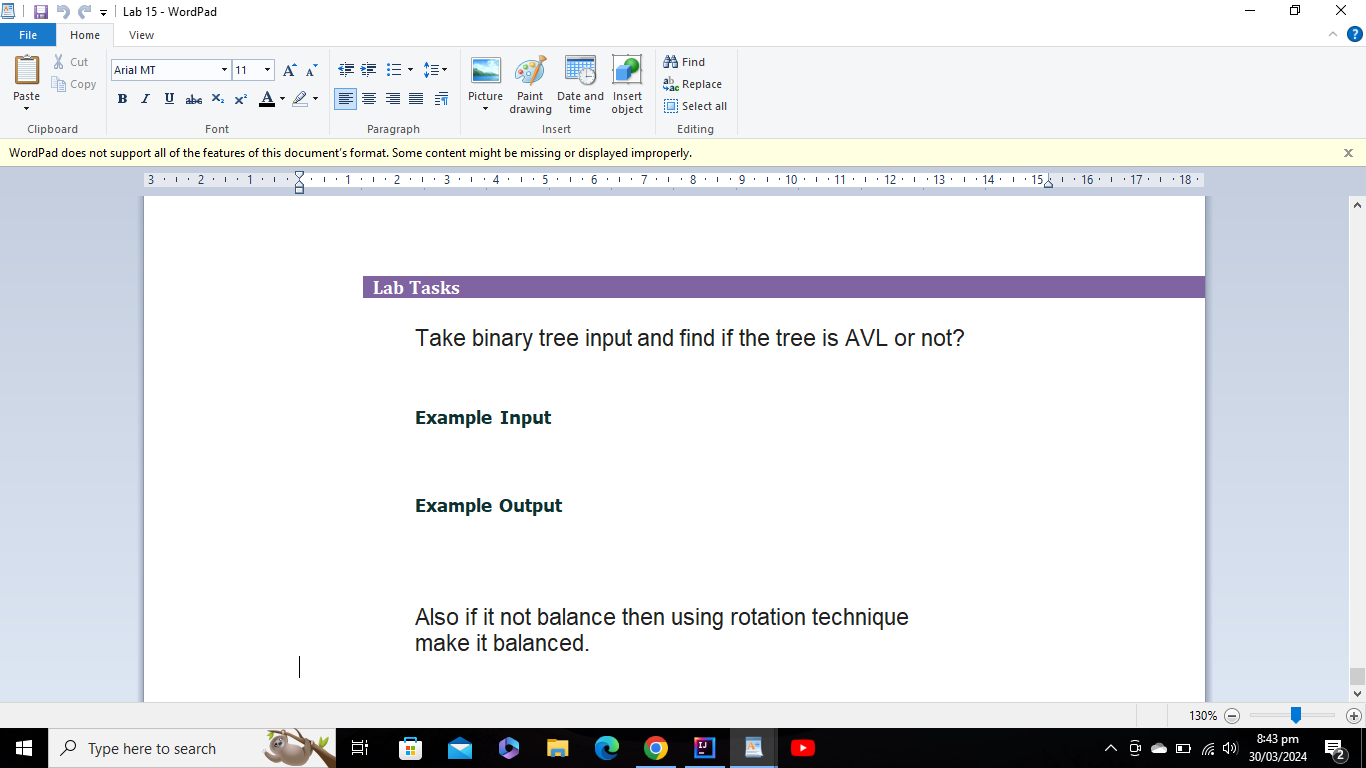
|  |  |
| --- | --- |
| ***Roll: No*** | ***22SW040*** |
| ***Section*** | ***01*** |
| ***DSA\_Lab*** | ***15(AVL Tree)*** |
| ***Teacher*** | ***Ma’am Muskan*** |

***Task***



***Code***

*import* java.util.Scanner;  
  
*public class* AVLTree {  
 *int* data;  
 AVLTree left,right;  
 *int* height = 0;  
 *private* AVLTree(){  
 left=right=*NIL*;  
 height=-1;  
 }  
  
 *public* AVLTree(*int* data) {  
 *this*.data = data;  
 left = right = *NIL*;  
 height=0;  
 }  
  
 *static* AVLTree *NIL* = *new* AVLTree();  
 *static* {  
 *NIL*.left = *NIL*.right = *NIL*;  
 *NIL*.height = -1;  
 }  
  
  
  
  
 *private int* max(*int* a, *int* b) {  
 *return* Math.*max*(a, b);  
 }  
  
 *private void* updateHeight(AVLTree avlTree) {  
 *int* leftChildHeight = height(avlTree.left);  
 *int* rightChildHeight = height(avlTree.right);  
 avlTree.height = max(leftChildHeight, rightChildHeight) + 1;  
 }  
  
 *private int* balanceFactor(AVLTree avlTree) {  
 *return* height(avlTree.right) - height(avlTree.left);  
 }  
  
 *private* AVLTree rotateRight(AVLTree avlTree) {  
 AVLTree leftChild = avlTree.left;  
 avlTree.left = leftChild.right;  
 leftChild.right = avlTree;  
 updateHeight(avlTree);  
 updateHeight(leftChild);  
 *return* leftChild;  
 }  
  
 *private* AVLTree rotateLeft(AVLTree avlTree) {  
 AVLTree rightChild = avlTree.right;  
 avlTree.right = rightChild.left;  
 rightChild.left = avlTree;  
 updateHeight(avlTree);  
 updateHeight(rightChild);  
 *return* rightChild;  
 }  
  
