SREE VIDYANIKETHAN ENGINEERING COLLEGE

Sree Sainath Nagar, A. Rangampet – 517 102

**Department of Computer Science and Engineering**

Lab Manual, II B. Tech. I Semester(2021-22)

**(20BT30531) DATA STRUCTURES LAB**

(Common to CSE, CSE(AI), CSE(DS), CSE (AI-ML) and IT)

**COURSE DESCRIPTION:** Hands on practice on implementation of Linked lists; Arrays; Stacks; Queues; Search algorithms; Sorting algorithms; Binary search tree representation and operations; Graph representation and operations; Hashing functions.

**COURSE OBJECTIVES:**

* To demonstrate practical knowledge on Java Programming constructs.
* To develop solutions for societal problems using Java Programming.

**COURSE OUTCOMES:** After successful completion of this course, the students will be able

to:

**CO1.** Implement linear data structures such as arrays, linked lists, stacks, queues for

efficient data organization and manipulation.

**CO2.** Develop solutions using data structures such as trees, graphs, heaps, hash tables

for efficient search and retrieval of data.

**CO3.** Select and apply appropriate techniques for searching and sorting problems.

**CO4.** Work independently and communicate effectively in oral and written forms.

**LIST OF EXPERIMENTS:**

1. A college has N number of students and the following details of all the studentsare maintained register number, name, branch, phone number. Write a program to store the details of the students using a singly linked list. Develop functions to perform the following operations on thedata.
   1. Insert new student’s details
   2. Display the details of thestudents
   3. Display the total number ofstudents
   4. Delete a given student’s information
2. Department of CSE has readers club named ‘Aalochana’. Students can be granted membership in readers club on their request. Similarly, one may cancel their membership of the club. Members of the club can rent books from the club. Write a program to create data structure to maintain readers club members information (Hall ticket number, name) using singly linked list. In singly linked list, the header node should store details of head of readers club and last node should store details of in- charge of readers club. Develop functions to perform the following operations on the data.
3. Store details of head and in-charge of the readersclub
4. Grant and cancel memberships ofstudents
5. Display total number ofmembers
6. Display the details of themembers
7. Displaythesortedlistofdetailsofthemembers(sortbasedontheirnames in alphabeticalorder)
8. A company has N number of employees and it maintains the following details of each of its employees: ID, department, salary, phone number. Develop a menu driven program using doubly linked list to store the employees data. Develop functions to perform the following operations on the data.
9. Add and deleteemployees
10. Display total number ofemployees
11. Display details of employees with salary more than Rs.50,000
12. Display the phone number of the employee given theID
13. a)Develop a menu driven program to perform the following operations on a stack of integers (Array and linked list implementations of stack with maximum size MAX)
14. Push anelement
15. Pop anelement
16. Display thestatus
17. Demonstrate overflow and underflow situations (in array implementation)
18. Writeaprogramtocheckwhetherastringispalindromeornotusingstackdata structure.
19. Mostly syntax errors in a computer program arise due to unbalanced braces (such as (), {}, []). Write a program using stack to check whether a given expression has balanced braces ornot.
20. a)Developamenudrivenprogramtoperformthefollowingoperationsonaqueue of characters (Array and linked list implementations of queue with maximum sizeMAX)
21. Insert anelement
22. Delete anelement
23. Display thestatus
24. Demonstrate overflow and underflow situations (in array implementation

b) A restaurant based on its human resources can accept a maximum of N number of food orders. The food orders are served in first come first serve basis. The food orders once placed cannot be cancelled. Write a program to simulate the food ordering and serving system in the restaurant using circular queue.

1. Write a program to perform the following operations on the binary searchtree.
2. Construct binary search tree by inserting the values {6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2} in the givenorder.
3. Displaythenodesofthetreeusinginorder,preorderandpostorder traversaltechniques.
4. Display the smallest number stored in thetree.
5. Search the tree for a givennumber.
6. There are train paths between cities. If there is a train between city A and city B thenthereisaroutebetweenthecities.Thecostoftherouteisthedistancebetween cityAandcityB.Representthetraintravelrouteinformationasagraph.Thenode canberepresentedbythenameofthecity.Writeaprogramtoperformthefollowing operations.
7. Store the details of train travel route information using adjacency list or adjacency matrixrepresentation.
8. Traversethegraphanddisplaythedetailsofalltrainsbetweenthecities along with the cost using breadth-firstmethod.
9. Traversethegraphanddisplaythedetailsofalltrainsbetweenthecities along with the cost using depth-firstmethod.
10. Store register numbers of students who attended placement training program in a randomorderinanarray.Writeafunctiontosearchwhetherastudenthasattended placement training program or notusing
11. Linearsearch
12. Binarysearch
13. Write a program to sort a given set of integersusing
14. Quicksort
15. Shellsort
16. a) Write a program to sort a given set of integers using mergesort.

b) Write a program to read the marks obtained by students in a mathematics examination and store the data using a heap data structure. Find out the maximum and minimum marks obtained by the students.

1. Write a program to implement the following hashingfunctions.
2. Separate ChainingMethod
3. Open AddressingMethod
4. Consider an online movie ticket booking system throughwhich customers can book tickets to watch movies at theatres. The database stores the details of each transaction of ticket booking with the details - ID, customer name, customer phone number, movie name, theatre name, date of show, time of show, number of tickets booked, starting seat number, total amount. Write a menu driven program to perform create the database and given an ID, display a client’s phone number. Use a hash table implementation to quickly search through the database.

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)**  **(20BT30531) DATA STRUCTURES LAB** |
| Exp. No: 1 |

1. A college has N number of students and the following details of all the studentsare maintained register number, name, branch, phone number. Write a program to store the details of the students using a singly linked list. Develop functions to perform the following operations on thedata.
   1. Insert new student’s details
   2. Display the details of thestudents
   3. Display the total number ofstudents
   4. Delete a given student’s information

**PROGRAM**  linkList2.java (Classes Link, LinkList, LinkList2App with Main)

// linkList2.java

// demonstrates linked list

// to run this program: C>java LinkList2App

////////////////////////////////////////////////////////////////

class Link

{

public int Reg\_No;

// data item (key)

public String Name;

public String Branch;

public long Phone\_no; // data item

public Link next; // next link in list

// -------------------------------------------------------------

public Link(final int Reg\_No, final String Name, final String Branch, final long Phone\_no) // constructor

{

this.Reg\_No = Reg\_No;

this.Name = Name;

this.Branch = Branch;

this.Phone\_no = Phone\_no;

}

// -------------------------------------------------------------

public void displayLink() // display ourself

{

System.out.print("{" + this.Reg\_No + ", " + this.Name + ", " + this.Branch + ",, " + this.Phone\_no + "} ");

}

} // end class Link

////////////////////////////////////////////////////////////////

class LinkList

{

private Link first; // ref to first link on list

// -------------------------------------------------------------

public LinkList() // constructor

{

first = null; // no links on list yet

}

// -------------------------------------------------------------

public void insertFirst(final int Reg\_No, final String Name, final String Branch, final long Phone\_no)

{ // make new link

final Link first = new Link(Reg\_No,Name,Branch,Phone\_no);

first.next = this.first; // it points to old first link

this.first = first; // now first points to this

}

// -------------------------------------------------------------

public Link find(int key) // find link with given key

{ // (assumes non-empty list)

Link current = first; // start at 'first'

while(current.Reg\_No != key) // while no match,

{

if(current.next == null) // if end of list,

return null; // didn't find it

else // not end of list,

current = current.next; // go to next link

}

return current; // found it

}

// -------------------------------------------------------------

public Link delete(int key) // delete link with given key

{ // (assumes non-empty list)

Link current = first; // search for link

Link previous = first;

while(current.Reg\_No != key)

{

if(current.next == null)

return null; // didn't find it

else

{

previous = current; // go to next link

current = current.next;

}

} // found it

if(current == first) // if first link,

first = first.next; // change first

else // otherwise,

previous.next = current.next; // bypass it

return current;

}

// -------------------------------------------------------------

public void displayList() // display the list

{

System.out.print("List (first-->last): ");

Link current = first; // start at beginning of list

while(current != null) // until end of list,

{

current.displayLink(); // print data

current = current.next; // move to next link

}

System.out.println("");

}

// -------------------------------------------------------------

} // end class LinkList

////////////////////////////////////////////////////////////////

class LinkList2App

{

public static void main(String[] args)

{

LinkList theList = new LinkList(); // make list

theList.insertFirst(1001,"Ganapathy","CSE",9876722222L); // insert 4 items

theList.insertFirst(1002,"Balaji","CSE",9874422222L);

theList.insertFirst(1003,"Shiva","CSE-AI",9876822222L);

theList.insertFirst(1004,"Subramaniya","ECE",9876733333L);

theList.displayList(); // display list

Link f = theList.find(1001); // find item

if( f != null)

System.out.println("Found link with key " + f.Name);

else

System.out.println("Can't find link");

Link d = theList.delete(1001); // delete item

if( d != null )

System.out.println("Deleted link with key " + d.Name);

else

System.out.println("Can't delete link");

theList.displayList(); // display list

} // end main()

