DSA LAB EXAM

NAME – GAGAN VIJAY BHANGALE

1)

```
import javax.lang.model.util.ElementScanner6;
import javax.swing.text.AbstractDocument.LeafElement;

import java.util.Stack;
import java.util.Queue;
import java.util.Scanner;
import java.util.LinkedList;

class BST{

   static class Node{
      private int data;
      private Node left;
      private Node right;
      private boolean visited;

      public Node(int data){
         this.data = data;
         this.left = null;
         this.right = null;
         this.visited = false;
      }

   }

   private Node root;
```

DSA LAB EXAM

```
public BST( ){
   root = null;
}

public boolean isBSTEmpty( ){
   return ( root == null );
}




public void addNode(int data){

   Node newNode = new Node(data);

   if( isBSTEmpty() ){
      root = newNode;
   }else{
      Node trav = root;

      while( true ){
         if( data < trav.data ){
            if( trav.left == null ){
               trav.left = newNode;
               break;
            }else{
               trav = trav.left;
            }
         }else{
            if( trav.right == null ){
               trav.right = newNode;
```

```
                break;
            }else{
                trav = trav.right;
            }
        }
    }
}

public void recAddNode(Node trav, int data){

    if( trav == null )
        return;

    if( data < trav.data ){
        if( trav.left == null ){

            trav.left = new Node(data);
        }else{
            recAddNode(trav.left, data);
        }
    }else{
        if( trav.right == null ){
            trav.right = new Node(data);
        }else{
            recAddNode(trav.right, data);
        }
    }
}

public void recAddNode(int data){
```

```
    if( isBSTEmpty() ){
        root = new Node(data);
    }else{
        recAddNode(root, data);
    }
}

public void preOrder(Node trav){

    if( trav == null )
        return;

    //VLR
    System.out.printf("%4d", trav.data);
    preOrder(trav.left);
    preOrder(trav.right);
}

public void preOrder(){
    if( !isBSTEmpty() ){
        System.out.print("preorder traversal is : ");;
        preOrder(root);
        System.out.println();
    }else
        System.out.println("bst is empty !!!");
}

public void inOrder(Node trav){

    if( trav == null )
```

```
        return;



    inOrder(trav.left);
    System.out.printf("%4d", trav.data);
    inOrder(trav.right);
}

public void inOrder(){
    if( !isBSTEmpty() ){
        System.out.print("inorder traversal is  : ");;
        inOrder(root);
        System.out.println();
    }else
        System.out.println("bst is empty !!!");
}

public void postOrder(Node trav){

    if( trav == null )
        return;

    //LRV
    postOrder(trav.left);
    postOrder(trav.right);
    System.out.printf("%4d", trav.data);
}

public void postOrder(){
    if( !isBSTEmpty() ){
        System.out.print("postorder traversal is: ");;
```

```
        postOrder(root);
        System.out.println();
    }else
        System.out.println("bst is empty !!!");
}

public void nonRecPreOrder( ){

    if( !isBSTEmpty() ){
        Stack<Node> s = new Stack<Node>();

        Node trav = root;
        System.out.print("preorder traversal is : ");
        while( trav != null || !s.empty( ) ){

            while( trav != null ){

                System.out.printf("%4d", trav.data);


                if( trav.right != null )
                    s.push(trav.right);


                trav = trav.left;
            }


            if( !s.empty() ){
                trav = s.pop();
            }
```

```
        }
        System.out.println();
    }else
        System.out.println("bst is empty !!!");
}

public void nonRecInOrder( ){

    if( !isBSTEmpty() ){
        Stack<Node> s = new Stack<Node>();

        Node trav = root;
        System.out.print("inorder traversal is  : ");
        while( trav != null || !s.empty( ) ){

            while( trav != null ){

                s.push(trav);



                trav = trav.left;
            }



            if( !s.empty() ){

                trav = s.pop();

                System.out.printf("%4d", trav.data);
```

```
            trav = trav.right;
          }

       }
       System.out.println();
    }else
       System.out.println("bst is empty !!!");
}

public void nonRecPostOrder( ){

    if( !isBSTEmpty() ){
       Stack<Node> s = new Stack<Node>();

       Node trav = root;
       System.out.print("postorder traversal is: ");
       while( trav != null || !s.empty( ) ){

          while( trav != null ){
             s.push(trav);


             trav = trav.left;
          }


          if( !s.empty() ){

             trav = s.pop();
```

```
            if( trav.right != null && trav.right.visited == false ){

                s.push(trav);

                trav = trav.right;
            }else{
                System.out.printf("%4d", trav.data);
                trav.visited = true;
                trav = null;
            }
          }
       }
       System.out.println();
    }else
       System.out.println("bst is empty !!!");
}

public void dfsTraversal( ){
   if( !isBSTEmpty() ){
       Stack<Node> s = new Stack<Node>();

       s.push(root);
       System.out.print("dfs traversal is : ");
       while( !s.empty( ) ){

          Node trav = s.pop();
          System.out.printf("%4d", trav.data);

          if( trav.right != null )
             s.push(trav.right);
```

```
            if( trav.left != null )
                s.push(trav.left);
        }
        System.out.println();
    }else
        System.out.println("bst is empty !!!");
}

public void bfsTraversal( ){
    if( !isBSTEmpty() ){
        Queue<Node> q = new LinkedList<Node>();

        q.offer(root);//enqueue
        System.out.print("bfs traversal is : ");
        while( !q.isEmpty() ){

            Node trav = q.poll();//dequeue
            System.out.printf("%4d", trav.data);



            if( trav.left != null )
                q.offer(trav.left);



            if( trav.right != null )
                q.offer(trav.right);

            }
        System.out.println();
```

```
    }else
        System.out.println("bst is empty !!!");
}

public boolean searchNode(int data, Node [] arr){

    Node trav = root;
    arr[ 0 ] = null;

    while( trav != null ){
        if( data == trav.data ){
            arr[ 1 ] = trav;
            return true;
        }

        arr[ 0 ] = trav;

        if( data < trav.data )
            trav = trav.left;
        else
            trav = trav.right;
    }


    arr[ 0 ] = null;
    return false;

}

public boolean deleteNode(int data){
    Node [] arr = { null, null };
```

