Name sineha darshan talreja

Cs231237

```
class Node {
   int key;
   Node left, right;

   public Node(int item) {
      key = item;
      left = right = null;
   }
}

class BST {
   Node root;

   BST(int key) {
      root = new Node(key);
   }

   BST() {
      root = null;
   }

   void printPostOrder(Node node) {
```

```
    if (node == null) return;
    printPostOrder(node.left);
    printPostOrder(node.right);
    System.out.print(node.key + " ");
}

void printInOrder(Node node) {
    if (node == null) return;
    printInOrder(node.left);
    System.out.print(node.key + " ");
    printInOrder(node.right);
}

void printPreOrder(Node node) {
    if (node == null) return;
    System.out.print(node.key + " ");
    printPreOrder(node.left);
    printPreOrder(node.right);
}

Node search(Node root, int key) {
    if (root == null || root.key == key) {
        return root;
    }
    if (root.key > key) {
        return search(root.left, key);
```

```
    }
    return search(root.right, key);
}

Node insertRec(Node root, int key) {
    if (root == null) {
        root = new Node(key);
        return root;
    } else {
        if (key < root.key) {
            root.left = insertRec(root.left, key);
        } else if (key > root.key) {
            root.right = insertRec(root.right, key);
        }
    }
    return root;
}

Node deleteRec(Node root, int key) {
    if (root == null) return root;
    if (key < root.key) {
        root.left = deleteRec(root.left, key);
    } else if (key > root.key) {
        root.right = deleteRec(root.right, key);
    } else {
        if (root.left == null) return root.right;
```

```
        else if (root.right == null) return root.left;
        root.key = minValue(root.right);
        root.right = deleteRec(root.right, root.key);
    }
    return root;
}

int minValue(Node root) {
    int minv = root.key;
    while (root.left != null) {
        minv = root.left.key;
        root = root.left;
    }
    return minv;
}

public static void main(String[] args) {
    BST tree = new BST();
    tree.root = new Node(8);
    Node a = new Node(3);
    tree.root.left = a;
    Node b = new Node(10);
    tree.root.right = b;
    a.left = new Node(1);
    a.right = new Node(6);
    b.left = new Node(9);
```

```
        b.right = new Node(14);

        System.out.println("Pre-order:");
        tree.printPreOrder(tree.root);
        System.out.println("\nPost-order:");
        tree.printPostOrder(tree.root);
        System.out.println("\nIn-order:");
        tree.printInOrder(tree.root);
    }
}
```