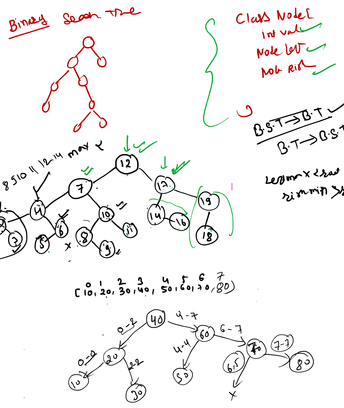
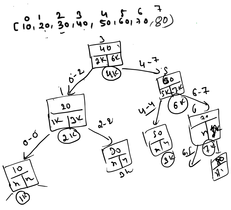
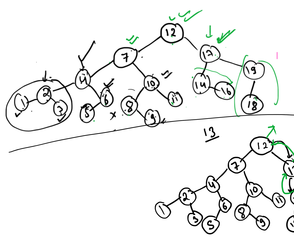


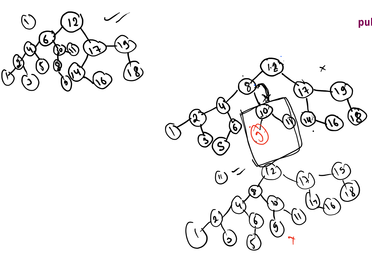

```
private Node createTree(int []in,int lo,int hi) {
      // TODO Auto-generated method stub
    if(lo>hi) {
       return null;
    }
     int mid=(lo+hi)/2;
     Node nn = new Node();
     nn.val=in[mid];
     nn.left=createTree(in, lo, mid-1);
     nn.right=createTree(in, mid+1, hi);
     return nn;

  }
```

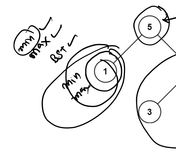




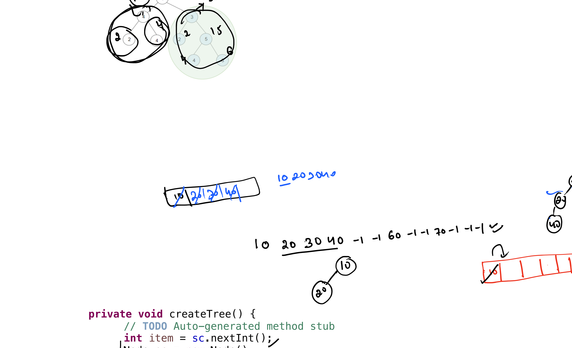
```
public TreeNode insertIntoBST(TreeNode root, int val) {
    if (root == null) {
        return new TreeNode(val);
    }
    if (root.val < val) {
        root.right = insertIntoBST(root.right, val);
    } else {
        root.left = insertIntoBST(root.left, val);
    }
    return root;
}
```

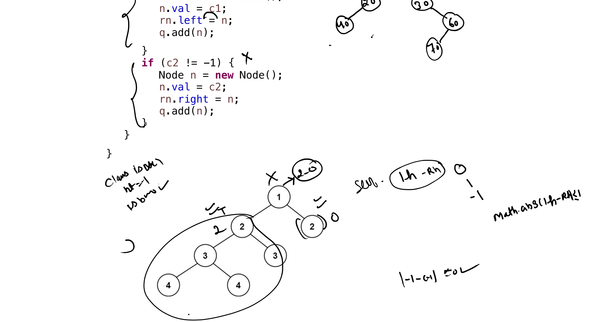


```
public TreeNode deleteNode(TreeNode root, int key) {
    if (root == null) {
        return null;
    }
    if (root.val < key) {
        root.right = deleteNode(root.right, key);
    } else if (root.val > key) {
        root.left = deleteNode(root.left, key);
    } else {
        // 0 or 1 child
        if(root.left==null) {
            return root.right;
        }
        else if(root.right==null) {
            return root.left;
        }
    }
}
```



```
public BstPair ValidBST(TreeNode root) {
        if (root == null) {
            return new BstPair();
        }
        BstPair lbp = ValidBST(root.left);
        BstPair rbp = ValidBST(root.right);
        BstPair sbp = new BstPair();
        sbp.max = Math.max(root.val, Math.max(lbp.max, rbp.max));
        sbp.min = Math.min(root.val, Math.min(lbp.min, rbp.min));
        sbp.isbst = lbp.isbst && rbp.isbst && lbp.max < root.val && rbp.min > root.val;
        return sbp;
    }
```






```
private void createTree() {
      // TODO Auto-generated method stub
     int item = sc.nextInt();
     Node nn = new Node();
     nn.val = item;
     root = nn;
     Queue<Node> q = new LinkedList<>();
     q.add(nn);
     while (!q.isEmpty()) {
        Node rn = q.poll();
        int c1 = sc.nextInt();
        int c2 = sc.nextInt();
        if (c1 != -1) {
            Node n = new Node();
            n.val = c1;
            rn.left = n;
            q.add(n);
        }
        if (c2 != -1) {
            Node n = new Node();
            n.val = c2;
            rn.right = n;
            q.add(n);
        }
    }
```