Library Management System Using Data Structures

Competitive Programming

A project Reported submitted by

M.Harshith-19BEV7022
M.Sravan Kumar-19BES7049

Under the Guidance of

Dr. Sumathi

Associate Professor, CSE,

VIT-AP

Table of content

INTRODUCTION

The majority of institutions have great infrastructural facilities and well-stocked libraries with a wide selection of literature. University libraries have an advantage over public libraries in terms of vast selection of books on engineering and science where readers may have the opportunity to learn a great deal about the relevant subjects. Because compared to the spread of information, this collection grows quickly every day.

Main motive of the project is library management where we can get the updates about the No.of books ,adding of books,deletion of books,update books.The system is divided into two sections: one for users and one for administrators. To manage accounts when the username and password have already been configured.

In a library, contextual information is very important as books are organized geographically based on subject and catalog number. Further, books are constantly being moved, loaned, or misplaced. Without additional support, a user may find the task of searching for books difficult, especially in larger libraries. On the other hand, librarians are tasked with the job of stack maintenance. Stack maintenance refers to the re-shelving of material that has been removed from a shelf, or shifting existing material within a stack to make room for new materials.

Background Study

Users may manage and arrange the library with the help of the Library System. This will facilitate and speed up the job of the library management. This approach is intended to make it easier for the librarian to recognise a book that the students or instructors desire to borrow. It will also make it simpler for the librarian to recognise unreturned books. The fundamental functions of a library system include adding new users, new books, and updated information, searching for books, allowing users and professors to check out books, and returning borrowed books.

This procedure would be made simpler by the Library System, which would help improve how quickly the library operates as a whole. It will also improve the efficiency of the library as a whole, from data entry to taking note of historical documents. The option for users or library members to not only browse the books but also to reserve books that have been issued by other users and request newer publications serves as the platform's main objective.

Problem Definition

A library management system is a framework that enables users to add/remove, download, and search books and book material. Given a constantly growing library, the issue is that users need a quick way to locate a certain book or keyword(s) inside a book.

Given a constantly growing library, the issue is that users need a quick way to locate a certain book or keyword(s) inside a book. Efficiency demands that, despite a growth in library material, processing times remain largely unchanged.

Objective

You will be able to:

- define library management;
- identify library management functions;
- list different structural components of a library system;
- Describe how the Acquisition Section, Technical Processing Section, Circulation Section, Reference Section, Periodicals Section, Maintenance Section, and Administration & Finance Section of a library operate.

<u>Methodology/Procedure</u>

List of Data Structures used with logic design

1) Binary Search Tree:

Binary search is used to search for a particular subject where the searching operation performs an in-order traversal on the created binary search tree to get the elements in sorted order, where the binary search tree is created using Double Linked List.

2) Hash-map:

Hash-maps often map keys to values without producing duplicate keys. Therefore, in our project, distinct keys are created using hash maps and allocated to each book so that we may store and retrieve records of books in a 2-D array using these keys.

3) 2-D array :

Here, we've used a 2-D array to keep the record of each book, with columns indicating the total number of books and rows denoting the total number of books that are now accessible.

<u>Operations to be performed on each data structure:</u>

Insertion: We are using insertion operation to add books in the binary search tree. And the tree will contain the name of the added books.

Deletion: To remove the node for that specific book from the binary search tree, we employ a deletion operation. The remaining items in the tree are once again reorganized once the node has been eliminated.

Updation: The librarian can update the quantity of already existing set of books which is stored in the 2D array

List or print all the values:
There is also the choice of printing all the book information. The title of the book, the number of copies that can be provided, and the total number of volumes the library has.

