IARE

IARE
INSTITUTE OF AERONAUTICAL ENGINEERING
(An Autonomous Institute affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad - 500 043

LABORATORY WORK BOOK

Name of the Student : Racherla Santhosh

Class IT-B Semester 03

Course Code : ACSD11 Course Name : DS Laboratory

Name of the Course Faculty Ms. K. Laxminarayanamma Faculty ID : IARE10033

Exercise Number : 11 Week Number : 11 Date : 19/11/2024

Roll Number									
2	3	9	5	1	A	1	2	C	3

S. No.	Exercise Number	EXERCISE NAME	MARKS AWARDED						
			Aim/ Preparation	Algorithm / Procedure / Performance in the Lab		Source Code / Calculations and Graphs	Program Execution / Results and Error Analysis	Viva - Voce	Total
			4	4		4	4	4	20
1	11.1	Insertion in an AVL Tree	✓	✓	✓	✓	✓	✓	20
2	11.2	Deletion in an AVL Tree							
3	11.3	Count Greater Node in AVL							
4	11.4	Minimum no. of Nodes in an AVL with given height							
5									
6									
7									
8									
9									
10									
11									
12									

Santhosh

Signature of the Student

Signature of the Faculty

11. AVL Tree.

11.1 Insertion In An AVL Tree :-

AIM :- Write a Program on Insertion in an AVL Tree.

PROGRAM :-

```
import java.util.Scanner;
class TreeNode {
    int val, height;
    TreeNode left, right;
    TreeNode(int d) {
        val = d;
        height = 1;
    }
}
class AVL_Tree {
    int height(TreeNode node) {
        if (node == null)
            return 0;
        return node.height;
    }
```

```
int max (int a, int b) {
    return (a > b) ? a : b;
}

TreeNode rightRotate (TreeNode z) {
    TreeNode y = z.left;
    TreeNode T2 = y.right;
    y.right = z;
    z.left = T2;
    z.height = max (height (z.left), height (z.right)) + 1;
    y.height = max (height (y.left), height (y.right)) + 1;
    return y;
}

TreeNode leftRotate (TreeNode z) {
    TreeNode y = z.right;
    TreeNode T2 = y.left;
    y.left = z;
    z.right = T2;
    z.height = max (height (z.left), height (z.right)) + 1;
    y.height = max (height (y.left), height (y.right)) + 1;
```

```
    return y;
}

int getBalance ( TreeNode node) {
  if (node == null)
    return 0;
  return height (node.left) - height (node.right);
}

TreeNode insert ( TreeNode node, int Key) {
  if (node == null)
    return new TreeNode (Key);
  if (Key < node.val)
    node.left = insert (node.left, Key);
  else if (Key > node.val)
    node.right = insert (node.right, Key);
  else
    return node;
  node.height = 1 + max (height (node.left),
                         height (node.right));
```

```
int balance = getBalance(node);
if (balance > 1 && key < node.left.val)
  return rightRotate(node);
if (balance < -1 && key > node.right.val)
  return leftRotate(node);
if (balance > 1 && key > node.left.val) {
  node.left = leftRotate(node.left);
  return rightRotate(node);
}
return node;
}
public static void main(String[] args) {
  AVL_Tree tree = new AVL_Tree();
  TreeNode root = null;
  Scanner scanner = new Scanner(System.in);
  while (true) {
    System.out.println("\n choose an Operation: ");
    System.out.println("1. Insert a Node");
    System.out.println("2. Display Preorder Traversal");
```

```
System.out.println("3. Exit");
int choice = Scanner.nextInt();
Switch (choice) {
 case 1:
  System.out.print("Enter the value to insert:");
  int insertValue = Scanner.nextInt();
  root = tree.insert(root, insertValue);
  System.out.println("Inserted" + insertValue);
  break;
case 2:
  System.out.println("preorder traversal of AVL tree:");
  tree.preOrder(root);
  System.out.println();
  break;
case 3:
  System.out.println("Exiting....");
  Scanner.close();
  System.exit(0);
default:
```

```
            System.out.println("Invalid choice. please try again");
        }
    }
}
}
```

OUTPUT :-

```
Choose an Operation :
1. Insert a node
2. Display Preorder traversal
3. Exit
1
Enter the value to insert : 30
Inserted 30
Enter the value to insert : 20
Inserted 20
Enter the value to insert : 40
Inserted 40
Enter the value to insert : 10
Inserted 10
Preorder traversal Of the AVL tree :
30 20 10 40
```

```
Choose an Operation : 3
Exiting ...
```

11.2 **Deletion In An AVL Tree :-**

AIM :- Write a Program on Deletion in an AVL Tree.

