**Introduction:-**

This Data Structures project is made as a Library Management System. There are two sections in the system, 1) the Librarian section, 2) the Student Section. There are various things the Librarian can do, for example- add a book, remove a book, edit a books details, issue a book, reclaim a book, add a student, etc. The students can browse the available books, search for books, and check their profile. The Data Structures that have been used are Linked List and Binary Search Trees.

**Problem Statement:-**

To create a library management system using Data Structures. It has following users:

1.) Librarian: To maintain and update records of books.

2.) Student: The one who claims book. The purpose is to create management system that stores the details of the book and allow add, search, issue , reclaim ,edit details of a book and the same time keeping track of books borrowed and quantity held at the library. Also to store the details of students. In our existing system, all the transactions of books are done manually, which takes more time for transactions like searching for a book, issuing a book or returning a book, for all these transactions a manual record has to be made. Also preparing list of books borrowed and available books in library will take time and to verify them all. So, a library management system on computers will tackle these problems, as it fast and no need to keep track of records on paper, it will save time will searching for book or while issuing a book.

**Explanation:-**

Binary Search Trees:-

The binary search trees have been used to store the details of the students as well as books. They have been used to reduce the time required to search for any member, as well as their dynamic property, hence reducing memory requirements.

Linked List:-

The Linked List has been used to store the details of any book being borrowed. They have been used due to their easy retrieval as well as their dynamic property.

**Data Structures used:-**

1. **Linked List:-**

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:

**Why Linked List?**

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 shifted.

**Advantages over arrays:**

1) Dynamic size

2) Ease of insertion/deletion

**Drawbacks:**

1) Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists efficiently with its default implementation. Read about it here.

2) Extra memory space for a pointer is required with each element of the list.

3) Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

**Operations on Linked List**

There are various operations which can be performed on singly linked list. A list of all operations used in this project is given below.

**Insertion**

The insertion into a singly linked list can be performed at different positions. In this project we have inserted new nodes at the end of the list.

Insertion at end of the list: It involves insertion at the last of the linked list. The new node can be inserted as the only node in the list or it can be inserted as the last one. Different logics are implemented in each scenario.

**Deletion**

The Deletion of a node from a singly linked list can be performed at different positions. In this project we have deleted specific nodes regardless of position.

Deletion of specified node: It involves deleting a specified node in the list. We need to skip the desired number of nodes to reach the node before the node to be deleted. This requires traversing through the list.

**Traversing**

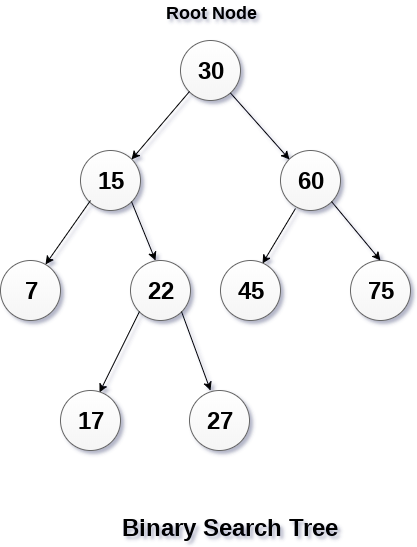
In traversing, we simply visit each node of the list at least once in order to perform some specific operation on it, for example, printing data part of each node present in the list.

**Searching**

In searching, we match each element of the list with the given element. If the element is found on any of the location then location of that element is returned otherwise null is returned.

1. **Binary Search Tree**

Binary Search tree can be defined as a class of binary trees, in which the nodes are arranged in a specific order. This is also called ordered binary tree. In a binary search tree, the value of all the nodes in the left sub-tree is less than the value of the root. Similarly, value of all the nodes in the right sub-tree is greater than or equal to the value of the root. This rule will be recursively applied to all the left and right sub-trees of the root.



A Binary search tree is shown in the above figure. As the constraint applied on the BST, we can see that the root node 30 doesn't contain any value greater than or equal to 30 in its left sub-tree and it also doesn't contain any value less than 30 in its right sub-tree.

**Advantages of using binary search tree**

Searching become very efficient in a binary search tree since, we get a hint at each step, about which sub-tree contains the desired element.

The binary search tree is considered as efficient data structure in compare to arrays and linked lists. In searching process, it removes half sub-tree at every step. Searching for an element in a binary search tree takes o(log2n) time. In worst case, the time it takes to search an element is 0(n).

It also speed up the insertion and deletion operations as compare to that in array and linked list.

**Disadvantages of Binary Search Tree**

The shape of the tree depends on the order of insertions, and it can be degenerated

When inserting or searching for an element, the key of each visited node has to be compared with the key of the element to be inserted/found. Keys may be long and the run time may increase much.

**Operations on Binary Search Tree**

There are many operations which can be performed on a binary search tree.

**Searching in BST**: Finding the location of some specific element in a binary search tree.

**Insertion in BST**: Adding a new element to the binary search tree at the appropriate location so that the property of BST do not violate.

**Deletion in BST**: Deleting some specific node from a binary search tree. However, there can be various cases in deletion depending upon the number of children, the node have.