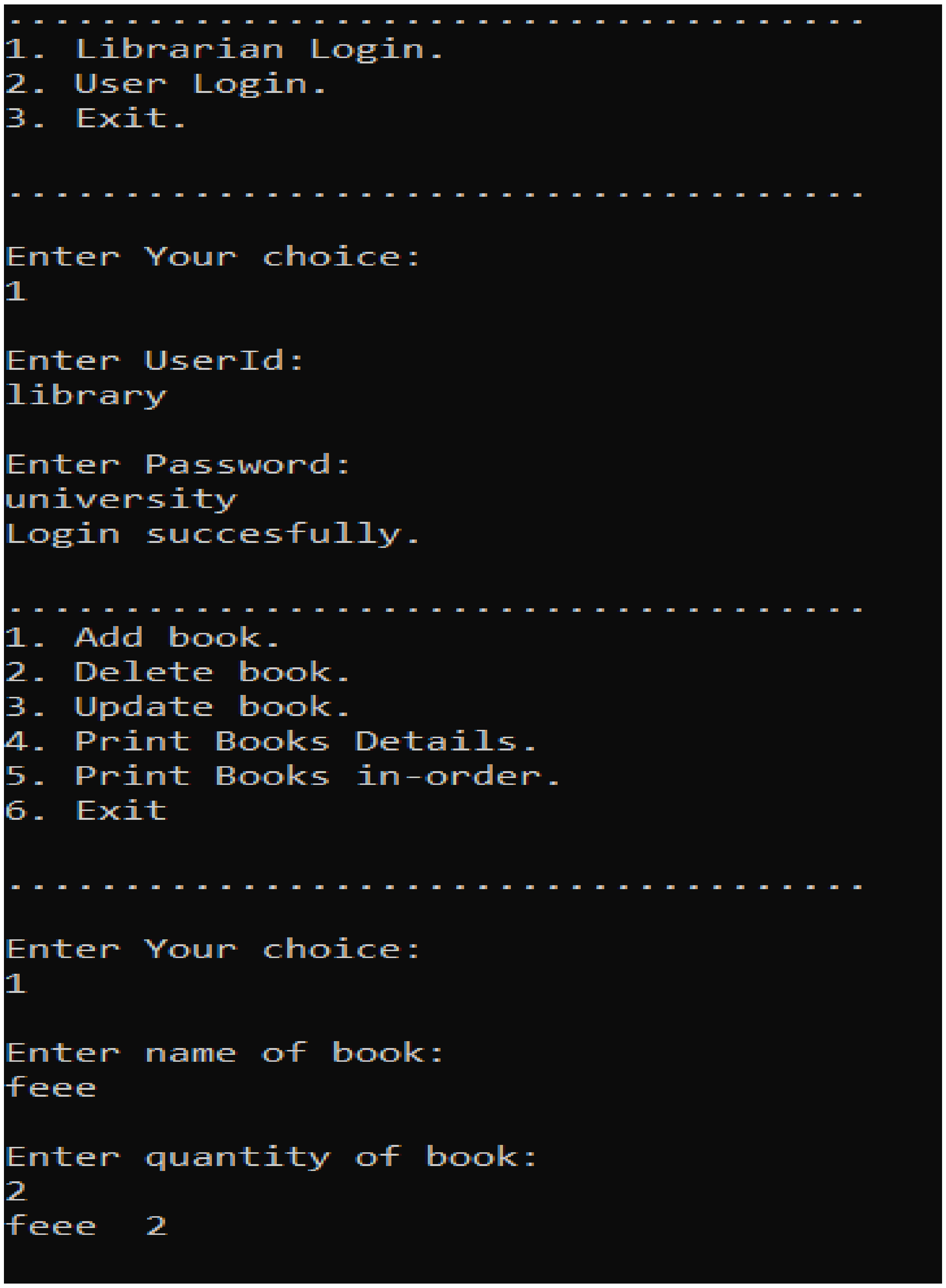
Print book in-order: Prints the contents of the binary search tree i.e. the names of the books in ascending order as the tree is balanced.

Requirements of processing on data structures :
Keeping elements in a specific order (ascending order) in the Binary Search tree is balanced.