} // end class LinkList2App

////////////////////////////////////////////////////////////////

**OUTPUT**

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>javac LinkList2.java

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java LinkList2App

List (first-->last): {1004, Subramaniya, ECE,, 9876733333} {1003, Shiva, CSE-AI,, 9876822222} {1002, Balaji, CSE,, 9874422222} {1001, Ganapathy, CSE,, 9876722222}

Found link with key Ganapathy

Deleted link with key Ganapathy

List (first-->last): {1004, Subramaniya, ECE,, 9876733333} {1003, Shiva, CSE-AI,, 9876822222} {1002, Balaji, CSE,, 9874422222}

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)**  **(20BT30531) DATA STRUCTURES LAB** |
| Exp. No: 2 |

1. Department of CSE has readers club named ‘Aalochana’. Students can be granted membership in readers club on their request. Similarly, one may cancel their membership of the club. Members of the club can rent books from the club. Write a program to create data structure to maintain readers club members information (Hall ticket number, name) using singly linked list. In singly linked list, the header node should store details of head of readers club and last node should store details of in- charge of readers club. Develop functions to perform the following operations on the data.
2. Store details of head and in-charge of the readersclub
3. Grant and cancel memberships ofstudents
4. Display total number ofmembers
5. Display the details of themembers
6. Displaythesortedlistofdetailsofthemembers(sortbasedontheirnames in alphabeticalorder)

**PROGRAM**  Node.java

import java.util.Scanner;

public class Node {

int prn;

String name;

Node next;

private static class panclub{

int num,cnt;

String nm;

Node head;

public panclub() {

num=cnt=0;

head=null;

}

public Node create() {

Scanner s=new Scanner(System.in);

Node temp,n1;

temp=n1=null;

System.out.println("\nHow many students data u want to insert in panclub database");

cnt=s.nextInt();

do {

n1=new Node();

System.out.println("\nEnter PRN number of student");

num=s.nextInt();

n1.prn=num;

System.out.println("\nEnter name of student");

nm=s.next();

n1.name=nm;

n1.next=null;

if(head==null) {

head=n1;

temp=head;

}

else {

temp=head;

while(temp.next!=null) {

temp=temp.next;

}

temp.next=n1;

}

cnt--;

}while(cnt>0);

return head;

}

public void display(Node head) {

Node temp;

temp=head;

while(temp!=null) {

if(temp.next==null) {

System.out.print("["+temp.prn+"|"+temp.name+"]->null");

}

else {

System.out.print("["+temp.prn+"|"+temp.name+"]->");

}

temp=temp.next;

}

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

}

public Node concat(Node head1,Node head2) {

Node head3,temp,temp1;

head3=temp=temp1=null;

temp=head1;

head3=head1;

while(temp.next!=null) {

temp=temp.next;

}

temp1=head2;

temp.next=temp1;

return head3;

}

public void reverse(Node head) {

Node temp;

temp=head;

if(temp==null)

return;

reverse(temp.next);

System.out.print("["+temp.prn+"|"+temp.name+"]->");

}

public Node insert\_president(Node head) {

Scanner s=new Scanner(System.in);

Node temp,n2;

temp=n2=null;

temp=head;

n2=new Node();

System.out.println("\nEnter PRN number of president");

n2.prn=s.nextInt();

System.out.println("\nEnter the name of president");

n2.name=s.next();

n2.next=temp;

head=n2;

return head;

}

public void insert\_member(Node head) {

Scanner s=new Scanner(System.in);

Node temp,n2;

int pn;

temp=head;

n2=new Node();

System.out.println("\nEnter the PRN number of Member");

n2.prn=s.nextInt();

System.out.println("\nEnter the name of the Member");

n2.name=s.next();

n2.next=null;

System.out.println("\nEnter the PRN number after which u want to add this member");

pn=s.nextInt();

while(temp!=null) {

if(temp.prn==pn) {

n2.next=temp.next;

temp.next=n2;

break;

}

temp=temp.next;

}

System.out.println("\nMember added successfully");

}

public void insert\_sec(Node head) {

Scanner s=new Scanner(System.in);

Node temp,n2;

temp=head;

n2=new Node();

System.out.println("\nEnter the PRN number of secretary");

n2.prn=s.nextInt();

System.out.println("\nEnter the name of secretary");

n2.name=s.next();

n2.next=null;

while(temp.next!=null) {

temp=temp.next;

}

temp.next=n2;

}

//Delete the president node from list

public Node del\_president(Node head) {

Node temp;

temp=head;

head=temp.next;

return head;

}

//Delete secretary

public Node del\_secretary(Node head) {

Node temp,t1 = null;

temp=head;

while(temp.next!=null) {

t1=temp;

temp=temp.next;

}

t1.next=null;

return head;

}

public Node del\_member(Node head) {

Scanner s=new Scanner(System.in);

Node temp,t1;

int pn;

temp=head;

System.out.println("\nEnter PRN number of member");

pn=s.nextInt();

while(temp!=null) {

if(temp.prn==pn) {

t1=temp.next;

temp.next=t1.next;

break;

}

temp=temp.next;

}

System.out.println("\nMember removed successfully");

return head;

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner s=new Scanner(System.in);

Node h1=new Node();

Node h2=new Node();

Node h3=new Node();

panclub p1=new panclub();

panclub p2=new panclub();

panclub p3=new panclub();

h1=h2=h3=null;

int ch;

System.out.println("!!!Group B:Assignment No:1!!!");

do {

System.out.println("1.Enter data of SE A division:\n2.Enter data of SE B division:");

System.out.println("3.Concatination of list..\n\tEnter your choice");

ch=s.nextInt();

switch(ch) {

case 1:

System.out.println("\nPlease enter the student info who is register memeber..");

System.out.println("\nEnter the Panclub Data of SE A Division:\n");

h1=p1.create();

System.out.println("\nSE Comp Division A List are as follows..\n");

p1.display(h1);

System.out.println("\nReverse List of SE Div A:\n");

p1.reverse(h1);

p1.insert\_sec(h1);

System.out.println("\nAfter insertion of Secretary: \n");

p1.display(h1);

h1=p1.insert\_president(h1);

System.out.println("\nAfter insertion of President: \n");

p1.display(h1);

p1.insert\_member(h1);

System.out.println("\n After insertion of member…\n");

p1.display(h1);

h1=p1.del\_president(h1);

System.out.println("\nAfter deletion of president…\n");

p1.display(h1);

h1=p1.del\_secretary(h1);

System.out.println("\nAfter deletion of secretary…\n");

p1.display(h1);

h1=p1.del\_member(h1);

System.out.println("\nAfter deletion of member…\n");

p1.display(h1);

break;

case 2:

System.out.println("\nEnter the Panclub Data of SE B Division:");

h2=p2.create();

System.out.println("\nSE Comp Division B List are as follows..");

p2.display(h2);

System.out.println("\nReverse List of SE Div B:");

p1.reverse(h2);

p2.insert\_sec(h2);

System.out.println("\nAfter insertion of Secretary: ");

p2.display(h2);

h2=p2.insert\_president(h2);

System.out.println("\nAfter insertion of President: ");

p2.display(h2);

p2.insert\_member(h2);

System.out.println("\nAfter insertion of member…");

p2.display(h2);

h2=p2.del\_president(h2);

System.out.println("\nAfter deletion of president…");

p1.display(h2);

h2=p2.del\_secretary(h2);

System.out.println("\nAfter deletion of secretary…");

p1.display(h2);

h2=p2.del\_member(h2);

System.out.println("\nAfter deletion of member…");

p2.display(h2);

break;

case 3:

h3=p3.concat(h1,h2);

System.out.println("\nThe concatenation of Div : A and Div : B of SE Comp Class are as follows.");

p3.display(h3);

break;

}

}while(ch!=4);

}

}

**OUTPUT**

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java Node

!!!Group B:Assignment No:1!!!

1.Enter data of SE A division:

2.Enter data of SE B division:

3.Concatination of list..

Enter your choice

1

Please enter the student info who is register memeber..

Enter the Panclub Data of SE A Division:

How many students data u want to insert in panclub database

1

Enter PRN number of student

1001

Enter name of student

Sai

SE Comp Division A List are as follows..

[1001|Sai]->null

Reverse List of SE Div A:

[1001|Sai]->

Enter the PRN number of secretary

1001

Enter the name of secretary

Akash

After insertion of Secretary:

[1001|Sai]->[1001|Akash]->null

Enter PRN number of president

1003

Enter the name of president

Kamal

After insertion of President:

[1003|Kamal]->[1001|Sai]->[1001|Akash]->null

Enter the PRN number of Member

1004

Enter the name of the Member

Mani

Enter the PRN number after which u want to add this member

1001

Member added successfully

After insertion of memberâ??

[1003|Kamal]->[1001|Sai]->[1004|Mani]->[1001|Akash]->null

After deletion of presidentâ??

[1001|Sai]->[1004|Mani]->[1001|Akash]->null

After deletion of secretaryâ??

[1001|Sai]->[1004|Mani]->null

Enter PRN number of member

1001

Member removed successfully

After deletion of memberâ??

[1001|Sai]->null

1.Enter data of SE A division:

2.Enter data of SE B division:

3.Concatination of list..

