*HARSH KASHYAP  
CSE 4*

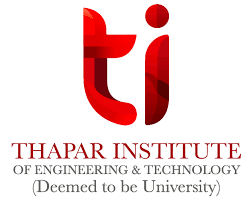
*101917088*

[*hkashyap\_be19@thapar.edu*](mailto:hkashyap_be19@thapar.edu)

A Practical activity Report submitted

for Data Structures (UCS301)

**DATA STRUCTURES**

****

Computer Science and Engineering

Patiala Campus

**2020**

Submitted to

Maninder Kaur

**Assignment 8**

**Question 1**

**Write a program for binary search tree (BST) having functions for the following**

**operations:**

* **Insert an element (no duplicates are allowed),**
* **Delete an existing element,**
* **Traverse the BST (in-order, pre-order, and post-order),**
* **Maximum depth, and**
* **Minimum depth.**

**SOLUTION CODE**

**/\***

**Write a program for binary search tree (BST) having functions for the following**

**operations:**

**Insert an element (no duplicates are allowed),**

**Delete an existing element,**

**Traverse the BST (in-order, pre-order, and post-order),**

**Maximum depth, and**

**Minimum depth.**

**\*/**

**import java.util.\*;**

**class BinarySearchTree**

**{**

**static Node root=null;**

**static class Node**

**{**

**int key;**

**Node left, right;**

**public Node(int d)**

**{**

**key=d;**

**left=right=null;**

**}**

**}**

**static Scanner scr= new Scanner(System.in);**

**public static void insert(int key)**

**{**

**root= insertRec(root, key);**

**}**

**public static Node insertRec(Node root, int key)**

**{**

**if(root == null)**

**{**

**root = new Node(key);**

**return root;**

**}**

**if(key== root.key)**

**{**

**System.out.println("Duplicate entry ");**

**}**

**if(key < root.key)**

**{**

**root.left = insertRec(root.left, key);**

**}**

**else if (key> root.key)**

**{**

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

**}**

**return root;**

**}**

**public static void inorder(Node root)**

**{**

**if(root!=null)**

**{**

**inorder(root.left);**

**System.out.println(root.key);**

**inorder(root.right);**

**}**

**}**

**public static void preorder(Node root)**

**{**

**if(root!=null)**

**{**

**System.out.println(root.key);**

**preorder(root.left);**

**preorder(root.right);**

**}**

**}**

**public static void postorder(Node root)**

**{**

**if(root!=null)**

**{**

**postorder(root.left);**

**postorder(root.right);**

**System.out.println(root.key);**

**}**

**}**

**public static int minValue(Node root)**

**{**

**int min =root.key;**

**Node copy=root;**

**while(copy!=null)**

**{**

**min=copy.key;**

**copy=copy.left;**

**}**

**return min;**

**}**

**public static int maxDepth(Node root)**

**{**

**if(root==null)**

**return 0;**

**else**

**{**

**int lDepth=maxDepth(root.left);**

**int rDepth=maxDepth(root.right);**

**return ((lDepth>rDepth)?