Snapshots of results

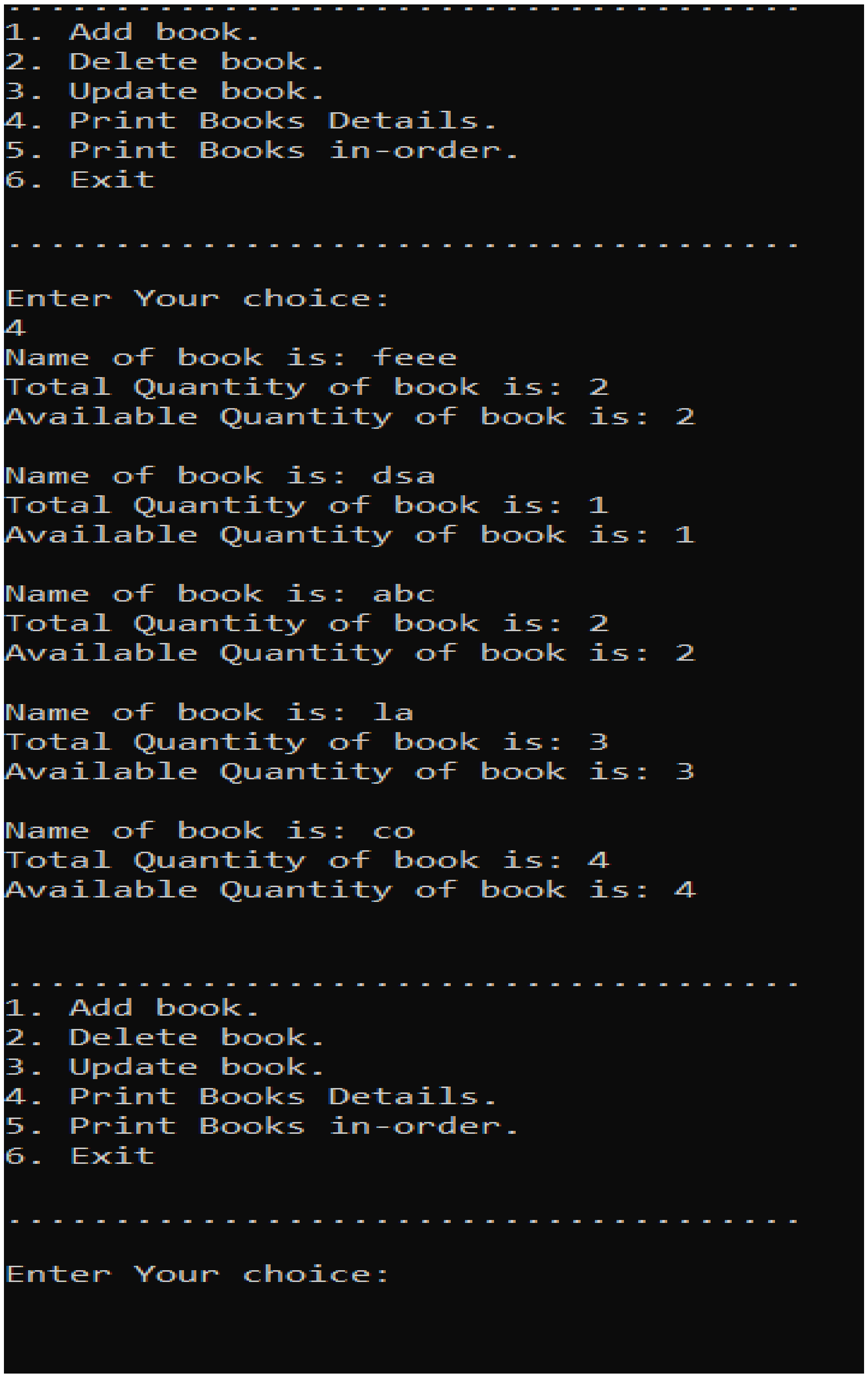
Librarian login

Adding book



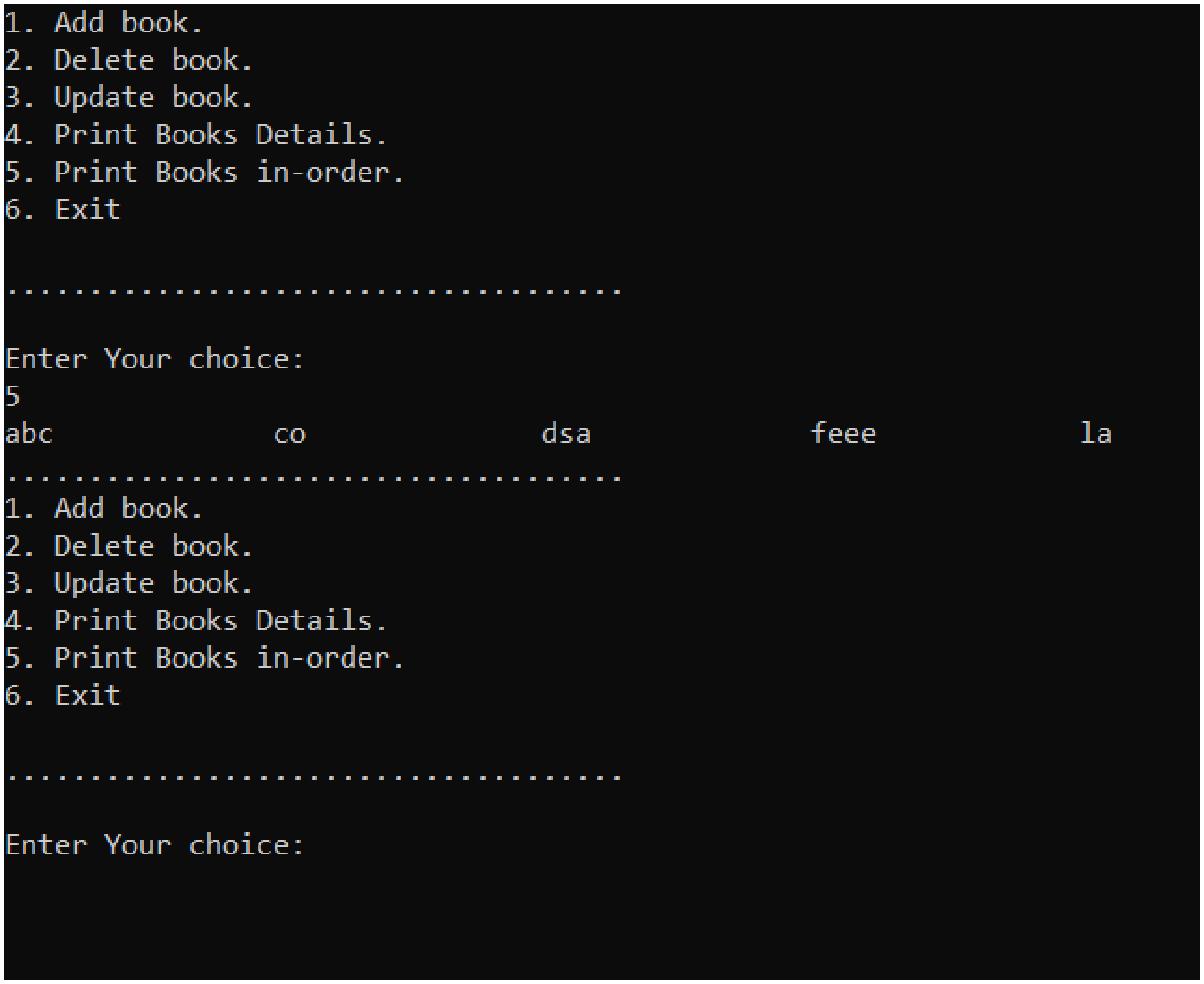
```
.....................................
1. Librarian Login.
2. User Login.
3. Exit.

.....................................

Enter Your choice:
1

Enter UserId:
library

Enter Password:
university
Login succesfully.

.....................................
1. Add book.
2. Delete book.
3. Update book.
4. Print Books Details.
5. Print Books in-order.
6. Exit

.....................................

Enter Your choice:
1

Enter name of book:
feee

Enter quantity of book:
2
feee  2
```