**Code:-**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct book

{ char name[30];

char a\_name[35];

int isbn;

int amount;

int borrowed;

struct book \*left, \*right;

}book;

typedef struct student

{ char name[35];

char uname[15];

char pass[15];

char contact[11];

int rollno;

struct borrows \*bor;

struct student \*left, \*right;

}student;

typedef struct libuser

{ char name[35];

char uname[15];

char pass[15];

}libuser;

libuser librar;

struct borrows

{ student \*s;

book \*b;

struct date

{ int day;

int month;

int year;

}issue,due;

struct borrows \*next;};

int libexist=0;

FILE \*fp;

void Linterface(book \*\*broot, student \*\*sroot, struct borrows \*\*bor\_head);

void Sinterface(book \*\*broot, student \*\*sroot);

void insertbook(book \*\*broot);

void insertstudent(student \*\*sroot);

void deletebook(book \*\*broot);

void deletestudent(student \*\*sroot);

void display\_book(book \*b);

void display\_student(student \*s);

void display\_allbooks(book \*broot);

void editbook(book \*broot);

void issue\_book(book \*broot, student \*sroot,struct borrows \*\*bor\_head);

void reclaim\_book(struct borrows \*\*bor\_head);

void display\_borrows(struct borrows \*bor\_head);

book\* search\_book(int isbn, book \*broot);

student\* search\_student(int rollno, student \*sroot);

void display\_allstudents(student \*sroot);

void editstudent(student \*sroot);

void book\_table(book \*broot);

void init(student \*\*sroot, book \*\*broot,struct borrows \*\*bor\_head);

void close(student \*sroot,book \*broot, struct borrows \*bor\_head);

void write\_student(student \*sroot);

void write\_book(book \*broot);

void write\_borrows(struct borrows \*bor\_head);

student\* read\_student();

book\* read\_book();

struct borrows\* read\_borrows(student \*sroot, book \*broot);

student\* student\_login(student \*sroot,char uname[],char pass[]);

int main()

{ int c=0;

book \*broot=NULL;

student \*sroot=NULL;

struct borrows \*bor\_head=NULL;

init(&sroot,&broot,&bor\_head);

do

{ if(c!=0)

{

printf("Press Enter to continue..");

getchar();

system("cls");

}

printf("Menu:-\n1. Login as Librarian\n2. Login as Student\n3. Exit\n");

printf("Please enter your choice: ");

scanf("%d",&c);

getchar(); //To deal with input buffer

switch (c)

{ case 1: Linterface(&broot,&sroot,&bor\_head);

break;

case 2: Sinterface(&broot,&sroot);

case 3: break;

default: printf("Invalid choice try again\n");}}while(c!=3);

close(sroot,broot,bor\_head);

return 0;}

void Linterface(book \*\*broot, student \*\*sroot,struct borrows \*\*bor\_head)

{ char uname[15],pass[15];

int c=0;

int rollno;

int isbn;

book \*sbook;

student \*sstudent;

system("cls");

if(libexist==0)

{ printf("No Librarian found, please create an account\n");

printf("Enter your name: ");

scanf("%[^\n]",librar.name);

printf("Enter your user name: ");

scanf("%s",librar.uname);

printf("Enter your password: ");

scanf("%s",librar.pass);

libexist=1;

system("cls");}

printf("Login as Librarian:-\nEnter your user name: ");

scanf("%s",uname);

printf("Enter your password: ");

scanf("%s",pass);

if(strcmp(uname,librar.uname)!=0||strcmp(pass,librar.pass)!=0)

{ printf("Invalid user name or password\n");

return;}

system("cls");

printf("Welcome %s\n",librar.name);

do

{ if(c!=0)

{ printf("Press Enter to continue..");

getchar();

system("cls");}

printf("Main Menu\n");

printf("1. Add a book\n2. Delete a book\n3. Search for a book\n4. Display all books\n");

printf("5. Edit a books details\n6. Issue a book\n7. Reclaim a book\n");

printf("8. Browse all issues\n9. Add a student\n10. Delete a student\n11. Search for a student\n");

printf("12. Display all students\n13. Edit a students details\n14. Log Out\n");

printf("Enter your choice: ");

scanf("%d",&c);

getchar(); //Input buffer

system("cls");

switch (c)

{ case 1: insertbook(broot);

break;

case 2: deletebook(broot);

break;

case 3: if(\*broot==NULL)

{ printf("No books found\n");

break;

}

printf("-------------------------------------------------\n");

printf("| ISBN No. | Book Name |\n");

printf("-------------------------------------------------\n");

book\_table(\*broot);

printf("ISBN of Required Book: ");

scanf("%d",&isbn);

getchar(); //Input Buffer

sbook=search\_book(isbn, \*broot);

if(sbook==NULL)

printf("Book Not found\n");

else

display\_book(sbook);

break;

case 4: if(\*broot!=NULL)

{ printf("Books are:-\n\n");

display\_allbooks(\*broot);

}

else

printf("No books to display\n");

break;

case 5: if(\*broot!=NULL)

editbook(\*broot);

else

printf("No books found\n");

break;

case 6: if(\*broot==NULL)

{ printf("No books found\n");

break;

}

issue\_book(\*broot,\*sroot,bor\_head);

break;

case 7: reclaim\_book(bor\_head);

break;

case 8: display\_borrows(\*bor\_head);

break;

case 9: insertstudent(sroot);

break;

case 10: deletestudent(sroot);

break;

case 11: printf("Roll Number of Required Student: ");

scanf("%d",&rollno);

getchar(); //Input Buffer

sstudent=search\_student(rollno,\*sroot);

if(sstudent==NULL)

printf("Student not found\n");

else

display\_student(sstudent);

break;

case 12: if(\*sroot!=NULL)

{ printf("Students are:-\n\n");

display\_allstudents(\*sroot);

}

else

printf("No students to display\n");

break;

case 13: editstudent(\*sroot);

break;

case 14: break;

default: printf("Invalid choice, try again\n");

}}while(c!=14);}

void Sinterface(book \*\*broot, student \*\*sroot)

{ int c=0;

int isbn;

student \*s1;

book \*sbook;

char uname[15],pass[15];

system("cls");

printf("Login as Student\nEnter your Username: ");

scanf("%s",uname);

printf("Enter your Password: ");

scanf("%s",pass);

s1=student\_login(\*sroot,uname,pass);

if(s1==NULL)