Enter your choice

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 3 |

A company has N number of employees and it maintains the following details of each of its employees: ID, department, salary, phone number. Develop a menu driven program using doubly linked list to store the employees data. Develop functions to perform the following operations on the data.

1. Add and deleteemployees
2. Display total number ofemployees
3. Display details of employees with salary more than Rs.50,000
4. Display the phone number of the employee given theID

**PROGRAM** EmployeeDetails.java

import java.util.\*;

class Employee {

String ID;

String dept;

int salary;

String cont;

public Employee(String ID, String dept, int salary, String cont) {

this.ID = ID;

this.dept= dept;

this.salary= salary;

this.cont= cont;

}

public String toString() {

return "Employee{" +

"ID='" + ID + '\'' +

", dept='" + dept + '\'' +

", salary=" + salary +

", cont='" + cont+ '\'' +

'}';

}

public static boolean displayM50000(Object o){

Employee emp=(Employee)o;

if (emp.salary>50000)

return true;

return false;

}

public boolean equals(Object o){

if (this==o)

return true;

if (o==null || getClass()!=o.getClass())

return false;

Employee emp=(Employee)o;

return salary==emp.salary&&Objects.equals(ID,emp.ID) &&Objects.equals(dept,emp.dept) &&Objects.equals(cont,emp.cont);

}

public static boolean displayPhoneNum(Object o,String ID) {

Employee emp = (Employee) o;

if (emp.ID.equals(ID)) {

System.out.println("The phone number of the employee with ID " + ID + " is " + emp.cont);

return true;

}

return false;

}

}

public class EmployeeDetails{

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

DLinkedList<Employee>employeeList = new DLinkedList<>();

char again = 'y';

while (again == 'y' || again == 'Y') {

System.out.println("Enter your choice:" + '\n' + "1:Add employees " + '\n' + "2:Remove employees"

+ '\n' + "3:Display employee count" + '\n' + "4: Employees with more than 50000 salary" + '\n' +

"5:Phone number with given ID");

int opt = s.nextInt();

switch (opt) {

case 1:

System.out.print("Enter the number of employees to enter details : ");

int N = s.nextInt();

for (int i = 0; i< N; i++) {

System.out.println("Enter the employee ID, Department, Salary and Phone Number : ");

employeeList.add(new Employee(s.next(), s.next(), s.nextInt(), s.next()));

}

break;

case 2:

System.out.println("Enter employee ID to remove details : ");

String ID = s.next();

DLinkedList.Node temp = employeeList.head;

boolean found = false;

for (int i = 0; i<employeeList.size(); i++) {

Employee emp = (Employee) temp.e;

found = Objects.equals(ID, emp.ID);

if (found) {

employeeList.remove(i);

System.out.println("Employee details with ID " + ID + " are removed successfully !");

break;

}

temp = temp.next;

}

if (!found)

System.out.println("Employee with ID " + ID + " is not found !");

break;

case 3:

System.out.println("The total number employees in the company is " + employeeList.size());

break;

case 4:

System.out.println("The Details of the employees with more than 50000 salary : ");

employeeList.display(1);

break;

case 5:

employeeList.display(2);

}

System.out.println('\n' + "Enter 'y' or 'Y' to get services again" + '\n' +

"Or any other key to deny : ");

again = s.next().charAt(0);

}

}

}

class DLinkedList<E> {

Node head;

int count;

class Node {

E e;

Node next;

Node previous;

Node(E e) {

this.e= e;

count++;

}

}

public void add(E e) {

Node adding = new Node(e);

if (head == null) {

head = adding;

return;

}

head.previous= adding;

adding.next= head;

head = adding;

}

public String toString() {

String s = "";

Node temp = head;

while (temp != null) {

s += temp.e+ ", ";

temp = temp.next;

}

String list = "[";

for (int i = 0; i<s.length() - 2; i++)

list += s.charAt(i);

return list + ']';

}

public int size() {

return count;

}

public void display(int opt) {

Node temp = head;

if (opt == 1) {

for (int i = 0; i< size(); i++) {

if (Employee.displayM50000(temp.e))

System.out.println(temp.e);

temp = temp.next;

}

} else if (opt == 2) {

System.out.print("Enter the employee ID to get phone number : ");

String ID = new Scanner(System.in).next();

for (int i = 0; temp!=null; i++) {

if (Employee.displayPhoneNum(temp.e, ID))

return;

temp = temp.next;

}

System.out.println("The employee with ID " + ID + " is not found.");

}

}

public boolean remove(int index) {

try{

if (index>=10 || index<0)

throw new IndexOutOfBoundsException();

}

catch(Exception e){

e.printStackTrace();

return false;

}

Node temp = head;

if (index == 0) {

head = head.next;

System.gc();

head.previous= null;

count--;

return true;

}

if (index == size() - 1) {

for (int i = 0; i< index - 1; i++)

temp = temp.next;

temp.next= null;

count--;

return true;

}

for (int i = 0; i< index - 1; i++)

temp = temp.next;

Node node1 = temp.next;

temp.next= node1.next;

Node node2 = node1.next;

node2.previous = temp;

count--;

return true;

}

public boolean remove(E e) {

Node temp = head;

int index = 0;

for (; temp != null; index++) {

if (e.equals(temp.e)) {

remove(index);

return true;

}

temp = temp.next;

}

return false;

}

}

**OUTPUT**

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>javac EmployeeDetails.java

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java EmployeeDetails.java

Error: Could not find or load main class EmployeeDetails.java

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java EmployeeDetails

Enter your choice:

1:Add employees

2:Remove employees

3:Display employee count

4: Employees with more than 50000 salary

5:Phone number with given ID

1

Enter the number of employees to enter details : 2

Enter the employee ID, Department, Salary and Phone Number :

1001

cse

10000

2345611111

Enter the employee ID, Department, Salary and Phone Number :

1002

ece

20000

2341522222

Enter 'y' or 'Y' to get services again

Or any other key to deny :

y

Enter your choice:

1:Add employees

2:Remove employees

3:Display employee count

4: Employees with more than 50000 salary

5:Phone number with given ID

3

The total number employees in the company is 2

Enter 'y' or 'Y' to get services again

Or any other key to deny :

y

Enter your choice:

1:Add employees

2:Remove employees

3:Display employee count

4: Employees with more than 50000 salary

5:Phone number with given ID

5

Enter the employee ID to get phone number : 1001

The phone number of the employee with ID 1001 is 2345611111

Enter 'y' or 'Y' to get services again

Or any other key to deny :

y

Enter your choice:

1:Add employees

2:Remove employees

3:Display employee count

4: Employees with more than 50000 salary

5:Phone number with given ID

4

The Details of the employees with more than 50000 salary :

Enter 'y' or 'Y' to get services again

Or any other key to deny :

y

Enter your choice:

1:Add employees

2:Remove employees

3:Display employee count

4: Employees with more than 50000 salary

5:Phone number with given ID

1

Enter the number of employees to enter details : 1

Enter the employee ID, Department, Salary and Phone Number :

1005

cse

50001

1234533333

Enter 'y' or 'Y' to get services again

Or any other key to deny :

y

Enter your choice:

1:Add employees

2:Remove employees

3:Display employee count

4: Employees with more than 50000 salary

5:Phone number with given ID

4

The Details of the employees with more than 50000 salary :

Employee{ID='1005', dept='cse', salary=50001, cont='1234533333'}

Enter 'y' or 'Y' to get services again

Or any other key to deny :

n

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 4 |

a)Develop a menu driven program to perform the following operations on a stack of integers (Array and linked list implementations of stack with maximum size MAX)

1. Push anelement
2. Pop anelement
3. Display thestatus
4. Demonstrate overflow and underflow situations (in array implementation)
5. Writeaprogramtocheckwhetherastringispalindromeornotusingstackdata structure.
6. Mostly syntax errors in a computer program arise due to unbalanced braces (such as (), {}, []). Write a program using stack to check whether a given expression has balanced braces ornot.

