VISVESVARAYA TECHNOLOGICAL UNIVERSITY JNANA SANGAMA,BELAGAVI-590018

A mini project report on ***"Library management system"***

submitted in partial fulfillment for the award of degree of

Bachelor of Engineering in

Electronics and Communication

Submitted by ,

AISHWARYA M HALLI-1NH18EC002

AISHWARYA NAGRAS-1NH18EC003

B GOWTHAMI-1NH18EC020

CHANDANA C-1NH18EC023

Under the guidance of

Ms.Mamta B S

Assistant Professor

Dept. of ECE

New Horizon College of Engineering, Bengaluru

NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade.

BENGALURU-560103

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the Mini project entitled "LIBRARY MANAGEMENT SYSTEM" is carried out by Ms. Aishwarya M(1NH18EC002) Ms.Aishwarya N(1NH18EC003) Ms.Gowthami B(1NH18EC020) and Ms.Chandana C(1NH18EC023), bonafide students of NHCE, Bengaluru in partial fulfillment for the award of Bachelor of Engineering in Electronics and Communication of the Visweswaraya Technological University, Belagavi during the year 2020-21. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The mini project report has been approved as it satisfies the academic requirements in respect of the mini project work prescribed for the said degree.

Signature of the Guide	Signature of the HOD
Ms Mamta B S	Dr. Sanjeev Sharma
Assistant Professor Department of ECE	Professor & HOD Department of ECE
NHCE, Bengaluru	NHCE, Bengaluru

External Viva

Name of the Examiners	Signature with Date
1. _____	1. _____
2. _____	2. _____

ACKNOWLEDGEMENT

We would like to express our sincere thanks to our principle Dr. Manjunatha sir for giving us the opportunity to take this project.

We also thank the HOD of the Electronics and communication department Mr.Sanjeev Sharma Sir for encouraging us to develop our application skills by undertaking such project.

A very warm gratitude for our project guide Ms.Mamta ma'am for encouraging us throughout while doing the project and also explaining unknown concepts in a very simple manner because of which we could make this project a success.

ABSTRACT

The paper reviews on the “Library Management System" which aims in developing a computerized system to maintain all the daily activity of library. It helps the librarian to make note of all the books which are newly added to the library, book issued and the details of the student. Library management system is a simple console application without graphics, developed using C programming

This report even emphasis on the data structure being involved in the design of the project i.e. Single linked list and even utilizes various aspects of C language. A clear description of operations which are used in the project, like sorting using linked list, deletion using linked list and searching using linked list with help of file concept is briefly explained. Additionally we explore the advantage of using the linked list over arrays.

The explanation of efficient code usage makes a clear point of learning the code through linked list and gives an idea about the application of library management. The code and implementation for the library management system is also depicted with the screenshots of its execution. It is hoped that the project study will implant and impart the importance of implementing the fast procedures for any task to be completed and invoke people to adopt to new technologies to make work easier and simpler.

TABLE OF CONTENTS:

LIST OF FIGURES

CHAPTER ONE: INTRODUCTION

Project aims and objectives:

We have studied C language, now we come to the real life problems and see how we can solve them. Here, it being our main aim, we will use C Program to develop one real life project with a simple project life cycle using Dev C++ platform.

The main objective of the application is to automate the existing system of manually maintained library records of the Book Issue, Book Return from the student, Stock Maintenance, Catalogue and Book Search to be computerized. So the Book Issue, Return, Searching will be a more refined and quicker process.

Background of project:

Library Management System is an application which refers to library systems which are generally small or medium in size. It is used by librarian to manage the library using a computerized system where he/she can record various transactions like issue of books, return of books, addition of new books, addition of new students etc. Books and student functions are also included in this system which would keep track of the students using the library and also a detailed description about the books a library contains. With this computerized system there will be no loss of book record or member record which generally happens when a non-computerized system is used.

It is a software application to maintain the records related to Book Purchase, Stock Maintenance, Book Search, Book Issue and all necessary requirements for the Library to manage the day to day operations. This application can be used by any Library to automate the process of manually maintaining the records related to the subject of maintaining the stock and Book Issues. It is seen that there is no loss of books and students don't waste time to search for the book required. It is preferred in college libraries where management of books is slightly complicated where students are always referring and borrowing them.

Access rights are divided into two main parts namely admin and the user

- If user's position is admin, the user is able to generate different kinds of reports like list of books and few more function like add, edit and delete book. All these functions are able to help librarian to manage and work efficiently. The admin access is password protected and only authorized people can access the functions under this.
- If the user is not an admin, he /she can enter his/her student id and name and can search for a book which they needed, thus can take a book from a library.

CHAPTER TWO: LITERATURE SURVEY

This involves listing down the important aspects that are supposed to be achieved in order obtain the required results:

Implementation using data structures:

The design of an efficient algorithm for the solution of the problem is called data structures. Data means combination of numbers and non-numbers like characters and structure means format, thus data structures means, and it's a collection of numbers and non-numbers and arrange them in a specific order.

Data structures are mainly classified into two types: Primitive data structures: Data structures are accessed the features directly from the system is called primitive data structures. Ex int ,float, char etc.Non primitive data structures: Data structures are accessed the features indirectly from the system is called non primitive data structures.Again Non primitive data structures are divided into two types, they are:Linear data structures and Nonlinear data structure.

Linked list:

A linked list is a dynamic data structure. The number of nodes in a list is not fixed and can grow and shrink on demand. Any application which has to deal with an unknown number of objects will need to use a linked list.

Each element of a list is comprising of two items - the data and a link to the next node. The last node has a link to null. The entry point into a linked list is called the head of the list. It should be noted that head is not a separate node, but the link to the first node. If the list is empty then the head is equal to null.