{ printf("Invalid Username or Password\n");

return;}

system("cls");

printf("Welcome %s\n",s1->name);

do

{ if(c!=0)

{ printf("Press Enter to continue..");

getchar();

system("cls");

}

printf("Main Menu\n");

printf("1. Display Profile\n2. Change Password\n3. Search for a book\n");

printf("4. Browse Books\n5. Log Out\n");

printf("Enter your choice: ");

scanf("%d",&c);

getchar(); //Input Buffer

system("cls");

switch (c)

{

case 1: display\_student(s1);

break;

case 2: printf("Enter new Password: ");

scanf("%s",s1->pass);

printf("Password changed\nLogin again to continue\n");

return;

break;

case 3: printf("ISBN of Required Book: ");

scanf("%d",&isbn);

getchar(); //Input Buffer

sbook=search\_book(isbn, \*broot);

if(sbook==NULL)

printf("Book not found\n");

else

display\_book(sbook);

break;

case 4: if(\*broot!=NULL)

{ printf("Books are:-\n\n");

display\_allbooks(\*broot);

}

else

printf("No books to display\n");

break;

case 5: break;

default: printf("Invalid Choice, try again\n");

break;

}}while(c!=5);}

void insertbook(book \*\*broot)

{ book \*ptr,\*prev,\*p;

ptr=(book \*)malloc(sizeof(book));

printf("Enter details to insert book\n");

printf("Book Name: ");

scanf("%[^\n]",ptr->name);

getchar(); //INput Buffer

printf("Author Name: ");

scanf("%[^\n]",ptr->a\_name);

printf("ISBN: ");

scanf("%d",&(ptr->isbn));

getchar(); //Input Buffer

printf("Availability: ");

scanf("%d",&(ptr->amount));

getchar(); //Input Buffer

ptr->borrowed=0;

ptr->left=ptr->right=NULL;

p=\*broot;

prev=NULL;

while(p!=NULL)

{ prev=p;

if((ptr->isbn)<(p->isbn))

p=p->left;

else

p=p->right;}

if(prev==NULL)

\*broot=ptr;

else

{ if((ptr->isbn)<(prev->isbn))

prev->left=ptr;

else

prev->right=ptr;}

}

void insertstudent(student \*\*sroot)

{ student \*ptr,\*prev,\*p;

ptr=(student \*)malloc(sizeof(student));

printf("Enter details to insert student\n");

printf("Student Name: ");

scanf("%[^\n]",ptr->name);

printf("User Name: ");

scanf("%s",ptr->uname);

printf("Password: ");

scanf("%s",ptr->pass);

printf("Contact Number: ");

scanf("%s",ptr->contact);

printf("Roll No: ");

scanf("%d",&(ptr->rollno));

getchar(); //Input Buffer

ptr->left=ptr->right=NULL;

ptr->bor=NULL;

p=\*sroot;

prev=NULL;

while(p!=NULL)

{ prev=p;

if((ptr->rollno)<(p->rollno))

p=p->left;

else

p=p->right;}

if(prev==NULL)

\*sroot=ptr;

else

{ if((ptr->rollno)<(prev->rollno))

prev->left=ptr;

else

prev->right=ptr;}

}

void deletebook(book \*\*broot)

{ int isbn;

char c;

book \*parent,\*del,\*delparent=NULL,\*rp;

if(\*broot==NULL)

{ printf("No Books found\n");

return;}

printf("-------------------------------------------------\n");

printf("| ISBN No. | Book Name |\n");

printf("-------------------------------------------------\n");

book\_table(\*broot);

printf("Enter ISBN of book to delete: ");

scanf("%d",&isbn);

getchar(); //Input Buffer

del=\*broot;

while(del!=NULL&&del->isbn!=isbn)

{ delparent=del;

if(isbn<del->isbn)

del=del->left;

else

del=del->right;}

if(del==NULL)

{ printf("Book not found\n");

return;}

printf("Book found:-\n");

display\_book(del);

printf("Do you really want to delete this book\nEnter y for yes n for no: ");

scanf("%c",&c);

getchar(); //Input Buffer

if(c!='y'&&c!='Y')

return;

if(del->left==NULL)

rp=del->right;

else if(del->right==NULL)

rp=del->left;

else

{ parent=del;

rp=del->right;

while(rp->left!=NULL)

{ parent=rp;

rp=rp->left;}

if(parent!=del)

{ parent->left=rp->right;

rp->right=del->right;}

rp->left=del->left;

}

if(delparent==NULL)

\*broot=rp;

else if(delparent->left==del)

delparent->left=rp;

else

delparent->right=rp;

free(del);

printf("Book Deleted\n");

}

void deletestudent(student \*\*sroot)

{ int rollno;

char c;

student \*parent,\*del,\*delparent=NULL,\*rp;

if(\*sroot==NULL)

{ printf("No Students found\n");

return;}

printf("Enter Roll Number of student to delete: ");

scanf("%d",&rollno);

getchar(); //Input Buffer

del=\*sroot;

while(del!=NULL&&del->rollno!=rollno)

{ delparent=del;

if(rollno<del->rollno)

del=del->left;

else

del=del->right;}

if(del==NULL)

{ printf("Student not found\n");

return;}

printf("Student found:-\n");

display\_student(del);

if(del->bor!=NULL)

{ printf("%s has yet to return a book please reclaim it first.\n",del->name);

return;}

printf("Do you really want to delete the records of %s?\n",del->name);

printf("Enter y for yes n for no: ");

scanf("%c",&c);

getchar(); //Input Buffer

if(c!='y'&&c!='Y')

return;

if(del->left==NULL)

rp=del->right;

else if(del->right==NULL)

rp=del->left;

else

{ parent=del;

rp=del->right;

while(rp->left!=NULL)

