Trees - Practice Problems

Do the given problems using JavaScript.

1. Write a Program to check if all the leaves of the binary tree are at the same level.
2. Write a Program to return the maximum depth of the binary tree.
3. Write a Program to return the height of the binary tree.
4. Write a Program to print the left side of the binary tree.

Solutions

1.

```
function checkLeafLevel(node, level, leafLevel)
{
    if (node == null)
        return true;
    // If a leaf node is seen
    if (node.left == null && node.right == null)
    {
        // When a leaf node is found first time
        if (leafLevel.leaflevel == 0)
        {
            // Set first found leaf's level
            leafLevel.leaflevel = level;
            return true;
        }

        // If this is not first leaf node,
        // compare its level with first leaf's level
        return (level == leafLevel.leaflevel);
    }
    // If this node is not leaf, recursively
    // check left and right subtrees
    return checkLeafLevel(node.left, level + 1, leafLevel) &&
           checkLeafLevel(node.right, level + 1, leafLevel);
}

function checkIfAllLeavesAtSameLevel(node)
{
    let level = 0;
    return checkIfAllLeavesAtSameLevel(node, level, mylevel);
}
```

2.

```
var maxDepth = function(root) {
    if(root === undefined || root===null){
        return 0;
    }
    return Math.max(maxDepth(root.left),maxDepth(root.right)) + 1;
};
```

3.

```
function height(node)
    {
        if(node == null) {
            return 0;
        }
        else{
```

```
        let leftHeight = this.height(node.left);
        let rightHeight = this.height(node.right);

        return Math.max(leftHeight, rightHeight)+1;
        }
}
```

4.

```
function leftView(root)
    {
        let q = [root]
        let res = []

        while(q[0]){
            let sub = []
            let qLen = q.length;
            for(let i=0; i<qLen; i++){
                let current = q.shift()
                sub.push(current.data)

                current.left && q.push(current.left)
                current.right && q.push(current.right)
            }
            res.push(sub)
        }

        return res.map(element => element.shift())
    }
```