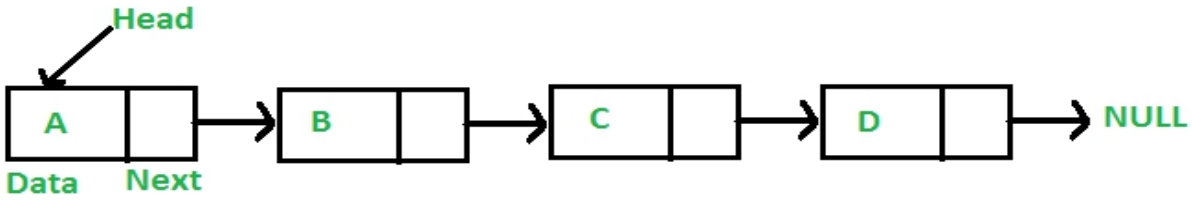

Fig 2.1

The linked list is of three types and we make use of single linked list. In a singly linked list each node in the list stores the contents of the node and a pointer or reference to the next node in the list. It does not store any pointer or reference to the previous node. It is called a singly linked list because each node only has a single link to another node. To store a single linked list, you only need to store a reference or pointer to the first node in that list. The last node has a pointer to nothingness to indicate that it is the last node. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.Arrays can be used to store linear data of similar types, but arrays have following limitations:
1) the size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.
2) Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to shift.

For example, in a system if we maintain a sorted list of IDs in an array id[].

Id[] = [1000, 1010, 1050, 2000, 2040].And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 .
Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id [], everything after 1010 has to be moved.

Uses over arrays:

1) Dynamic size
2) Ease of insertion/deletion

Representation in C:

A linked list is represented by a pointer to the first node of the linked list. The first node is called head. If the linked list is empty, then value of head is NULL.
Each node in a list consists of at least two parts.
1) data
2) pointer to the next node
In C, we can represent a node using structures. Below is an example of a linked list node with an integer data.

```
// A linked list node

struct Node
```

```
{
  int data;
  struct Node *next;
};
```

Insertion

A node can be added in three ways:

1) At the front of the linked list
2) After a given node.
3) At the end of the linked list.

Add a node at the front:

The new node is always added before the head of the given Linked List. And newly added node becomes the new head of the Linked List. For example if the given Linked List is 10->15->20->25 and we add an item 5 at the front, then the Linked List becomes 5->10->15->20->25. Let us call the function that adds at the front of the list is push(). The push() must receive a pointer to the head

pointer, because push must change the head pointer to point to the new node.

```
/* Given a reference (pointer to pointer) to the head of a list
   and an int,  inserts a new node on the front of the list. */
void push(struct Node** head_ref, intnew_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));

    /* 2. put in the data  */
    new_node->data  =new_data;

    /* 3. Make next of new node as head */
```

```
    new_node->next = (*head_ref);


    /* 4. move the head to point to the new node */

    (*head_ref)    = new_node;

}
```

Add a node after a given node:

We are given pointer to a node, and the new node is inserted after the given node.

```
/* Given a node prev_node, insert a new node after the given
   prev_node */

voidinsertAfter(structNode* prev_node, intnew_data)

{

    /*1. check if the given prev_node is NULL */

    if(prev_node == NULL)

    {

      printf("the given previous node cannot be NULL");

      return;

    }

         /* 2. allocate new node */

    structNode* new_node =(structNode*) malloc(sizeof(structNode));


    /* 3. put in the data  */

    new_node->data  =new_data;


    /* 4. Make next of new node as next of prev_node */

    new_node->next = prev_node->next;
```

```
   /* 5. move the next of prev_node as new_node */
   prev_node->next = new_node;
}
```

Add a node at the end:

The new node is always added after the last node of the given Linked List. For example if the given Linked List is 5->10->15->20->25 and we add an item 30 at the end, then the Linked List becomes 5->10->15->20->25->30.

Since a Linked List is typically represented by the head of it, we have to traverse the list till end and then change the next of last node to new node.

Following are the 6 steps to add node at the end.

```
/* Given a reference (pointer to pointer) to the head
  of a list and an int, appends a new node at the end  */
voidappend(structNode** head_ref, intnew_data)
{
   /* 1. allocate node */
   structNode* new_node = (structNode*) malloc(sizeof(structNode));

   structNode *last = *head_ref;  /* used in step 5*/

   /* 2. put in the data  */
   new_node->data  =new_data;

   /* 3. This new node is going to be the last node, so make next
        of it as NULL*/
   new_node->next = NULL;
```

```
 /* 4. If the Linked List is empty, then make the new node as head */
   if(*head_ref == NULL)
   {
     *head_ref = new_node;
     return;
   }
     /* 5. Else traverse till the last node */
   while(last->next != NULL)
      last = last->next;
   /* 6. Change the next of last node */
   last->next = new_node;
   return;
}
```

Deletion

Linked lists provide us the great feature of deleting a node. The process of deletion is also easy to implement. The basic structure is to declare a temporary pointer which points the node to be deleted. Then a little bit of working on links of nodes. There are also three cases in which a node can be deleted:

1. Deletion at the start
2. Deletion at the end
3. Deletion at a particular position

Deletion at the Start: In this case, the first node of the linked list is deleted. I know, you remember that the first node is called the head. So, we are going to delete the head node.

The process of deletion includes:Declare a **temp** pointer and pass the address of the first node, i.e. head to this pointer.

1. Declare the second node of the list as head as it will be the first node of linked list after deletion.
2. Delete the temp node.

Deletion at the End: Deletion of the last node is a bit difficult to understand than the first node. In the case of the first node, you just need access to the head and you can delete it. But in the case of the last node, you also need access to the second to the last node of the linked list as you will delete the last node and make the previous node as the tail of linked list.

Our algorithm should be capable of finding a node that comes before the last node. This can be achieved by traversing the linked list. We would make two temporary pointers and let them move through the whole linked list. At the end, the previous node will point to the second to the last node and the current node will point to the last node, i.e. node to be deleted. We would delete this node and make the previous node as the tail.

