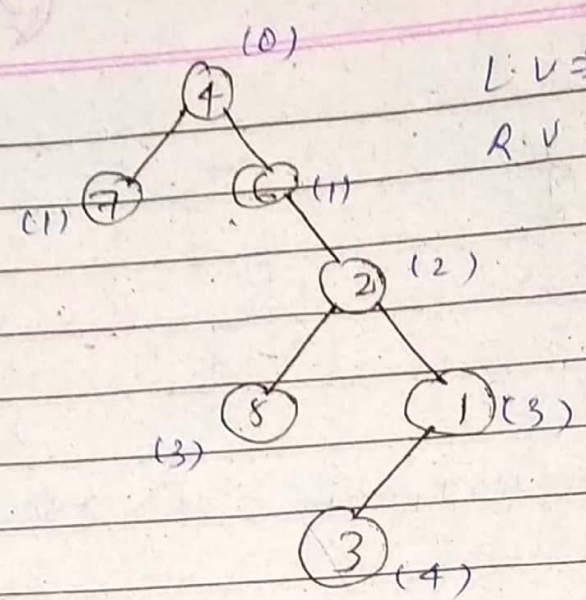
Q print left view / right view of a binary tree



L.V ⇒ 4 7 2 5 3

R.V ⇒ 4 6 2 1 3

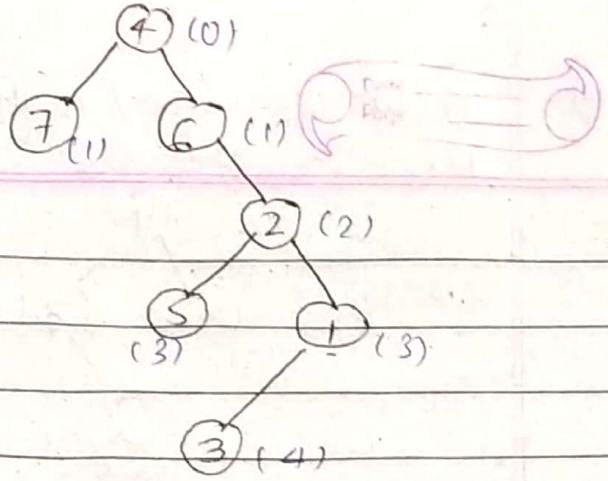
~~void~~

[left-view]

```
void printLeftView (Node root){
    ArrayList<Node> a = new ArrayList<>();
    printLeftViewUtil(root, a, 0);
    for (Node curr : list){
        System.out.print(curr.data + " ");
    }
}

void printLeftViewUtil(Node root, ArrayList<Integer>
                       a, int level){
    if (root == null) return;
    if (a.get(level) == null){
        a.add(root);
    }
    printLeftViewUtil(root.left, a, level+1);
    printLeftView Util(root.right, a, level+1);
}
```

[right-view]



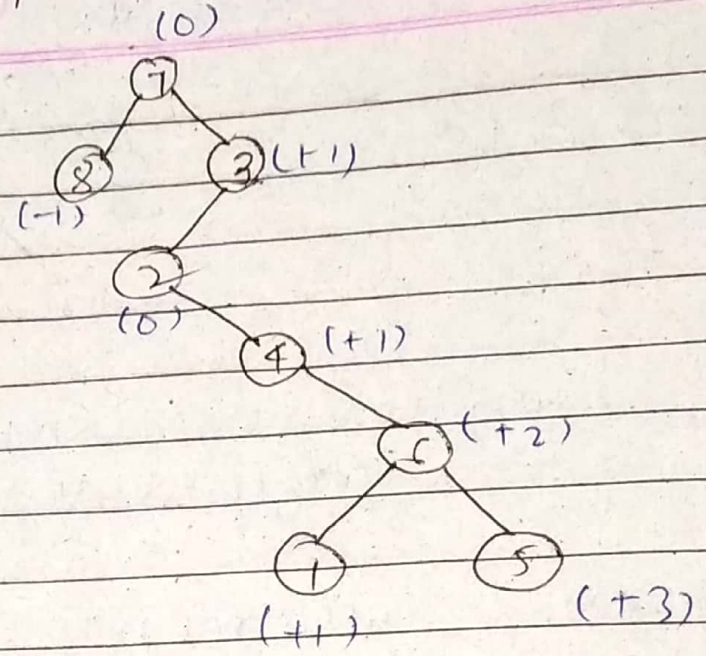
[method:1]

```
void printRightViewUtil(Node root,
                        ArrayList<Integer> a, int level)
{
    if(root == null) return;
    if(a.get(level) == null){
        a.add(root);
    }
    printRightViewUtil(root.right, a, level+1);
    printRightViewUtil(root.left, a, level+1);
}
```

[method:2]

```
void printRightViewUtil(Node root, ArrayList a
                        int level)
{
    if(root == null) return;
    a.set(level, root);
    printRightViewUtil(root.left, a, level+1);
    printRightViewUtil(root.right, a, level+1);
}
```

print top-view / bottom-view of Binary tree.



← steps →

-2	-1	0	1	2	3
	8	7	3	6	5

```
void topView (Node, root, map<Integer, Node>,
                    int steps) {
    if(root == null) return;
    map.putIfAbsent (steps, root);
    topView (root.left, map, steps -1);
    topView (root.right, map, steps +1);
}
```