Printing book details



```
-------------------------------------
1. Add book.
2. Delete book.
3. Update book.
4. Print Books Details.
5. Print Books in-order.
6. Exit

-------------------------------------

Enter Your choice:
4
Name of book is: feee
Total Quantity of book is: 2
Available Quantity of book is: 2

Name of book is: dsa
Total Quantity of book is: 1
Available Quantity of book is: 1

Name of book is: abc
Total Quantity of book is: 2
Available Quantity of book is: 2

Name of book is: la
Total Quantity of book is: 3
Available Quantity of book is: 3

Name of book is: co
Total Quantity of book is: 4
Available Quantity of book is: 4


-------------------------------------
1. Add book.
2. Delete book.
3. Update book.
4. Print Books Details.
5. Print Books in-order.
6. Exit

-------------------------------------

Enter Your choice:
```

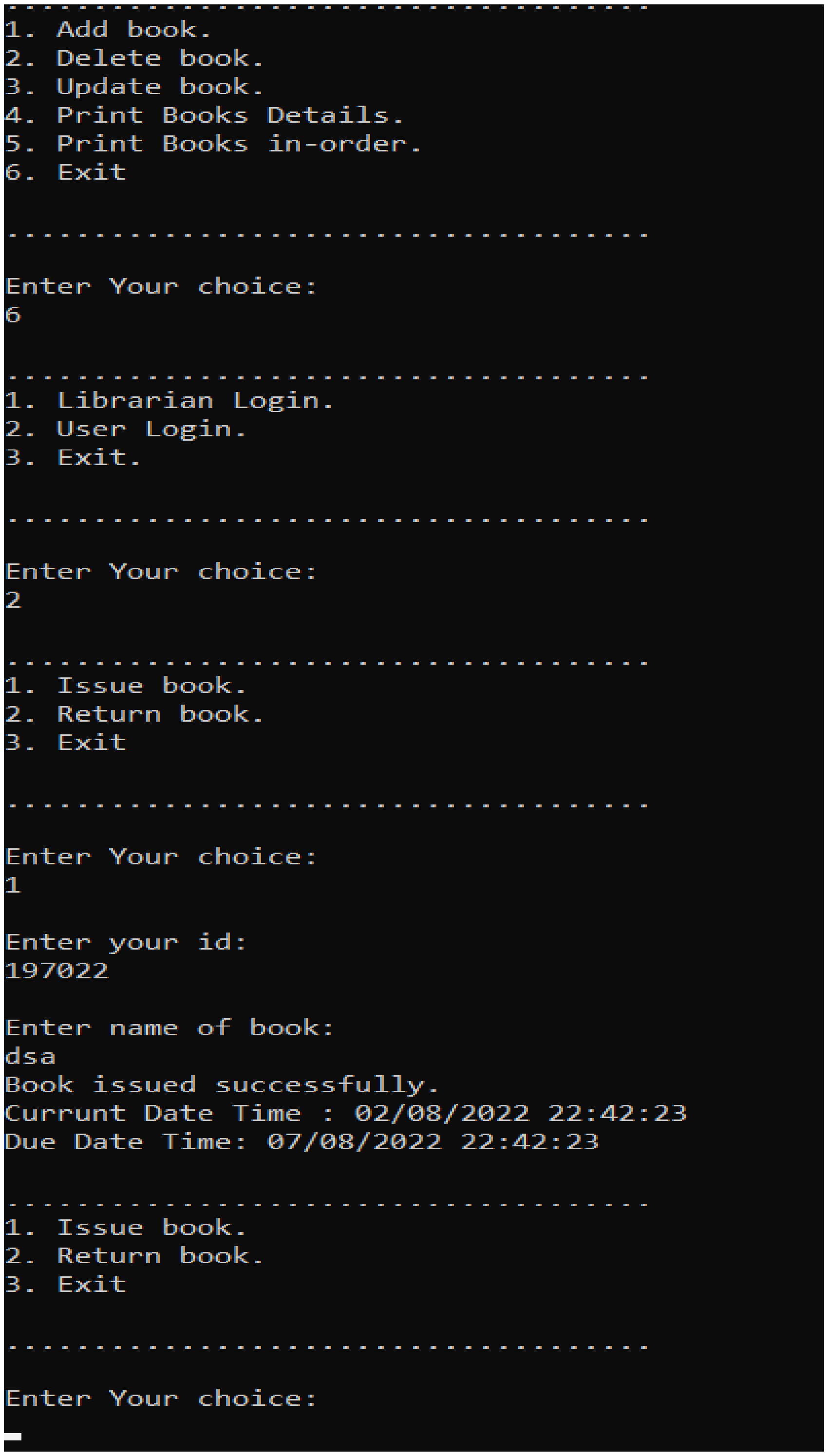
Printing books in order



```
1. Add book.
2. Delete book.
3. Update book.
4. Print Books Details.
5. Print Books in-order.
6. Exit

....................................

Enter Your choice:
5
abc             co              dsa             feee            la
....................................
1. Add book.
2. Delete book.
3. Update book.
4. Print Books Details.
5. Print Books in-order.
6. Exit

....................................

Enter Your choice:
```

