import java.util.LinkedList;  
import java.util.Queue;  
import java.util.Scanner;  
  
public class Prac {  
 static class Node{  
 int data;  
 Node left;  
 Node right;  
  
 Node(int data){  
 this.data = data;  
 }  
 }  
 public static Node insert(Node root, int val) {  
 if(root == null) {  
 root = new Node (val);  
 return root;  
 }  
 if(root.data>val) {  
 root.left = *insert*(root.left, val);  
  
 }  
 else {  
 root.right = *insert*(root.right, val);  
 }  
 return root;  
 }  
  
 public static void inorder(Node root){  
 if(root == null){  
 return ;  
 }  
 *inorder*(root.left);  
 System.*out*.print(root.data+ " ");  
 *inorder*(root.right);  
 }  
  
 public static boolean search(Node root, int key){  
 if(root == null){  
 return false;  
 }  
 if(root.data > key){ //let subtree  
 return *search*(root.left,key);  
 }  
 else if(root.data < key){  
 return *search*(root.right, key);  
 }  
 else if(root.data == key){  
 return true;  
 }  
 return false;  
 }  
  
 public static Node delete(Node root,int val) {  
 if(root.data > val) {  
 root.left = *delete*(root.left, val);  
 }  
  
 else if(root.data < val) {  
 root.right = *delete*(root.right, val);  
 }  
  
 else{ //root.data == val  
 // case 1  
 if(root.left == null && root.right == null){  
 return null;  
 }  
 // case 2  
 else if(root.left == null){  
 return root.right;  
 }  
 else if(root.right == null){  
 return root.left;  
 }  
 // case 3  
 else {  
 root.data = *minValue*(root.right);  
 root.right = *delete*(root.right, root.data );  
 }  
 }  
 return root;  
 }  
 static int minValue(Node root) {  
 int minval = root.data;  
 while(root.left!=null) {  
 minval = root.left.data;  
 root = root.left;  
  
 }  
 return minval;  
 }  
  
 public static void printLevelOrder(Node data)  
 {  
 Queue<Node> queue = new LinkedList<Node>();  
 queue.add(data);  
 while (!queue.isEmpty()) {  
 // poll() removes the present head.  
 Node tempNode = queue.poll();  
 System.*out*.print(tempNode.data + " ");  
 // Enqueue left child  
 if (tempNode.left != null) {  
 queue.add(tempNode.left);  
 }  
 if (tempNode.right != null) {  
 queue.add(tempNode.right);  
 }  
 }  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n;  
 Node root = null;  
 int a[] = new int[100];  
 int ch;  
 do {  
 System.*out*.println("Enter your Choice\n1.Insert \n2.Delete \n3.search \n4.Display \n5.BFS");  
 ch = sc.nextInt();  
 switch (ch) {  
 case 1:  
 System.*out*.println("enter no. of elements");  
 n = sc.nextInt();  
 for (int i = 0; i < n; i++) {  
 a[i] = sc.nextInt();  
 root = *insert*(root, a[i]);  
 }  
 break;  
 case 2:  
 System.*out*.println("enter the value that you have to delete");  
 int val = sc.nextInt();  
 *delete*(root, val);  
 break;  
 case 3:  
 System.*out*.println("enter the value that you have to search");  
 int ser = sc.nextInt();  
  
 if (*search*(root, ser)) {  
 System.*out*.println(ser + " is Found");  
 } else {  
 System.*out*.println("Value not found");  
 }  
 break;  
 case 4:  
 System.*out*.println("Display inorder Traversal");  
 *inorder*(root);  
 break;  
 case 5:  
 System.*out*.println("Level order Traversal-Breadth First Search");  
 *printLevelOrder*(root);  
 break;  
 default:  
 System.*out*.println("Something went wrong");  
 }  
 }  
 while (ch != 6);  
 }  
}