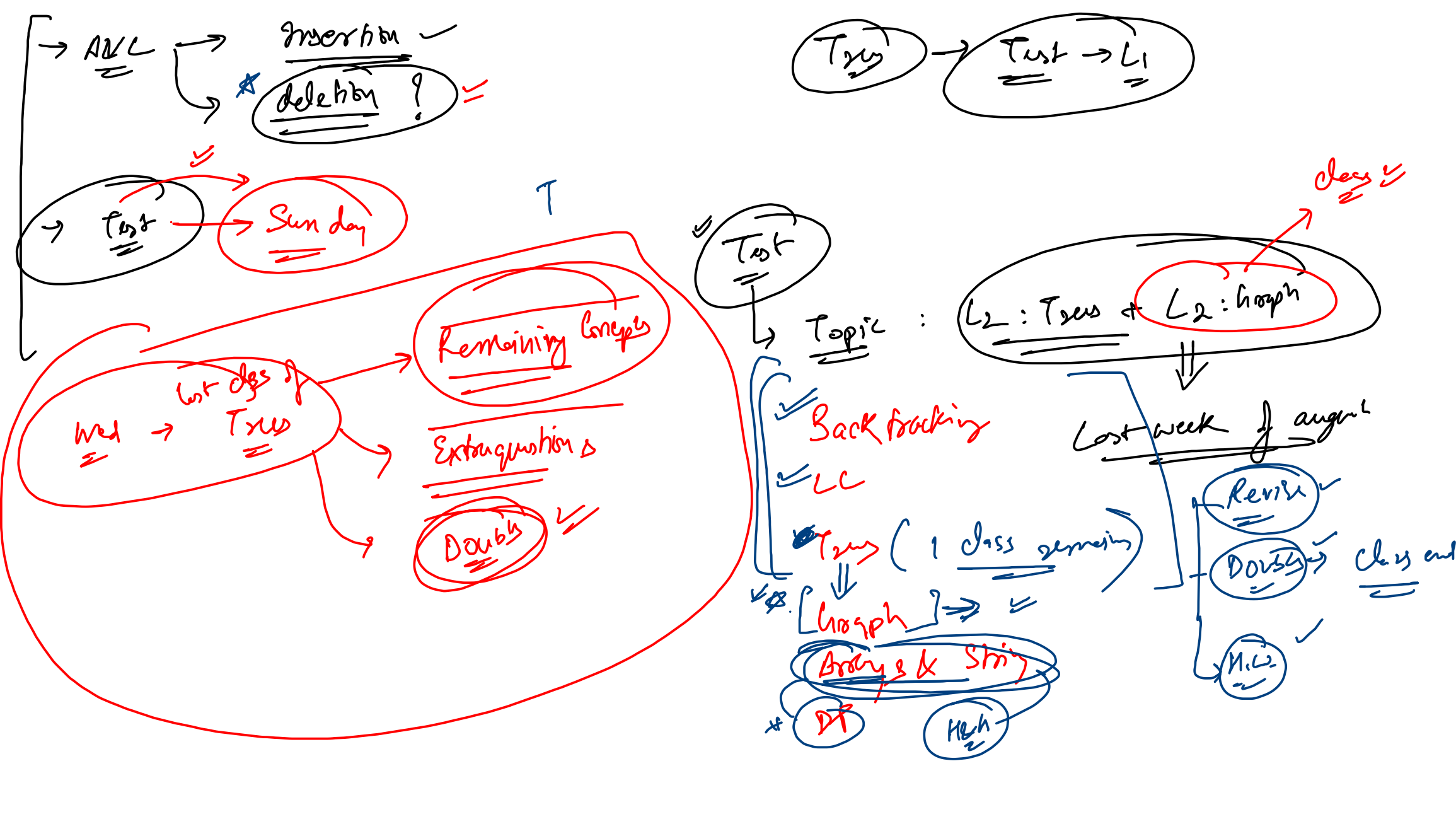
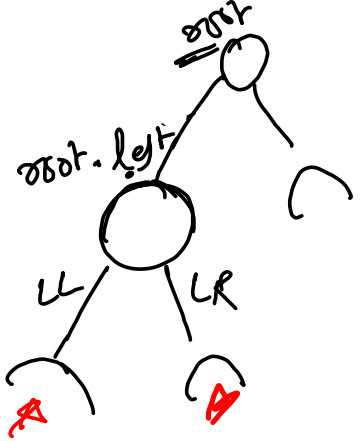

AVL
Insertion
deletion ?
Test
Sunday
T
Test
Trees
Test → L1
Remaining Concepts
Last days of
Web → Trees
Extinguishers
Doubts
Test
Topic :
L2 : Trees + L2 : Graph
days
Backtracking
LC
Trees (1 class remains
[Graph] ⇒
Arrays & Strings
DP
High
Last week of August
Revise
Doubts
Class end
M.L.