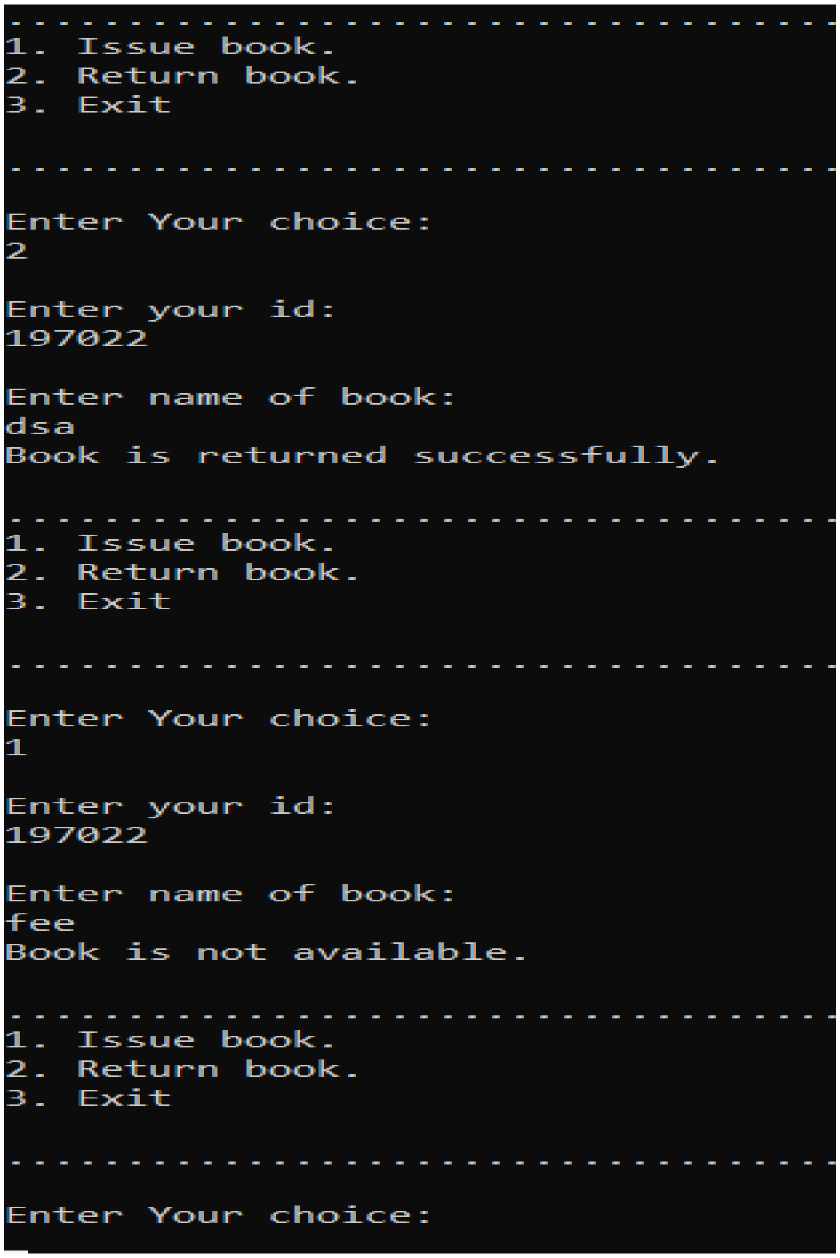
<u>**User ID**</u>

Issue book



```
......................................
1. Add book.
2. Delete book.
3. Update book.
4. Print Books Details.
5. Print Books in-order.
6. Exit

......................................

Enter Your choice:
6

......................................
1. Librarian Login.
2. User Login.
3. Exit.

......................................

Enter Your choice:
2

......................................
1. Issue book.
2. Return book.
3. Exit

......................................

Enter Your choice:
1

Enter your id:
197022

Enter name of book:
dsa
Book issued successfully.
Currunt Date Time : 02/08/2022 22:42:23
Due Date Time: 07/08/2022 22:42:23

......................................
1. Issue book.
2. Return book.
3. Exit

......................................

Enter Your choice:
```

Return book



```
-----------------------------------
1. Issue book.
2. Return book.
3. Exit

-----------------------------------

Enter Your choice:
2

Enter your id:
197022

Enter name of book:
dsa
Book is returned successfully.

-----------------------------------
1. Issue book.
2. Return book.
3. Exit

-----------------------------------

Enter Your choice:
1

Enter your id:
197022

Enter name of book:
fee
Book is not available.

-----------------------------------
1. Issue book.
2. Return book.
3. Exit

-----------------------------------

Enter Your choice:
```

