Searching catalogues in library using binary search tree

➢ Team members:

M.Nithin Sai-19BCD7099

K.Saranya-19BCN7187

● Introduction :

Our main objective in this project is to create a library management system wherein students can issue books and the admin or librarian can update/delete the record of books kept in the library.
So we have the system into two parts : from user's perspective and from admin's perspective.

First of all, the admin must login to handle the accounts where the username and password are already set. After he has logged in successfully, he can add, delete and update the books. He can add any new book in the already existing list of books. Similarly he can also delete any existing book. In the update option, the admin can update the quantity of books as well as the name of the book. As and when the admin adds the books, a binary search tree will be created where the nodes contain the name of books and are put in sorted order.

Now if a student wants to issue/return any book, then he/she must login into the system, by enetering their valid university ID. The student will be allowed to issue/return only if their ID matches the list of university ID's of students. When the student enters the name of book to be issued, that particular book will be searched by it's name, in the already created binary search tree. If the book is not found in the tree, then a message will be printed "Book is not available in the library". And if the book is out of stock, then this message will be printed, "This book is currently unavailable. Please try after some days." Moreover, the student cannot issue more than 2 books simultaneously. When the student issues a book, the issuing date and time is recorded by the librarian. And if the student misses the due date of returning the book, then he has to pay that particular fine.

● 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 Linked List.

➢ **Doubly linked list**

2) Hash-map:

Generally Hash-map, maps keys to values such that no duplicate keys are generated. So, in our project, unique keys are generated using hash map and it is assigned to every book such that using this keys we can store and fetch records of books in 2-D array.

3) 2-D array

Here, we have used 2-D array to store the record of each book where rows indicate unique value for particular book which hash-map returns and columns indicate total quantity of books and availbale quantity of books.

• Operations to be perform on each data structure:

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:

We use deletion operation to delete the node of that particular

book from the binary search tree. After the node is deleted, the remaining elements are again rearranged in the tree.

Updation:

The librarian can upate the quantity of already existing set of books which is stored in the 2D array.

List or print all the values:

There is also an option where all the details of the books are printed. Name of the book, available quantity of the books that can be issued and total number of books that library has.

Print book in-order:

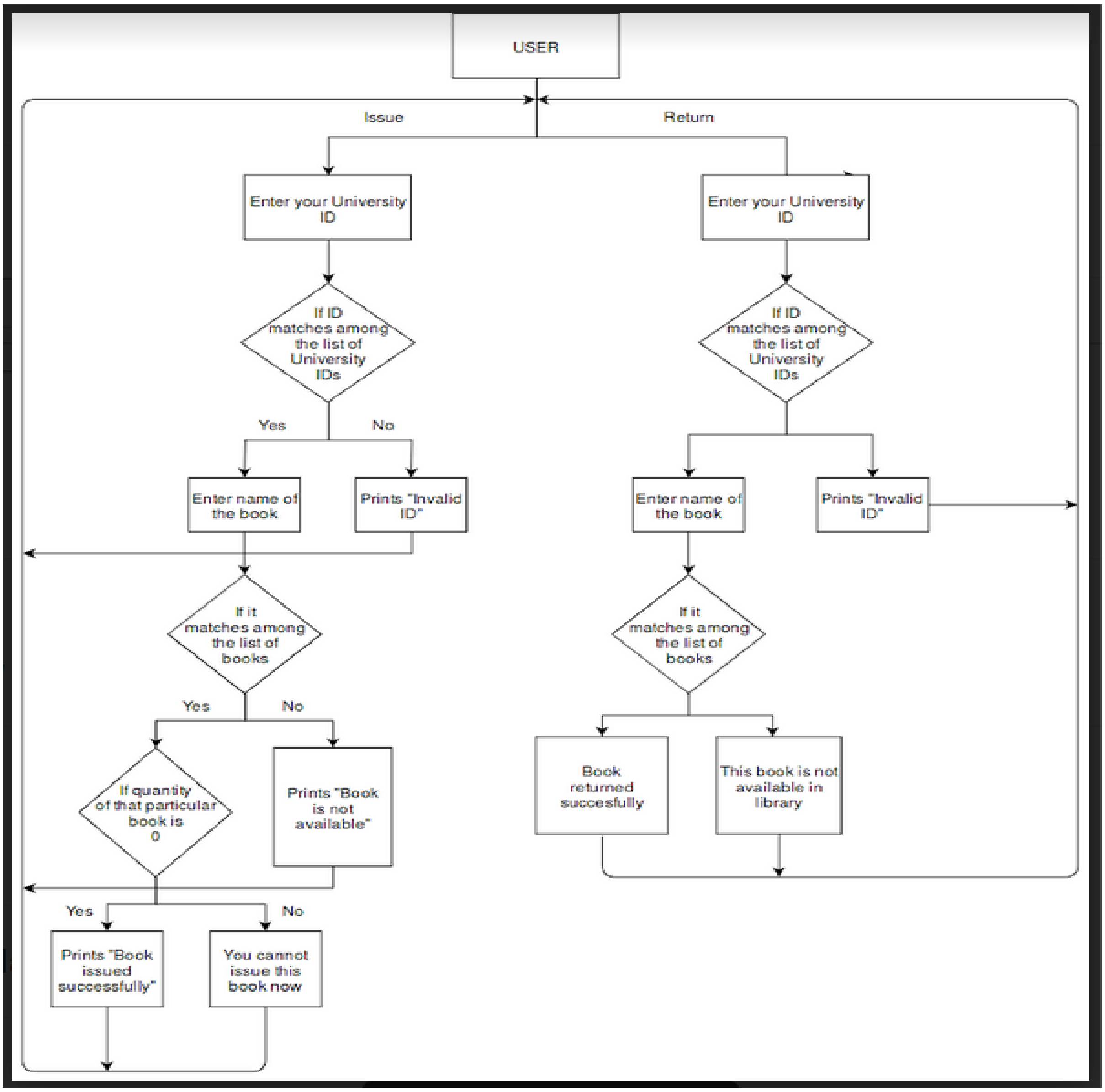
Prints the contents of the binary search tree i.e. the names of the books in ascending order as the tree is balanced.

Requirements of processing on data structures

Keeping elements in a specific order (ascending order) in Binary Search tree is balanced.