Deletion at a Particular Position:In linked list, we can delete a specific node. The process of deletion is simple. Here we don't use the head and tail nodes. We ask the user to input the position of the node to be deleted. After that, we just move two temporary pointers through the linked list until we reach our specific node. Now, we delete our current node and pass the address of the node after it to the previous pointer. This way, the current node is removed from the linked list and the link is established between its previous and next node.

Display

1. In order to write the program for display, We must create a linked list using create(). Then and then only we can traverse through the linked list
2. Traversal Starts from Very First node. We cannot modify the address stored inside global variable **"**start" thus we have to declare one temporary variable -"temp" of type node.
3. In order to traverse from start to end you should assign Address of Startingnode in Pointer variable i.e temp

CHAPTER THREE: TECHNOLOGY USED

HARDWARE SYSTEM CONFIGURATION:

Processor - Intel Core i5

Speed - 1.7GHz

RAM - 4GB

Hard Disk - 1 TB

SOFTWARE SYSTEM CONFIGURATION:

Operating System - windows 10

Programming Language - C

Platform - Dev C++

Compiler - GCC

CHAPTER FOUR: PROPOSED METHODOLOGY

The project involves sequential execution of steps to see the required result in a proper order and the methodology that has been followed is:

- Defining the functions: The type of operations to be performed are listed and the following codes are generated. These operations are performed with the concepts of data structures like linked lists,stacks,queues etc.

- Function for the admin: The admin can perform certain operations like loading details of a new book, modify the details of an existing book, delete the book details if required,search the book,keep a record of the availability of the book with the help of specific serial numbers allotted to each book.

- Function for the user:The user can also have access to certain operations like issuing a book,deposit a book,search for a book and if the user has to issue a book the user's details are filled according to the format specified in order to create a record of the book details when issued.

- Combining both the modules: The final step is have the admin module and user module into a single code where the end result appears as a single module with two subdivisions for the admin and user and the other subdivisions under the two main modules.

CHAPTER FIVE: RESULT OBTAINED

Welcome screen/login screen:

fig 4.1

Choice 1: Asks for the password.

If successful proceeds or shows up as incorrect password.

Fig 4.2

Password being correct, admin is directed to this page:

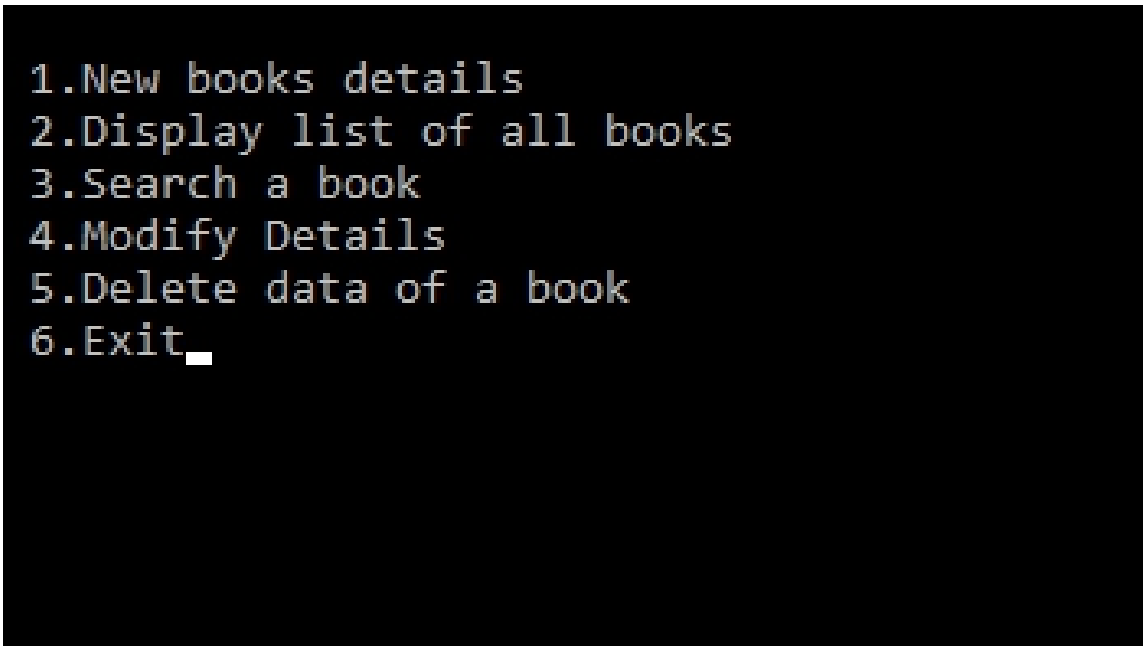

Fig 4.3

On choosing opt 1(new book entry)

Its asks for following details and writes in the file.

```
   1.New books details
   2.Display list of all books
   3.Search a book
   4.Modify Details
   5.Delete data of a book
   6.Exit
1



1.Book_No : 5
2.Book_Name : comp
3.Author : eee
4.Year_Of_Publication : 2000
5.Price : 35
```

Fig 4.4

For opt 2:(display list of books)

For choosing opt 1: (list of all books in the file)

```
1:List of all books
2:List of issued books
3:List of deposited books

Enter your choice : 1



Book_no         Book_name       Author  Year_of_publication     Price

1               math            aaa             1990            367
2               sci             bbb             1980            250
3               eng             ccc             1670            190
4               kan             ddd             1234            45
5               comp            eee             2000            35
```

Fig 4.5

For opt 3: (search a book)

Shows the output if book id is matched,

If not show the output as data not found.