DSA LAB EXAM

```
if( !searchNode(data, arr) )
   return false;



Node temp = arr[ 1 ];
Node parent = arr[ 0 ];



if( parent == null )
   System.out.println("node is found at root position => temp.data : "+temp.data);
else
   System.out.println("parent.data : "+parent.data+"\ttemp.data : "+temp.data);



if( temp.left != null && temp.right != null ){

   Node succ = temp.right;
   parent = temp;
   while( succ.left != null ){
      parent = succ;
      succ = succ.left;
   }

   temp.data = succ.data;

   temp = succ;
}



if( temp.left == null ){
```

```
        if( temp == root )
            root = temp.right;
        else if( temp == parent.left )
            parent.left = temp.right;
        else
            parent.right = temp.right;
    }else{
        if( temp == root )
            root = temp.left;
        else if( temp == parent.left )
            parent.left = temp.left;
        else
            parent.right = temp.left;
    }

    return true;
}

public int max( int n1, int n2){
    return ( ( n1 > n2 ) ? n1 : n2 );
}

public int height(Node trav){

    if( trav == null )
        return -1;

    return ( max( height(trav.left), height(trav.right) ) + 1 );
}
```

DSA LAB EXAM

```
public int height( ){
   if( isBSTEmpty() )
      return -1;
   else
      return ( height(root) );
}

public Node leftRotation(Node axis, Node parent){
   if( axis == null || axis.right == null )
      return null;

   Node newaxis = axis.right;
   axis.right = newaxis.left;
   newaxis.left = axis;

   if( axis == root )
      root = newaxis;
   else if( axis == parent.left )
      parent.left = newaxis;
   else
      parent.right = newaxis;

   return newaxis;
}

public Node rightRotation(Node axis, Node parent){
   if( axis == null || axis.left == null )
      return null;

   Node newaxis = axis.left;
   axis.left = newaxis.right;
```

```
    newaxis.right = axis;

    if( axis == root )
        root = newaxis;
    else if( axis == parent.left )
        parent.left = newaxis;
    else
        parent.right = newaxis;

    return newaxis;
}

public void balance(Node trav, Node parent){

    if( trav == null )
        return;

    int bf = height(trav.left) - height(trav.right);



    while( bf < -1 || bf > +1 ){
        if( bf < -1 ){
            trav = leftRotation(trav, parent);
            bf += 2;
        }else{
            trav = rightRotation(trav, parent);
            bf -= 2;
        }
    }
```