{ parent=rp;

rp=rp->left;}

if(parent!=del)

{ parent->left=rp->right;

rp->right=del->right;}

rp->left=del->left;

}

if(delparent==NULL)

\*sroot=rp;

else if(delparent->left==del)

delparent->left=rp;

else

delparent->right=rp;

free(del);

printf("Student Deleted\n");

}

void display\_book(book \*b)

{ printf("Book Name: %s\n",b->name);

printf("Author Name: %s\n",b->a\_name);

printf("ISBN: %d\n",b->isbn);

printf("Quantity held by Library: %d\n",b->amount);

printf("Quantity Borrowed by members: %d\n",b->borrowed);}

void display\_student(student \*s)

{ printf("Student Name: %s\n",s->name);

printf("Roll Number: %d\n",s->rollno);

printf("Contact Number: %s\n",s->contact);

if(s->bor!=NULL)

{ printf("Borrowed Book:-\n");

display\_book(s->bor->b);

printf("Issue Date: %d/%d/%d\n",s->bor->issue.day,s->bor->issue.month,s->bor->issue.year);

printf("Due Date: %d/%d/%d\n",s->bor->due.day,s->bor->due.month,s->bor->due.year);}

}

void display\_allbooks(book \*broot)

{ if(broot!=NULL)

{ display\_allbooks(broot->left);

display\_book(broot);

printf("\n\n");

display\_allbooks(broot->right);}

}

void editbook(book \*broot)

{ int isbn,c,qty;

printf("-------------------------------------------------\n");

printf("| ISBN No. | Book Name |\n");

printf("-------------------------------------------------\n");

book\_table(broot);

printf("Enter ISBN of book to edit: ");

scanf("%d",&isbn);

getchar(); //Input Buffer

while(broot!=NULL&&broot->isbn!=isbn)

{ if(isbn<broot->isbn)

broot=broot->left;

else

broot=broot->right;}

if(broot==NULL)

{ printf("Book not found\n");

return;}

display\_book(broot);

do

{ printf("Options:-\n");

printf("1. Change Book Name\n2. Change Author Name\n");

printf("3. Change Quantity held by library\n4. Finish Editing\n");

printf("Enter your choice: ");

scanf("%d",&c);

getchar(); //Input Buffer

switch (c)

{case 1: printf("New Book Name: ");

scanf("%[^\n]",broot->name);

break;

case 2: printf("New Author Name: ");

scanf("%[^\n]",broot->a\_name);

break;

case 3: printf("New quantity held by library: ");

scanf("%d",&qty);

getchar();

if(qty<broot->borrowed)

printf("Please reclaim the required amount of books to decrease the amount\n");

else

broot->amount=qty;

break;

case 4: break;

default:

printf("Invalid choice, please try again\n");

break;

}}while(c!=4);

}

void issue\_book(book \*broot, student \*sroot,struct borrows \*\*bor\_head)

{ int rollno,isbn;

struct borrows \*ptr,\*prev,\*p;

printf("Roll Number of Borrower: ");

scanf("%d",&rollno);

getchar(); //Input Buffer

while(sroot!=NULL&&sroot->rollno!=rollno)

{ if(rollno<sroot->rollno)

sroot=sroot->left;

else

sroot=sroot->right;}

if(sroot==NULL)

{ printf("Student Not found\n");

return;}

printf("Student Found:-\n");

display\_student(sroot);

if(sroot->bor!=NULL)

{ printf("%s has yet to return a book, please reclaim it first\n",sroot->name);

return;}

printf("-------------------------------------------------\n");

printf("| ISBN No. | Book Name |\n");

printf("-------------------------------------------------\n");

book\_table(broot);

printf("ISBN of book to issue: ");

scanf("%d",&isbn);

getchar(); //Input Buffer

while(broot!=NULL&&broot->isbn!=isbn)

{ if(isbn<broot->isbn)

broot=broot->left;

else

broot=broot->right;}

if(broot==NULL)

{ printf("Book not found\n");

return;}

printf("Book found:-\n");

display\_book(broot);

if(broot->amount<=broot->borrowed)

{ printf("Not enough copies of book to issue\n");

return;}

ptr=(struct borrows\*)malloc(sizeof(struct borrows));

ptr->s=sroot;

ptr->b=broot;

sroot->bor=ptr;

broot->borrowed=(broot->borrowed)+1;

printf("Enter issue date as follows, DD MM YYYY: ");

scanf("%d%d%d", &(ptr->issue.day),&(ptr->issue.month),&(ptr->issue.year));

getchar(); //Input Buffer

printf("Enter due date as follows, DD MM YYYY: ");

scanf("%d%d%d", &(ptr->due.day),&(ptr->due.month),&(ptr->due.year));

getchar(); //Input Buffer

ptr->next=NULL;

p=\*bor\_head;

prev=NULL;

while(p!=NULL)

{ prev=p;

p=p->next;}

if(prev==NULL)

\*bor\_head=ptr;

else

prev->next=ptr;

}

void reclaim\_book(struct borrows \*\*bor\_head)

{ int rollno;

struct borrows \*reclaim,\*prev;

if(\*bor\_head==NULL)

{ printf("No books to reclaim\n");

return;}

printf("Roll Number of Borrower: ");

scanf("%d",&rollno);

getchar(); //Input Buffer

reclaim=\*bor\_head;

while(reclaim!=NULL&&reclaim->s->rollno!=rollno)

{ prev=reclaim;

reclaim=reclaim->next;}

if(reclaim==NULL)