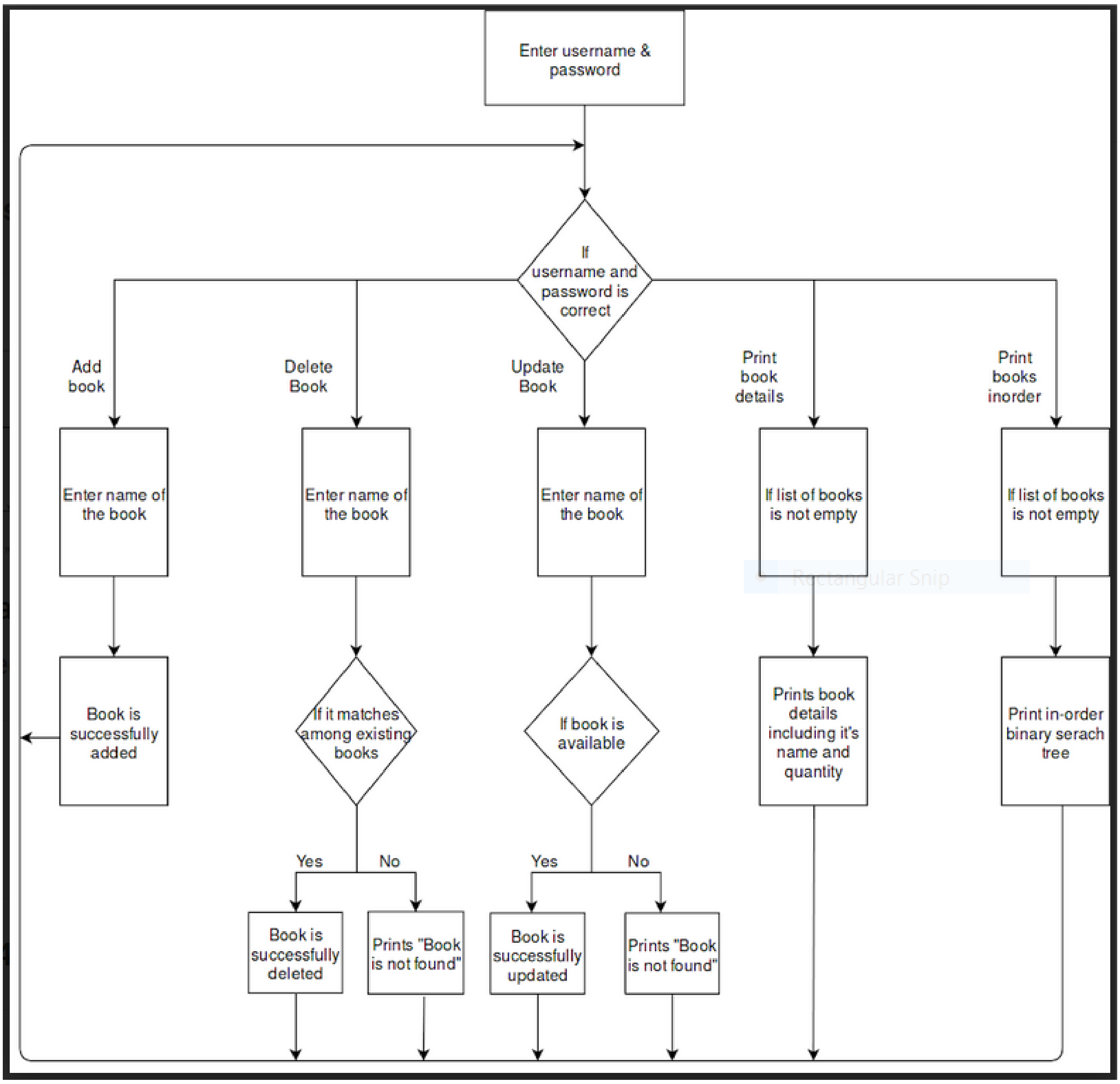
● Flowcharts

1) User :



USER
Issue
Return
Enter your University ID
If ID matches among the list of University IDs
Yes
No
Enter name of the book
Prints "Invalid ID"
If it matches among the list of books
Yes
No
If quantity of that particular book is 0
Prints "Book is not available"
Yes
No
Prints "Book issued successfully"
You cannot issue this book now
Enter your University ID
If ID matches among the list of University IDs
Enter name of the book
Prints "Invalid ID"
If it matches among the list of books
Book returned succesfully
This book is not available in library

2) Librarian/ admin :


Enter username & password
If username and password is correct
Add book
Delete Book
Update Book
Print book details
Print books inorder
Enter name of the book
Enter name of the book
Enter name of the book
If list of books is not empty
If list of books is not empty
Book is successfully added
If it matches among existing books
If book is available
Prints book details including it's name and quantity
Print in-order binary serach tree
Yes
No
Yes
No
Book is successfully deleted
Prints "Book is not found"
Book is successfully updated
Prints "Book is not found"

• List of programs

Filename:finaldsa.java

Source codes:

Insertion:

```
void insert(String key) {

root = insertRec(root, key); }

Node insertRec(Node root, String key) {
```

```
if(root == null) {

root = new Node(key);

return root; }

//If book name < root then place it as left child

if (key.compareTo(root.key)<0) root.left = insertRec(root.left, key);

//If book name > root then place it as Right child else if (key.compareTo(root.key)>0)

root.right = insertRec(root.right, key); return root;

}
```

Deletion:

```
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; }
```

Updation:

```
void update(String key,String key1) {

deleteKey(key);

insert(key1); }
```