(lDepth+1):(rDepth+1));**

**}**

**}**

**public static int minDepth(Node root)**

**{**

**if(root==null)**

**return 0;**

**else if(root.left==null &&root.right==null)**

**return 1;**

**else if (root.left==null)**

**return minDepth(root.right)+1;**

**else if (root.right == null)**

**return minDepth(root.left)+1;**

**else**

**return Math.min(minDepth(root.left),minDepth(root.right))+1;**

**}**

**public static void delete(Node root,int key)**

**{**

**root=deleteRecur(root,key);**

**}**

**public static Node deleteRecur(Node root, int key)**

**{**

**if(root==null)**

**return root;**

**else if (key < root.key)**

**root.left=deleteRecur(root.left, key);**

**else if(key > root.key)**

**root.right=deleteRecur(root.right, key);**

**else**

**{**

**System.out.println("Found and deleted. ");**

**if(root.left==null)**

**return root.right;**

**else if(root.right == null)**

**return root.left;**

**else**

**{**

**root.key= minValue(root.right); //minimum value**

**root.right = deleteRecur(root.right, root.key);**

**}**

**}**

**return root;**

**}**

**public static int menu()**

**{**

**System.out.println("\nMENU");**

**System.out.println("1. Insert an element\n2. Delete an existing element\n3. Traverse the BST - in-order. ");**

**System.out.println("4.Traverse the BST - pre-order.\n5. Traverse the BST - post-order");**

**System.out.println("6. Maximum depth.\n7. Minimum depth.\n8. Exit\nEnter your choice");**

**int choice=scr.nextInt();**

**return choice;**

**}**

**public static void main(String[] args)**

**{**

**BinarySearchTree tree = new BinarySearchTree();**

**int ch;**

**int num;**

**do**

**{**

**ch=menu();**

**switch(ch)**

**{**

**case 1:**

**do{**

**System.out.println("Enter element. Press 0 to exit insertion.");**

**num=scr.nextInt();**

**if(num!=0)**

**tree.insert(num);**

**}while(num!=0);**

**break;**

**case 2:**

**System.out.println("Enter element you wish to delete.");**

**num=scr.nextInt();**

**tree.delete(root, num);**

**break;**

**case 3:**

**System.out.println("Inorder Traversal ");**

**tree.inorder(root);**

**break;**

**case 4:**

**System.out.println("Preorder Traversal ");**

**tree.preorder(root);**

**break;**

**case 5:**

**System.out.println("Postorder Traversal ");**

**tree.postorder(root);**

**break;**

**case 6:**

**System.out.println("The maximum depth is "+maxDepth(root));**

**break;**

**case 7:**

**System.out.println("The minimum depth is "+minDepth(root));**

**break;**

**case 8:**

**System.out.println("Thank You. Exitting");**

**break;**

**default:**

**System.out.println("Not an option. ");**

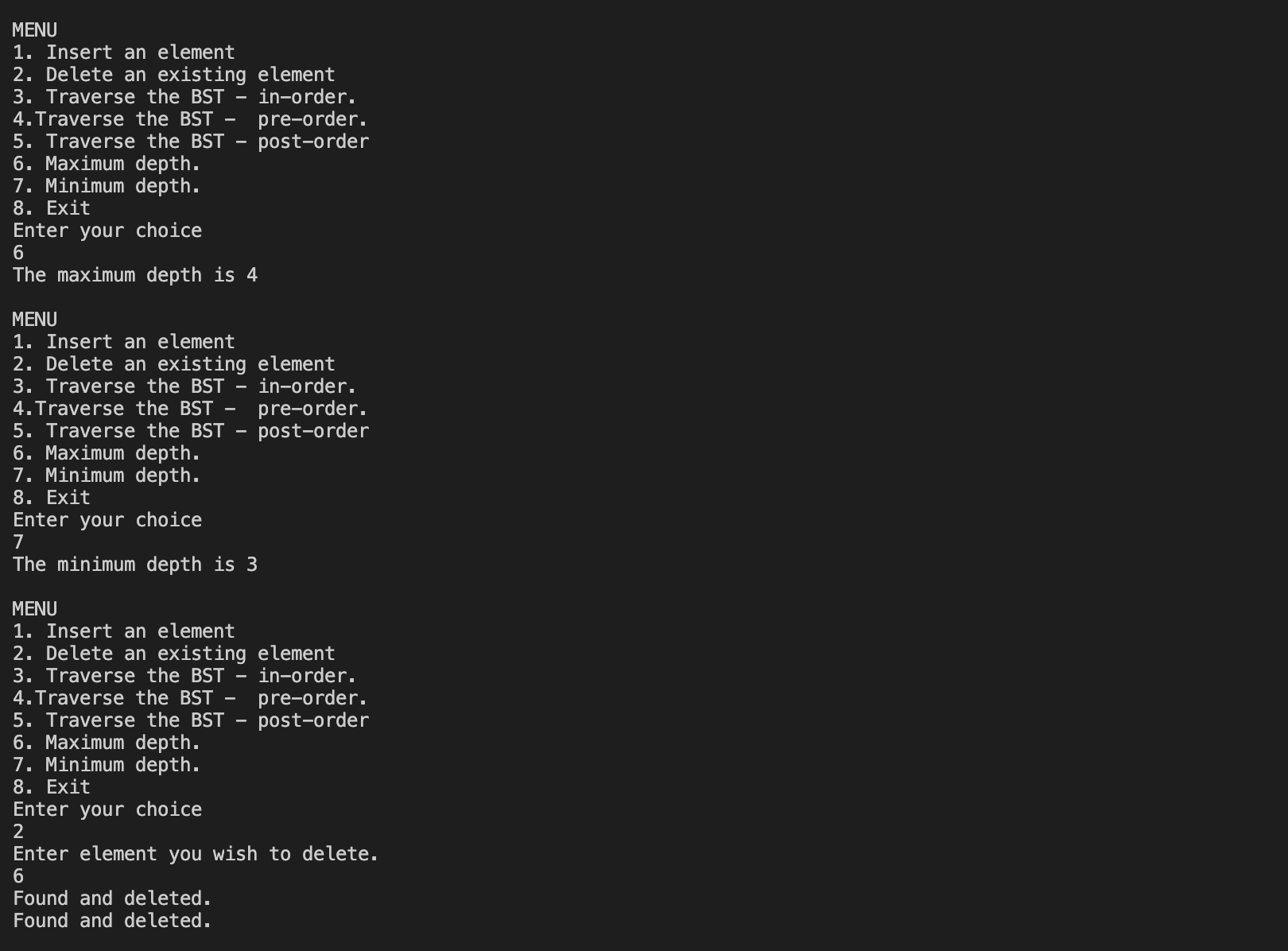
**}**

**}while(ch!=8);**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 2**

**Develop a menu driven program demonstrating the following operations on Circular Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().**

**Note: Use arrays to implement circular queue**

**SOLUTION CODE**

**/\***

**Develop a menu driven program demonstrating the following operations on Circular Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().**