 *private* AVLTree reBalance(AVLTree avlTree) {  
 *int* balanceFactor = balanceFactor(avlTree);  
 *if* (balanceFactor < -1) {  
 *if* (balanceFactor(avlTree.left) <= 0) {  
 avlTree = rotateRight(avlTree);  
 } *else* {  
 avlTree.left = rotateLeft(avlTree.left);  
 avlTree = rotateRight(avlTree);  
 }  
 }  
  
 *if* (balanceFactor > 1) {  
 *if* (balanceFactor(avlTree.right) >= 0) {  
 avlTree = rotateLeft(avlTree);  
 } *else* {  
 avlTree.right = rotateRight(avlTree.right);  
 avlTree = rotateLeft(avlTree);  
 }  
 }  
 *return* avlTree;  
 }  
  
 *public boolean* isAVL(AVLTree avlTree) {  
 *if* (avlTree == *NIL*) {  
 *return true*;  
 }  
  
 *int* leftHeight = height(avlTree.left);  
 *int* rightHeight = height(avlTree.right);  
  
 *if* (Math.*abs*(leftHeight - rightHeight) > 1) {  
 *return false*;  
 }  
  
 *return* isAVL(avlTree.left) && isAVL(avlTree.right);  
 }  
  
 *private int* height(AVLTree avlTree) {  
 *if* (avlTree == *NIL*) {  
 *return* -1;  
 }  
 *return* avlTree.height;  
 }  
  
  
 *public int* size(){  
 *if*(*this*==*NIL*) *return* 0;  
 *return* (1+ left.size()+ right.size());  
 }  
 *public boolean* add(*int* key) {  
 AVLTree newNode = *new* AVLTree(key);  
 *if* (*this* == *NIL*) {  
 *this*.data = key;  
 *this*.left = *NIL*;  
 *this*.right = *NIL*;  
 *this*.height = 0;  
 *return true*;  
 }  
  
 AVLTree currentNode = *this*;  
 AVLTree parentNode = *null*;  
 *while* (currentNode != *NIL*) {  
 parentNode = currentNode;  
 *if* (key < currentNode.data) {  
 currentNode = currentNode.left;  
 } *else if* (key > currentNode.data) {  
 currentNode = currentNode.right;  
 } *else* {  
 *return false*;  
 }  
 }  
  
 *if* (key < parentNode.data) {  
 parentNode.left = newNode;  
 } *else* {  
 parentNode.right = newNode;  
 }  
 newNode.height = Math.*max*(height(newNode.left), height(newNode.right)) + 1;  
 *return true*;  
 }  
  
 *public* AVLTree grow(*int* key){  
 *if*(*this*==*NIL*) *return new* AVLTree(key);  
 *if*(*this*.data==key) *return this*;  
 *if*(key>*this*.data) right=right.grow(key);  
 *if*(key<*this*.data) left = left.grow(key);  
 height = 1+Math.*max*(left.height ,right.height);  
 *return this*;  
 }  
  
 *public* AVLTree balanceTree(AVLTree avlTree) {  
 *if* (avlTree == *NIL*) {  
 *return NIL*;  
 }  
  
 avlTree.left = balanceTree(avlTree.left);  
 avlTree.right = balanceTree(avlTree.right);  
  
 updateHeight(avlTree);  
 avlTree = reBalance(avlTree);  
  
 *return* avlTree;  
 }  
  
 *public void* printInOrder(AVLTree avlTree) {  
 *if* (avlTree == *NIL*) {  
 *return*;  
 }  
  
 printInOrder(avlTree.left);  
 System.***out***.print(avlTree.data + " ");  
 printInOrder(avlTree.right);  
 }  
 *public void* printPreOrder(AVLTree avlTree) {  
 *if* (avlTree == *NIL*) {  
 *return*;  
 }  
  
 System.***out***.print(avlTree.data + " ");  
 printInOrder(avlTree.left);  
 printInOrder(avlTree.right);  
 }  
 *public void* printPostOrder(AVLTree avlTree) {  
 *if* (avlTree == *NIL*) {  
 *return*;  
 }  
  
 printInOrder(avlTree.left);  
 printInOrder(avlTree.right);  
 System.***out***.print(avlTree.data + " ");  
 }  
  
  
 *public static void* main(String[] args) {  
 AVLTree avlTree = *new* AVLTree();  
 System.***out***.print("Enter Size of the tree : ");  
 Scanner sc = *new* Scanner(System.***in***);  
 *int* size = sc.nextInt();  
 System.***out***.println("Enter Elements in the tree --> ");  
 *for*(*int* i=0;i<size;i++) {  
 *int* element = sc.nextInt();  
 avlTree.add(element);  
 }  
 System.***out***.println("Printing the Avl tree : " );  
 System.***out***.print("In order : ");  
 avlTree.printInOrder(avlTree);  
 System.***out***.println();  
 System.***out***.print("post order : ");  
 avlTree.printPostOrder(avlTree);  
 System.***out***.println();  
 System.***out***.print("pre order : ");  
 avlTree.printPreOrder(avlTree);  
 System.***out***.println();  
 *boolean* result = avlTree.isAVL(avlTree);  
  
 *if*(result){  
 System.***out***.println("Tree is Balance/AVL tree ");  
 }  
 *else*{  
 System.***out***.println("The tree is not Balance");  
 avlTree = avlTree.balanceTree(avlTree);  
 System.***out***.println("Now Our tree is Balance ");  
 System.***out***.print("In order : ");  
 avlTree.printInOrder(avlTree);  
 System.***out***.println();  
 System.***out***.print("post order : ");  
 avlTree.printPostOrder(avlTree);  
 System.***out***.println();  
 System.***out***.print("pre order : ");  
 avlTree.printPreOrder(avlTree);  
 System.***out***.println();  
 }  
  
  
  
 }  
 }

***Output***