**PROGRAM** 4a i) ArrayStack.java

import java.util.Scanner;

public class ArrayStack{

int MAX=10;

int stack[]=new int[MAX];

int top=-1;

void push(int data)

{

if(top==MAX-1)

{

System.out.println("stack overflow");

}

else

{

top++;

stack[top]=data;

}

}

void pop()

{

if(top==-1)

{

System.out.println("stack underflow");

}

else

{

System.out.println(stack[top]+" is deleted from the stack");

top--;

}

}

void disp()

{

if(top==-1)

{

System.out.println("stack is empty");

}

else

{

for(int i=top;i>=0;i--)

{

System.out.print(stack[i]+" \t");

}

System.out.println();

}

}

void peek()

{

if(top==-1)

{

System.out.println("stack is empty");

}

else

{

System.out.println(stack[top]+" is the top of the stack");

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

ArrayStack s=new ArrayStack();

Scanner sc=new Scanner(System.in);

System.out.println("1.push\t 2.pop \t3.disp \t4.peek \t5.exit");

int ch;

do

{

System.out.println("enter your choice");

ch=sc.nextInt();

switch(ch)

{

case 1:

System.out.println("enter data");

int data=sc.nextInt();

s.push(data);

break;

case 2:

s.pop();

break;

case 3:

s.disp();

break;

case 4:

s.peek();

break;

}

}while(ch<=4);

}

}

**OUTPUT**

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>javac ArrayStack.java

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java ArrayStack

1.push 2.pop 3.disp 4.peek 5.exit

enter your choice

1

enter data

12

enter your choice

1

enter data

34

enter your choice

1

enter data

56

enter your choice

1

enter data

55

enter your choice

3

55 56 34 12

enter your choice

4

55 is the top of the stack

enter your choice

2

55 is deleted from the stack

enter your choice

3

56 34 12

enter your choice

4

56 is the top of the stack

enter your choice

5

**PROGRAM** 4a ii) LinkStack.java

import java.util.Scanner;

class Element

{

int data;

Element next;

Element(int data)

{

this.data=data;

this.next=null;

}

}

public class LinkStack{

Element top;

void push(int data)

{

Element newelement=new Element(data);

if(top==null)

{

top=newelement;

}

else

{

newelement.next=top;

top=newelement;

}

}

void pop()

{

if(top==null)

{

System.out.println("stack is underflow");

}

else

{

System.out.println(top.data+"is deleted from the stack");

top=top.next;

}

}

void peek()

{

if(top==null)

{

System.out.println("stack is underflow");

}

else

{

System.out.println("top element in the stack is"+top.data);

}

}

void disp()

{

Element t=top;

while(t!=null)

{

System.out.print(t.data+"\t");

t=t.next;

}

System.out.println();

}

public static void main(String[] args) {

// TODO Auto-generated method stub

LinkStack s=new LinkStack();

Scanner sc=new Scanner(System.in);

System.out.println("1.push\t 2.pop \t3.disp \t4.peek \t5.exit");

int ch;

do

{

System.out.println("enter your choice");

ch=sc.nextInt();

switch(ch)

{

case 1:

System.out.println("enter data");

int data=sc.nextInt();

s.push(data);

break;

case 2:

s.pop();

break;

case 3:

s.disp();

break;

case 4:

s.peek();

break;

}

}while(ch<=4);

}

}

**OUTPUT**

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>javac LinkStack.java

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java LinkStack

1.push 2.pop 3.disp 4.peek 5.exit

enter your choice

1

enter data

12

enter your choice

1

enter data

45

enter your choice

1

enter data

67

enter your choice

1

enter data

89

enter your choice

2

89is deleted from the stack

enter your choice

3

67 45 12

enter your choice

1

enter data

78

enter your choice

3

78 67 45 12

enter your choice

2

78is deleted from the stack

enter your choice

3

67 45 12

enter your choice

4

top element in the stack is67

enter your choice

5

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>

b) Write a program to check whether a string is palindrome or not using stack data

structure.

**PROGRAM** 4b Palindromestack.java

import java.util.Stack;

import java.util.Scanner;

class Palindromestack {

public static void main(String[] args) {

System.out.print("Enter any string:");

Scanner in=new Scanner(System.in);

String inputString = in.nextLine();

Stack stack = new Stack();

for (int i = 0; i < inputString.length(); i++) {

stack.push(inputString.charAt(i));

}

String reverseString = "";

while (!stack.isEmpty()) {

reverseString = reverseString+stack.pop();

}

if (inputString.equals(reverseString))

System.out.println("The input String is a palindrome.");

else

System.out.println("The input String is not a palindrome.");

}

}

**OUTPUT**

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>javac Palindromestack.java

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java Palindromestack

Enter any string:SVEC

The input String is not a palindrome.

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java Palindromestack

Enter any string:MALAYALAM

The input String is a palindrome.

c) Mostly syntax errors in a computer program arise due to unbalanced braces

(such as (), {}, []). Write a program using stack to check whether a given

expression has balanced braces or not.

**PROGRAM** 4c

import java.util.\*;

public class Balancingsymbols {

// function to check if brackets are balanced

static boolean aresymbolsBalanced(String expr)

{

// Using ArrayDeque is faster than using Stack class

Deque<Character> stack

= new ArrayDeque<Character>();

// Traversing the Expression

for (int i = 0; i < expr.length(); i++)

{

char x = expr.charAt(i);

if (x == '(' || x == '[' || x == '{')

{

// Push the element in the stack

stack.push(x);

continue;

}

// If current character is not opening

// bracket, then it must be closing. So stack

// cannot be empty at this point.

if (stack.isEmpty())

return false;

char check;

switch (x) {

case ')':

check = stack.pop();

if (check == '{' || check == '[')

return false;

break;

case '}':

check = stack.pop();

if (check == '(' || check == '[')

return false;

break;

case ']':

check = stack.pop();

if (check == '(' || check == '{')

return false;

break;

}

}

// Check Empty Stack

return (stack.isEmpty());

}

// Driver code

public static void main(String[] args)

{

Scanner sc= new Scanner(System.in); //System.in is a standard input stream

System.out.print("Enter a string: ");

String expr= sc.nextLine(); //reads string

System.out.print("You have entered: "+expr);

// Function call

if (aresymbolsBalanced(expr))

System.out.println("Balanced ");

else

System.out.println("Not Balanced ");

}

}

**OUTPUT**

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java Balancingsymbols

Enter a string: [(a+b]

You have entered: [(a+b]Not Balanced

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java Balancingsymbols

Enter a string: {([])}

You have entered: {([])}Balanced

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java Balancingsymbols

Enter a string: [a+b]

You have entered: [a+b]Balanced

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java Balancingsymbols

Enter a string: [a+b/c]

You have entered: [a+b/c]Balanced

C:\Program Files (x86)\Java\jdk1.8.0

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 5 |

a)Developamenudrivenprogramtoperformthefollowingoperationsonaqueue of characters (Array and linked list implementations of queue with maximum sizeMAX)

1. Insert anelement
2. Delete anelement
3. Display thestatus
4. Demonstrate overflow and underflow situations (in array implementation

b) A restaurant based on its human resources can accept a maximum of N number of food orders. The food orders are served in first come first serve basis. The food orders once placed cannot be cancelled. Write a program to simulate the food ordering and serving system in the restaurant using circular queue.

**PROGRAM**5a)

// Circular Queue implementation in Java

import java.util.Scanner;

public class QArray {

int SIZE = 5; // Size of Circular Queue

int front, rear;

int items[] = new int[SIZE];

QArray() {

front = -1;

rear = -1;

}

// Check if the queue is full

boolean isFull() {

if (front == 0 && rear == SIZE - 1) {

return true;

}

if (front == rear + 1) {

return true;

}

return false;

}

// Check if the queue is empty

boolean isEmpty() {

if (front == -1)

return true;

else

return false;

}

// Adding an element

void enQueue(int element) {

if (isFull()) {

System.out.println("Queue is full");

} else {

if (front == -1)

front = 0;

rear = (rear + 1) % SIZE;

items[rear] = element;

System.out.println("Inserted " + element);

}

}

// Removing an element

int deQueue() {

int element;

if (isEmpty()) {

System.out.println("Queue is empty");

return (-1);

} else {

element = items[front];

if (front == rear) {

front = -1;

rear = -1;

} /\* Q has only one element, so we reset the queue after deleting it. \*/

else {

front = (front + 1) % SIZE;

}

return (element);

}

}

void display() {

/\* Function to display status of Circular Queue \*/

int i;

if (isEmpty()) {

System.out.println("Empty Queue");

} else {

System.out.println("Front -> " + front);

System.out.println("Items -> ");

for (i = front; i != rear; i = (i + 1) % SIZE)

System.out.print(items[i] + " ");

System.out.println(items[i]);

System.out.println("Rear -> " + rear);

}

}

public static void main(String[] args) {

QArray q = new QArray();

Scanner sc=new Scanner(System.in);

System.out.println("1.Enqueue\t 2.Dequeue \t3.Display \t4.exit");

int ch;

do

{

System.out.println("enter your choice");

ch=sc.nextInt();

switch(ch)

{

case 1:

System.out.println("enter data");

int data=sc.nextInt();

q.enQueue(data);

break;

case 2:

int elem = q.deQueue();

if (elem != -1) {

System.out.println("Deleted Element is " + elem);

}

break;

case 3:

q.display();

break;

}

}while(ch<=3);

}

}

**OUTPUT**

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>javac QArray.java

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java QArray

1.Enqueue 2.Dequeue 3.Display 4.exit

enter your choice

1

enter data

12

Inserted 12

enter your choice

1

enter data

34

Inserted 34

enter your choice

3

Front -> 0

Items ->

12 34

Rear -> 1

enter your choice

1

enter data

55

Inserted 55

enter your choice

3

Front -> 0

Items ->

12 34 55

Rear -> 2

enter your choice

2

Deleted Element is 12

enter your choice

3

Front -> 1

Items ->

34 55

Rear -> 2

enter your choice

4

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>

**PROGRAM**

b) A restaurant based on its human resources can accept a maximum of N number of food orders. The food orders are served in first come first serve basis. The food orders once placed cannot be cancelled. Write a program to simulate the food ordering and serving system in the restaurant using circular queue.