{ printf("Found no Student with Roll Number %d having borrowed any book\n",rollno);

return;}

printf("Student Found:-\n");

display\_student(reclaim->s);

reclaim->b->borrowed=(reclaim->b->borrowed)-1;

if(reclaim==\*bor\_head)

\*bor\_head=(\*bor\_head)->next;

else

prev->next=reclaim->next;

reclaim->s->bor=NULL;

free(reclaim);

printf("Reclaimed Book\n");

}

void display\_borrows(struct borrows \*bor\_head)

{ if(bor\_head==NULL)

{ printf("No books have been issued\n");

return;}

printf("Details of Borrows:-\n\n");

while(bor\_head!=NULL)

{ display\_student(bor\_head->s);

printf("\n\n");

bor\_head=bor\_head->next;}

}

book\* search\_book(int isbn, book \*broot)

{ while(broot!=NULL&&isbn!=broot->isbn)

{ if(isbn<broot->isbn)

broot=broot->left;

else

broot=broot->right;}

return broot;

}

student\* search\_student(int rollno, student \*sroot)

{ while(sroot!=NULL&&rollno!=sroot->rollno)

{ if(rollno<sroot->rollno)

sroot=sroot->left;

else

sroot=sroot->right;}

return sroot;

}

void display\_allstudents(student \*sroot)

{ if(sroot!=NULL)

{

display\_allstudents(sroot->left);

display\_student(sroot);

printf("\n\n");

display\_allstudents(sroot->right);}

}

void editstudent(student \*sroot)

{ int rollno,c;

printf("Enter Roll No of student to edit: ");

scanf("%d",&rollno);

getchar(); //Input Buffer

while(sroot!=NULL&&sroot->rollno!=rollno)

{ if(rollno<sroot->rollno)

sroot=sroot->left;

else

sroot=sroot->right;}

if(sroot==NULL)

{ printf("Student not found\n");

return;}

display\_student(sroot);

do

{ printf("Options:-\n");

printf("1. Change Student Name\n2. Change User Name\n");

printf("3. Change Password\n4. Change Contact Number\n5. Finish editing\n");

printf("Enter your choice: ");

scanf("%d",&c);

getchar(); //Input Buffer

switch (c)

{

case 1: printf("Changed Student Name: ");

scanf("%[^\n]",sroot->name);

break;

case 2: printf("Changed User Name: ");

scanf("%s",sroot->uname);

break;

case 3: printf("Changed Password: ");

scanf("%s",sroot->pass);

break;

case 4: printf("Changed Contact Number: ");

scanf("%s",sroot->contact);

break;

case 5: break;

default: printf("Invalid choice, please try again\n");

break;

}}while(c!=5);

}

student\* student\_login(student \*sroot,char uname[],char pass[])

{ student \*temp=NULL;

if(sroot==NULL)

return NULL;

if((strcmp(uname,sroot->uname)==0)&&(strcmp(pass,sroot->pass)==0))

return sroot;

else

{ temp=student\_login(sroot->left,uname,pass);

if(temp==NULL)

temp=student\_login(sroot->right,uname,pass);}

return temp;

}

void book\_table(book \*broot)

{ if(broot->left!=NULL)

book\_table(broot->left);

printf("| %-12d| %-30s|\n",broot->isbn,broot->name);

printf("-------------------------------------------------\n");

if(broot->right!=NULL)

book\_table(broot->right);

}

void init(student \*\*sroot, book \*\*broot,struct borrows \*\*bor\_head)

{ fp=NULL;

int i=0;

fp=fopen("./librar.bin","rb");

if(fp==NULL)

{ fp=fopen("./librar.bin","wb");

fwrite(&libexist,sizeof(int),1,fp);

fclose(fp);

fp=fopen("./librar.bin","rb");}

fread(&libexist,sizeof(int),1,fp);

if(libexist==1)

fread(&librar,sizeof(libuser),1,fp);

fclose(fp);

fp=fopen("./students.bin","rb");

i=0;

if(fp==NULL)

{ fp=fopen("./students.bin","wb");

fwrite(&i,sizeof(int),1,fp);

fclose(fp);

fp=fopen("./students.bin","rb");}

fread(&i,sizeof(int),1,fp);

if(i==1)

\*sroot=read\_student();

fclose(fp);

fp=fopen("./books.bin","rb");

i=0;

if(fp==NULL)

{ fp=fopen("./books.bin","wb");

fwrite(&i,sizeof(int),1,fp);

fclose(fp);

fp=fopen("./books.bin","rb");}

fread(&i,sizeof(int),1,fp);

if(i==1)

\*broot=read\_book();

fclose(fp);

fp=fopen("./borrows.bin","rb");

i=0;

if(fp==NULL)

{ fp=fopen("./borrows.bin","wb");

fwrite(&i,sizeof(int),1,fp);

fclose(fp);

fp=fopen("./borrows.bin","rb");}

fread(&i,sizeof(int),1,fp);

if(i==1)

\*bor\_head=read\_borrows(\*sroot,\*broot);

fclose(fp);

}

void close(student \*sroot,book \*broot, struct borrows \*bor\_head)

{ fp=NULL;

int i=0;

fp=fopen("./librar.bin","wb");

if(fp==NULL)

printf("Something went wrong\n");

else

{ fwrite(&libexist,sizeof(int),1,fp);

if(libexist==1)

fwrite(&librar,sizeof(libuser),1,fp);

fclose(fp);

}

fp=fopen("./students.bin","wb");

if(fp==NULL)

printf("Something went wrong\n");

else

{ if(sroot!=NULL)

i=1;

fwrite(&i,sizeof(int),1,fp);

if(sroot!=NULL)

write\_student(sroot);

i=0;

fclose(fp);