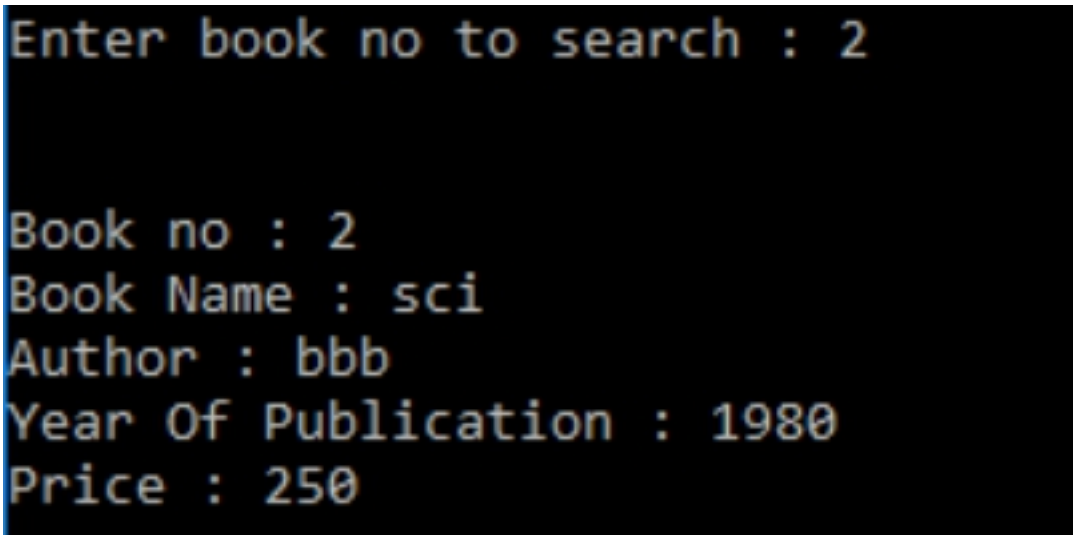

Fig 4.6

For opt4.(modify the details)

I have changed the year of publication of the book.

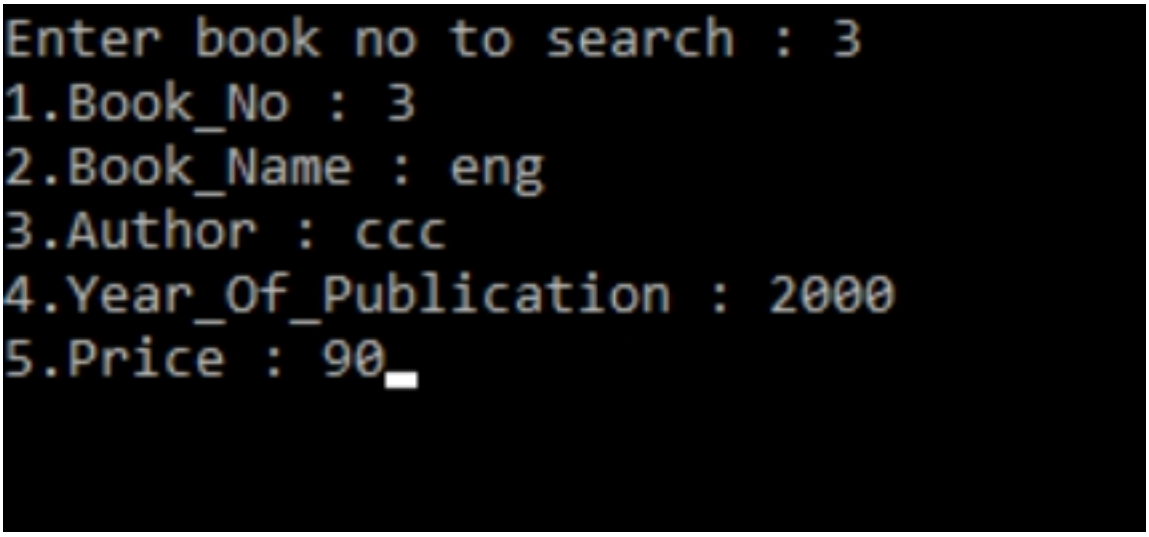

Fig 4.7

On choosing option 5. (Delete) the can deleted any record.

If record not found shows an error.

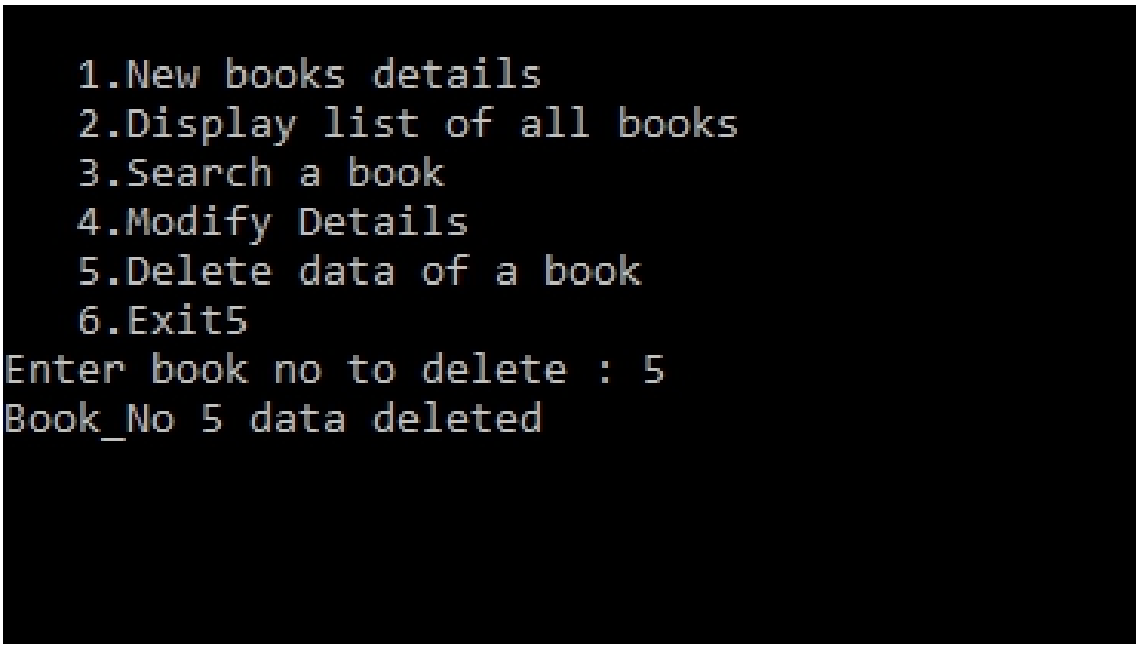

Fig 4.8

Now press 6 to exit to login screen:

1. choose opt 2 to user mode:

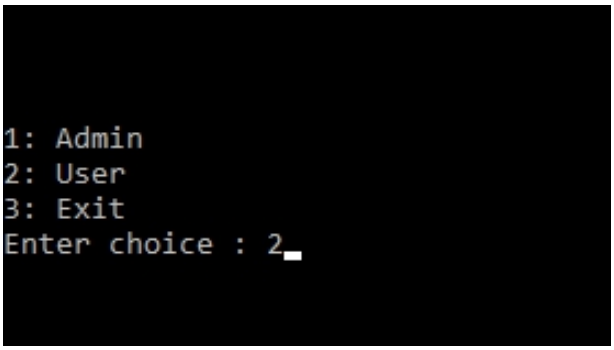

Fig 4.9

These the available options:

For opt 1:

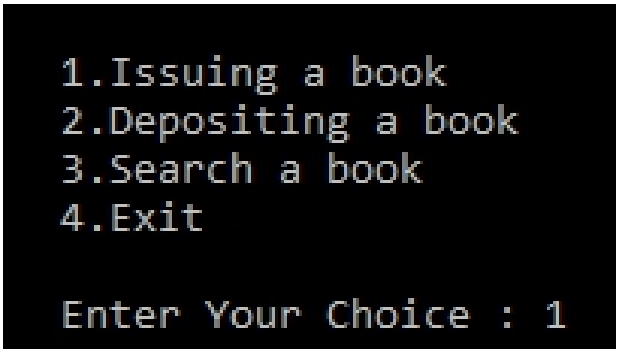

Fig 4.10

It accepts student name and USN number,

Ask for book name and id

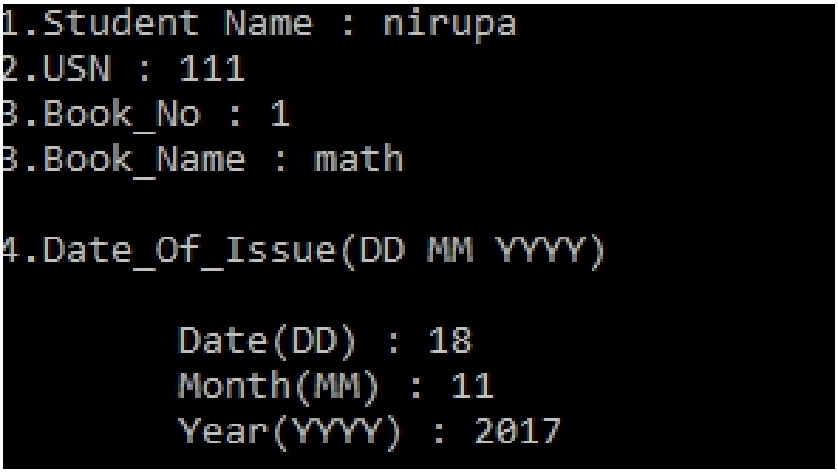

Fig 4.11

For opt 2: (deposit a book)

```
1.Student Name : nirupa
2.USN : 111
3.Book_No : 1
3.Book_Name : math

4.Date_Of_Issue(DD MM YYYY)

        Date(DD) : 18
        Month(MM) : 11
        Year(YYYY) : 2017

   7.Date_Of_Deposit(DD MM YYYY)

           Date(DD) : 20
           Month(MM) : 11
           Year(YYYY) : 2017
```

Fig 4.12

For opt 3: (search a book)

```
Enter book no to search : 4


Book no : 4
Book Name : kan
Author : ddd
Year Of Publication : 1998
Price : 50
```

Fig 4.13

Exit to main menu:

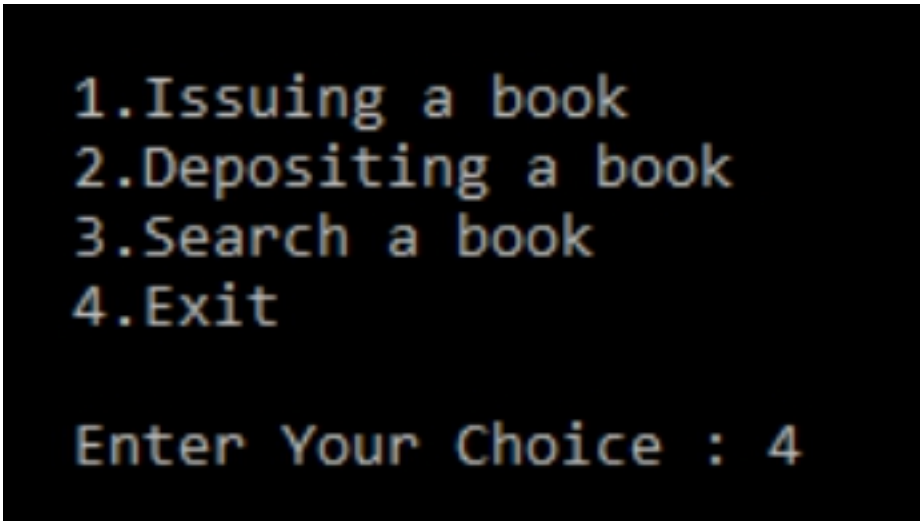

Fig 4.14

Exits from the program execution.

