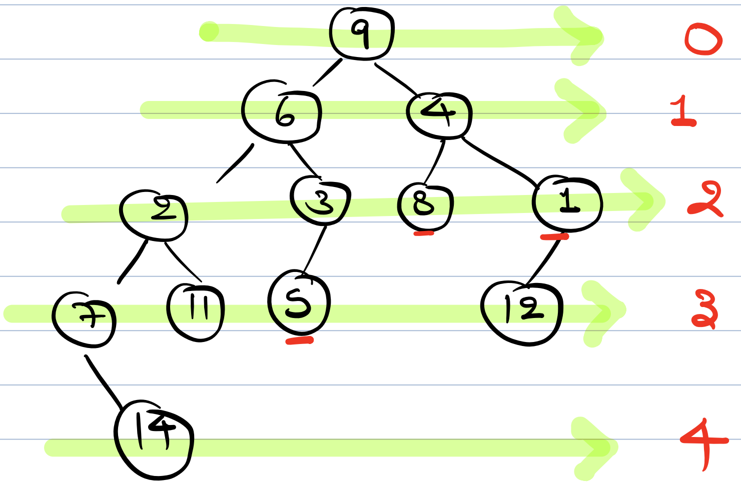
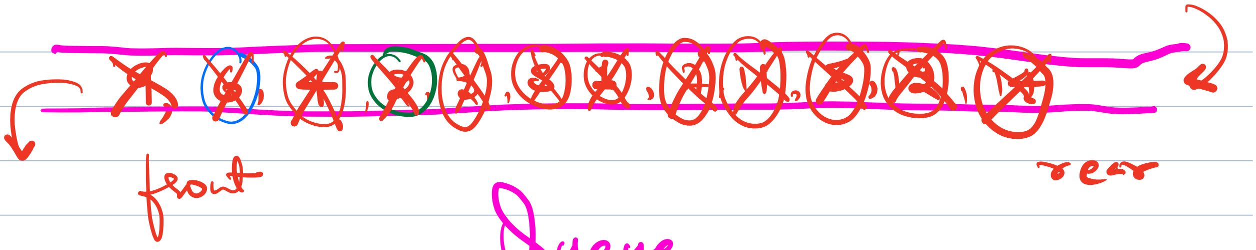
Q Given a BT. Print the level order traversal (BFS)



9, 6, 4, 2, 3, 8, 1, 7, 11, 5, 12, 14



Queue

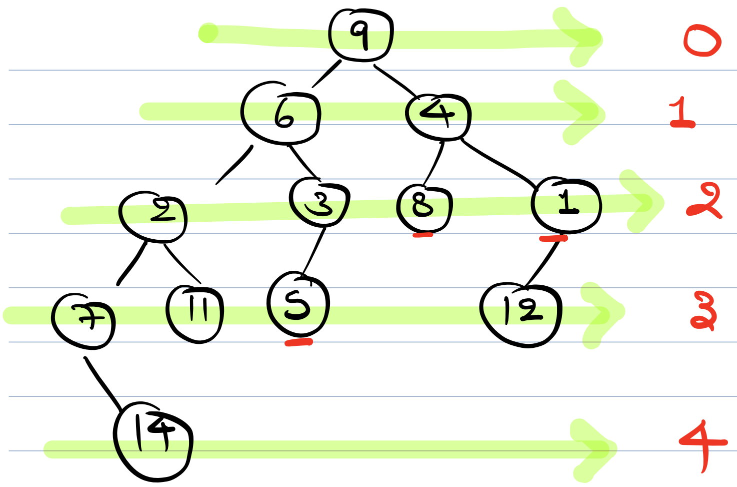
1) Insert root into queue

2) While the queue is not empty.

