**Assignment**

**Pre-requisite :** You must have completed the course upto phase 3 including assignments.

**Pls follow the instructions as instructed in the assignment**

**Codes : git clone** [**https://github.com/sachinites/MemoryLeakDetector**](https://github.com/sachinites/MemoryLeakDetector)

**Hash code : ab795d55e5dd87f47f27c9a3b0f4fe8450507f84**

**git reset --hard ab795d55e5dd87f47f27c9a3b0f4fe8450507f84 (same hash code as above)**

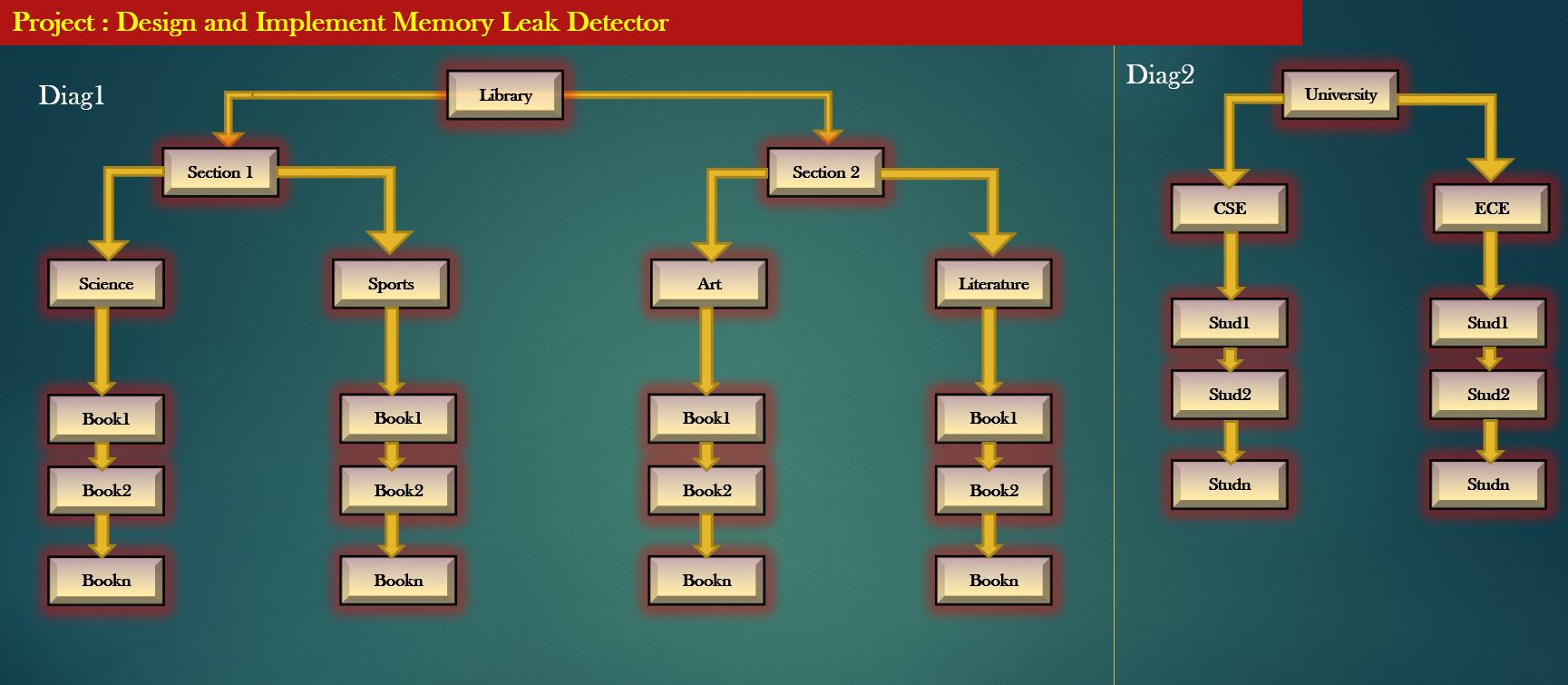
**Goal:**

**By the end of the assignment, you will be able to write an application harnessing the benefits of MLD library that you have written and catch memory leaks if any**

**Problem Statement:**

In this assignment, we will going to create an application in **ass1.c** file, which have the following data structure organization. We already had discussed this in our course Lecture Videos.