PROGRAM :-

```
import java.util.Scanner;
class TreeNode {
  int val, height;
  TreeNode left, right;
  TreeNode(int d) {
    val = d;
    height = 1; } }
class AVL_Tree {
  int height(TreeNode node) {
    if( node == null)
      return 0;
    return node.height;
}
```

```
int max ( int a, int b) {
  return (a > b) ? a : b ; }
TreeNode RightRotate ( TreeNode z) {
  TreeNode y = z.left;
  TreeNode T2 = y.right;
  y.right = z;
  z.left = T2;
  z.height = max( height (z.left), height (z.right)) + 1;
  y.height = max ( height (y.left), height ( y.right)) + 1;
  return y;
}
TreeNode LeftRotate (TreeNode z) {
  TreeNode y = z.right;
  TreeNode T2 = y.left;
  y.left = z;
  z.right = T2;
  z.height = max (height (z.left), height (z.right)) + 1;
  y.height = max (height (y.left), height (y.right)) + 1;
  return y;
}
```

```
TreeNode insert ( TreeNode node, int key) {
  if (node == null)
    return new TreeNode (key);
  if (key < node.val)
    node.left = insert (node.left, key);
  else if (key > node.val)
    node.right = insert (node.right, key);
  else
    return node;
node.height = 1 + max (height (node.left), height (node.right));
TreeNode deleteNode ( TreeNode root, int key) {
  if (root == null)
    return root;
  if (key < root.val)
    root.left = deleteNode (root.left, key);
  else if ( key > root.val)
    root.right = deleteNode (root.right, key);
  else {
```

```
if ((root.left == null) || (root.right == null)) {
    TreeNode temp = (root.left != null) ? root.left : root.right;
    if (temp == null) {
        temp = root;
        root = null;
    } else
        root = temp;
} else {
    TreeNode temp = minValueNode(root.right);
    root.val = temp.val;
    root.right = deleteNode(root.right, temp.val); } }
TreeNode minValueNode(TreeNode node) {
    TreeNode current = node;
    while (current.left != null)
        current = current.left;
    return current;
}

Public Static void main(String[] args) {
    AVL_Tree tree = new AVL_Tree();
    TreeNode root = null;
```

```
Scanner Scanner = new Scanner (System.in);
while (true) {
 System.out.println ("\n choose an operation :");
 System.out.println ("1. Insert a Node");
 System.out.println ("2. Delete a Node");
 System.out.println ("3. Display Preorder traversal");
 System.out.println ("4. Exit");
 int choice = Scanner.nextInt();
 Switch (choice) {
  case 1 :
   System.out.print (" Enter the value to insert :");
   int insertValue = Scanner.nextInt();
   root = tree.insert (root, insertValue);
   System.out.println (" Inserted" + insertValue);
   break;
  case 2 :
   System.out.print (" Enter the Value to delete: ");
   int deleteValue = Scanner.nextInt();
   root = tree.deleteNode (root, deleteValue);
```

```
        System.out.println("Deleted " + deleteValue);
        break;
    case 3:
        System.out.println("Preorder traversal of AVL tree:");
        tree.preOrder(root);
        System.out.println();
        break;
    case 4:
        System.out.println("Exiting...");
        scanner.close();
        System.exit(0);
    default:
        System.out.println("Invalid choice. Please
                            Try Again");
    }
  }
}
```

OUTPUT :-

```
choose an operation:
1. Insert a node
2. Delete a node
```

3. Display Preorder Traversal

4. Exit

1

Enter the value to insert : 10.

