

```
1  /**
2   * public class TreeNode {
3   *     public int key;
4   *     public TreeNode left;
5   *     public TreeNode right;
6   *     public TreeNode(int key) {
7   *         this.key = key;
8   *     }
9   * }
10 */
11 public class Solution {
12     public boolean isBalanced(TreeNode root) {
13         if (root == null) {
14             return true;
15         }
16         if (Math.abs(getHeight(root.left) - getHeight(root.right)) > 1) {
17             return false;
18         }
19         return isBalanced(root.right) && isBalanced(root.left);
20     }
21     private int getHeight(TreeNode root) {
22         if (root == null) {
23             return 0;
24         }
25         int L = getHeight(root.left);
26         int R = getHeight(root.right);
27         return Math.max(L, R) + 1;
28     }
29 }
30
```

```
1  /**
2  * public class TreeNode {
3  *     public int key;
4  *     public TreeNode left;
5  *     public TreeNode right;
6  *     public TreeNode(int key) {
7  *         this.key = key;
8  *     }
9  * }
10 */
11 public class Solution {
12     public int findHeight(TreeNode root) {
13         if(root == null){
14             return 0;
15         }
16         int leftHeight = findHeight(root.left);
17         int rightHeight = findHeight(root.right);
18         return Math.max(findHeight(root.left),findHeight(root.right))+1;
19     }
20 }
21
22
23
```

```
1  /**
2   * public class TreeNode {
3   *     public int key;
4   *     public TreeNode left;
5   *     public TreeNode right;
6   *     public TreeNode(int key) {
7   *         this.key = key;
8   *     }
9   * }
10 */
11 public class Solution {
12     public int countNodes(TreeNode root) {
13         if (root == null) {
14             return 0;
15         }
16         int left = countNodes(root.left);
17         int right = countNodes(root.right);
18         return countNodes(root.left) + countNodes(root.right) + 1;
19     }
20 }
21
```

```
1  /**
2  * public class TreeNode {
3  *     public int key;
4  *     public TreeNode left;
5  *     public TreeNode right;
6  *     public TreeNode(int key) {
7  *         this.key = key;
8  *     }
9  * }
10 */
11 public class Solution {
12     public List<Integer> inOrder(TreeNode root) {
13         List<Integer> res = new ArrayList<>();
14         helper(root,res);
15         return res;
16     }
17     private void helper(TreeNode root ,List<Integer> res){
18         if (root == null){
19             return;
20         }
21         helper(root.left,res);
22         res.add(root.key);
23         helper(root.right,res);
24     }
25 }
26
27
```

```
1  /**
2  * public class TreeNode {
3  *     public int key;
4  *     public TreeNode left;
5  *     public TreeNode right;
6  *     public TreeNode(int key) {
7  *         this.key = key;
8  *     }
9  * }
10 */
11 public class Solution {
12     public List<Integer> preOrder(TreeNode root) {
13         List<Integer> res = new ArrayList<>();
14         helper(root,res);
15         return res;
16     }
17     private void helper(TreeNode root, List<Integer> res){
18         if (root == null ){
19             return;
20         }
21         res.add(root.key);
22         helper(root.left,res);
23         helper(root.right,res);
24     }
25 }
26
```

```
1  /**
2  * public class TreeNode {
3  *     public int key;
4  *     public TreeNode left;
5  *     public TreeNode right;
6  *     public TreeNode(int key) {
7  *         this.key = key;
8  *     }
9  * }
10 */
11 public class Solution {
12     public List<Integer> postOrder(TreeNode root) {
13         List<Integer> res = new ArrayList<>();
14         helper(root, res);
15         return res;
16     }
17     private void helper(TreeNode root, List<Integer> res) {
18         if (root == null) {
19             return;
20         }
21         helper(root.left, res);
22         helper(root.right, res);
23         res.add(root.key);
24     }
25 }
26
```