|  |
| --- |
| public class CircularQueue implements Queue { |
|  |

|  |
| --- |
| public final int capacity; |
|  |

|  |
| --- |
| public PrintJob[] jobs; |
|  |

|  |
| --- |
| public int index; |
|  |

|  |
| --- |
| public Queue queue; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| public CircularQueue(int capacity) { |
|  |

|  |
| --- |
| this.capacity = capacity; |
|  |

|  |
| --- |
| this.index = 0; |
|  |

|  |
| --- |
| this.jobs = new PrintJob[capacity]; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public int getNumberOfJobs() { |
|  |

|  |
| --- |
| return index--; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public boolean isEmpty() { |
|  |

|  |
| --- |
| return index == 0; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public void AddBck(PrintJob job) throws QueueException { |
|  |

|  |
| --- |
| synchronized (this) { |
|  |

|  |
| --- |
| if (index + 1 > capacity) { |
|  |

|  |
| --- |
| throw new QueueException("Overflow"); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| this.jobs[index] = job; |
|  |

|  |
| --- |
| notify(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @Override |
|  |

|  |
| --- |
| public PrintJob removeFront() throws InterruptedException { |
|  |

|  |
| --- |
| synchronized (this) { |
|  |

|  |
| --- |
| if (isEmpty()) { |
|  |

|  |
| --- |
| wait(); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| PrintJob job = this.jobs[0]; |
|  |

|  |
| --- |
| PrintJob aux[] = new PrintJob[capacity]; |
|  |

|  |
| --- |
| System.arraycopy(jobs, 1, aux, 0, index); |
|  |

|  |
| --- |
| this.jobs = aux; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if (index > 0) { |
|  |

|  |
| --- |
| index--; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| notify(); |
|  |

|  |
| --- |
| return job; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

}

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 6 |

Write a program to perform the following operations on the binary searchtree.

1. Construct binary search tree by inserting the values {6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2} in the givenorder.
2. Displaythenodesofthetreeusinginorder,preorderandpostorder traversaltechniques.
3. Display the smallest number stored in thetree.
4. Search the tree for a givennumber.

class BSN

{

int data;

BSN left,right;

BSN(int data)

{

this.data=data;

this.left=null;

this.right=null;

}

}

public class BST {

BSN root=null;

boolean f=false;

void insert(int data)

{

root=addNode(root,data);

}

BSN addNode(BSN root,int data)

{

if(root==null)

{

root=new BSN(data);

return root;

}

if(data<root.data)

{

root.left=addNode(root.left,data);

}

else if(data>root.data)

{

root.right=addNode(root.right,data);

}

return root;

}

boolean search(BSN r,int data)

{

if(r==null)

{

}

else if(data==r.data)

{

f=true;

}

else if(data<r.data)

{

search(r.left,data);

}

else

{

search(r.right,data);

}

return f;

}

void inorder(BSN root)

{

if(root==null)

{

}

else

{

inorder(root.left);

System.out.print(root.data+",");

inorder(root.right);

}

}

void preorder(BSN root)

{

if(root==null)

{

}

else

{

System.out.print(root.data+",");

preorder(root.left);

preorder(root.right);

}

}

void postorder(BSN root)

{

if(root==null)

{

}

else

{

postorder(root.left);

postorder(root.right);

System.out.print(root.data+",");

}

}

void del(int data)

{

root=delNode(root,data);

}

BSN delNode(BSN root,int data)

{

if(root==null)

{

return root;

}

if(data<root.data)

{

root.left=delNode(root.left,data);

}

else if(data>root.data)

{

root.right=delNode(root.right,data);

}

else

{

if(root.left==null)

{

return root.right;

}

else if(root.right==null)

{

return root.left;

}

root.data=minvalue(root.right);

root.right=delNode(root.right,root.data);

}

return root;

}

int minvalue(BSN root)

{

int m=root.data;

while(root.left!=null)

{

m=root.left.data;

root=root.left;

}

return m;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

BST b=new BST();

b.insert(20);

b.insert(30);

b.insert(10);

b.insert(5);

b.insert(15);

b.insert(40);

b.inorder(b.root);

System.out.println();

b.preorder(b.root);

System.out.println();

b.postorder(b.root);

System.out.println(b.search(b.root,15));

b.del(10);

b.inorder(b.root);

System.out.println();

b.del(5);

b.inorder(b.root);

System.out.println();

}

}

**OUTPUT**

C:\Program Files (x86)\Java\jdk1.8.0\_151\bin>java BST

5,10,15,20,30,40,

20,10,5,15,30,40,

5,15,10,40,30,20,true

5,15,20,30,40,

15,20,30,40,

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 7 |

There are train paths between cities. If there is a train between city A and city B thenthereisaroutebetweenthecities.Thecostoftherouteisthedistancebetween cityAandcityB.Representthetraintravelrouteinformationasagraph.Thenode canberepresentedbythenameofthecity.Writeaprogramtoperformthefollowing operations.

* 1. Store the details of train travel route information using adjacency list or adjacency matrixrepresentation.
  2. Traversethegraphanddisplaythedetailsofalltrainsbetweenthecities along with the cost using breadth-firstmethod.
  3. Traversethegraphanddisplaythedetailsofalltrainsbetweenthecities along with the cost using depth-firstmethod.

class BSN

{

int data;

BSN left,right;

BSN(int data)

{

this.data=data;

this.left=null;

this.right=null;

}

}

public class BST {

BSN root=null;

boolean f=false;

void insert(int data)

{

root=addNode(root,data);

}

BSN addNode(BSN root,int data)

{

if(root==null)

{

root=new BSN(data);

return root;

}

if(data<root.data)

{

root.left=addNode(root.left,data);

}

else if(data>root.data)

{

root.right=addNode(root.right,data);

}

return root;

}

boolean search(BSN r,int data)

{

if(r==null)

{

}

else if(data==r.data)

{

f=true;

}

else if(data<r.data)

{

search(r.left,data);

}

else

{

search(r.right,data);

}

return f;

}

void inorder(BSN root)

{

if(root==null)

{

}

else

{

inorder(root.left);

System.out.print(root.data+",");

inorder(root.right);

}

}

void preorder(BSN root)

{

if(root==null)

{

}

else

{

System.out.print(root.data+",");

preorder(root.left);

preorder(root.right);

}

}

void postorder(BSN root)

{

if(root==null)

{

}

else

{

postorder(root.left);

postorder(root.right);

System.out.print(root.data+",");

}

}

void del(int data)

{

root=delNode(root,data);

}

BSN delNode(BSN root,int data)

{

if(root==null)

{

return root;

}

if(data<root.data)

{

root.left=delNode(root.left,data);

}

else if(data>root.data)

{

root.right=delNode(root.right,data);

}

else

{

if(root.left==null)

{

return root.right;

}

else if(root.right==null)

{

return root.left;

}

root.data=minvalue(root.right);

root.right=delNode(root.right,root.data);

}

return root;

}

int minvalue(BSN root)

{

int m=root.data;

while(root.left!=null)

{

m=root.left.data;

root=root.left;

}

return m;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

BST b=new BST();

b.insert(20);

b.insert(30);

b.insert(10);

b.insert(5);

b.insert(15);

b.insert(40);

b.inorder(b.root);

System.out.println();

b.preorder(b.root);

System.out.println();

b.postorder(b.root);

System.out.println(b.search(b.root,15));

b.del(10);

b.inorder(b.root);

System.out.println();

b.del(5);

b.inorder(b.root);

System.out.println();

}

}

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 8 |

Store register numbers of students who attended placement training program in a randomorderinanarray.Writeafunctiontosearchwhetherastudenthasattended placement training program or notusing

1. Linearsearch
2. Binarysearch

**Linearsearch**

// Java code for linearly searching x in arr[]. If x // is present then return its // location, otherwise return -1

class GFG

{

public static int search(int arr[], int x)

{

int n = arr.length;

for (int i = 0; i < n; i++)

{

if (arr[i] == x)

return i;

}

return -1;

}

// Driver code

public static void main(String args[])

{

int arr[] = { 2, 3, 4, 10, 40 };

int x = 10;

// Function call

int result = search(arr, x);

if (result == -1)

System.out.print(

"Element is not present in array");

else

System.out.print("Element is present at index "

+ result);

}

}

**Binarysearch**

// Binary Search in Java

class BinarySearch {

int binarySearch(int array[], int x, int low, int high) {

// Repeat until the pointers low and high meet each other

while (low <= high) {

int mid = low + (high - low) / 2;

if (array[mid] == x)

return mid;

if (array[mid] < x)

low = mid + 1;

else

high = mid - 1;

}

return -1;

}

public static void main(String args[]) {

BinarySearch ob = new BinarySearch();

int array[] = { 3, 4, 5, 6, 7, 8, 9 };

int n = array.length;

int x = 4;

int result = ob.binarySearch(array, x, 0, n - 1);

if (result == -1)

System.out.println("Not found");

else

System.out.println("Element found at index " + result);

}

}

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 9 |

Write a program to sort a given set of integersusing

1. Quicksort
2. Shellsort

// Quick sort in Java

import java.util.Arrays;

class Quicksort {

// method to find the partition position

static int partition(int array[], int low, int high) {

// choose the rightmost element as pivot

int pivot = array[high];

