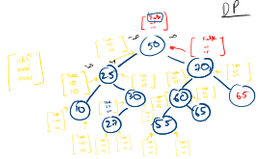
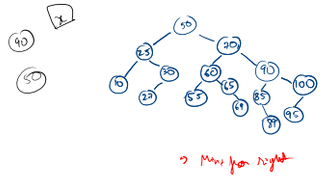
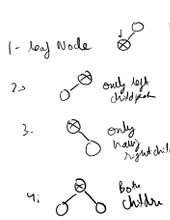


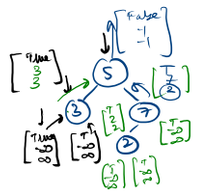


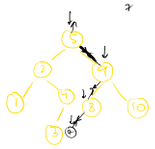



```
Helper dfs(Node root){
    if(root == null){
        return new Helper(true,Integer.MIN_VALUE,Integer.MAX_VALUE);
    }
    Helper left = dfs(root.left);
    Helper right = dfs(root.right);

    boolean isBst = left.isBst && right.isBst && root.data > left.maxi && root.data < right.mini;

    if(!isBst) return new Helper(false,-1,-1);
    int myMax = Math.max(root.data,right.maxi);
    int myMin = Math.min(root.data,left.mini);
    return new Helper(true,myMax,myMin);
}
```

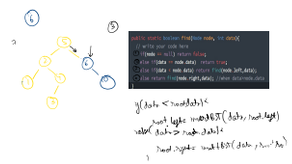




```
public Node insertNode(Node root, int val){
    // WRITE YOUR CODE HERE
    //base case, when we are standing on null
    if(root == null){
        Node nn = new Node(val);
        return nn;
    }

    if(val < root.val){
        //insert to left
        root.left = insertNode(root.left,val);
    }else{
        //insert in right
        root.right = insertNode(root.right,val);
    }
    return root;
}
```

```
public static boolean find(Node node, int data){
  // write your code here
    if(node == null) return false;
    else if(data == node.data) return true;
    else if(data < node.data) return find(node.left,data);
    else return find(node.right,data); //when data>node.data
}
```



```
if(data < root.data){
    root.left = deleteNode(root.left,data);
}else if(data > root.data){
    root.right = deleteNode(root.right,data);
}
```