 → Remove the first ele
 → Print the first ele

 → Use this ele to add the ele of next level (its children)

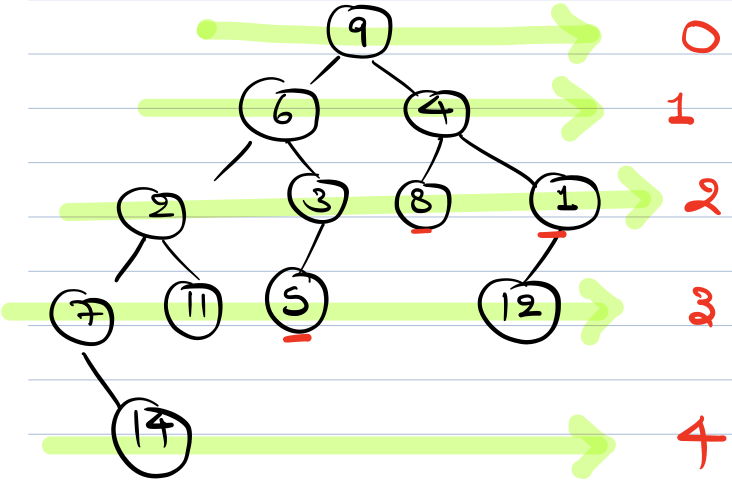
{



[
[9]
[6, 4]
[2, 3, 8, 1]
[7, 11, 5, 12]
[14]
]

~~9~~, ~~#~~, ~~6~~, ~~4~~, #, 2, 3, 8, 1, ~~#~~

NULL NULL



9
6, 4

> q.size().

~~9~~, ~~6~~, ~~4~~, 2, 3, 8, 1

Ele in level 0 = 1

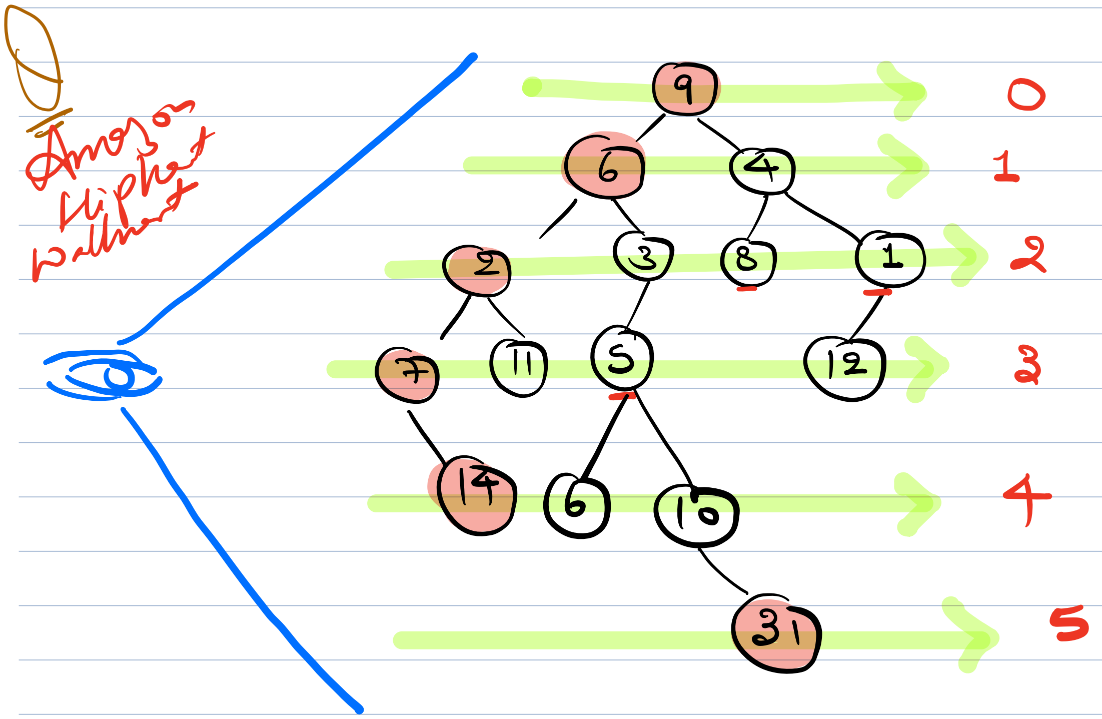
Ele in level 1 = 2

Ele in level 2 = 4

Code

```
List<List<Int>> levelOrder (root) {
    if (root == NULL) { return NULL;}
    List <List<Int>> result;
    Queue <Node > Q;
    Q.add (root);
    while (!Q.isEmpty()) {
        size = Q.size();
        List<Int> level;
        for (i=0; i<size; i++) {
            Node temp = Q.dequeue();
            level.add (temp.data);
            if (temp.left != NULL) {
                Q.enque (temp.left);
            }
            if (temp.right != NULL) {
                Q.enque (temp.right);
            }
        }
        result.add(level);
    }
    return    result;
}
```