// pointer for greater element

int i = (low - 1);

// traverse through all elements

// compare each element with pivot

for (int j = low; j < high; j++) {

if (array[j] <= pivot) {

// if element smaller than pivot is found

// swap it with the greatr element pointed by i

i++;

// swapping element at i with element at j

int temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

// swapt the pivot element with the greater element specified by i

int temp = array[i + 1];

array[i + 1] = array[high];

array[high] = temp;

// return the position from where partition is done

return (i + 1);

}

static void quickSort(int array[], int low, int high) {

if (low < high) {

// find pivot element such that

// elements smaller than pivot are on the left

// elements greater than pivot are on the right

int pi = partition(array, low, high);

// recursive call on the left of pivot

quickSort(array, low, pi - 1);

// recursive call on the right of pivot

quickSort(array, pi + 1, high);

}

}

}

// Main class

class Main {

public static void main(String args[]) {

int[] data = { 8, 7, 2, 1, 0, 9, 6 };

System.out.println("Unsorted Array");

System.out.println(Arrays.toString(data));

int size = data.length;

// call quicksort() on array data

Quicksort.quickSort(data, 0, size - 1);

System.out.println("Sorted Array in Ascending Order: ");

System.out.println(Arrays.toString(data));

}

}

**Shell Sort in Java**

class ShellSort {

/\* function to implement shellSort \*/

static void shell(int a[], int n)

{

    /\* Rearrange the array elements at n/2, n/4, ..., 1 intervals \*/

    for (int interval = n/2; interval > 0; interval /= 2)

    {

     for (int i = interval; i < n; i += 1)

        {

            /\* store a[i] to the variable temp and make

the ith position empty \*/

            int temp = a[i];

            int j;

            for (j = i; j >= interval && a[j - interval] >

temp; j -= interval)

                a[j] = a[j - interval];

            /\* put temp (the original a[i]) in its correct

position \*/

            a[j] = temp;

        }

    }

}

static void printArr(int a[], int n) /\* function to print the array elements \*/

{

    int i;

    for (i = 0; i < n; i++)

        System.out.print(a[i] + " ");

}

public static void main(String args[])

{

    int a[] = { 30, 28, 37, 5, 9, 14, 22, 39 };

    int n = a.length;

    System.out.print("Before sorting array elements are - \n");

    printArr(a, n);

    shell(a, n);

    System.out.print("\nAfter applying shell sort, the array elements are - \n");

    printArr(a, n);

}

}

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 10 |

a) Write a program to sort a given set of integers using mergesort.

b) Write a program to read the marks obtained by students in a mathematics examination and store the data using a heap data structure. Find out the maximum and minimum marks obtained by the students.

**Merge Sort**

import java.util.Arrays;

// Merge sort in Java

class Main {

// Merge two sub arrays L and M into array

void merge(int array[], int p, int q, int r) {

int n1 = q - p + 1;

int n2 = r - q;

int L[] = new int[n1];

int M[] = new int[n2];

// fill the left and right array

for (int i = 0; i < n1; i++)

L[i] = array[p + i];

for (int j = 0; j < n2; j++)

M[j] = array[q + 1 + j];

// Maintain current index of sub-arrays and main array

int i, j, k;

i = 0;

j = 0;

k = p;

// Until we reach either end of either L or M, pick larger among

// elements L and M and place them in the correct position at A[p..r]

// for sorting in descending

// use if(L[i] >= <[j])

while (i < n1 && j < n2) {

if (L[i] <= M[j]) {

array[k] = L[i];

i++;

} else {

array[k] = M[j];

j++;

}

k++;

}

// When we run out of elements in either L or M,

// pick up the remaining elements and put in A[p..r]

while (i < n1) {

array[k] = L[i];

i++;

k++;

}

while (j < n2) {

array[k] = M[j];

j++;

k++;

}

}

// Divide the array into two sub arrays, sort them and merge them

void mergeSort(int array[], int left, int right) {

if (left < right) {

// m is the point where the array is divided into two sub arrays

int mid = (left + right) / 2;

// recursive call to each sub arrays

mergeSort(array, left, mid);

mergeSort(array, mid + 1, right);

// Merge the sorted sub arrays

merge(array, left, mid, right);

}

}

public static void main(String args[]) {

// created an unsorted array

int[] array = { 6, 5, 12, 10, 9, 1 };

Main ob = new Main();

// call the method mergeSort()

// pass argument: array, first index and last index

ob.mergeSort(array, 0, array.length - 1);

System.out.println("Sorted Array:");

System.out.println(Arrays.toString(array));

}

}

**MAXHEAP**

import java.util.Arrays;

import java.util.Vector;

// A class for implementing the priority queue

class PriorityQueue

{

// vector to store heap elements

private Vector<Integer> A;

// constructor: use the default initial capacity of a vector

public PriorityQueue() {

A = new Vector();

}

// constructor: set a custom initial capacity for vector

public PriorityQueue(int capacity) {

A = new Vector(capacity);

}

// return parent of `A[i]`

private int parent(int i)

{

// if `i` is already a root node

if (i == 0) {

return 0;

}

return (i - 1) / 2;

}

// return left child of `A[i]`

private int LEFT(int i) {

return (2\*i + 1);

}

// return right child of `A[i]`

private int RIGHT(int i) {

return (2\*i + 2);

}

// swap values at two indexes

void swap(int x, int y)

{

// swap with a child having greater value

Integer temp = A.get(x);

A.setElementAt(A.get(y), x);

A.setElementAt(temp, y);

}

// Recursive heapify-down procedure. Here, the node at index `i`

// and its two direct children violate the heap property

private void heapify\_down(int i)

{

// get left and right child of node at index `i`

int left = LEFT(i);

int right = RIGHT(i);

int largest = i;

// compare `A[i]` with its left and right child

// and find the largest value

if (left < size() && A.get(left) > A.get(i)) {

largest = left;

}

if (right < size() && A.get(right) > A.get(largest)) {

largest = right;

}

if (largest != i)

{

// swap with a child having greater value

swap(i, largest);

// call heapify-down on the child

heapify\_down(largest);

}

}

// Recursive heapify-up procedure

private void heapify\_up(int i)

{

// check if the node at index `i` and its parent violates

// the heap property

if (i > 0 && A.get(parent(i)) < A.get(i))

{

// swap the two if heap property is violated

swap(i, parent(i));

// call heapify-up on the parent

heapify\_up(parent(i));

}

}

// return size of the heap

public int size() {

return A.size();

}

// check if the heap is empty or not

public Boolean isEmpty() {

return A.isEmpty();

}

// insert a specified key into the heap

public void add(Integer key)

{

// insert a new element at the end of the vector

A.addElement(key);

// get element index and call heapify-up procedure

int index = size() - 1;

heapify\_up(index);

}

// Function to remove and return an element with the highest priority

// (present at the root). It returns null if the queue is empty

public Integer poll()

{

try {

// if the heap is empty, throw an exception

if (size() == 0) {

throw new Exception("Index is out of range (Heap underflow)");

}

// element with the highest priority

int root = A.firstElement(); // or A.get(0);

// replace the root of the heap with the last element of the vector

A.setElementAt(A.lastElement(), 0);

A.remove(size() - 1);

// call heapify-down on the root node

heapify\_down(0);

// return root element

return root;

}

// catch and print the exception

catch (Exception ex)

{

System.out.println(ex);

return null;

}

}

// Function to return an element with the highest priority

// (present at the root). It returns null if the queue is empty

public Integer peek()

{

try {

// if the heap has no elements, throw an exception

if (size() == 0) {

throw new Exception("Index out of range (Heap underflow)");

}

// otherwise, return the top (first) element

return A.firstElement(); // or A.get(0);

}

// catch the exception and print it, and return null

catch (Exception ex)

{

System.out.println(ex);

return null;

}

}

// Function to remove all elements from the priority queue

public void clear()

{

System.out.print("Emptying queue: ");

while (!A.isEmpty()) {

System.out.print(poll() + " ");

}

System.out.println();

}

// Returns true if the queue contains the specified element

public Boolean contains(Integer i) {

return A.contains(i);

}

// Returns an array containing all elements in the queue

public Integer[] toArray() {

return A.toArray(new Integer[size()]);

}

}

class Main

{

public static void main (String[] args)

{

// create a priority queue with an initial capacity of 10.

// The value of an element decides the priority of it.