**Note: Use arrays to implement circular queue**

**\*/**

**import java.util.\*;**

**import java.io.\*;**

**class a5ques2**

**{**

**static Scanner scr = new Scanner(System.in);**

**static int SIZE = 5;**

**static int arr[] = new int[SIZE];**

**// Lets keep size as 100**

**static int front =-1;**

**static int rear = -1;**

**public static void main(final String[] args)**

**{**

**int ch = 0;**

**do**

**{**

**ch = menu();**

**switch (ch) {**

**case 1:**

**enqueue();**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**peek();**

**break;**

**case 4:**

**if (isEmpty())**

**System.out.println("Stack is empty .Underflow\n");**

**else**

**System.out.println("Stack is not empty \n");**

**break;**

**case 5:**

**if (isFull())**

**System.out.println("Stack is full. Overflow\n");**

**else**

**System.out.println("Stack is not full \n");**

**break;**

**case 6:**

**display();**

**break;**

**case 7:**

**System.out.println("We are done ");**

**break;**

**default:**

**System.out.println("Not an option \n");**

**}**

**} while (ch != 7);**

**}**

**static int menu() {**

**System.out.println(**

**"\n——MENU——- \n1.ENQUEUE\n2.DEQUEUE\n3.PEEK\n4.EMPTY\n5.FULL\n6.DISPLAY\n7.EXIT\nEnter your choice");**

**int ch = scr.nextInt();**

**return ch;**

**}**

**static void enqueue()**

**{**

**if (isFull())**

**{**

**System.out.println("Overflow ");**

**return;**

**}**

**System.out.print("Enter element to be pushed : ");**

**int pos = scr.nextInt();**

**if (front == -1 && rear == -1)**

**front = rear = 0;**

**else {**

**rear = (rear + 1) % SIZE;**

**}**

**arr[rear] = pos;**

**}**

**static void dequeue()**

**{**

**if (isEmpty())**

**{**

**System.out.println("Underflow ");**

**return;**

**}**

**System.out.println("Popping out element... : "+arr[front]);**

**if(front==rear)**

**front=rear=-1;**

**else**

**front=(front+1)%SIZE;**

**}**

**static void peek()**

**{**

**if (isEmpty())**

**{**

**System.out.println("Underflow ");**

**return;**

**}**

**System.out.println("Viewing the top element ... : "+arr[front]);**

**}**

**static boolean isFull()**

**{**

**if((front==0 && rear==SIZE-1) || (rear==front-1))**

**return true;**

**return false;**

**}**

**static boolean isEmpty()**

**{**

**if(front==-1 )**

**return true;**

**return false;**

**}**

**static void display()**

**{**

**if (isEmpty())**

**{**

**System.out.println("Underflow ");**

**return;**

**}**

**System.out.println("Viewing the Queue ");**

**int i=front;**

**do**

**{**

**System.out.println(" | "+arr[i]+" |");**

**System.out.println(" -----");**

**if(i==rear)**

**break;**

**if(i==SIZE-1)**

**i=0;**

**i++;**

**}while(true);**

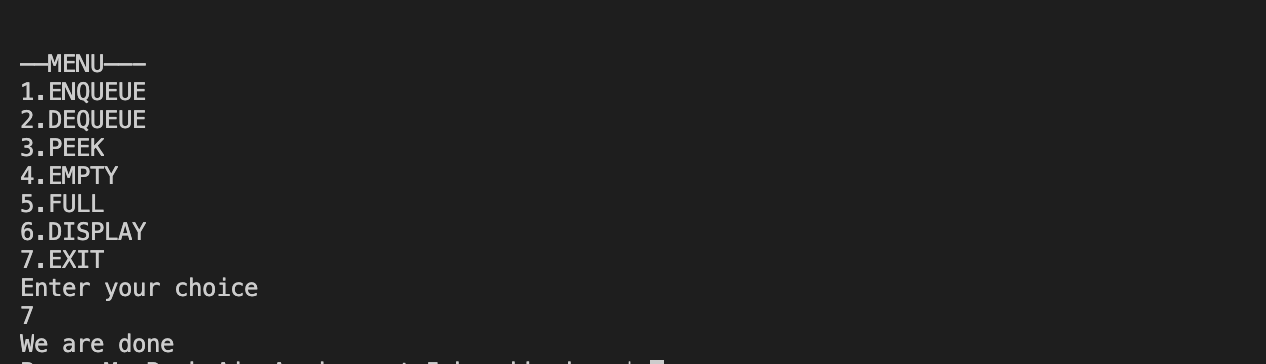
**}**

**}**

**OUTPUT**

****

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 3**

**Write a program interleave the first half of the queue with the second half.**

**SOLUTION CODE**

**/\***

**Write a program interleave the first half of the queue with the second half.**

**\*/**

**import java.util.\*;**

**import java.io.\*;**

**class a5ques3**

**{**

**static Scanner scr = new Scanner(System.in);**

**public static void main(String[] args)**

**{**

**Queue<Integer> q=new LinkedList<Integer>();**

**while(true)**

**{**

**System.out.println("Insert element. Press 0 to terminate");**

**int num=scr.nextInt();**

**if(num==0)**

**{**

**System.out.println("Done with taking input. ");**

**break;**

**}**

**else**

**{**

**q.add(num);**

**}**

**}**

**int size=q.size();**

**interLeave(q);**

**System.out.println("Displaying the new queue");**

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

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

**System.out.println();**

**}**

**static void interLeave(Queue<Integer> q)**

**{**

**int size =q.size();**

**Stack<Integer> s= new Stack<>();**

**for(int i=0;i<size/2;i++)**

**{**

**s.push(q.poll());**

**}**

**while(!s.empty())**

**{**

**q.add(s.pop());**

**}**

**for(int i=0;i<size/2;i++)**

**{**

**q.add(q.poll());**

**}**

**for(int i=0;i<size/2;i++)**

**{**

**s.push(q.poll());**

**}**

**while(!s.empty())**

**{**

**q.add(s.pop());**

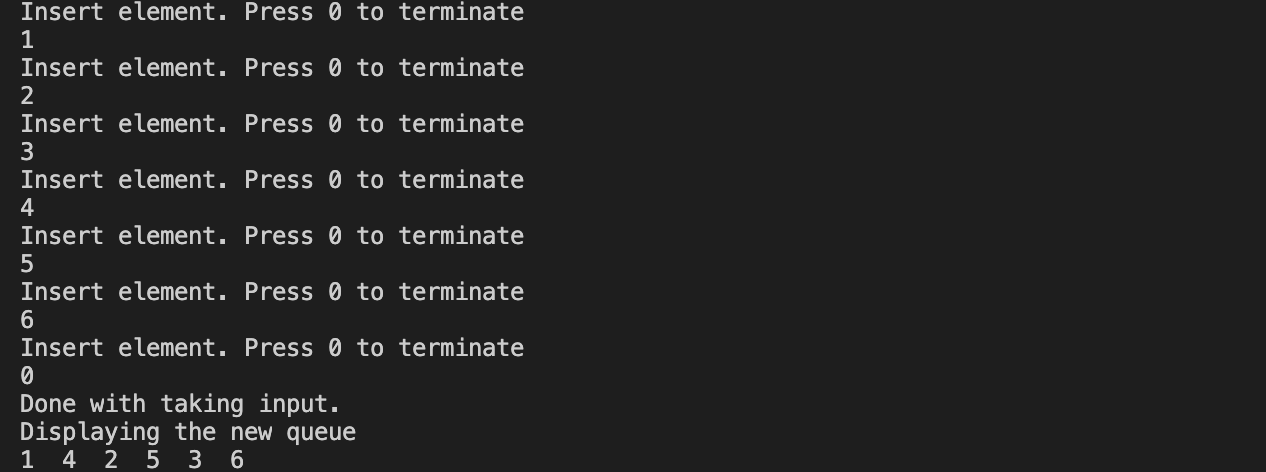
**q.add(q.poll());**

**}**

**}**

**}**

***OUTPUT***

******

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 4**