Print tree in 2D:

```
void printTree() {

root = printTreeRec(root, 0); }

Node printTreeRec(Node t , int space) {

if(t == null) return root;

space += 5; printTreeRec(t.right ,space);

System.out.println();

for(int i = 5 ; i < space ; i++)

System.out.print( " ");

System.out.print("[" +t.key+ "]");

return printTreeRec(t.left ,space); }
```

Print in-order:

```
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);

} }
```

Searching the 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); }
```

• Input data and output generated

1) Add book

- **Input data**
  Enter name of the book: Enter quantity of book:
- **Output generated**
  Book added successfully...

2) Delete book

- **Input data**
  Enter name of the book: Enter quantity of book:
- **Output generated**
  Book deleted successfully...

3) Update book

- **Input data**
  Enter name of book:
  Enter quantity of book to add more:
- **Output generated** Successfully updated...

---

4) Print book details
Output generated

All the details of book will be printed

5) Print books in-order

Output generated
The list of all books will be printed in ascending order of their

names as it is balanced binary search tree (Balanced BST).

6) Print tree

Output generated

Balanced binary search tree is printed

7) Isuue book by user

- **Input data**
  Enter your university ID: Enter name of book:
- **Output generated**

  Book issued successfully!! Current date time:
  Due date and time:

8) Return book:

- **Input data**
  Enter your university ID: Enter name of book:
- **Output generated**

  Book returned successfully!! Current date time:
  Return date and time:

References:

1. **How to get Key From Value in Hashtable, HashMap or Map in**

   Javahttps://javarevisited.blogspot.com/2013/02/how-to-get-key-from-value-in- hashtable.html#axzz5VJAsUOtP

2. **Number of days between dates (Beginning Java forum at Coderanch)**

   https://coderanch.com/t/543495/java/Number-days-dates

3. **Masking password input from the console : Java - Stack Overflow**

   https://stackoverflow.com/questions/8138411/masking-password-input-from- the-console-java

4. **Binary Search Tree | Set 1 (Search and Insertion) – GeeksforGeeks**

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

5. **How to Get Current Date and Time in Java – Javatpoint**

   https://www.javatpoint.com/java-get-current-date

6. **How to compare dates in Java? - Stack Overflow**

   https://stackoverflow.com/questions/2592501/how-to-compare-dates-in-java

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;
library_management()
{
    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);

      }


      //print tree in 2D


      void printTree()

  {

    root = printTreeRec(root, 0);

  }


      Node printTreeRec(Node t , int space)

  {

    if(t == null)
```

```
        return root;

    space += 5;

    printTreeRec(t.right ,space);

    System.out.println();

    for(int i = 5 ; i < space ; i++)
        System.out.print( " ");
    System.out.print("[" +t.key+ "]");

    return printTreeRec(t.left ,space);
}



        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[2];

                //Add student Details

                array[0]=new Student("Nithin",197099,"B.Tech-DA");

                array[1]=new Student("SARANYA",197187,"B.Tech-NS");


                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="abc123";


        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. Print tree ");
                        System.out.println("7. 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:");
                                        }
                                        else
                                        {

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

    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://To print binary search
tree
                                tree.printTree();
                            break;

                            case 7:
                                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<2;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.SECOND, 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;
```