}

fp=fopen("./books.bin","wb");

if(fp==NULL)

printf("Something went wrong\n");

else

{ if(broot!=NULL)

i=1;

fwrite(&i,sizeof(int),1,fp);

if(broot!=NULL)

write\_book(broot);

i=0;

fclose(fp);}

fp=fopen("./borrows.bin","wb");

if(fp==NULL)

printf("Something went wrong\n");

else

{ if(bor\_head!=NULL)

i=1;

fwrite(&i,sizeof(int),1,fp);

if(bor\_head!=NULL)

write\_borrows(bor\_head);

i=0;

fclose(fp);}

}

void write\_student(student \*sroot)

{ fwrite(sroot,sizeof(student),1,fp);

if(sroot->left!=NULL)

write\_student(sroot->left);

if(sroot->right!=NULL)

write\_student(sroot->right);

}

void write\_book(book \*broot)

{ fwrite(broot,sizeof(book),1,fp);

if(broot->left!=NULL)

write\_book(broot->left);

if(broot->right!=NULL)

write\_book(broot->right);

}

void write\_borrows(struct borrows \*bor\_head)

{ fwrite(&(bor\_head->s->rollno),sizeof(int),1,fp);

fwrite(&(bor\_head->b->isbn),sizeof(int),1,fp);

fwrite(&(bor\_head->issue),sizeof(bor\_head->issue),1,fp);

fwrite(&(bor\_head->due),sizeof(bor\_head->due),1,fp);

fwrite(&(bor\_head->next),sizeof(bor\_head->next),1,fp);

if(bor\_head->next!=NULL)

write\_borrows(bor\_head->next);

}

student\* read\_student()

{ student \*sroot;

sroot=(student \*)malloc(sizeof(student));

fread(sroot,sizeof(student),1,fp);

if(sroot->left!=NULL)

sroot->left=read\_student();

if(sroot->right!=NULL)

sroot->right=read\_student();

return sroot;

}

book\* read\_book()

{ book \*broot;

broot=(book \*)malloc(sizeof(book));

fread(broot,sizeof(book),1,fp);

if(broot->left!=NULL)

broot->left=read\_book();

if(broot->right!=NULL)

broot->right=read\_book();

return broot;

}

struct borrows\* read\_borrows(student \*sroot, book \*broot)

{ int rollno,isbn;

struct borrows \*bor\_head;

bor\_head=(struct borrows\*)malloc(sizeof(struct borrows));

fread(&rollno,sizeof(int),1,fp);

bor\_head->s=search\_student(rollno,sroot);

bor\_head->s->bor=bor\_head;

fread(&isbn,sizeof(int),1,fp);

bor\_head->b=search\_book(isbn,broot);

fread(&(bor\_head->issue),sizeof(bor\_head->issue),1,fp);

fread(&(bor\_head->due),sizeof(bor\_head->due),1,fp);

fread(&(bor\_head->next),sizeof(bor\_head->next),1,fp);

if(bor\_head->next!=NULL)

bor\_head->next=read\_borrows(sroot, broot);

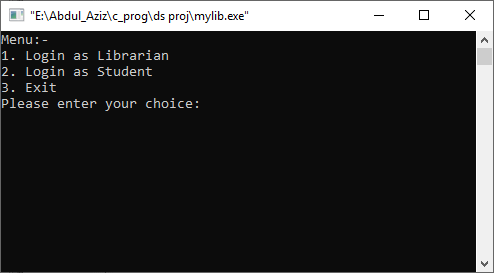
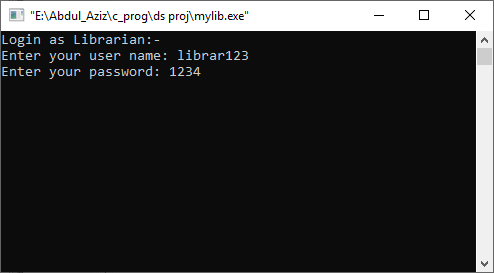
return bor\_head;

}

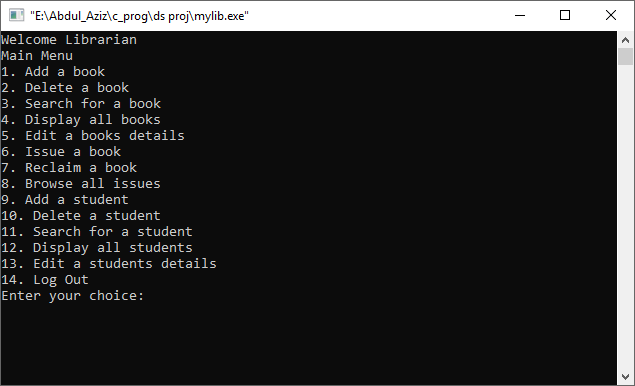
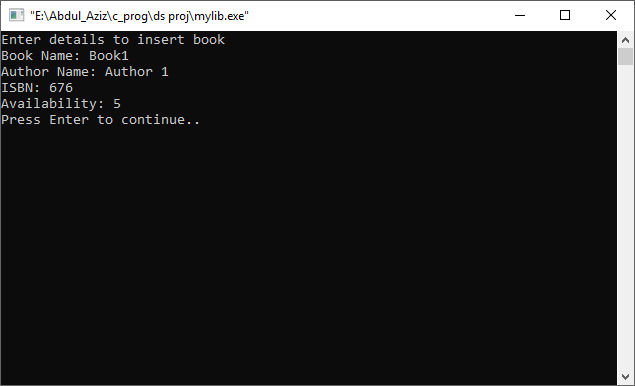
**Screenshots:-**

