```
package com.jinglin;

import java.io.*;
import java.util.Scanner;

public class Main{
    public static void main(String[] args) throws IOException {
        File treeInput = new File("c:/treeInput.txt");
        Scanner sc = new Scanner(treeInput);
        int i = 1;

        while(sc.hasNext()){
            System.out.println("Set#" + i++);
            setOperation(new Tree(), sc);
            System.out.println("============================================");
        }
    }

    public static void setOperation(Tree tree, Scanner sc){
        tree.fillTree(sc);
        setPrint(tree, sc);
        if(tree.insertOrDelete(sc)) setPrint(tree, sc);
        tree.freeTree();
    }

    public static void setPrint(Tree tree, Scanner sc){
        tree.inOrderPrint();
        System.out.println();
        tree.preOrderPrint();
        System.out.println();
        tree.postOrderPrint();
        System.out.println();
        System.out.println("The number of nodes in the tree is " + tree.countNodes());
        System.out.println("The number of children in the tree is " + tree.countChildren());
        tree.showChildren();
        System.out.println();
    }

}
```

```
package com.jinglin;

import java.util.Scanner;

public class Tree {
    TreeNode root;
    private int countChildren;
    private int countNodes;

    public TreeNode getRoot() {
        return root;
    }

    public void fillTree(Scanner sc){
        int num;
        num = sc.nextInt();
        while(num != -999){
            insertInt(num);
            num = sc.nextInt();
        }
    }

    private void insertInt(int value){
        if(root == null) root = new TreeNode(value);
        else insert(value, root);
    }

    private void insert(int value, TreeNode root){
        if(value == root.getNum()) return;
        if(value < root.getNum()) {
            if (root.getLeft() == null) root.setLeft(new TreeNode(value));
            else insert(value, root.getLeft());
        }else{
            if(root.getRight() == null) root.setRight(new TreeNode(value));
            else insert(value, root.getRight());
        }
    }

    public void inOrderPrint(){inOrderPrint(root);}

    private void inOrderPrint(TreeNode root){
        if(root == null) return;
        inOrderPrint(root.getLeft());
        System.out.print(root.getNum() + " ");
        inOrderPrint(root.getRight());
    }

    public void preOrderPrint(){preOrderPrint(root);}

    private void preOrderPrint(TreeNode root){
        if(root == null) return;
        System.out.print(root.getNum() + " ");
        preOrderPrint(root.getLeft());
        preOrderPrint(root.getRight());
    }

    public void postOrderPrint(){postOrderPrint(root);}

    private void postOrderPrint(TreeNode root){
        if(root == null) return;
        postOrderPrint(root.getLeft());
        postOrderPrint(root.getRight());
        System.out.print(root.getNum() + " ");
    }
```

```
public int countChildren(){
    countChildren = 0;
    countChildren(root);
    return countChildren;
}

private void countChildren(TreeNode root) {
    if (root == null) return;
    countChildren(root.getLeft());
    if(root.getLeft() != null) countChildren++;
    if(root.getRight() != null) countChildren++;
    countChildren(root.getRight());
}

public void showChildren(){
    System.out.println("Number of children each node has:");
    showChildren(root);
    System.out.println();
}

private void showChildren(TreeNode root){
    if(root == null) return;
    showChildren(root.getLeft());
    int children = 0;
    if(root.getLeft() != null) children++;
    if(root.getRight() != null) children++;
    System.out.print(root.getNum() + ":" + children + " ");
    showChildren(root.getRight());
}

public int countNodes(){
    countNodes = 0;
    if(root != null) {
        countNodes++;
        countNodes(root);
    }
    return countNodes;
}

private void countNodes(TreeNode root) {
    if (root == null) return;
    countNodes(root.getLeft());
    if(root.getLeft() != null) countNodes++;
    if(root.getRight() != null) countNodes++;
    countNodes(root.getRight());
}

private void deleteInt(int value){
    root = delete(value, root);
}

private TreeNode delete(int value, TreeNode root){
    if(root == null) return root;
    if(value < root.getNum()) root.setLeft(delete(value, root.getLeft()));
    else if(value > root.getNum()) root.setRight(delete(value, root.getRight()));
    else{

        if(root.getLeft() != null && root.getRight() != null){
            TreeNode p = root.getLeft();
            TreeNode q = root;
            while(p.getRight() != null){
                q = p;
                p = p.getRight();
```

```
                }
                if(q == root) p.setRight(root.getRight());
                else if(p.getLeft() == null){
                    q.setRight(null);
                    p.setLeft(root.getLeft());
                    p.setRight(root.getRight());
                }else if(p.getLeft() != null){
                    q.setRight(p.getLeft());
                    p.setLeft(root.getLeft());
                    p.setRight(root.getRight());
                }
                return p;
            }

            else if(root.getLeft() == null) return root.getRight();
            else if(root.getRight() == null) return root.getLeft();
        }
        return root;
    }

    public boolean insertOrDelete(Scanner sc){
        String routine;
        int num;
        routine = sc.next();
        if(routine.equals("-999")) return false;
        while(!routine.equals("-999")){
            if(routine.equals("Delete")){
                num = sc.nextInt();
                deleteInt(num);
            }else if(routine.equals("Insert")){
                num = sc.nextInt();
                insertInt(num);
            }
            routine = sc.next();
        }
        return true;
    }

    public void freeTree(){
        freeTree(root);
        root = null;
        System.out.println("Tree has been freed");
    }

    private void freeTree(TreeNode root){
        if(root == null) return;
        freeTree(root.getLeft());
        freeTree(root.getRight());
        root.setLeft(null);
        root.setRight(null);
    }
}
```

```java
package com.jinglin;

public class TreeNode {
    private int num;
    private TreeNode left;
    private TreeNode right;

    public TreeNode(int num) {
        this.num = num;
    }

    public int getNum() {
        return num;
    }

    public TreeNode getLeft() {
        return left;
    }

    public void setLeft(TreeNode left) {
        this.left = left;
    }

    public TreeNode getRight() {
        return right;
    }

    public void setRight(TreeNode right) {
        this.right = right;
    }
}
```