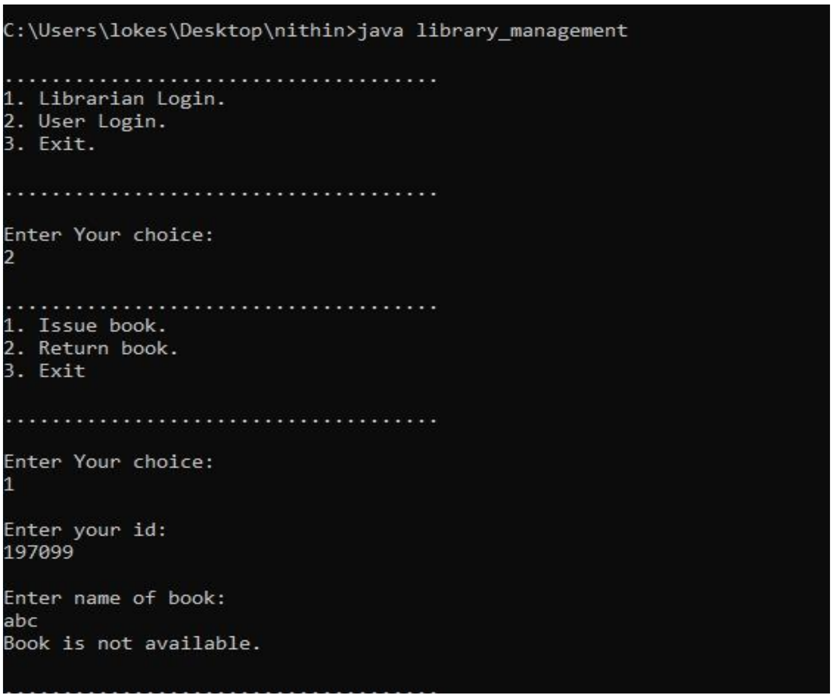
```
                                                                cal =
Calendar.getInstance();
                                                                Rday2=cal.getTime();

        //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("Book is returned successfully.");
                                                                }
```

```
                                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;
```

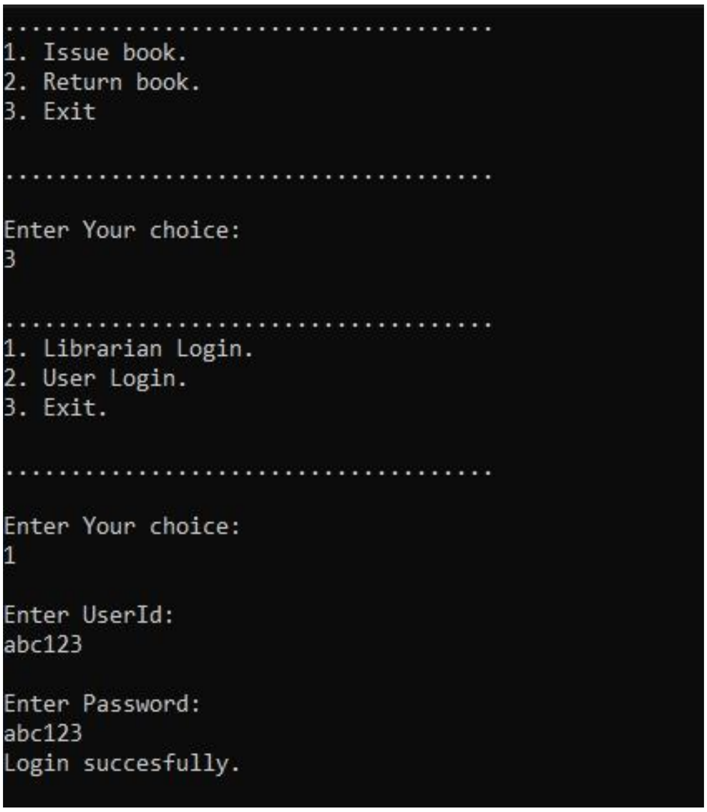
```
                }

                }

        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();
```

```
        }

}
```