Fig 4.15

CHAPTER SIX: FUTURE SCOPE

- Dynamic size : Linked list is a dynamic data structure so it can grow and shrink at runtime by allocating and deallocating memory. So there is no need to give initial size of linked list.
- Ease of insertion/deletion : Insertion and deletion of nodes are really easier. Unlike array here we don't have to shift elements after insertion or deletion of an element. In linked list we just have to update the address present in next pointer of a node.
- No Memory Wastage:As size of linked list can increase or decrease at run time so there is no memory wastage. In case of array there is lot of memory wastage, like if we declare an array of size 10 and store only 6 elements in it then space of 4 elements are wasted. There is no such problem in linked list as memory is allocated only when required.
- Implementation: Data structures such as stack and queues can be easily implemented using linked list
- To make the Library system more efficient and effective.
- To provide a user friendly environment where user can be serviced better.
- Make functioning of library faster.
- Provide a system where the library staff can catch defaulters and not let them escape.
- To minimize the loss of books

CHAPTER SEVEN: REFERENCE

- Geeksforgeeks article for data structures implementation
- c4learn.com
- Codescope for debugging methods
- codementor.io
- For linked list implementation thecrazyprogrammer.com
- For code implementation codeforge.com

APPENDIX:

```
#include<conio.h>
#include<process.h>
#include<stdlib.h>
#include<string.h>
#include<stdio.h>
struct date
{
        int day;
        int month;
        int year;
};
struct library
{
        intbook_no;
        charbook_name[40];
        char author[30];
        intyear_of_publication;
        int price;
        struct library *ptr;
};
struct issue
{
        charsd_name[20];
        charusn[20];
        intbook_no;
```

```
        charbook_name[25];
        struct date d_issued;
        struct issue *ptr;
};
struct deposit
{
        charsd_name[20];
        charusn[20];
        intbook_no;
        charbook_name[25];
        struct date d_issued;
        struct date d_deposite;
        struct deposit *ptr;
};
typedefstruct library lnode;
typedefstruct issue inode;
typedefstruct deposit dnode;
lnode *h1=NULL,*s1,*p1,*n1;
inode *h2=NULL,*s2,*p2,*n2;
dnode *h3=NULL,*s3,*p3,*n3;
voidinput_book_detail()
{
        n1=(lnode*)malloc(sizeof(lnode));
        printf("\n\n\n");
        printf("1.Book_No : ");
        scanf("%d",&n1->book_no);
```

```
        printf("2.Book_Name : ");
        scanf("%s",&n1->book_name);
        printf("3.Author : ");
        scanf("%s",&n1->author);
        printf("4.Year_Of_Publication : ");
        scanf("%d",&n1->year_of_publication);
        printf("5.Price : ");
        scanf("%d",&n1->price);
        n1->ptr=NULL;
        if(h1==NULL)
                h1=n1;
        else
        p1->ptr=n1;
        p1=n1;
}
voidissuing_book()
{
        n2=(inode*)malloc(sizeof(inode));

        printf("1.Student Name : ");
        scanf("%s",&n2->sd_name);
        printf("2.USN : ");
        scanf("%s",&n2->usn);
        printf("3.Book_No : ");
        scanf("%d",&n2->book_no);
        printf("3.Book_Name : ");
```

```
        scanf("%s",&n2->book_name);

        printf("\n4.Date_Of_Issue(DD MM YYYY)\n\n");
        printf("\tDate(DD) : ");
        scanf("%d",&n2->d_issued.day);
        printf("\tMonth(MM) : ");
        scanf("%d",&n2->d_issued.month);
        printf("\tYear(YYYY) : ");
        scanf("%d",&n2->d_issued.year);
        n2->ptr=NULL;
        if(h2==NULL)
                h2=n2;
        else
        p2->ptr=n2;
        p2=n2;
}
voiddeposit_book()
{
        n3=(dnode*)malloc(sizeof(dnode));

        printf("1.Student Name : ");
        scanf("%s",&n3->sd_name);
        printf("2.USN : ");
        scanf("%s",&n3->usn);
        printf("3.Book_No : ");
        scanf("%d",&n3->book_no);
```

```
        printf("3.Book_Name : ");
        scanf("%s",&n3->book_name);
        printf("\n4.Date_Of_Issue(DD MM YYYY)\n\n");
        printf("\tDate(DD) : ");
        scanf("%d",&n3->d_issued.day);
        printf("\tMonth(MM) : ");
        scanf("%d",&n3->d_issued.month);
        printf("\tYear(YYYY) : ");
        scanf("%d",&n3->d_issued.year);

        printf("\n   7.Date_Of_Deposit(DD MM YYYY)\n\n");
        printf("\t   Date(DD) : ");
        scanf("%d",&n3->d_deposite.day);
        printf("\t   Month(MM) : ");
        scanf("%d",&n3->d_deposite.month);
        printf("\t   Year(YYYY) : ");
        scanf("%d",&n3->d_deposite.year);
        n3->ptr=NULL;
        if(h3==NULL)
                h3=n3;
        else
                p3->ptr=n3;
        p3=n3;
}
voidsearch_book()
{
```

```
        int key;
        if(h1==NULL)
                printf("Empty");
        else
        {       s1=h1;
                printf("Enter book no to search : ");
                scanf("%d",&key);
                while(s1->book_no!=key && s1->ptr!=NULL)
                {
                        s1=s1->ptr;
                }
                if(s1->book_no!=key && s1->ptr==NULL)
                        printf("Data not found");
                        else
                {
                                printf("\n\nBook no : %d\n",s1->book_no);
                                printf("Book Name : %s\n",s1->book_name);
                                printf("Author : %s\n",s1->author);
                                printf("Year Of Publication : %d\n",s1->year_of_publication);
                                printf("Price : %d\n",s1->price);
                }
                getch();
        }
}
voidmodify_book()
{
```

```
int key;
if(h1==NULL)
        printf("Empty\n");
        else
{
        s1=h1;
        printf("Enter book no to search : ");
        scanf("%d",&key);
        while(s1->book_no!=key && s1->ptr!=NULL)
        {
                s1=s1->ptr;
        }
        if(s1->book_no!=key && s1->ptr==NULL)
