Right/Left view of binary tree
Problem Statement: Given a Binary Tree, find the **Right/Left view** of it. The right view of a Binary Tree is a set of nodes visible when the tree is viewed from the **right** side. The left view of a Binary Tree is a set of nodes visible when the tree is viewed from the **left** side.
Example 1:

Input:

```
        1
      /   \
     3     2
```

Output: Right view- 1 2
Left view- 1 3

Explanation: Seeing through the left side it sees only 1 and 3 while through the right side we only see 1 and 2.
Example 2:

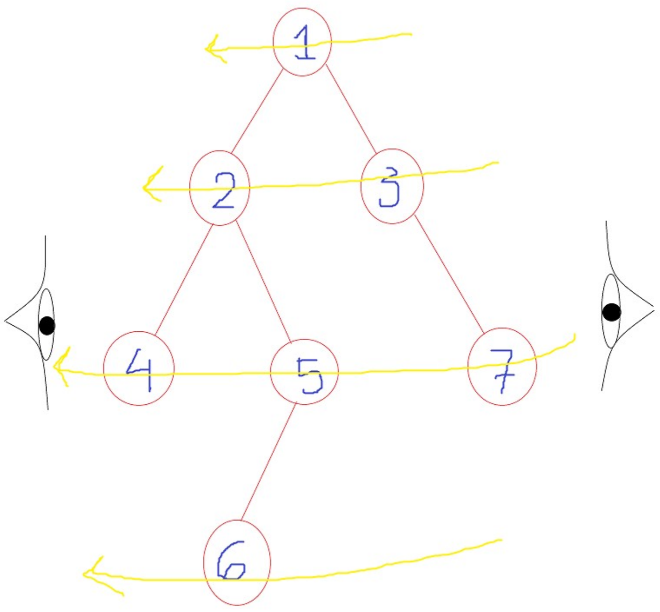
Input:

```
          10
        /    \
       20     30
      /  \
     40   60
```

Output: Right View- 10 30 60
Left view- 10 20 40

Explanation: Seeing through the left side it sees only 10, 20, and 40 while through the right side we only see 10, 30, and 60.
Solution:
***Disclaimer**: Don't jump directly to the solution, try it out yourself first.*

Intuition: We have to do a Recursive Level Order Traversal.
Root Right Left —-> for Right view
Root Left Right —-> for Left view



Approach:
- Create an vector data structure inside both the left and the right side view function
- Call for the recursive _left and recursive_right function respectively with the (root,level,vector). Here level will be initially passed as 0.

- Return the vector.
- Now in the recursive_left function
- If vector size is equal to the level then push_back its node value to the vector data structure.
- Otherwise call recursive_left for (node->left,level+1,vector)
- Call recursive_left for (node->right,level+1,vector)
- Now in the recursive_right function
- If vector size is equal to the level then push_back its node value to the vector data structure.
- Otherwise call recursive_right for (node->right,level+1,vector)
- Call recursive_right for (node->left,level+1,vector)

Tip: The Code for the Left and the Right View is almost identical.
In the Right view code first, you have to call the recursive function for the right then the left node
AND
In the Right view code first, you have to call the recursive function for the Left than the right node
Code: For the right view of the binary tree.

- C++ Code
- Java Code

```
class Solution {
public:
    void recursion(TreeNode *root, int level, vector<int> &res)
    {
        if(root==NULL) return ;
        if(res.size()==level) res.push_back(root->val);
        recursion(root->right, level+1, res);
        recursion(root->left, level+1, res);
    }

    vector<int> rightSideView(TreeNode *root) {
        vector<int> res;
        recursion(root, 0, res);
        return res;
    }
};
```

Time Complexity: O(N)

Space Complexity: O(H)   (H -> Height of the Tree)

Code: For the left view of binary tree.

- C++ Code
- Java Code

```
class Solution {
public:
    void recursion(TreeNode *root, int level, vector<int> &res)
    {
        if(root==NULL) return ;
        if(res.size()==level) res.push_back(root->val);
        recursion(root->left, level+1, res);
        recursion(root->right, level+1, res);
    }

    vector<int> leftSideView(TreeNode *root) {
        vector<int> res;
        recursion(root, 0, res);
        return res;
    }
};
```

Time Complexity: O(N)

Space Complexity: O(H)   (H -> Height of the Tree)