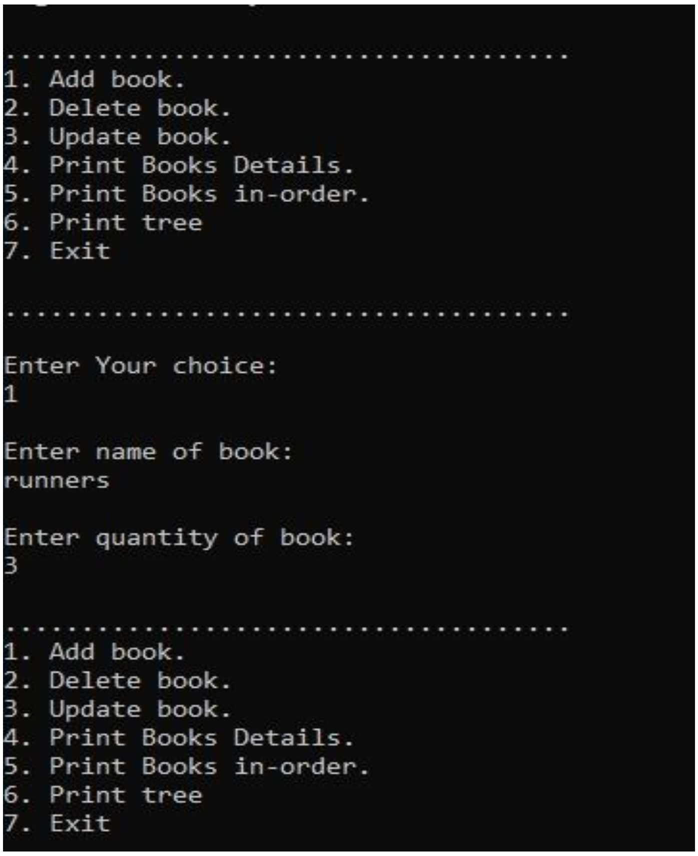
Output :



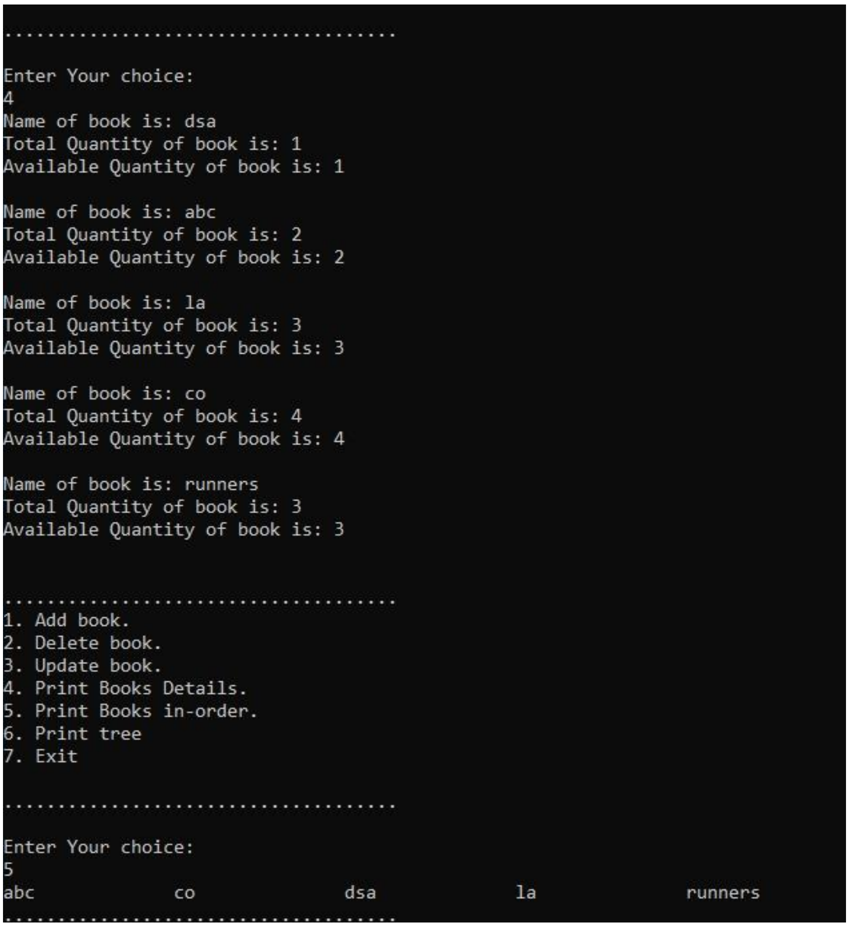
```
C:\Users\lokes\Desktop\nithin>java library_management

.....................................
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:
197099

Enter name of book:
abc
Book is not available.

.....................................
```