                printf("Data not found");
                else
        {
                printf("1.Book_No : %d\n",key);
                printf("2.Book_Name : ");
                scanf("%s",&s1->book_name);
                printf("3.Author : ");
                scanf("%s",&s1->author);
                printf("4.Year_Of_Publication : ");
                scanf("%d",&s1->year_of_publication);
                printf("5.Price : ");
                scanf("%d",&s1->price);
        }
```

```
                getch();}
}
voiddel_book()
{
        int key;
        if(h1==NULL)
                printf("Empty\n");
                else
        {
                s1=h1;
                printf("Enter book no to search : ");
                scanf("%d",&key);
                while(s1->book_no!=key && s1->ptr!=NULL)
                {
                        p1=s1;
                        s1=s1->ptr;
                }
                if(s1->book_no!=key && s1->ptr==NULL)
                        printf("Data not found");
                        else
                {
                        printf("Book_No %d data deleted",key);
                        if(h1==s1)
                                h1=s1->ptr;
                        else
                                p1->ptr=s1->ptr;
```

```
                        free(s1);
                }
        }
        getch();
}

voiddisplay_books()
{
        int choice;
        printf("\n\n1:List of all books\n2:List of issued books\n3:List of deposited books\n\nEnter your choice : ");
        scanf("%d",&choice);
        if(choice==1)
        {
                if(h1==NULL)
                printf("Empty");
                else
                {
                        s1=h1;
                        printf("\n\n\nBook_no \tBook_name\tAuthor\tYear_of_publication\tPrice\n\n");
                        while(s1!=NULL)
                        {
                                printf("%d\t\t",s1->book_no);
                                printf("%s\t\t",s1->book_name);
                                printf("%s\t\t",s1->author);
                                printf("%d\t\t",s1->year_of_publication);
```

```
                printf("%d\n",s1->price);
                s1=s1->ptr;
            }
        }
}
else if(choice==2)
{
        if(h2==NULL)
        printf("Empty");
        else
        {
            s2=h2;
printf("\n\n\nName\tUSN\tB_no\tB_name\tDate_of_Issue\n\n");
            while(s2!=NULL)
            {
                printf("%s\t",s2->sd_name);
                printf("%s\t",s2->usn);
                Printf ("%d\t",s2->book_no);
                printf("%s\t",s2->book_name);
                printf("%d-",s2->d_issued.day);
                printf("%d-",s2->d_issued.month);
                printf("%d\n",s2->d_issued.year);
                s2=s2->ptr;
            }
        }
}
```

```
else if(choice==3)
{
        if(h3==NULL)
        printf("Empty");
        else
        {s3=h3;
printf("\n\n\nName\tUSN\t\tB_no\tB_name\tDate_of_Issue\tDate_of_Deposit\n\n");
                while(s3!=NULL)
                {
                        printf("%s\t",s3->sd_name);
                        printf("%s\t",s3->usn);
                        printf("%d\t",s3->book_no);
                        printf("%s\t",s3->book_name);
                        printf("%d-",s3->d_issued.day);
                        printf("%d-",s3->d_issued.month);
                        printf("%d\t",s3->d_issued.year);
                        printf("%d-",s3->d_deposite.day);
                        printf("%d-",s3->d_deposite.month);
                        printf("%d\n",s3->d_deposite.year);
                        s3=s3->ptr;
                }
        }
}
else
printf("\n\nINVALID INPUT");
getch();
```

```
}
int password(char a[25])
{
charpwd[25], ch;
        int i=0;
        printf("Enter password : ");
        while (1)
        {
                ch = getch();
                if (ch == 13)  {
                        pwd[i]='\0';
break;
}
if (ch==8)
        if(i==0)                                                                {
continue;
}
else{
printf("\b \b");
i--;
continue;
}
}
pwd[i++] = ch;
printf("*");
}
```

```
if(strcmp(a,pwd)==0)
        return(1);
        else
        return(0);
}

voidload_lib()
{
        intlbook_no;
        charlbook_name[40];
        charlauthor[30];
        intlyear_of_publication;
        intlprice;
        FILE *p;
        p=fopen("total.txt","r");

        while((fscanf(p,"%d",&lbook_no),fscanf(p,"%s",lbook_name),fscanf(p,"%s",lauthor),fsca
nf(p,"%d",&lyear_of_publication),fscanf(p,"%d",&lprice))==1)
        {
                n1=(lnode*)malloc(sizeof(lnode));
                n1->book_no=lbook_no;
                strcpy(n1->book_name,lbook_name);
                strcpy(n1->author,lauthor);
                n1->year_of_publication=lyear_of_publication;
                n1->price=lprice;
                n1->ptr=NULL;
```

```
                if(h1==NULL)
                {
                        h1=n1;
                }
                else
                {
                        p1->ptr=n1;
                }
                p1=n1;
        }
        fclose(p);
}

voidwrite_lib()
{
        FILE *p;
        p=fopen("total.txt","w");
        s1=h1;
        if(h1!=NULL)
        while(s1!=NULL)
        {
                fprintf(p,"%d\t",s1->book_no);
                fprintf(p,"%s\t",s1->book_name);
                fprintf(p,"%s\t",s1->author);
                fprintf(p,"%d\t",s1->year_of_publication);
                fprintf(p,"%d\n",s1->price);
```

```
                s1=s1->ptr;
        }
        fclose(p);
}
voidload_issue()
{
        charisd_name[20];
        chariusn[20];
        intibook_no;
        charibook_name[25];
        struct date id_issued;
        FILE *i;
        i=fopen("issue.txt","r");
        while((fscanf(i,"%s\t",isd_name),fscanf(i,"%s\t",iusn),fscanf(i,"%d\t",&ibook_no),fscanf(i,