**Write a program to find first non-repeating character in a string using Queue.**

**SOLUTION CODE**

**import java.util.Scanner;**

**import java.util.Queue;**

**import java.util.LinkedList;**

**import java.io.\*;**

**public class a5ques4**

**{**

**static Scanner scr= new Scanner(System.in);**

**static int SIZE=26;**

**static Queue<Character> q = new LinkedList<Character>();**

**static String str="";**

**public static void main(String[] args)**

**{**

**System.out.print("Enter String on which the operation is to be checked : ");**

**str=scr.nextLine();**

**check();**

**}**

**static void check()**

**{**

**int ch[]=new int[SIZE];**

**for(int i=0;i<SIZE;i++)**

**{**

**ch[i]=0;**

**}**

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

**{**

**char temp= str.charAt(i);**

**if(temp==' ')**

**continue;**

**q.add(temp);**

**//System.out.println(temp-65+" "+temp+" "+ch[temp-65]);**

**ch[temp-'a']++;**

**while(!q.isEmpty())**

**{**

**if(ch[q.peek()-'a']==1)**

**{**

**System.out.print(q.peek()+" ");**

**break;**

**}**

**else**

**{**

**q.poll();**

**}**

**}**

**if(q.isEmpty())**

**System.out.print("-1 ");**

**}**

**System.out.println();**

**}**

**}**

***OUTPUT***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**EXTRA**

**Question**

**WAP to Split a Circular Linked List into two halves.**

**SOLUTION CODE**

**//Java program to generate binary numbers from 1 to n**

**import java.util.LinkedList;**

**import java.util.Queue;**

**import java.util.Scanner;;**

**class a5BinaryExtra**

**{**

**static Scanner scr=new Scanner(System.in);**

**static void generatePrintBinary(int n)**

**{**

**Queue<String> q = new LinkedList<String>();**

**q.add("1");**

**for(int i=n;i>0;i--)**

**{**

**String s1 = q.peek();**

**q.remove();**

**System.out.println(s1);**

**String s2 = s1;**

**q.add(s1 + "0");**

**q.add(s2 + "1");**

**}**

**}**

**public static void main(String[] args)**

**{**

**System.out.print("Enter range : ");**

**int n=scr.nextInt();**

**generatePrintBinary(n);**

**}**

**}**

***OUTPUT***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**