|  |
| --- |
| package labsession5; |
|  | import java.util.Scanner; |
|  |  |
|  | public class BST { |
|  | class Node { |
|  | int key; |
|  | Node left, right; |
|  | public Node(int d){ |
|  | key = d; |
|  | left = right = null; |
|  | } |
|  | } |
|  | Node root; |
|  | BST(){ |
|  | root = null; |
|  | } |
|  |  |
|  | void insert(int key) { |
|  | root = insertRec(root, key); |
|  | } |
|  |  |
|  | Node insertRec(Node root, int key) { |
|  | if (root == null) { |
|  | root = new Node(key); |
|  | return root; |
|  | } |
|  | if (key < root.key) { |
|  | root.left = insertRec(root.left, key); |
|  | } |
|  | else if (key > root.key) { |
|  | root.right = insertRec(root.right, key); |
|  | } |
|  | return root; |
|  | } |
|  |  |
|  | void deleteKey(int key) |
|  | { |
|  | root = del(root, key); |
|  | } |
|  | Node del(Node root, int key) |
|  | { |
|  | if (root == null) return root; |
|  | if (key < root.key) |
|  | root.left = del(root.left, key); |
|  | else if (key > root.key) |
|  | root.right = del(root.right, key); |
|  | else |
|  | { |
|  | if (root.left == null) |
|  | return root.right; |
|  | else if (root.right == null) |
|  | return root.left; |
|  | root.key = minV(root.right); |
|  | root.right = del(root.right, root.key); |
|  | } |
|  |  |
|  | return root; |
|  | } |
|  |  |
|  | int minV(Node root) |
|  | { |
|  | int minv = root.key; |
|  | while (root.left != null) |
|  | { |
|  | minv = root.left.key; |
|  | root = root.left; |
|  | } |
|  | return minv; |
|  | } |
|  |  |
|  | void inorder() { |
|  | inorderTraversal(root); |
|  | } |
|  |  |
|  | void inorderTraversal(Node root){ |
|  | if(root!=null){ |
|  | inorderTraversal(root.left); |
|  | System.out.println(root.key); |
|  | inorderTraversal(root.right); |
|  | } |
|  | } |
|  | public static void main(String[] args){ |
|  |  |
|  | BST tree = new BST(); |
|  | Scanner sc = new Scanner(System.in); |
|  | int n, dat, e, j; |
|  | System.out.println("How many nodes do you want to enter?"); |
|  | n=sc.nextInt(); |
|  | System.out.println("Enter the values : "); |
|  | for(int i=0;i<n;i++){ |
|  | dat=sc.nextInt(); |
|  | tree.insert(dat); |
|  | } |
|  |  |
|  | System.out.println("The INORDER traversal of the tree is : "); |
|  | tree.inorder(); |
|  | System.out.println("How many elements you want to delete?"); |
|  | e=sc.nextInt(); |
|  | for(j=0;j<e;j++){ |
|  | System.out.println("Delete a node"); |
|  | int kj = sc.nextInt(); |
|  | tree.deleteKey(kj); |
|  | tree.inorder(); |
|  | } |
|  |  |
|  | } |
|  |  |
|  | } |