T.C. = O(N)



Amazon
Flipkart
Walmart

Given a BT. Print the left view of the BT

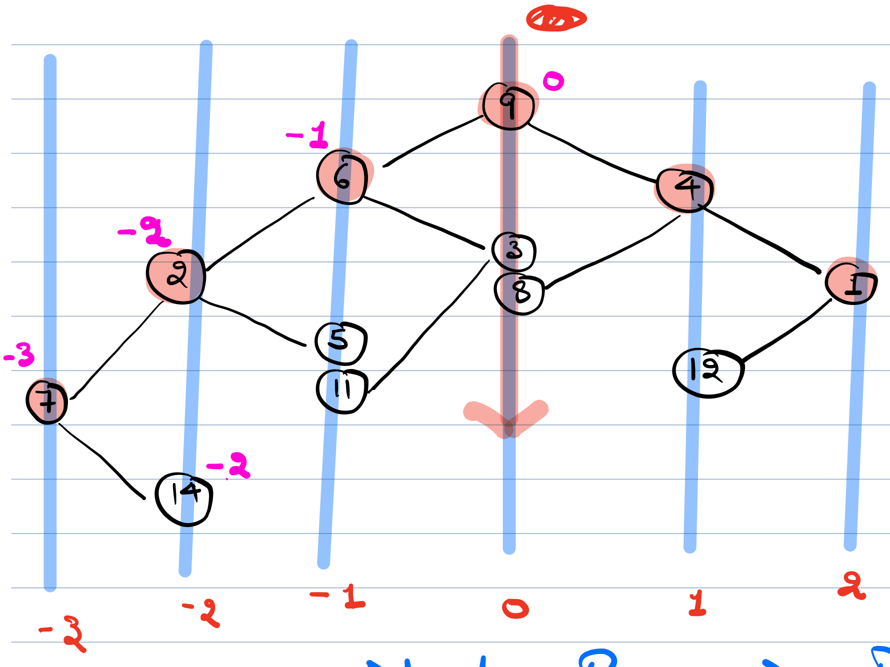
Ans = 9, 6, 2, 7, 14, 31

Sol^n Print the 1st node of every level

Q Right View.

Sol^n Last Node of every level.

Q Given a BT.
Print the vertical order traversal



[
-3 ← [7],
-2 ← [2, 14],
-1 ← [6, 5, 11],
0 ← [9, 3, 8],
1 ← [4, 12],
2 ← [1],
]

N, L, R → Pre Order

HashMap < level, List <Nodes>>

Code

```
HashMap < int, list<Nodes> >

                  0
preorder ( root, dist )
      if (root == NULL) { return; }

      if ( ! map.contains(dist) ) {
            map.put(dist, new ArrayList<Int>());
      }
      map.get(dist).add(root.data);
      preorder(root.left, dist-1);
      preorder(root.right, dist+1);
}
```

T.C. = O(N)

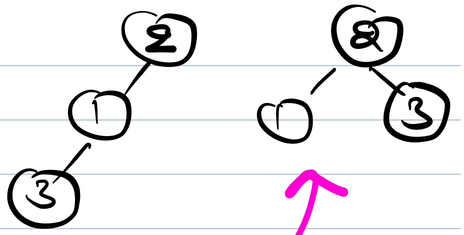
Q) Given the preorder & inorder traversal of a tree

Construct the tree (No duplicates)

1) Pre : [2, 1, 3]

N, L, R

Root



→ In : [1, 2, 3]