CODE

```
import java.util.Date;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Map.Entry;
import java.util.Scanner;
import java.util.Set;


class Student
{
        String name;
        int id_no;
        String Stream;
        String book1,book2;
        int book_no,issuedbook;

        Student(String name,int id_no,String Stream)
        {
        this.name=name;
        this.id_no=id_no;
        this.Stream=Stream;

        }
}

public class library_management
{
        static void Selectionsort( Student arr[])
        {
            int n = arr.length;

            for (int i = 0; i < n-1; i++)
            {
                int min_idx = i;
                for (int j = i+1; j < n; j++)
                    if (arr[j].id_no < arr[min_idx].id_no)//Sort according to ID number of student
                        min_idx = j;


                String temp1 = arr[min_idx].name;
                arr[min_idx].name = arr[i].name;
                arr[i].name = temp1;
        String temp2 = arr[min_idx].Stream;
        arr[min_idx].Stream = arr[i].Stream;
        arr[i].Stream = temp2;

    }
}

static void display(Student arr[])
{
 for(int i=0;i<arr.length;i++)
 {
 System.out.println("\nName of Student:" + arr[i].name);
 System.out.println("\nId of Student:" + arr[i].id_no);
 System.out.println("\nStream of Student:" + arr[i].Stream);
 }
}


class Node
{
String key;
Node left, right;



public Node(String item)
{
key = item;
left = null;
right=null;
}
}



Node root;
private static Scanner input;
void finaldsa()
{
root = null;
}



//Insert Book
void insert(String key)
{
root = insertRec(root, key);
}



Node insertRec(Node root, String key)
{
if(root == null)
```

```
{
root = new Node(key);
return root;
}

if (key.compareTo(root.key)<0) //If book name < root then place it as left child
root.left = insertRec(root.left, key);
else if (key.compareTo(root.key)>0) //If book name > root then place it as Right child
root.right = insertRec(root.right, key);
else
System.out.println("error.");


return root;
}

void update(String key,String key1)
{
deleteKey(key);
insert(key1);
}



//Search Book
public boolean containsNode(String value)
{
return containsNodeRecursive(root, value);
}



private boolean containsNodeRecursive(Node current, String key)
{
if (current == null)
{
return false;
}
//If book name < root then place it as left child
if (key.equalsIgnoreCase(current.key))
{
return true;
}

//If book name < root then search at left side of root else right side
return key.compareTo(current.key)<0
? containsNodeRecursive(current.left, key)
: containsNodeRecursive(current.right, key);
}





void deleteKey(String key)
{
    root = deleteRec(root, key);
}

Node deleteRec(Node root, String key)
{
    if (root == null)  return root;

  //If book name < root then search it at left side and delete
    if (key.compareTo(root.key)<0)
        root.left = deleteRec(root.left, key);
  //If book name > root then search it at right side and delete
    else if (key.compareTo(root.key)<0)
        root.right = deleteRec(root.right, key);

    else
    {
        if (root.left == null)
            return root.right;
        else if (root.right == null)
            return root.left;

        root.key = minValue(root.right);

        root.right = deleteRec(root.right, root.key);
    }

    return root;
}

String minValue(Node root)
{
    String minv = root.key;
    while (root.left != null)
    {
        minv = root.left.key;
        root = root.left;
    }
    return minv;
}
     //Print Books Inorder
     void printInorder(Node node)
     {
     if (node == null)
     return;



     printInorder(node.left);



     System.out.print(node.key + "                              ");



     printInorder(node.right);
```

```
}


void printInorder()
{
printInorder(root);
}


void inorder()
{
inorderRec(root);
}

void inorderRec(Node root)
{
if (root != null)
{
inorderRec(root.left);
System.out.println(root.key);
inorderRec(root.right);
}
}


public static void main(String[] args) throws Exception
{

input = new Scanner(System.in);
library_management tree = new library_management();
HashMap<String, Integer> hashmapping = new HashMap<>();
SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
Calendar cal = Calendar.getInstance();
Student[] array =new Student[3];
//Add student Details
array[0]=new Student("Harshith",197022,"B.Tech-ECE");
array[1]=new Student("Sravan",197049,"B.Tech-ECE");
array[2]=new Student("Neeraj",197259,"B.Tech-CSE");
int [][] arr=new int[100][2];

//Create file to store data of students.
  FileWriter fr = new FileWriter("append.txt", true);
BufferedWriter br = new BufferedWriter(fr);

FileWriter fr1 = new FileWriter("append.txt", true);
BufferedWriter br1 = new BufferedWriter(fr1);

FileWriter fr2 = new FileWriter("append.txt", true);
BufferedWriter br2 = new BufferedWriter(fr2);

FileWriter fr3 = new FileWriter("append.txt", true);
BufferedWriter br3 = new BufferedWriter(fr3);
```

```
FileReader file = new FileReader("x.txt");
BufferedReader reader = new BufferedReader(file);

FileReader file2= new FileReader("y.txt");
BufferedReader reader2 = new BufferedReader(file2);

FileReader file3= new FileReader("z.txt");
BufferedReader reader3 = new BufferedReader(file3);



Date Rday1 = null,Rday2 = null,Cday=null;
boolean e1=false;


int i=0;

while(e1==false)
{
System.out.println("\n....................................." );
System.out.println("1. Librarian Login. ");
System.out.println("2. User Login. ");
System.out.println("3. Exit. ");
System.out.println("\n....................................." );

System.out.println("\nEnter Your choice:");
int ch1 = input.nextInt();

switch(ch1)
{
case 1:                    //For Librarian login
String pwd1="abc123";
String id1="dsa@1";

System.out.println("\nEnter UserId:" );
String id2 = input.next();

System.out.println("\nEnter Password:" );
String pwd2=input.next();

if(!id1.equals(id2))
System.out.println("Invalid Userid.");
else if(!pwd1.equals(pwd2))
System.out.println("Invalid Password.");
else
{
System.out.println("Login succesfully.");
boolean e2=false;
while(e2==false)
{
System.out.println("\n....................................." );