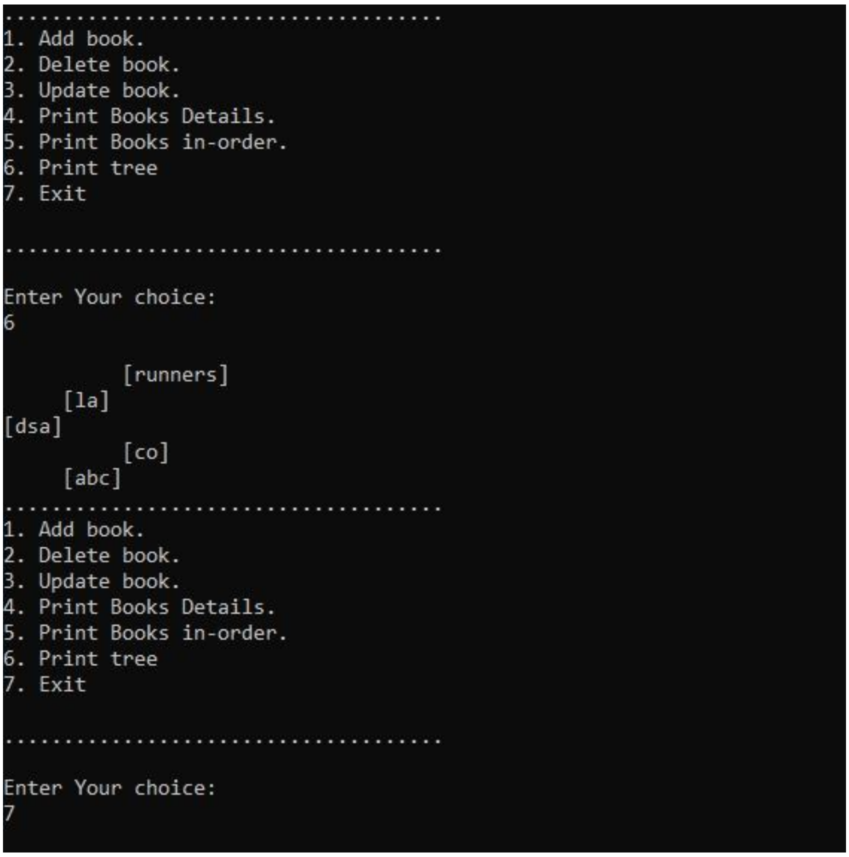
```
....................................
1. Issue book.
2. Return book.
3. Exit

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

Enter Your choice:
3

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

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

Enter Your choice:
1

Enter UserId:
abc123

Enter Password:
abc123
Login succesfully.
```