BST → Insert → Balanced ? / Unbalanced ? → self balance AVL



diff = LHT − RHT

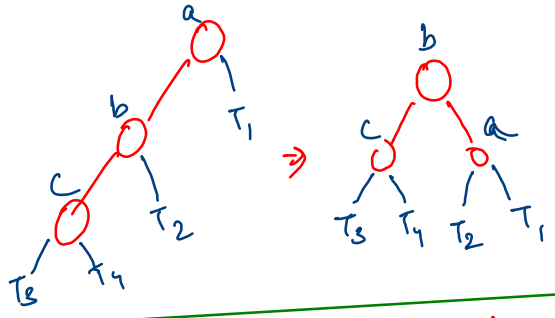
```
if ( diff > 1) {
    if ( data < root.left) { LL
    } else if (data > root.left) { LR
} }
} ( diff < -1 ) {
```

RR

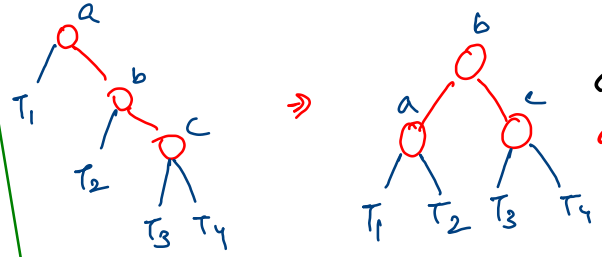
RL

{ }

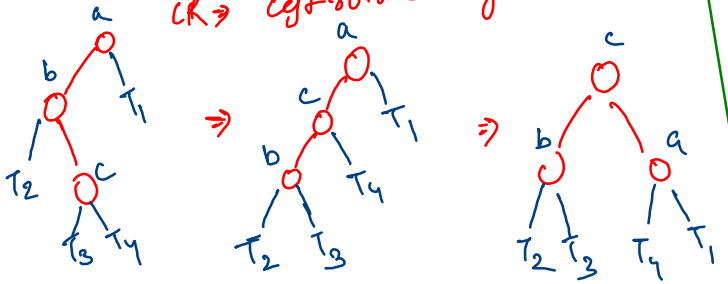
LL ⇒ right rotation



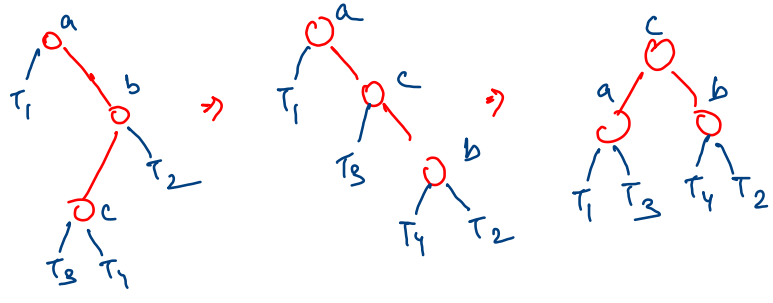
RR ⇒ left rotation



LR ⇒ left rotate + right rotate

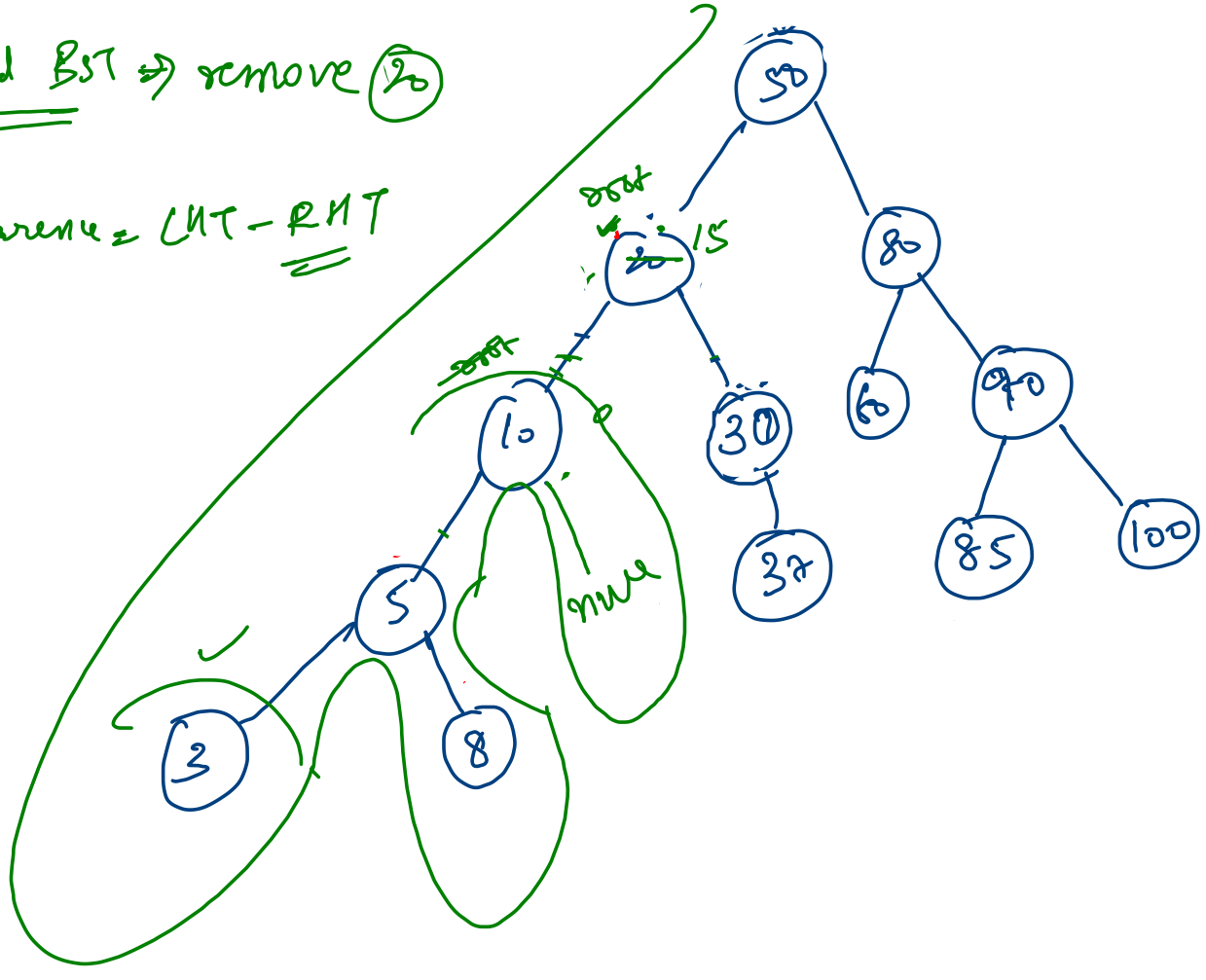


RL ⇒ right rotate + left rotate

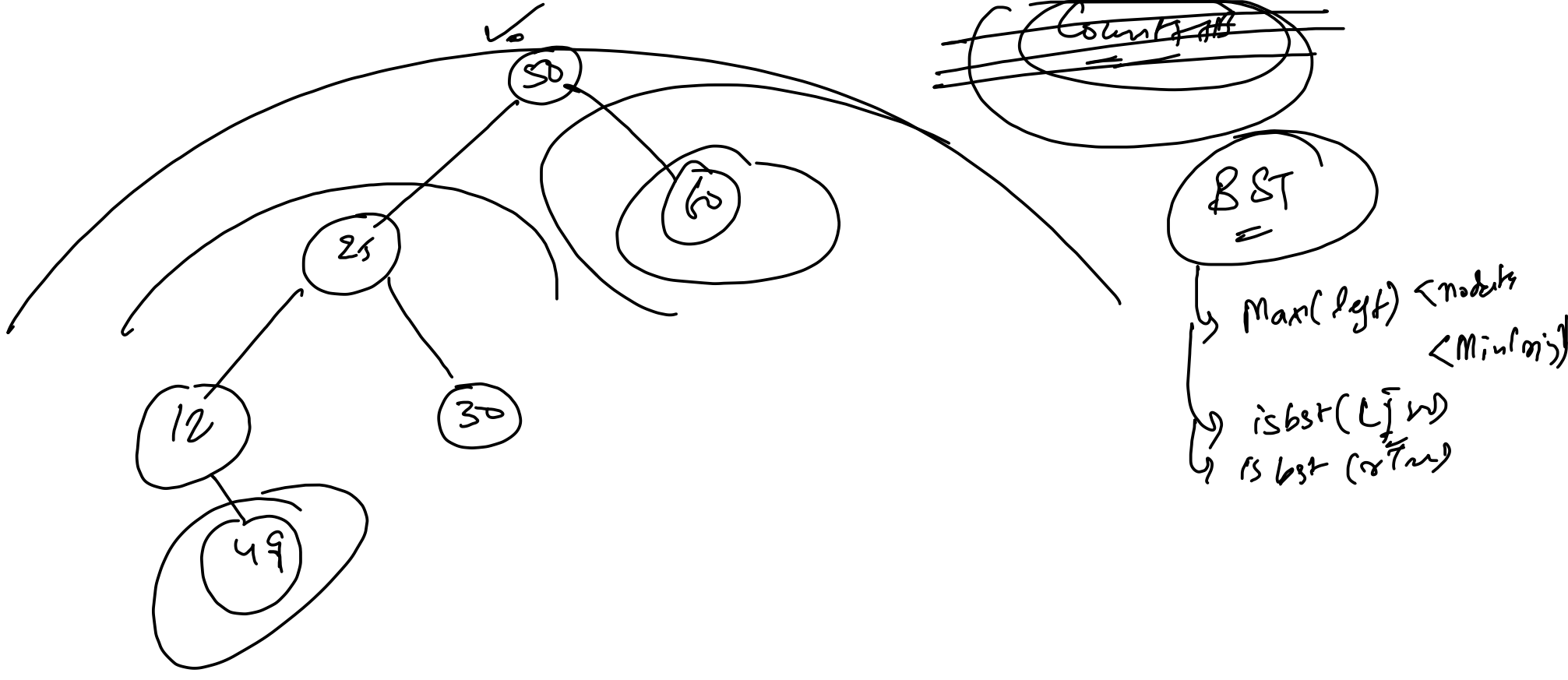


Balanced BST ⇒ remove (20)

difference = LHT − RHT




```
if (difference(root) > 1) {          L
    int delta = difference(root.left);
    if (delta >= 0) {
        LL
    } else if (delta < 0) {
        LR
    }
} else if (difference(root) < -1) {  R
    int delta = difference(root.right);
    if (delta > 0) {
        RL
    } else if (delta <= 0) {
