BST IN-CLASS PROBLEMS :
(Mentor : Gladden Rumao)

1. **Check BST :**
   https://leetcode.com/problems/validate-binary-search-tree

```
class Solution {
    public boolean check(TreeNode root , long min ,
long max){
        if(root==null)
            return true;
        if(root.val<=min || root.val>=max)
            return false;
        return check(root.left,min,root.val) &&
check(root.right,root.val,max);
    }
    public boolean isValidBST(TreeNode root) {
        return
check(root,Long.MIN_VALUE,Long.MAX_VALUE);
    }
}
```

2. **Search in a Binary Search Tree :**
   https://leetcode.com/problems/search-in-a-binary-search-tree/

```
class Solution {
    public TreeNode searchBST(TreeNode root, int
val) {
        if(root==null)
            return null;
        if(root.val==val)
```

```
            return root;
        if(val < root.val)
            return searchBST(root.left,val);
        else
            return searchBST(root.right,val);
    }
}
```

3. **Insert into a Binary Search Tree :**
   https://leetcode.com/problems/insert-into-a-binary-search-tree/

```
class Solution {
    public TreeNode insertIntoBST(TreeNode root, int
val) {
        if(root==null){
            //create a node that is be inserted
            TreeNode node = new TreeNode(val);
            return node;
        }
        if(val < root.val){
            root.left =
insertIntoBST(root.left,val);
        }
        else{
            root.right =
insertIntoBST(root.right,val);
        }
        return root;
    }
}
```

4. **Delete Node in a BST**
   https://leetcode.com/problems/delete-node-in-a-bst/

```
class Solution {
    public TreeNode getMin(TreeNode curr){
        while(curr.left!=null){
            curr = curr.left;
        }
        return curr;
    }
    public TreeNode deleteNode(TreeNode root, int
key) {
        //search the node to be deleted
        TreeNode parent = null;
        TreeNode curr = root;
        while(curr!=null && curr.val != key){
//doesnt exist or found
            parent = curr;
            if(key<curr.val)
                curr = curr.left;
            else
                curr = curr.right;
        }
        if(curr==null) //stop
            return root;
        //case1 : node has 0 children -> leaf node
        if(curr.left==null && curr.right==null){
            if(curr==root){
                root = null;
            }
            else{
                if(parent.left==curr)
                    parent.left = null;
                else
                    parent.right = null;
            }
        }
        //case 2 : node has 1 child node
        else if(curr.left==null ||
curr.right==null){
            TreeNode child;
            if(curr.left == null )
                child = curr.right;
            else
                child = curr.left;
            if(curr==root){
                root = child;
            }
            else{
                if(curr == parent.left)
                    parent.left = child;
                else
                    parent.right = child;
            }
        }
        //case 3 : node has 2 child node
        else{
            TreeNode min = getMin(curr.right);
            int minval = min.val;
            deleteNode(root,minval);
            curr.val = minval;
        }
        return root;

    }
}
```

5. **Kth Smallest Element in a BST**
   https://leetcode.com/problems/kth-smallest-element-in-a-bst/

```
class Solution {
    int ans=0;
    int count=0;
    public int kthSmallest(TreeNode root, int k) {
        inOrder(root,k);
        return ans;
    }

    public void inOrder(TreeNode root,int k){
        if(root==null) return;

        inOrder(root.left,k);
        count++;
        if(count==k){
            ans=root.val;
            return;
        }
        inOrder(root.right,k);
    }


}
```