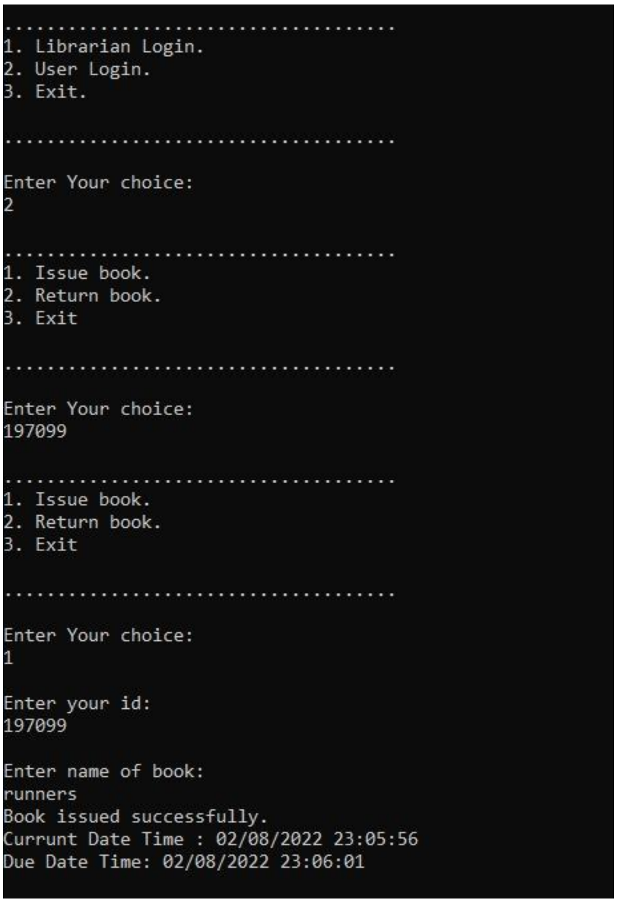
```
.....................................
1. Add book.
2. Delete book.
3. Update book.
4. Print Books Details.
5. Print Books in-order.
6. Print tree
7. Exit

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

Enter Your choice:
1

Enter name of book:
runners

Enter quantity of book:
3

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



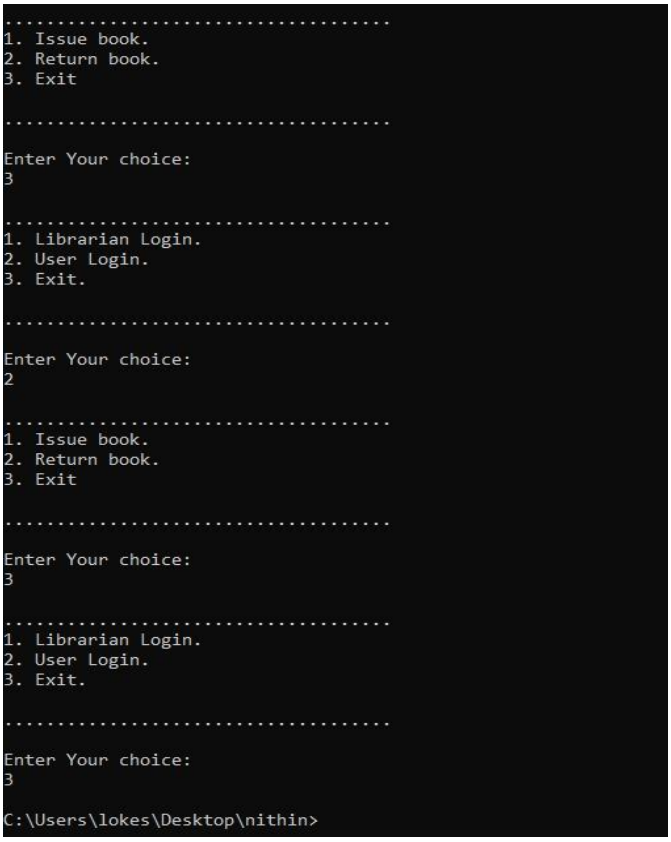
```
.....................................

Enter Your choice:
4
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

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


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

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

Enter Your choice:
5
abc             co              dsa             la              runners
.....................................
```



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

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

Enter Your choice:
6

          [runners]
     [la]
[dsa]
          [co]
     [abc]
.....................................
1. Add book.
2. Delete book.
3. Update book.
4. Print Books Details.
5. Print Books in-order.
6. Print tree
7. Exit

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

Enter Your choice:
7
```



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

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

Enter Your choice:
2

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

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

Enter Your choice:
197099

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

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

Enter Your choice:
1

Enter your id:
197099

Enter name of book:
runners
Book issued successfully.
Currunt Date Time : 02/08/2022 23:05:56
Due Date Time: 02/08/2022 23:06:01
```



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

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

Enter Your choice:
3

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

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

Enter Your choice:
2

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

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

Enter Your choice:
3

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

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

Enter Your choice:
3

C:\Users\lokes\Desktop\nithin>
```