to the instructor or TAs to get credit for the lab.