Given a binary tree, print the bottom view from left to right.  
A node is included in bottom view if it can be seen when we look at the tree from bottom.

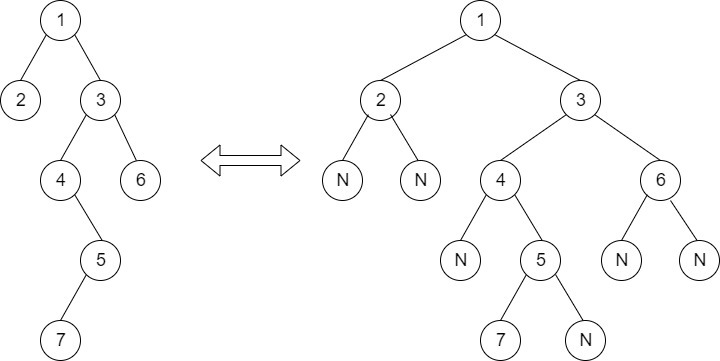
                      20  
                    /    \  
                  8       22  
                /   \        \  
              5      3       25  
                    /   \        
                  10    14

For the above tree, the bottom view is 5 10 3 14 25.  
If there are **multiple**bottom-most nodes for a horizontal distance from root, then print the later one in level traversal. For example, in the below diagram, 3 and 4 are both the bottommost nodes at horizontal distance 0, we need to print 4.

                      20  
                    /    \  
                  8       22  
                /   \     /   \  
              5      3 4     25  
                     /    \        
                 10       14

For the above tree the output should be 5 10 4 14 25.

**Input Format:**  
First line of input contains the number of test cases T. For each test case, there will be only a single line of input which is a string representing the tree as described below:

1. The values in the string are in the order of level order traversal of the tree where, numbers denotes node values, and a character “N” denotes NULL child.
2. For example:  
     
   For the above tree, the string will be: 1 2 3 N N 4 6 N 5 N N 7 N

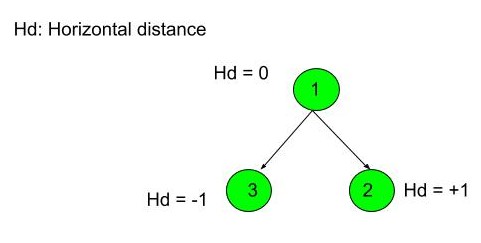
**Output Format:**  
The function should print nodes in the bottom view of Binary Tree. Your code should not print a newline, it is added by the caller code that runs your function.

**Your Task:**  
This is a functional problem, you **don't**need to care about input, just complete the function **bottomView**() which takes the root node of the tree as input and returns an array containing the bottom view of the given tree.

**Expected Time Complexity:**O(N).  
**Expected Auxiliary Space:**O(N).

**Constraints:**  
1 <= T<= 30  
1 <= Number of nodes <= 105  
1 <= Data of a node <= 105

**Example:  
Input:**  
2  
1 3 2  
10 20 30 40 60  
**Output:**  
3 1 2  
40 20 60 30

**Explanation:  
Testcase 1:**  First case represents a tree with 3 nodes and 2 edges where root is 1, left child of 1 is 3 and right child of 1 is 2.  


Thus nodes of the binary tree will be printed as such 3 1 2.

// { Driver Code Starts

vector <int> bottomView(Node \*root)

{

//this is map of hd and node value

//<hd,node value>

map<int,int>mp;

vector<int> ans;

if(root==NULL)

return ans;

int hd=0;

queue<pair<Node\*,int>>pnodes;

//rootnode <rootnodedata,0th level>

pnodes.push(make\_pair(root,hd));

Node\*temp;

while(!pnodes.empty())

{

//update node value in Map

pair<Node\*,int> front=pnodes.front();

pnodes.pop();

hd=front.second;

temp=front.first;

mp[hd]=temp->data;

if(temp->left!=NULL)

{

//push left children with hd level hd-1

pnodes.push(make\_pair(temp->left,hd-1));

mp[hd-1]=temp->left->data;

}

if(temp->right!=NULL)

{

//push hd level with level hd+1

pnodes.push(make\_pair(temp->right,hd+1));

mp[hd+1]=temp->right->data;

}

}

for (auto it:mp)

{

//push updates value in queue

ans.push\_back(it.second);

}

return ans;

}