System.out.println("1. Add book. ");
System.out.println("2. Delete book. ");
System.out.println("3. Update book. ");
System.out.println("4. Print Books Details. ");
System.out.println("5. Print Books in-order. ");
System.out.println("6. Exit");

System.out.println("\n...................................." );

System.out.println("\nEnter Your choice:");
int ch2 = input.nextInt();

switch(ch2)
{
case 1:     //To add a book

String line;
 while((line = reader.readLine()) != null)
{

tree.insert(line);
hashmapping.put(line, i);

       i++;
}
int j=i;

int o = 0;
String number;
while((number = reader2.readLine()) != null)
{
int result = Integer.parseInt(number);
if(j!=o)
    arr[o][0] = result;
o++;
}

int pq=0;
String number1;
while((number1 = reader3.readLine()) != null)
{
int result1 = Integer.parseInt(number1);
if(j!=pq)
    arr[pq][1] = result1;
pq++;
}

             System.out.println("\nEnter name of book:");
String name = input.next();
boolean z1=tree.containsNode(name);

if(z1==true)
{
System.out.println("\nIt is already exists:");
br1.write("Already exist");
 }
else
```

```
{
System.out.println("\nEnter quantity of book:");
int quantity = input.nextInt();
System.out.println(name+"  "+quantity+" ");


br1.write(name);
br2.write(quantity);
br3.write(quantity);

tree.insert(name);
hashmapping.put(name, i);
hashmapping.get(name);
arr[i][0]+=quantity;//Total quantity of books
arr[i][1]+=quantity;//Available quantity of books
        i++;
}
break;

case 2:                                   //To delete a book

System.out.println("\nEnter name of book:");
String b1 = input.next();

boolean x=tree.containsNode(b1);
if(x==true)
{
tree.deleteKey(b1);
hashmapping.remove(b1);
}

break;
case 3:                    //To update any book

System.out.println("\nEnter name of book:");
String b2 = input.next();

boolean z=tree.containsNode(b2);
if(z==true)
{
int a=hashmapping.get(b2);
System.out.println("\nEnter quantity of book to add more:");
int q = input.nextInt();
arr[a][0]+=q;

}
break;

case 4:                    //Print Books Details

Set<Entry<String, Integer>> setOfEntries = hashmapping.entrySet();

for(Entry<String, Integer> entry : setOfEntries)
{
```

```
int r=entry.getValue();
            System.out.println("Name of book is: " + entry.getKey());
            System.out.println("Total Quantity of book is: " + arr[r][0]);
System.out.println("Available Quantity of book is: " + arr[r][1]);
System.out.println();
        }

break;

case 5:     //To Print Books in-order
   tree.printInorder();
break;



case 6:
e2=true;
break;

}
}
}
break;

case 2:                     //For User Login

boolean e3=false;
while(e3==false)
{
System.out.println("\n...................................." );
System.out.println("1. Issue book. ");
System.out.println("2. Return book. ");
System.out.println("3. Exit");
System.out.println("\n...................................." );

System.out.println("\nEnter Your choice:");
int ch3 = input.nextInt();

switch(ch3)
{
case 1://To issue a book
int index = -1;
System.out.println("\nEnter your id:");
int id = input.nextInt();

//display(array);
 for(int k=0;k<3;k++)
{
if(array[k].id_no==id)
index=k;

}
if(index!=-1)
{
```

```
if(array[index].book_no==2)
{
System.out.println("\nYou can't issue more than two books.");
}
else
{
System.out.println("\nEnter name of book:");
String book = input.next();
boolean x=tree.containsNode(book);
if(x==true)
{
int a=hashmapping.get(book);
if(arr[a][1]>0)
{
if(array[index].book1==null)
array[index].book1=book;
else
array[index].book2=book;
System.out.println("Book issued successfully.");
arr[a][1]--;
Cday=cal.getTime();
System.out.println("Currunt Date Time : " + formatter.format(cal.getTime()));
cal.add(Calendar.DATE, 5);
Rday1 = cal.getTime();
System.out.println("Due Date Time: " + formatter.format(Rday1));
array[index].book_no++;

br.write("\nStudent name:       " + array[index].name);
br.write("\nStudent ID  :       " + array[index].id_no);
br.write("\nIssued Book :       " + book);
br.write("\nIssued date :       " + formatter.format(Cday));
br.write("\nReturn date :       " + formatter.format(Rday1));

}
else
System.out.println("You can not issue book now.\nTry again after some days");
}
else
System.out.println("Book is not available.");
}

}
else
System.out.println("Invalid ID");
break;

case 2://to return a book

try
{
int ind = -1;
System.out.println("\nEnter your id:");
int s_id = input.nextInt();
System.out.println("\nEnter name of book:");
```

```
String Rbook = input.next();
for(int k=0;k<3;k++)
 {
if(array[k].id_no==s_id && (array[k].book1.equalsIgnoreCase(Rbook)==true || array[k].book2.equalsIgnoreCase(Rbook)==true))
ind=k;

 }

if(ind!=-1)
{

boolean y=tree.containsNode(Rbook);

if(y==true)
{

if(array[ind].book1.equalsIgnoreCase(Rbook)==true)
array[ind].book1=null;
else
array[ind].book2=null;

cal = Calendar.getInstance();
Rday2=cal.getTime();
//System.out.println(Rday2 + "&"+ Rday1);

if(Rday2.after(Rday1))
{
                System.out.println("Book is overdue.");
                long diff=Rday2.getTime()-Rday1.getTime();
                int noofdays=(int)(diff/(2000/**24*60*60*/));
                System.out.println("Due Date Time: " + formatter.format(Rday2));
                System.out.println("book is delayed by " + noofdays + "seconds." + diff);
                double charge =noofdays*5;
                System.out.println("Your charge is: " + charge + "Rs." );
            }
else
{
System.out.println("Bookireturnedsuccessfully.");
}


int a=hashmapping.get(Rbook);
arr[a][1]++;
array[ind].book_no--;
}
}

else
System.out.println("Invalid ID");
}
catch(Exception e)
{
System.out.println("Something is going to be wrong.");
}
```

```
         break;

        case 3:
        e3=true;

        break;
        }
        }
        break;

        case 3:
        e1=true;

        break;
        }

        }
        br.close();
        fr.close();
        br1.close();
        fr1.close();
        br2.close();
        fr2.close();
        br3.close();
        fr3.close();
        reader.close();
        reader2.close();
        reader3.close();

        }

}
```