        RR
    }
}
```


50
25
60
12
30
49
BST
Max(left) < node.data
< Min(right)
isbst(L, M)
isbst(R, M)

A nation is at a brink of war & only way is to prevent it by dectecting all enemies within a mile radius. Remember a whole lot of casualities will be at your hands if any of the enemy remains undetected.
Well to conclude ... everything is upto you now , will the war break out or you will be able to prevent it ?.

you will be given location(central location) of a enemy lookout & your task is to find number of enemy lookouts in a given distance radius so that our nation's special forces subdues them before they make the move.

Note the given facts about your mission :

1. The rooted binary tree will be provided as input.
2. central location name of enemy lookout will also be provided.
3. task is to find number of enemy lookouts in a given distance(d) radius.
4. d is a positive integer.
5. traveling an edge means 1 unit distance has been travelled.

```
printKLevelsDown(list.get(0),k);

for(int idx = 1 ; idx < list.size() ; idx++){
    countOfEnemyLookout++;
    int remDist = k - idx;
    Node curr = list.get(idx);
    Node prev = list.get(idx-1);
    if(remDist == 0){
        // System.out.println(curr.locName);
        break;
    }else{
        if(curr.left == prev){
            printKLevelsDown(curr.right , remDist-1);
        }else if(curr.right == prev){
            printKLevelsDown(curr.left , remDist-1);
        }
    }
}
```

```
public static void printKLevelsDown(Node node, int k){
    if(node == null){
        return;
    }
    if(k == 0){
        // System.out.println(node.locName);
        countOfEnemyLookout++;
        return;
    }
    // System.out.println(node.locName);
    countOfEnemyLookout++;
    printKLevelsDown(node.left,k-1);
    printKLevelsDown(node.right,k-1);
}
```