In addition, In this assignment, we will use one more library – LinkedList library. LinkedList implementation is present in **LinkedList** folder. Linked list is defined by structure **ll\_t**;



**STEP 1: Application Structures**

**Following should be the definition of structures the application ass1.c should define.**

**Include following header files in ass1.c**

#include <stdio.h>

#include <stdlib.h>

#include "mld.h"

#include "LinkedList/LinkedListApi.h"

**Define the following structures in ass1.c file**

typedef struct student\_ {

char stud\_name[32];

ll\_t \*book\_lst; /\*List of books borrowed by this student\*/

struct student\_ \*left;

struct student\_ \*right;

} student\_t ;

typedef struct university\_ {

student\_t \*student\_lst; /\*Doubly linked list of students in the university\*/

} university\_t;

typedef struct book\_ {

char book\_name[32];

float price;

student\_t \*stud; /\*Student who borrowed this book\*/

struct book\_ \*left;

struct book\_ \*right;

} book\_t;

typedef struct section\_ {

char section\_name[32];

book\_t \*book\_list; /\* Doubly linked list of books in this section \*/

} section\_t;

typedef struct library\_ {

Char lib\_name[32];

section\_t \*sections[2];

} library\_t;

**Step 2: Application Structure Registration**

Write the following function in ass1.c file which takes the responsibility to register all structures with mld library.

**void init\_register\_application\_structures\_with\_mld(object\_db\_t \*object\_db);**

**Step 3: Preparing Linked List Library for MLD**

Now that since, Linkedlist also being used by the application, Linkedlibrary use standard functions for memory allocations. Check LinkedListApi.c. So, MLD library will not be able to track objects such as linkedlist or linkedlist nodes.

To enable MLD library to track Linkedlist specific objects, we need to do the following :

1. #include "../mld.h" in LinkedList/LinkedListApi.c file
2. In LinkedListApi.c, define below function and write code to register LinkedList structures with MLD library. Below fn should be defined in LinkedListApi.c

**void init\_register\_linked\_list\_structures\_with\_mld(object\_db\_t \*\_object\_db);**

1. Replace all standard calloc/malloc/free in LinkedListApi.c with xcalloc and xfree
2. declare a static variable - "static object\_db\_t \*object\_db" in LinkedListApi.c and initialize it in API **init\_register\_linked\_list\_structures\_with\_mld (object\_db\_t \*\_object\_db);**

**APIs to be used by your application to maintain LinkedList. In your assignment, you need to make use of following APIs to create a linked-list, insert or remove the data from linked-list.**

**ll\_t\* init\_singly\_ll();** // create a new linked list using and return a pointer to it.

**int singly\_ll\_add\_node\_by\_val(ll\_t \*ll, void\* data);** // Create a new node in a linked list, and new holds the ‘data’. Insert this new node into a linkedlist. Return 0 on success, -1 on failure

**void delete\_singly\_ll(ll\_t \*ll);** /\*Delete and free all nodes of a slinked list , but not the data held by the nodes if the linked list. Application must take the responsibility to release the data first before calling this API\*/

**void singly\_ll\_delete\_node\_by\_data\_ptr(ll\_t \*ll, void \*data);** /\*Delete and free the nodes of a linked list which holds the ‘data’. This function do not free ‘data’.\*/

**bool\_t is\_singly\_ll\_empty(ll\_t \*ll);** /\*to test of the linked-list is empty or not. Return 0 if empty\*/

If you want to iterate over a linked list, you can implement your own manual while or for loops.

**Step 4: Preparing an Application Main function to get started**

See file ass1.c. I have implemented main() for you. You should create Structure and object database of MLD library and register root objects of the application.

**Step 5: Write a Menu-Driven Program to perform below operations**

Write a Menu Driven Program as follows :

1. Search a particular book in Library >> Take a book name in library and search in both sections if book is present or not.
2. Assign book to Student >> Take two inputs : book name and student name. If both exists, allocate book to student. Report error if book is already borrowed by some other student already.
3. De-assign book >> Take one Input : book name., and mark it as unborrowed and available to be borrowed
4. Delete book >> Take one input : Book name. Delete the book from the library only if found and not assigned to any student.
5. Delete a student >> Take one input : student name. Delete the student from university, and release all books borrowed by him/her
6. Run MLD Algorithm
7. Report Leaked Objects
8. Exit >> clean up the entire application state, before and exit the program

**Step 6: Intentionally introduce Memory Leaks and see if MLD library report it**

Introduce some memory leak intentionally, and then choose option 6 followed by option 7 to see if MLD library reports leak.

**Compilation Steps :**

gcc -g -c mld.c -o mld.o

gcc -g -c LinkedList/LinkedListApi.c -o LinkedList/LinkedListApi.o

gcc -g -c ass1.c -o ass1.o

gcc -g -o exe1 ass1.o LinkedList/LinkedListApi.o mld.o

./exe1