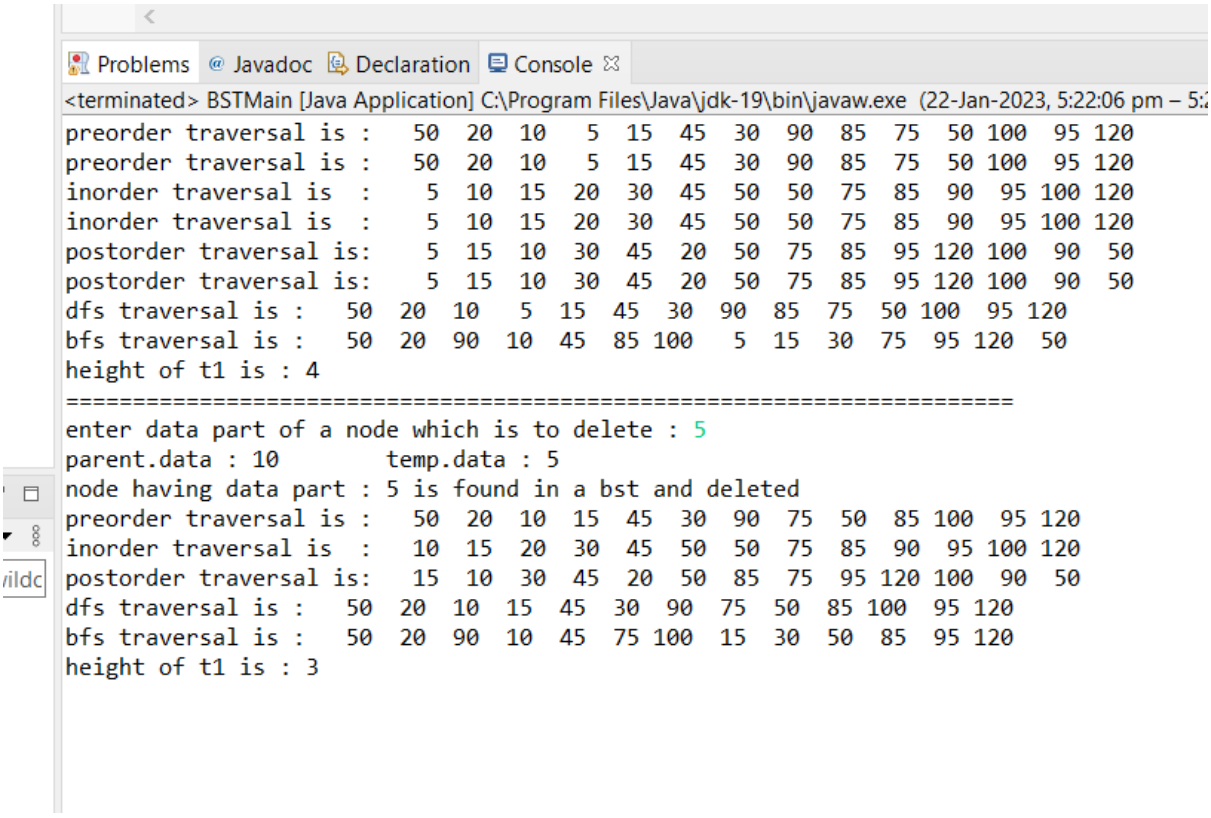
```
        balance(trav.left, trav);
        balance(trav.right, trav);

    }

    //warpper function
    public void balance( ){
        if( !isBSTEmpty( ) )
            balance(root, null);//initialization
    }
}

public class BSTMain {
    public static void main(String[] args) {
        //create an empty BST
        BST t1 = new BST();
        //Binary Search Tree => Input Order : 50 20 90 85 10 45 30 100 15 75 95 120 5 50
        /*
        t1.recAddNode(50);
        t1.recAddNode(20);
        t1.recAddNode(90);
        t1.recAddNode(85);
        t1.recAddNode(10);
        t1.recAddNode(45);
        t1.recAddNode(30);
        t1.recAddNode(100);
        t1.recAddNode(15);
        t1.recAddNode(75);
        t1.recAddNode(95);
        t1.recAddNode(120);
        t1.recAddNode(5);
```

```
t1.recAddNode(50);
*/

t1.addNode(50);
t1.addNode(20);
t1.addNode(90);
t1.addNode(85);
t1.addNode(10);
t1.addNode(45);
t1.addNode(30);
t1.addNode(100);
t1.addNode(15);
t1.addNode(75);
t1.addNode(95);
t1.addNode(120);
t1.addNode(5);
t1.addNode(50);


// t1.addNode(10);
// t1.addNode(20);
// t1.addNode(30);
// t1.addNode(40);
// t1.addNode(50);
// t1.addNode(60);
//t1.addNode(70);


t1.preOrder();
t1.nonRecPreOrder();
t1.inOrder();
```

```
        t1.nonRecInOrder();
        t1.postOrder();
        t1.nonRecPostOrder();
        t1.dfsTraversal();
        t1.bfsTraversal();
        System.out.println("height of t1 is : "+t1.height( ));


System.out.println("================================================================
======");

        //accept data part of node which is to delete
        System.out.print("enter data part of a node which is to delete : ");
        Scanner sc = new Scanner(System.in);
        int data = sc.nextInt();

        if( !t1.isBSTEmpty() ){
           if( t1.deleteNode(data) )
              System.out.println("node having data part : "+data+" is found in a bst and deleted");
           else
              System.out.println("node having data part : "+data+" is not found in a bst");
        }



        t1.balance();


        t1.preOrder();
        t1.inOrder();
        t1.postOrder();
        t1.dfsTraversal();
        t1.bfsTraversal();
        System.out.println("height of t1 is : "+t1.height( ));
```