PriorityQueue pq = new PriorityQueue(10);

// insert three integers

pq.add(3);

pq.add(2);

pq.add(15);

// print priority queue size

System.out.println("Priority queue size is " + pq.size());

// search 2 in priority queue

Integer searchKey = 2;

if (pq.contains(searchKey)) {

System.out.println("Priority queue contains " + searchKey + "\n");

}

// empty queue

pq.clear();

if (pq.isEmpty()) {

System.out.println("The queue is empty");

}

System.out.println("\nCalling remove operation on an empty heap");

System.out.println("The element with the highest priority is " + pq.poll());

System.out.println("\nCalling peek operation on an empty heap");

System.out.println("The element with the highest priority is " + pq.peek() +

System.lineSeparator());

// again insert three integers

pq.add(5);

pq.add(4);

pq.add(45);

// construct an array containing all elements present in the queue

Integer[] I = pq.toArray();

System.out.println("Printing array: " + Arrays.toString(I));

System.out.println("\nThe element with the highest priority is " + pq.poll());

System.out.println("The element with the highest priority is " + pq.peek());

}

}

**MINHEAP**

import java.util.Arrays;

import java.util.Vector;

// A class for implementing the Priority queue

class PriorityQueue

{

// vector to store heap elements

private Vector<Integer> A;

// constructor: use the default initial capacity of a vector

public PriorityQueue() {

A = new Vector();

}

// constructor: set a custom initial capacity for vector

public PriorityQueue(int capacity) {

A = new Vector(capacity);

}

// return parent of `A[i]`

private int parent(int i)

{

// if `i` is already a root node

if (i == 0) {

return 0;

}

return (i - 1) / 2;

}

// return left child of `A[i]`

private int LEFT(int i) {

return (2\*i + 1);

}

// return right child of `A[i]`

private int RIGHT(int i) {

return (2\*i + 2);

}

// swap values at two indexes

void swap(int x, int y)

{

// swap with a child having greater value

Integer temp = A.get(x);

A.setElementAt(A.get(y), x);

A.setElementAt(temp, y);

}

// Recursive heapify-down procedure. Here, the node at index `i`

// and its two direct children violate the heap property

private void heapify\_down(int i)

{

// get left and right child of node at index `i`

int left = LEFT(i);

int right = RIGHT(i);

int smallest = i;

// compare `A[i]` with its left and right child

// and find the smallest value

if (left < size() && A.get(left) < A.get(i)) {

smallest = left;

}

if (right < size() && A.get(right) < A.get(smallest)) {

smallest = right;

}

if (smallest != i)

{

// swap with a child having lesser value

swap(i, smallest);

// call heapify-down on the child

heapify\_down(smallest);

}

}

// Recursive heapify-up procedure

private void heapify\_up(int i)

{

// check if the node at index `i` and its parent violates

// the heap property

if (i > 0 && A.get(parent(i)) > A.get(i))

{

// swap the two if heap property is violated

swap(i, parent(i));

// call heapify-up on the parent

heapify\_up(parent(i));

}

}

// return size of the heap

public int size() {

return A.size();

}

// check if the heap is empty or not

public Boolean isEmpty() {

return A.isEmpty();

}

// insert a specified key into the heap

public void add(Integer key)

{

// insert a new element at the end of the vector

A.addElement(key);

// get its index and call the heapify-up procedure

int index = size() - 1;

heapify\_up(index);

}

// Function to remove and return an element with the highest priority

// (present at the root). It returns null if the queue is empty

public Integer poll()

{

try {

// if the heap is empty, throw an exception

if (size() == 0) {

throw new Exception("Index is out of range (Heap underflow)");

}

// element with the highest priority

int root = A.firstElement(); // or A.get(0);

// replace the root of the heap with the last element of the vector

A.setElementAt(A.lastElement(), 0);

A.remove(size() - 1);

// call heapify-down on the root node

heapify\_down(0);

// return root element

return root;

}

// catch and print the exception

catch (Exception ex)

{

System.out.println(ex);

return null;

}

}

// Function to return an element with the highest priority

// (present at the root). It returns null if the queue is empty

public Integer peek()

{

try {

// if the heap has no elements, throw an exception

if (size() == 0) {

throw new Exception("Index out of range (Heap underflow)");

}

// otherwise, return the top (first) element

return A.firstElement(); // or A.get(0);

}

// catch the exception and print it, and return null

catch (Exception ex)

{

System.out.println(ex);

return null;

}

}

// Function to remove all elements from the priority queue

public void clear()

{

System.out.print("Emptying queue: ");

while (!A.isEmpty()) {

System.out.print(poll() + " ");

}

System.out.println();

}

// Returns true if the queue contains the specified element

public Boolean contains(Integer i) {

return A.contains(i);

}

// Returns an array containing all elements in the queue

public Integer[] toArray() {

return A.toArray(new Integer[size()]);

}

}

class Main

{

public static void main (String[] args)

{

// create a priority queue with an initial capacity of 10.

// The value of an element decides the priority of it.

PriorityQueue pq = new PriorityQueue(10);

// insert three integers

pq.add(3);

pq.add(2);

pq.add(15);

// print priority queue size

System.out.println("Priority queue size is " + pq.size());

// search 2 in priority queue

Integer searchKey = 2;

if (pq.contains(searchKey)) {

System.out.println("Priority queue contains " + searchKey + "\n");

}

// empty queue

pq.clear();

if (pq.isEmpty()) {

System.out.println("The queue is empty");

}

System.out.println("\nCalling remove operation on an empty heap");

System.out.println("The element with the highest priority is " + pq.poll());

System.out.println("\nCalling peek operation on an empty heap");

System.out.println("The element with the highest priority is " + pq.peek() +

System.lineSeparator());

// again insert three integers

pq.add(5);

pq.add(4);

pq.add(45);

// construct an array containing all elements present in the queue

Integer[] I = pq.toArray();

System.out.println("Printing array: " + Arrays.toString(I));

System.out.println("\nThe element with the highest priority is " + pq.poll());

System.out.println("The element with the highest priority is " + pq.peek());

}

}

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 11 |

Write a program to implement the following hashingfunctions.

1. Separate ChainingMethod
2. Open AddressingMethod

// Java program to demonstrate implementation of our

// own hash table with chaining for collision detection

import java.util.ArrayList;

import java.util.Objects;

// A node of chains

class HashNode<K, V> {

K key;

V value;

final int hashCode;

// Reference to next node

HashNode<K, V> next;

// Constructor

public HashNode(K key, V value, int hashCode)

{

this.key = key;

this.value = value;

this.hashCode = hashCode;

}

}

// Class to represent entire hash table

class Map<K, V> {

// bucketArray is used to store array of chains

private ArrayList<HashNode<K, V>> bucketArray;

// Current capacity of array list

private int numBuckets;

// Current size of array list

private int size;

// Constructor (Initializes capacity, size and

// empty chains.

public Map()

{

bucketArray = new ArrayList<>();

numBuckets = 10;

size = 0;

// Create empty chains

for (int i = 0; i < numBuckets; i++)

bucketArray.add(null);

}

public int size() { return size; }

public boolean isEmpty() { return size() == 0; }

private final int hashCode (K key) {

return Objects.hashCode(key);

}

// This implements hash function to find index

// for a key

private int getBucketIndex(K key)

{

int hashCode = hashCode(key);

int index = hashCode % numBuckets;

// key.hashCode() coule be negative.

index = index < 0 ? index \* -1 : index;

return index;

}

// Method to remove a given key

public V remove(K key)

{

// Apply hash function to find index for given key

int bucketIndex = getBucketIndex(key);

int hashCode = hashCode(key);

// Get head of chain

HashNode<K, V> head = bucketArray.get(bucketIndex);

// Search for key in its chain

HashNode<K, V> prev = null;

while (head != null) {

// If Key found

if (head.key.equals(key) && hashCode == head.hashCode)

break;

// Else keep moving in chain

prev = head;

head = head.next;

}

// If key was not there

if (head == null)

return null;

// Reduce size

size--;

// Remove key

if (prev != null)

prev.next = head.next;

else

bucketArray.set(bucketIndex, head.next);

return head.value;

}

// Returns value for a key

public V get(K key)

{

// Find head of chain for given key

int bucketIndex = getBucketIndex(key);

int hashCode = hashCode(key);

HashNode<K, V> head = bucketArray.get(bucketIndex);

// Search key in chain

while (head != null) {

if (head.key.equals(key) && head.hashCode == hashCode)

return head.value;

head = head.next;

}

// If key not found

return null;

}

// Adds a key value pair to hash

public void add(K key, V value)

{

// Find head of chain for given key

int bucketIndex = getBucketIndex(key);

int hashCode = hashCode(key);

HashNode<K, V> head = bucketArray.get(bucketIndex);

// Check if key is already present

while (head != null) {

if (head.key.equals(key) && head.hashCode == hashCode) {

head.value = value;

return;

}

head = head.next;

}

// Insert key in chain

size++;

head = bucketArray.get(bucketIndex);

HashNode<K, V> newNode

= new HashNode<K, V>(key, value, hashCode);

newNode.next = head;

bucketArray.set(bucketIndex, newNode);

// If load factor goes beyond threshold, then

// double hash table size

if ((1.0 \* size) / numBuckets >= 0.7) {

ArrayList<HashNode<K, V>> temp = bucketArray;

bucketArray = new ArrayList<>();

numBuckets = 2 \* numBuckets;

size = 0;

for (int i = 0; i < numBuckets; i++)

bucketArray.add(null);

for (HashNode<K, V> headNode : temp) {

while (headNode != null) {

add(headNode.key, headNode.value);

headNode = headNode.next;

}

}

}

}

// Driver method to test Map class

public static void main(String[] args)

{

Map<String, Integer> map = new Map<>();

map.add("this", 1);

map.add("coder", 2);

map.add("this", 4);

map.add("hi", 5);

System.out.println(map.size());

