DSA LAB ASSIGNMENT-5

21BCE7371

RADHA KRISHNA GARG

INPUT

Code:

```
class Node {
    int item, height;
    Node left, right;

    Node(int d) {
      item = d;
      height = 1;
    }
  }

  class AVLTree {
    Node root;

    int height(Node N) {
      if (N == null)
        return 0;
      return N.height;
    }

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

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

    Node leftRotate(Node x) {
      Node y = x.right;
      Node T2 = y.left;
      y.left = x;
```

```
    x.right = T2;
    x.height = max(height(x.left), height(x.right)) + 1;
    y.height = max(height(y.left), height(y.right)) + 1;
    return y;
  }
  int getBalanceFactor(Node N) {
    if (N == null)
      return 0;
    return height(N.left) - height(N.right);
  }


  Node insertNode(Node node, int item) {


    if (node == null)
      return (new Node(item));
    if (item < node.item)
      node.left = insertNode(node.left, item);
    else if (item > node.item)
      node.right = insertNode(node.right, item);
    else
      return node;

    node.height = 1 + max(height(node.left), height(node.right));
    int balanceFactor = getBalanceFactor(node);
    if (balanceFactor > 1) {
      if (item < node.left.item) {
        return rightRotate(node);
      } else if (item > node.left.item) {
        node.left = leftRotate(node.left);
        return rightRotate(node);
      }
    }
    if (balanceFactor < -1) {
      if (item > node.right.item) {
        return leftRotate(node);
      } else if (item < node.right.item) {
        node.right = rightRotate(node.right);
        return leftRotate(node);
      }
    }
    return node;
  }

  Node nodeWithMimumValue(Node node) {
    Node current = node;
    while (current.left != null)
```

```
      current = current.left;
    return current;
  }

  Node deleteNode(Node root, int item) {

    if (root == null)
      return root;
    if (item < root.item)
      root.left = deleteNode(root.left, item);
    else if (item > root.item)
      root.right = deleteNode(root.right, item);
    else {
      if ((root.left == null) || (root.right == null)) {
        Node temp = null;
        if (temp == root.left)
          temp = root.right;
        else
          temp = root.left;
        if (temp == null) {
          temp = root;
          root = null;
        } else
          root = temp;
      } else {
        Node temp = nodeWithMimumValue(root.right);
        root.item = temp.item;
        root.right = deleteNode(root.right, temp.item);
      }
    }
    if (root == null)
      return root;

    root.height = max(height(root.left), height(root.right)) + 1;
    int balanceFactor = getBalanceFactor(root);
    if (balanceFactor > 1) {
      if (getBalanceFactor(root.left) >= 0) {
        return rightRotate(root);
      } else {
        root.left = leftRotate(root.left);
        return rightRotate(root);
      }
    }
    if (balanceFactor < -1) {
      if (getBalanceFactor(root.right) <= 0) {
        return leftRotate(root);
      } else {
        root.right = rightRotate(root.right);
```

```
        return leftRotate(root);
      }
    }
    return root;
  }

  void preOrder(Node node) {
    if (node != null) {
      System.out.print(node.item + " ");
      preOrder(node.left);
      preOrder(node.right);
    }
  }

  private void printTree(Node currPtr, String indent, boolean last) {
    if (currPtr != null) {
      System.out.print(indent);
      if (last) {
        System.out.print("R----");
        indent += "    ";
      } else {
        System.out.print("L----");
        indent += "|   ";
      }
      System.out.println(currPtr.item);
      printTree(currPtr.left, indent, false);
      printTree(currPtr.right, indent, true);
    }
  }

  public static void main(String[] args) {
    AVLTree tree = new AVLTree();
    tree.root = tree.insertNode(tree.root,3 );
    tree.root = tree.insertNode(tree.root, 2);
    tree.root = tree.insertNode(tree.root, 1);
    tree.root = tree.insertNode(tree.root, 4);
    tree.root = tree.insertNode(tree.root, 5);
    tree.root = tree.insertNode(tree.root, 6);
    tree.root = tree.insertNode(tree.root, 7);
    tree.root = tree.insertNode(tree.root, 8);
    tree.root = tree.insertNode(tree.root, 9);
    System.out.println("AVL tree: ");
    tree.printTree(tree.root, "", true);
  }}
```

OUTPUT

```
        R----9
PS C:\Users\krish\Documents\java>  c:; cd 'c:\Users\krish\Documents\java'; & 'C:\Program File
InExceptionMessages' '-cp' 'C:\Users\krish\AppData\Roaming\Code\User\workspaceStorage\2c70a5e
0e910\bin' 'AVLTree'
AVL tree:
R----4
   L----2
   |  L----1
   |  R----3
   R----6
      L----5
      R----8
         L----7
         R----9
PS C:\Users\krish\Documents\java>
```