Inserted 10

Enter the value to insert : 20

Inserted 20

Enter the value to insert : 30

Inserted 30

Enter the value to insert : 40

Inserted 40

3

Preorder traversal of the AVL tree : 20 10 30 40

2

Enter the value to delete : 10

Deleted 10

3

Preorder traversal of the AVL tree : 20 30 40

4

Exiting ...

11.3 Count Greatest Nodes in AVL Tree :-

AIM :- Write a program to calculate number of elements which are greater than given value in AVL.

PROGRAM :-

```
import java.util.Scanner;
class TreeNode {
  int Key, height, desc;
  TreeNode left, right;
  TreeNode (int d) {
    Key = d;
    height = 1;
    desc = 0; } }
class AVL-Tree {
  int height (TreeNode node) {
    return (node == null) ? 0 : node.height; }
  int max (int a, int b) {
    return (a > b) ? a : b; }
  TreeNode insert (TreeNode node, int key) {
```

```
if (node == null)
    return new TreeNode(key);
if (key < node.key)
    node.left = insert(node.left, key);
else if (key > node.key)
    node.right = insert(node.right, key);
else
    return node;
if (balance > 1 && key < node.left.key)
    return rightRotate(node);
if (balance < -1 && key > node.right.key)
    return leftRotate(node);
if (balance > 1 && key > node.left.key) {
    node.left = leftRotate(node.left);
    return rightRotate(node);
}
if (balance < -1 && key < node.right.key) {
    node.right = rightRotate(node.right);
    return leftRotate(node);
}
return node;
}
```

```
Public Static void main (String[] args) {
    Scanner sc = new Scanner (System.in);
    AVL_Tree tree = new AVL_Tree();
    TreeNode root = null;
    System.out.println("Enter the number of elements to
                        insert into the AVL Tree:");
    int n = sc.nextInt();
    System.out.println("Enter the elements:");
    for (int i=0; i<n; i++) {
        int el = sc.nextInt();
        root = tree.insert(root, el); }
    System.out.println("Enter the value of x to count the number
                        of elements greater than it:");
    int x = sc.nextInt();
    System.out.println("Preorder traversal of the constructed
                        AVL Tree is:");
    tree.preorder(root)
    System.out.println();
    System.out.println("No. of elements greater than "+x+":"+
                        tree.countGreater(root, x));
```

```
        sc.close();
    }
}
```

OUTPUT :-

Enter the number of elements to insert into the AVL Tree : 4

Enter the elements : 10 20 30 40

Enter the value of x to count the number of elements greater than it : 25

PreOrder traversal of the constructed AVL tree is :

20 10 30 40

Number of elements greater than 25 : 2.

11.4 Minimum Number Of Nodes in an AVL Tree with Given Height :-

AIM :- Write a program to find the minimum number of nodes the tree can have.

PROGRAM :-

```
import java.util.Scanner;
Public class AVLTree_MinimumNodes {
```

```
Public static int AVLnodes (int height) {
  if (height == 0) {
    return 1;
  } else if (height == 1) {
    return 2;
  }
  return 1 + AVLnodes (height - 1) + AVLnodes (height - 2);
}
Public static void main (String [] args) {
  Scanner sc = new Scanner (System. in);
  System. out. print (" Enter the height of the AVL tree:");
  int H = sc. nextInt();
  System. out. println ("The minimum number of nodes in the
  AVL tree with height "+ H + " is: "+ AVLnodes (H));
  sc. close();
  }
}
```

OUTPUT:-

Enter the height of the AVL tree : 0

The minimum number of nodes in AVL tree is : 1

VIVA VOCE :-

1) What is an AVL Tree ?

A) An AVL Tree is a Self-balancing Binary Search Tree where the height difference between left & right Subtrees atmost 1.

2) What are the two basic operations used for rebalancing an AVL tree ?

A) Left Rotation & Right Rotation

3) How can the number of elements greater than a given value be counted in an AVL tree ?

A) By traversing the tree and counting the nodes with values greater than the given value using the Subtree size & balance conditions.

4) What is the time complexity for inserting a node in an AVL tree ?

A) The time complexity for insertion in an AVL tree is O(log n).