System.out.println(map.remove("this"));

System.out.println(map.remove("this"));

System.out.println(map.size());

System.out.println(map.isEmpty());

}

}

**OPEN ADDRESSING**

**publicclass** HashEntry {

**privateint**key;

**privateint**value;

      HashEntry(**int** key, **int** value) {

**this**.key = key;

**this**.value = value;

      }

**publicint** getValue() {

**return**value;

      }

**publicvoid** setValue(**int** value) {

**this**.value = value;

      }

**publicint** getKey() {

**return**key;

      }

}

**publicclass** DeletedEntry **extends** HashEntry {

**privatestatic** DeletedEntry entry = **null**;

**private** DeletedEntry() {

**super**(-1, -1);

      }

**publicstatic** DeletedEntry getUniqueDeletedEntry() {

**if** (entry == **null**)

                  entry = **new** DeletedEntry();

**return**entry;

      }

}

**publicclass** HashMap {

**privatefinalstaticint**TABLE\_SIZE = 128;

      HashEntry[] table;

      HashMap() {

            table = **new** HashEntry[TABLE\_SIZE];

**for** (**int** i = 0; i <TABLE\_SIZE; i++)

                  table[i] = **null**;

      }

**publicint** get(**int** key) {

**int** hash = (key % TABLE\_SIZE);

**int** initialHash = -1;

**while** (hash != initialHash

                        && (table[hash] == DeletedEntry.getUniqueDeletedEntry() || table[hash] != **null**

                                   &&table[hash].getKey() != key)) {

**if** (initialHash == -1)

                        initialHash = hash;

                  hash = (hash + 1) % TABLE\_SIZE;

            }

**if** (table[hash] == **null** || hash == initialHash)

**return** -1;

**else**

**return**table[hash].getValue();

      }

**publicvoid** put(**int** key, **int** value) {

**int** hash = (key % TABLE\_SIZE);

**int** initialHash = -1;

**int** indexOfDeletedEntry = -1;

**while** (hash != initialHash

                        && (table[hash] == DeletedEntry.getUniqueDeletedEntry() || table[hash] != **null**

                                   &&table[hash].getKey() != key)) {

**if** (initialHash == -1)

                        initialHash = hash;

**if** (table[hash] == DeletedEntry.getUniqueDeletedEntry())

                        indexOfDeletedEntry = hash;

                  hash = (hash + 1) % TABLE\_SIZE;

            }

**if** ((table[hash] == **null** || hash == initialHash)

                        && indexOfDeletedEntry != -1)

                  table[indexOfDeletedEntry] = **new** HashEntry(key, value);

**elseif** (initialHash != hash)

**if** (table[hash] != DeletedEntry.getUniqueDeletedEntry()

                             &&table[hash] != **null**&&table[hash].getKey() == key)

                        table[hash].setValue(value);

**else**

                        table[hash] = **new** HashEntry(key, value);

      }

**publicvoid** remove(**int** key) {

**int** hash = (key % TABLE\_SIZE);

**int** initialHash = -1;

**while** (hash != initialHash

                        && (table[hash] == DeletedEntry.getUniqueDeletedEntry() || table[hash] != **null**

                                   &&table[hash].getKey() != key)) {

**if** (initialHash == -1)

                        initialHash = hash;

                  hash = (hash + 1) % TABLE\_SIZE;

            }

**if** (hash != initialHash &&table[hash] != **null**)

                  table[hash] = DeletedEntry.getUniqueDeletedEntry();

      }

}

**/\*\***

**\* @file**

**\* Hash table with open addressing Version 1**

**\***

**\*/**

public class HashTableOpenAddressing {

/\*\*

\* Properties of hash table.

\*/

private int max = 20;

private HashTableNode[] vector = new HashTableNode[max];

/\*\*

\* Element of hash table.

\*/

class HashTableNode {

public int key;

HashTableNode(int key) {

this.key = key;

}

}

/\*\*

\* Constructor method.

\*/

HashTableOpenAddressing() {

for (int i = 0; i < max; i++) {

this.vector[i] = null;

}

}

/\*\*

\* Add method.

\*

\* Add an element to the hash table.

\*

\* @param int key

\* Integer key.

\* @param int i

\* Stepsize.

\* @return void

\*/

public void add(int key, int i) {

int hashcode = this.h(key, i);

if ((i > 0) && (hashcode == this.h(key, 0))) {

System.out.println("It's not possible to add the key " + key + " in hash table.");

return;

}

if (this.vector[hashcode] != null) {

this.add(key, i + 1);

}

else {

this.vector[hashcode] = new HashTableNode(key);

}

}

/\*\*

\* Overloading of add() method.

\*

\* Simplifies the call of method.

\*/

public void add(int key) {

this.add(key, 0);

}

/\*\*

\* Search method.

\*

\* Search an element in hash table.

\*

\*/

public int search(int key, int i) {

int hashcode = this.h(key, i);

if ((i > 0) && (hashcode == this.h(key, 0))) {

System.out.println("Key " + key + " not found in hash table. It was made " + i + " stepsizes.");

return -99;

}

if ((this.vector[hashcode] != null) && (this.vector[hashcode].key == key)) {

System.out.println("Key " + key + " found in hash table in " + (i + 1) + "-esim stepsize.");

return hashcode;

}

return this.search(key, i + 1);

}

/\*\*

\* Overloading of search() method.

\*

\* Simplifies the call of method.

\*/

public int search(int key) {

return this.search(key, 0);

}

/\*\*

\* Remove method.

\*

\* Remove an element from the hash table.

\*/

public void remove(int key) {

int i = this.search(key);

if (i != -99) {

System.out.println("The key " + key + " was removed from hash table.");

this.vector[i] = null;

}

else {

System.out.println("The key " + key + " was not removed, because it does not exists in hash table.");

}

}

/\*\*

\* Linear probing.

\*/

public int h\_linear(int key, int i) {

return (key + i) % this.max;

}

/\*\*

\* Quadratic probing.

\*/

public int h\_quadratic(int key, int i) {

return (key + (i \* i)) % this.max;

}

/\*\*

\* Double hashing.

\*/

public int h\_double(int key, int i) {

return (this.h\_double2(key) + i) % this.max;

}

/\*\*

\* Auxiliar hash function to Double hashing made by h\_double() method.

\*/

public int h\_double2(int key) {

return (1 + key) % (this.max - 1);

}

/\*\*

\* Hash function.

\*/

public int h(int key, int i) {

return this.h\_linear(key, i);

}

/\*\*

\* toString method.

\*

\* Provide some visual funcionality to see the elements inside the hash table.

\*

\* @return String

\* Representation of the hash table in the moment by a string.

\*/

public String toString() {

String description = "Hash table: [ ";

for (int i = 0; i < this.max; i++) {

if (this.vector[i] == null) {

description += "\_\_ ";

}

else {

description += String.format("%2d ", this.vector[i].key);

}

}

description += "]";

return description;

}

}

|  |
| --- |
| **SREE VIDYANIKETHAN ENGINEERING COLLEGE**  **Sree Sainath Nagar, A. Rangampet – 517 102**  **Department of Computer Science and Engineering**  **II B. Tech. I Semester (2021-22)** (20BT30531) DATA STRUCTURES LAB |
| Exp. No: 12 |

Consider an online movie ticket booking system throughwhich customers can book tickets to watch movies at theatres. The database stores the details of each transaction of ticket booking with the details - ID, customer name, customer phone number, movie name, theatre name, date of show, time of show, number of tickets booked, starting seat number, total amount. Write a menu driven program to perform create the database and given an ID, display a client’s phone number. Use a hash table implementation to quickly search through the database.

// Java skeleton code to design an online movie

// booking system.

Enums :

public enum SeatStatus {

SEAT\_BOOKED,

SEAT\_NOT\_BOOKED;

}

public enum MovieStatus {

Movie\_Available,

Movie\_NotAvailable;

}

public enum MovieType {

ENGLISH,

HINDI;

}

public enum SeatType {

NORMAL,

EXECUTIVE,

PREMIUM,

VIP;

}

public enum PaymentStatus {

PAID,

UNPAID;

}

class User {

int userId;

String name;

Date dateOfBirth;

String mobNo;

String emailId;

String sex;

}

class Movie {

int movieId;

int theaterId;

MovieType movieType;

MovieStatus movieStatus;

}

class Theater {

int theaterId;

String theaterName;

Address address;

List<Movie> movies;

float rating;

}

class Booking {

int bookingId;

int userId;

int movieId;

List<Movie> bookedSeats;

int amount;

PaymentStatus status\_of\_payment;

Date booked\_date;

Time movie\_timing;

}

class Address {

String city;

String pinCode;

String state;

String streetNo;

String landmark;

}

Class SeatBook

{

Transaction transaction\_obj;

bool seats[total\_seats];

String place;

String ticketType;

bool check\_availability();

int position\_of\_seat()

{

return seat\_pos\_in\_theater;

}

void multiple tickets();

void final\_booking()

{

place = positon\_of\_seat();

if (single\_ticket)

continue;

else

mutliple\_ticket\_booking();

Transaction\_obj.pay(ticketType, seats\_booked, place);

}

}