Given below are some screenshots of our project:-

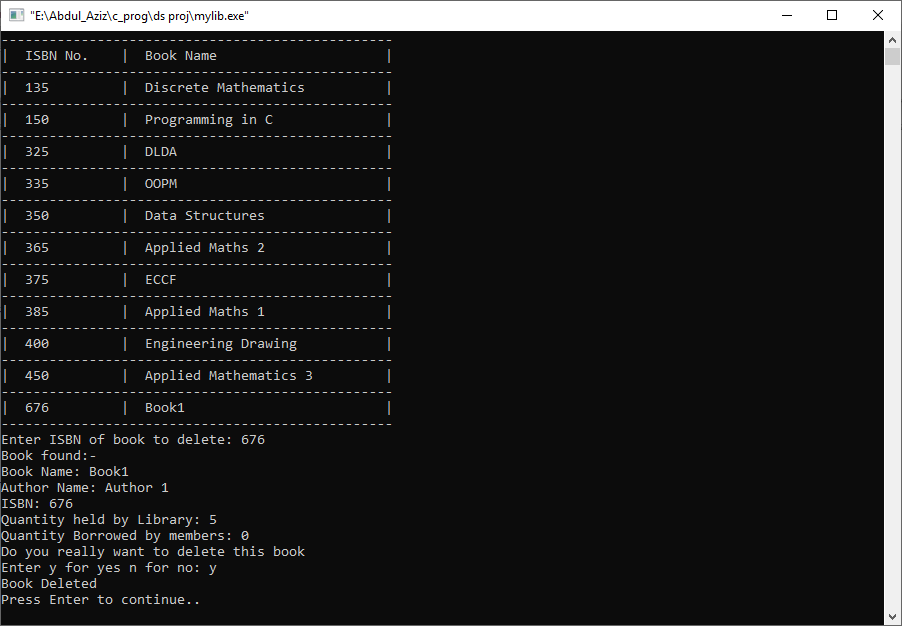
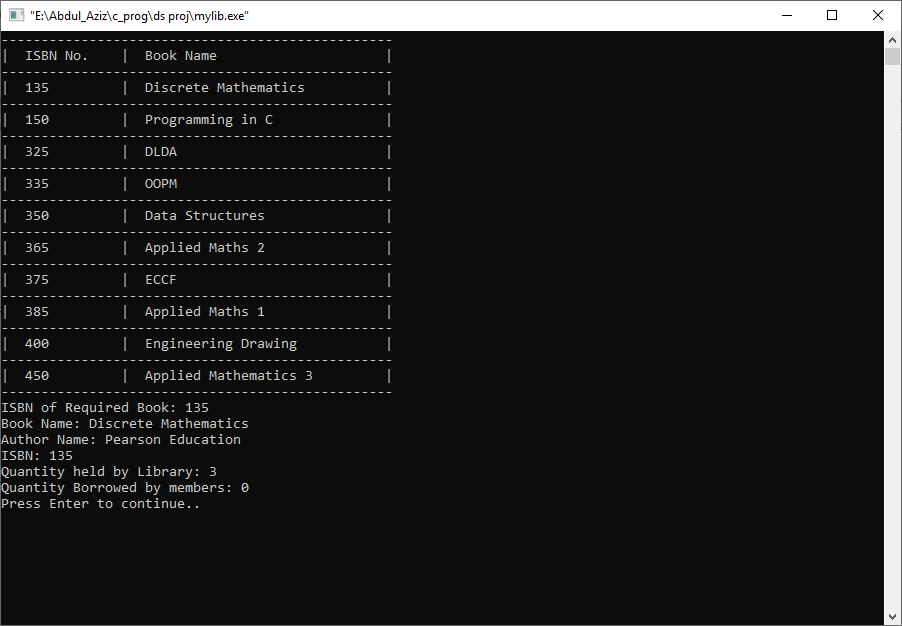
Start Page:- Login as librarian

