# Data Structures and Algorithms - Assignment 9

1. Leetcode Problem : https://leetcode.com/problems/balanced-binary-tree/description/

```
class Solution {

    public int height(TreeNode trav){
        if(trav == null)
            return 0;
        int hl = height(trav.left);
        int hr = height(trav.right);
        if(hl == -1)
            return -1;
        if(hr == -1)
            return -1;
        if(Math.abs(hl - hr) > 1)
            return -1;

        return (hl < hr ? hr : hl) + 1;
    }

    public boolean isBalanced(TreeNode root) {
        return height(root) != -1;
    }
}
```

2. Leetcode Problem : https://leetcode.com/problems/validate-binary-search-tree/description/

```
class Solution {
    public boolean isValidBST(TreeNode root) {

        return isValid(root, null, null);
    }

    public boolean isValid(TreeNode trav, TreeNode minNode, TreeNode maxNode){
        if(trav == null)
            return true;
        if((minNode != null && trav.val <= minNode.val) || (maxNode != null &&
trav.val >= maxNode.val))
        return false;

        return isValid(trav.left, minNode, trav) && isValid(trav.right, trav,
maxNode);
    }
}
```

3. Leetcode Problem : https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/description/

```
class Solution {
    public List<List<Integer>> zigzagLevelOrder(TreeNode root) {
        if(root == null)
            return new ArrayList<List<Integer>>();
        Deque<TreeNode> q = new ArrayDeque<>();
        q.add(root);

        List<List<Integer>> res = new ArrayList<List<Integer>>();
        List<Integer> l;
        boolean flag = true;
        while(!q.isEmpty()){
            int size = q.size();
            l = new ArrayList<Integer>();
            if(flag){
                for(int i = 0 ; i < size ; i++){
                    TreeNode temp = q.pollFirst();
                    l.add(temp.val);
                    if(temp.left != null)
                        q.addLast(temp.left);
                    if(temp.right != null)
                        q.addLast(temp.right);
                }
            }else{
                for(int i = 0 ; i < size ; i++){
                    TreeNode temp = q.pollLast();
                    l.add(temp.val);
                    if(temp.right != null)
                        q.addFirst(temp.right);
                    if(temp.left != null)
                        q.addFirst(temp.left);
                }
            }
            flag = !flag;
            res.add(l);
        }
        return res;
    }
}
```

4. Leetcode Problem : https://leetcode.com/problems/maximum-depth-of-binary-tree/description/

5. Leetcode Problem : https://leetcode.com/problems/minimum-depth-of-binary-tree/description/

```
class Solution {
    public int minDepth(TreeNode root) {
        if(root == null)
            return 0;
```

```
        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);
        int depth = 1;
        while(!q.isEmpty()){
            int size = q.size();
            for(int i = 0 ; i < size ; i++){
                TreeNode temp = q.poll();
                if(temp.left == null && temp.right == null)
                    return depth;
                if(temp.left != null)
                    q.add(temp.left);
                if(temp.right != null)
                    q.add(temp.right);
            }
            depth++;
        }
        return 0;
    }
}
```