Conclusion

We are certain that the issues with the current system will be resolved after we have finished the project. To decrease human error and boost efficiency, the "LIBRARY MANAGEMENT SYSTEM" procedure was digitized. This project's primary goal is to reduce human effort. Because all of the records are kept in the access database which makes it simple to access data, the maintenance of the records is efficient. All forms have a navigation button that may be used to go through a lot of records. If there are a lot of records, the user only has to fill in the search term to obtain the results right away. Additionally, the editing is simplified.

To update the desired field, the user only needs to enter text in the appropriate field and click the update button. A specific, unique ID number is supplied to the books and students. so that they may be appropriately and error-free accessed. The primary goal of the project is to gather accurate data on a specific student and the books that are accessible in the library. The issues that plagued the previous system have been mostly resolved. Additionally, it is anticipated that this initiative will go a long way toward meeting consumer needs. The computerization of library management will boost productivity while also lowering stress levels among staff members, indirectly enhancing human resources.

References

1. https://d1wqtxts1xzle7.cloudfront.net/60558025/Journal_2018-ijca-91670720190911-90087-18bkqru-with-cover-page-v2.pdf?Expires=1659454852&Signature=HO7nWxXS1mG8N1eq9oNSeFkhZ1Zr0HuTilCMMitfrHXFSX1HyI0B5fWug4KQQoyqVZaHp84h01NIMfn3uJTdYpxaa~03DVKXG~Fx9FEy5T66UIIF0rv5wAII0S1kRUqGSXRwaKvONOxRgI38yiTWFETj5bPnhqt4Ri3XEYfx2jCzPq8LSOy0OeyaSq-esxhlhr74YU88YesXSHu-NWuxtA5HvQyl~aFJN~uB6KVDTiDmXDl89OceBpaKTLfQbus1RwkAteN-IerRqXIThFo7WDKlop6TFFB8-ZDdjvgaoKFAWYWmHw1RNNoHCQ-JUflGbifG2aTPk2r5P4hxu5ClCQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

2. https://www.nios.ac.in/media/documents/SrSecLibrary/LCh-015A.pdf

3. https://d1wqtxts1xzle7.cloudfront.net/83737447/Library_Management_System_LMS_Using_JAVA-with-cover-page-v2.pdf?Expires=1659460529&Signature=ZMKhMpk~0tgYcbTxA5~qlxeBtEjHZVndyoXaGGQsLe0WQHlm-IYKpZ3JjHzrcM9VAiwy1jjq9WbhhTU2i8ICiseJ7X~E8wU1Y9WY1YeB~YfWPu~BpOVx7dID6URChh9GSo5jOyjvXrQSwMaxQI110oYmcDFCf~iYi7NYM7i4JjQwmBWSbrRwSheY3wAVuc~zxPw8T7Czw9Nvc6bbK3Iy0CIgBLgDpHqYAunlCU7ogIXwI59VcM8mZlAf0Ovh0ERl5SYt2gu6B4UTneWt3u5SKArptsrO9AX5-iEGT-V6q7y1R13nObX3ZxHvFXO1ylGhAOyOFLGd~hYL~VglpcwwGQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

4. https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/

Teamwork and Work Management

S.No	Name	Work done
1	**M.Harshith**	Binary tree & Hashmap implementation Buffer reader and writer Project Report
2	**M.Sravan Kumar**	Student class User login code and cases Librarian login code and cases