L N R

Root

$x - (S_{pre}+1)+1 = LLST$

$x - S_{pre} = LLST$

$x = LLST + S_{pre}$

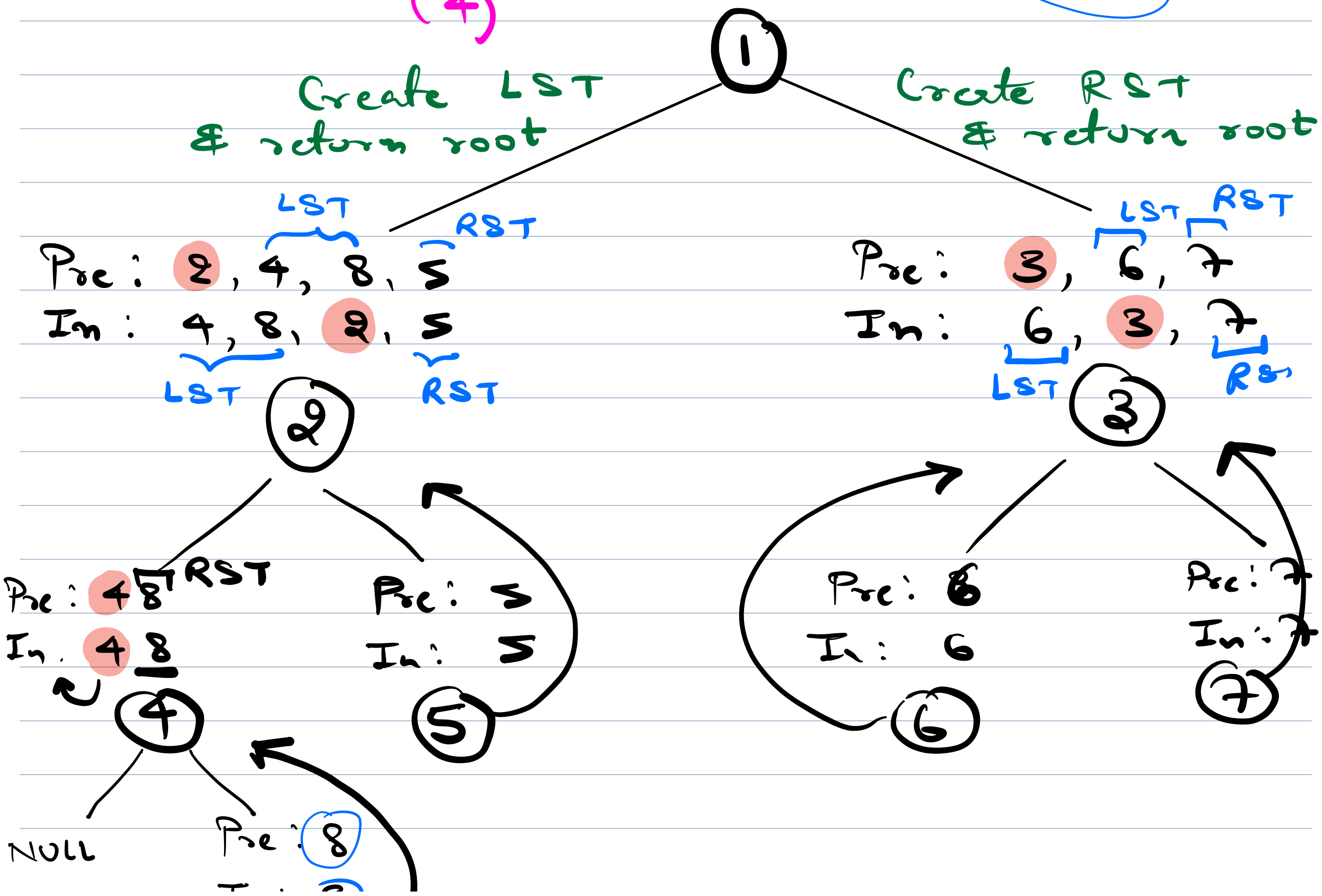
e_{pre}

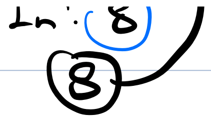
Spre, LST, 4, x, RST

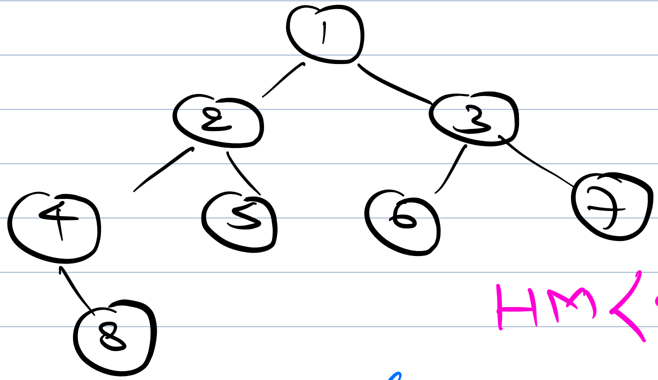
Pre: 1, 2, 4, 8, 5, 3, 6, 7

In: 4, 8, 2, 5, 1, 6, 3, 7

Sin, LST (4), idx-1, idx, RST, ein







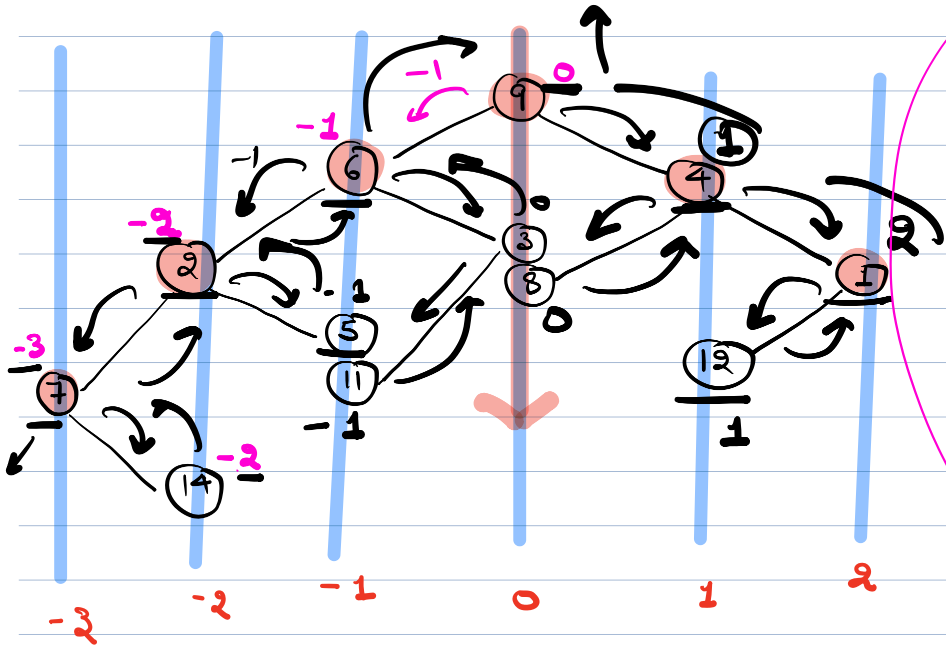
HM<Element, Index>

pre[], in[] ⇒ globSol.

```
Node   buildTree(Sin, ein, Spre, epre){

      if (Sin > ein){ return NULL;}
      Node  root = new Node(pre[Spre]);
      int  idx = map.get(pre[Spre]);
      int  sizeLST = (idx-1) - Sin + 1
                   = idx - Sin;

      root.left = buildTree(Sin, idx-1,
                   Spre+1, Spre+sizeLST);

      root.right = buildTree(idx+1, ein,
                   Spre+sizeLST+1, epre)
```

```
    return root;
}
```

Doubts



```
[
 -3 ← [7],
 -2 ← [2, 14],
 -1 ← [6, 5, 11],
  0 ← [9, 3, 8],
  1 ← [4, 12],
  2 ← [1],
]
```

Key	Value
0	[9, 3, 8]
-1	[6, 5, 11]
-2	[2, 14,]
-3	[7,]
1	[4, 12]
2	[1]

```
for (i = min ; i <= max ; i++)
```