Assignment

Name:

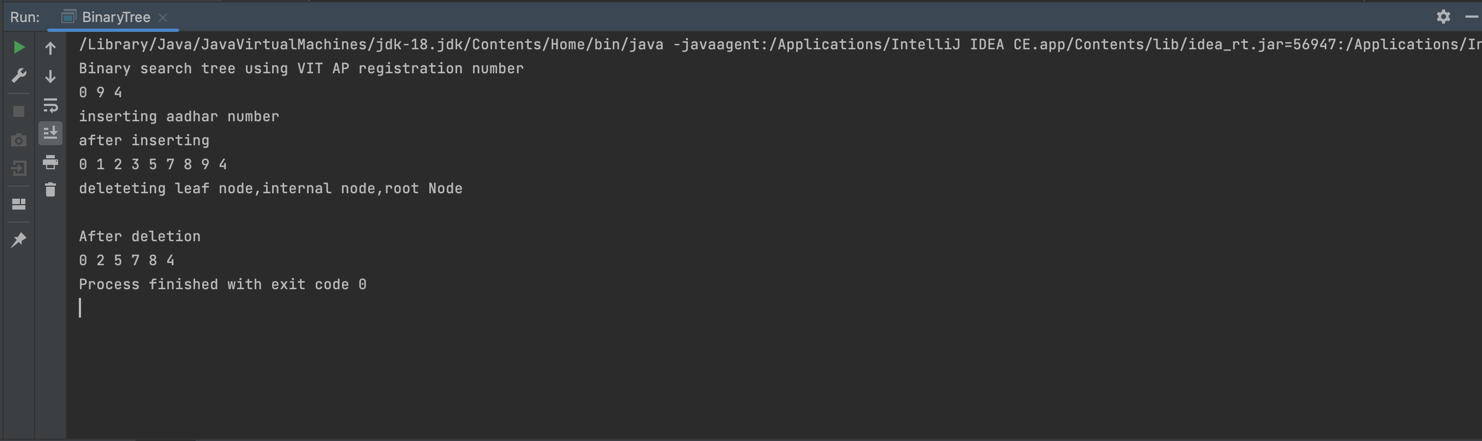
Slot:

Reg No:

Code:

class Binary {  
 Node root;  
 static class Node {  
 int data;  
 Node left;  
 Node right;  
 public Node(int data){  
 this.data = data;  
 left = right = null;  
 }  
 }  
 Binary(){  
 root = null;  
 }  
 void deleteKey(int key) {  
 root = delete\_Recursive(root, key);  
 }  
 Node delete\_Recursive(Node root, int key) {  
 if (root == null) return root;  
 if (key < root.data)  
 root.left = delete\_Recursive(root.left, key);  
 else if (key > root.data)  
 root.right = delete\_Recursive(root.right, key);  
 else {  
 if (root.left == null)  
 return root.right;  
 else if (root.right == null)  
 return root.left;  
  
 root.data = minValue(root.right);  
 root.right = delete\_Recursive(root.right, root.data);  
 }  
 return root;  
 }  
  
  
 int minValue(Node root) {  
 int minval = root.data;  
 while (root.left != null) {  
 minval = root.left.data;  
 root = root.left;  
 }  
 return minval;  
 }  
  
 int maxValue(Node root) {  
 int maxval = root.data;  
 while (root.right != null) {  
 maxval = root.right.data;  
 root = root.right;  
 }  
 return maxval;  
 }  
  
 void insert(int key) {  
 root = insert(root, key);  
 }  
 Node insert(Node root, int key) {  
 if (root == null) {  
 root = new Node(key);  
 return root;  
 }  
 if (key < root.data)  
 root.left = insert(root.left, key);  
 else if (key > root.data)  
 root.right = insert(root.right, key);  
 return root;  
 }  
  
  
  
  
 void inorder() {  
 inorder(root);  
 }  
 void inorder(Node root) {  
 if (root != null) {  
 inorder(root.left);  
 System.*out*.print(root.data + " ");  
 inorder(root.right);  
 }  
 }  
 void createTree(){  
 Node first=new Node(9);  
 Node second=new Node(0);  
 Node third=new Node(4);  
 root=first;  
 root.left=second;  
 root.right=third;  
  
 }  
  
 boolean search(int key) {  
 root = search\_Recursive(root, key);  
 if(root!= null)  
 return true;  
 else  
 return false;  
 }  
 Node search\_Recursive(Node root, int key) {  
 if (root==null || root.data==key)  
 return root;  
 if (root.data > key)  
 return search\_Recursive(root.left, key);  
 return search\_Recursive(root.right, key);  
 }  
}  
public class BinaryTree{  
 public static void main(String[] args) {  
 Binary bst = new Binary();  
 System.*out*.println("Binary search tree using VIT AP registration number");  
 bst.createTree();  
 bst.inorder();  
 System.*out*.println("\ninserting aadhar number");  
 bst.insert(7);  
 bst.insert(1);  
 bst.insert(8);  
 bst.insert(5);  
 bst.insert(2);  
 bst.insert(3);  
 System.*out*.println("after inserting ");  
 bst.inorder();  
 System.*out*.println("\ndeleteting leaf node,internal node,root Node");  
 bst.deleteKey(3);  
 bst.deleteKey(1);  
 bst.deleteKey(9);  
 System.*out*.println("\nAfter deletion");  
 bst.inorder();  
  
  
  
  
 }  
}

Output:



/Library/Java/JavaVirtualMachines/jdk-18.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea\_rt.jar=56947:/Applications/IntelliJ IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath /IdeaProjects/Assignment/out/production/Assignment BinaryTree

Binary search tree using VIT AP registration number

0 9 4

inserting Aadhar number

after inserting

0 1 2 3 5 7 8 9 4

delete leaf node, internal node, root Node

After deletion

0 2 5 7 8 4

Process finished with exit code 0