"%s\t",ibook_name),fscanf(i,"%d\t",&id_issued.day),fscanf(i,"%d\t",&id_issued.month),fscanf(i,
"%d\n",&id_issued.year))==1)
        {
                n2=(inode*)malloc(sizeof(inode));

                strcpy(n2->sd_name,isd_name);
                strcpy(n2->usn,iusn);
                n2->book_no=ibook_no;
                strcpy(n2->book_name,ibook_name);
                n2->d_issued.day=id_issued.day;
                n2->d_issued.month=id_issued.month;
                n2->d_issued.year=id_issued.year;
```

```
                n2->ptr=NULL;


                if(h2==NULL)
                {
                        h2=n2;
                }
                else
                {
                        p2->ptr=n2;
                }
                p2=n2;
        }
        fclose(i);
}
voidwrite_issue()
{
        FILE *i;
        i=fopen("issue.txt","w");
        s2=h2;
        if(h2!=NULL)
        while(s2!=NULL)
        {

                fprintf(i,"%s\t",s2->sd_name);
                fprintf(i,"%s\t",s2->usn);
                fprintf(i,"%d\t",s2->book_no);
```

```
                fprintf(i,"%s\t",s2->book_name);
                fprintf(i,"%d\t",s2->d_issued.day);
                fprintf(i,"%d\t",s2->d_issued.month);
                fprintf(i,"%d\n",s2->d_issued.year);
        s2=s2->ptr;
        }
        fclose(i);
}

voidload_dep()
{
        charisd_name[20];
        chariusn[20];
        intibook_no;
        charibook_name[25];
        struct date id_issued;
        struct date id_deposite;
        FILE *i;
        i=fopen("deposit.txt","r");

        while((fscanf(i,"%s\t",isd_name),fscanf(i,"%s\t",iusn),fscanf(i,"%d\t",&ibook_no),fscanf(i,
"%s\t",ibook_name),fscanf(i,"%d\t",&id_issued.day),fscanf(i,"%d\t",&id_issued.month),fscanf(i,
"%d\n",&id_issued.year),fscanf(i,"%d\t",&id_deposite.day),fscanf(i,"%d\t",&id_deposite.month)
,fscanf(i,"%d\n",&id_deposite.year))==1)
        {
                n3=(dnode*)malloc(sizeof(dnode));
```

```
                strcpy(n3->sd_name,isd_name);
                strcpy(n3->usn,iusn);
                n3->book_no=ibook_no;
                strcpy(n3->book_name,ibook_name);
                n3->d_issued.day=id_issued.day;
                n3->d_issued.month=id_issued.month;
                n3->d_issued.year=id_issued.year;
                n3->d_deposite.day=id_deposite.day;
                n3->d_deposite.month=id_deposite.month;
                n3->d_deposite.year=id_deposite.year;

                n3->ptr=NULL;

                if(h3==NULL)
                {
                        h3=n3;
                }
                else
                {
                        p3->ptr=n3;
                }
                p3=n3;
        }
        fclose(i);
}
voidwrite_dep()
```

```
{
        FILE *i;
        i=fopen("deposit.txt","w");
        s3=h3;
        if(h3!=NULL)
        while(s3!=NULL)
        {
                fprintf(i,"%s\t",s3->sd_name);
                fprintf(i,"%s\t",s3->usn);
                fprintf(i,"%d\t",s3->book_no);
                fprintf(i,"%s\t",s3->book_name);
                fprintf(i,"%d\t",s3->d_issued.day);
                fprintf(i,"%d\t",s3->d_issued.month);
                fprintf(i,"%d\t",s3->d_issued.year);
                fprintf(i,"%d\t",s3->d_deposite.day);
                fprintf(i,"%d\t",s3->d_deposite.month);
                fprintf(i,"%d\n",s3->d_deposite.year);

                s3=s3->ptr;
        }
        fclose(i);
}

//Main program

int main()
```

```
{
        inta_choice=1,u_choice=1,ch=1;
        charpwd[25] = {"nirupa"};

        load_lib();
        load_issue();
        load_dep();

        while(1)
        {
                system("cls");
                a_choice=1,u_choice=1;
                printf("\n\t\t\t\t\tWELCOME TO LIBRARY\n");

                printf("\n\n1: Admin \n2: User \n3: Exit \nEnter choice : ");
                scanf("%d",&ch);

                if(ch==1)
                {if(password(pwd))
                        {while(a_choice!=6)
                                {system("cls");
                                        printf("\n   1.New books details");
                                        printf("\n   2.Display list of all books");
                                        printf("\n   3.Search a book");
                                        printf("\n   4.Modify Details");
                                        printf("\n   5.Delete data of a book ");
```

```
printf("\n   6.Exit");
scanf("%d",&a_choice);

if(a_choice==1)
{
        input_book_detail();
}
else if(a_choice==2)
{
        system("cls");
        display_books();
}
else if(a_choice==3)
{
        system("cls");
        search_book();
}
else if(a_choice==4)
{
        system("cls");
        modify_book();
}
else if(a_choice==5)
{
        del_book();
}
```

```
                }
        }
        else
        printf("\n\nIncorrectpwd\n\n");
        getch();
}
else if(ch==2)
{
        while(u_choice!=4)
        {
                system("cls");
                printf("\n   1.Issuing a book");
                printf("\n   2.Depositing a book");
                printf("\n   3.Search a book");
                printf("\n   4.Exit");
                printf("\n\n   Enter Your Choice : ");
                scanf("%d",&u_choice);
                if(u_choice==1)
                {
                        system("cls");
                        issuing_book();
                }
                else if(u_choice==2)
                {
                        system("cls");
                        deposit_book();
```

```
                }
                else if(u_choice==3)
                {
                        system("cls");
                        search_book();
                }
                getch();
        }
    }
    else if(ch==3)
    {
        write_lib();
        write_issue();
        write_dep();
        printf("\n\n\t\t\t\tBYE HAVE A NICE DAY\n");
        exit(0);
    }
    else
    {
        printf("\n\nInvalid\n\n");
    }
    getch();
}
```