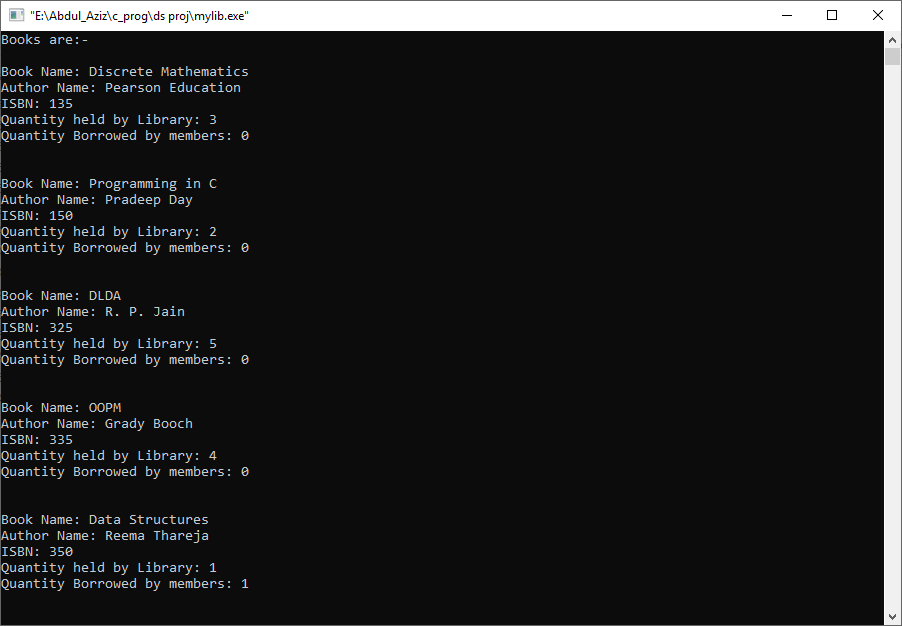
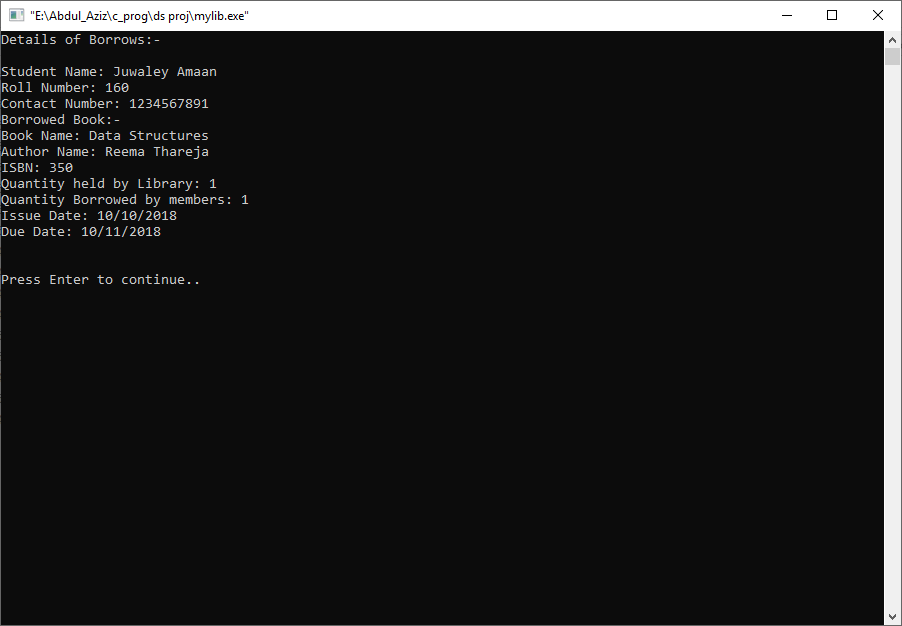
Librarian interface:- Adding a new book:-

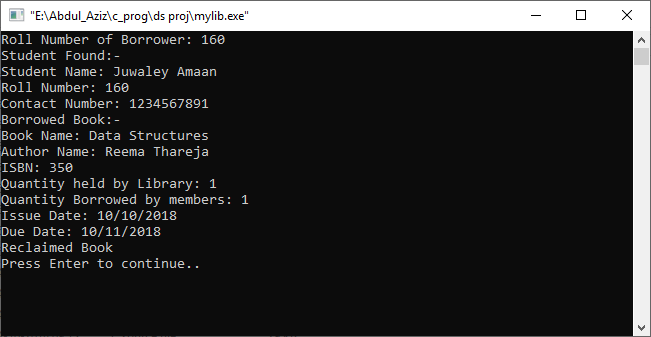
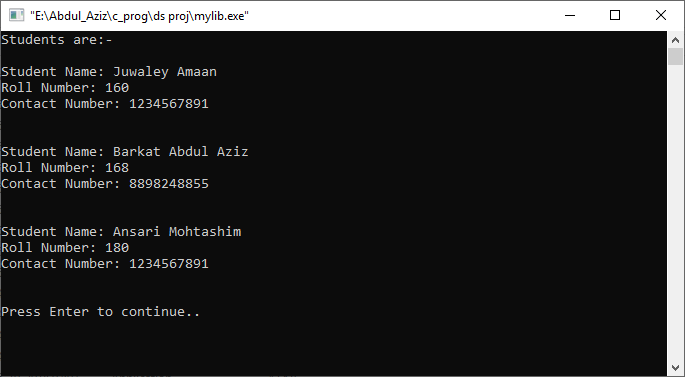
Deleting a book:- Searching for a book:-

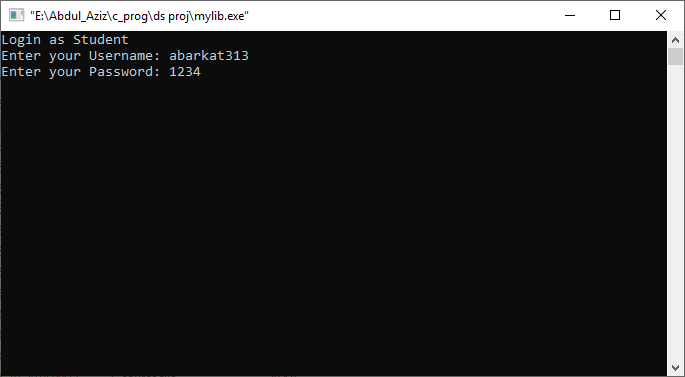
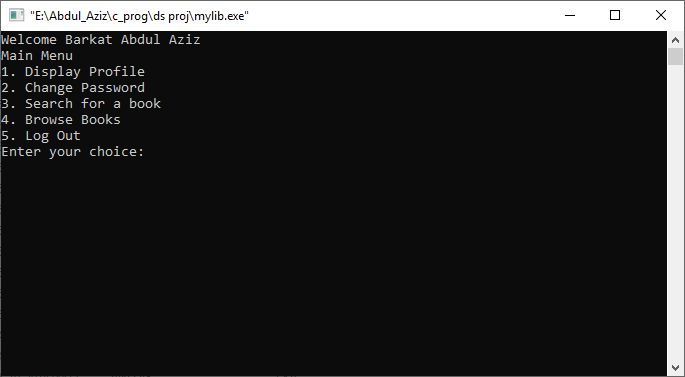
Displaying all books:- Displaying all borrows:-

Reclaiming a book:- displaying all students:-

Login as student:- Student interface:-

Changing password:-