DSA LAB EXAM

```
    }
}
```



Problems | Javadoc | Declaration | Console

<terminated> BSTMain [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (22-Jan-2023, 5:22:06 pm – 5:

```
preorder traversal is :   50  20  10   5  15  45  30  90  85  75  50 100  95 120
preorder traversal is :   50  20  10   5  15  45  30  90  85  75  50 100  95 120
inorder traversal is  :    5  10  15  20  30  45  50  50  75  85  90  95 100 120
inorder traversal is  :    5  10  15  20  30  45  50  50  75  85  90  95 100 120
postorder traversal is:    5  15  10  30  45  20  50  75  85  95 120 100  90  50
postorder traversal is:    5  15  10  30  45  20  50  75  85  95 120 100  90  50
dfs traversal is :   50  20  10   5  15  45  30  90  85  75  50 100  95 120
bfs traversal is :   50  20  90  10  45  85 100   5  15  30  75  95 120  50
height of t1 is : 4
========================================================================
enter data part of a node which is to delete : 5
parent.data : 10        temp.data : 5
node having data part : 5 is found in a bst and deleted
preorder traversal is :   50  20  10  15  45  30  90  75  50  85 100  95 120
inorder traversal is  :   10  15  20  30  45  50  50  75  85  90  95 100 120
postorder traversal is:   15  10  30  45  20  50  85  75  95 120 100  90  50
dfs traversal is :   50  20  10  15  45  30  90  75  50  85 100  95 120
bfs traversal is :   50  20  90  10  45  75 100  15  30  50  85  95 120
height of t1 is : 3
```

DSA LAB EXAM

2)

```
import java.util.Scanner;

import javax.lang.model.util.ElementScanner6;

class Stack {
	private int[] arr;
	private int top;

	Stack() {
		arr = new int[5];
		top = -1;
	}

	Stack(int size) {
		arr = new int[size];
		top = -1;
	}

	public boolean isStackFull() {
		return (top == arr.length - 1);
	}

	public boolean isStackEmpty() {
		return (top == -1);
	}

	public void pushElement(int element) {

		top++;

		arr[top] = element;
	}

	public int peekElement() {

		return (arr[top]);

	}

	public void popElement() {

		top--;

	}
}

public class StaticStackMain {
	public static int menu() {

		System.out.println("****** static stack ******");
		System.out.println("0. exit");
		System.out.println("1. push element");
		System.out.println("2. pop element");
		System.out.println("3. peek element");

		System.out.print("enter the choice : ");
```

DSA LAB EXAM

```
		Scanner sc = new Scanner(System.in);
		int choice = sc.nextInt();

		return choice;
	}

	public static void main(String[] args) {
		Stack s1 = new Stack();
		int element;

		while (true) {
			int choice = menu();
			switch (choice) {
			case 0:
				System.exit(0);

			case 1:
				if (!s1.isStackFull()) {
					System.out.print("enter an element : ");
					Scanner sc = new Scanner(System.in);
					element = sc.nextInt();
					s1.pushElement(element);
					System.out.println(element + " inserted/pushed
onto the stack....");
				} else
					System.out.println("stack overflow !!!");
				break;

			case 2:
				if (!s1.isStackEmpty()) {
					element = s1.peekElement();
					s1.popElement();
					System.out.println(element + " popped from the
the stack....");
				} else
					System.out.println("stack underflow !!!");

				break;

			case 3:
				if (!s1.isStackEmpty()) {
					element = s1.peekElement();
					System.out.println("topmost ele is : " +
element);
				} else
					System.out.println("stack underflow !!!");
				break;
			}

		}

	}
}
```

DSA LAB EXAM

```
StaticStackMain [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe  (22-Jan-2023, 5:31:39 pm)
0. exit
1. push element
2. pop element
3. peek element
enter the choice : 1
enter an element : 10
10 inserted/pushed onto the stack....
****** static stack ******
0. exit
1. push element
2. pop element
3. peek element
enter the choice : 1
enter an element : 20
20 inserted/pushed onto the stack....
****** static stack ******
0. exit
1. push element
2. pop element
3. peek element
enter the choice : 1
enter an element : 30
30 inserted/pushed onto the stack....
****** static stack ******
0. exit
1. push element
2. pop element
3. peek element
enter the choice : 2
30 popped from the the stack....
****** static stack ******
0. exit
1. push element
2. pop element
3. peek element
enter the choice : 3
topmost ele is : 20
****** static stack ******
0. exit
1. push element
2. pop element
3. peek element
enter the choice :
```