PROBLEM

Level order traversal

Easy Accuracy: 70.31% Submissions: 185K+ Points: 2

Given a **root** of a binary tree with **n** nodes, find its level order traversal.
Level order traversal of a tree is breadth-first traversal for the tree.

Example 1:

Input:

```
    1
  /   \
 3     2
```

Output:
1 3 2

Example 2:

Input:

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

Output:
10 20 30 40 60

Your Task:
You don't have to take any input. Complete the function **levelOrder()** that takes the root node as input parameter and returns a list of integers containing the level order traversal of the given Binary Tree.

Expected Time Complexity: O(n)
Expected Auxiliary Space: O(n)

Constraints:
$1 \leq n \leq 10^5$
$0 \leq$ Data of a node $\leq 10^9$

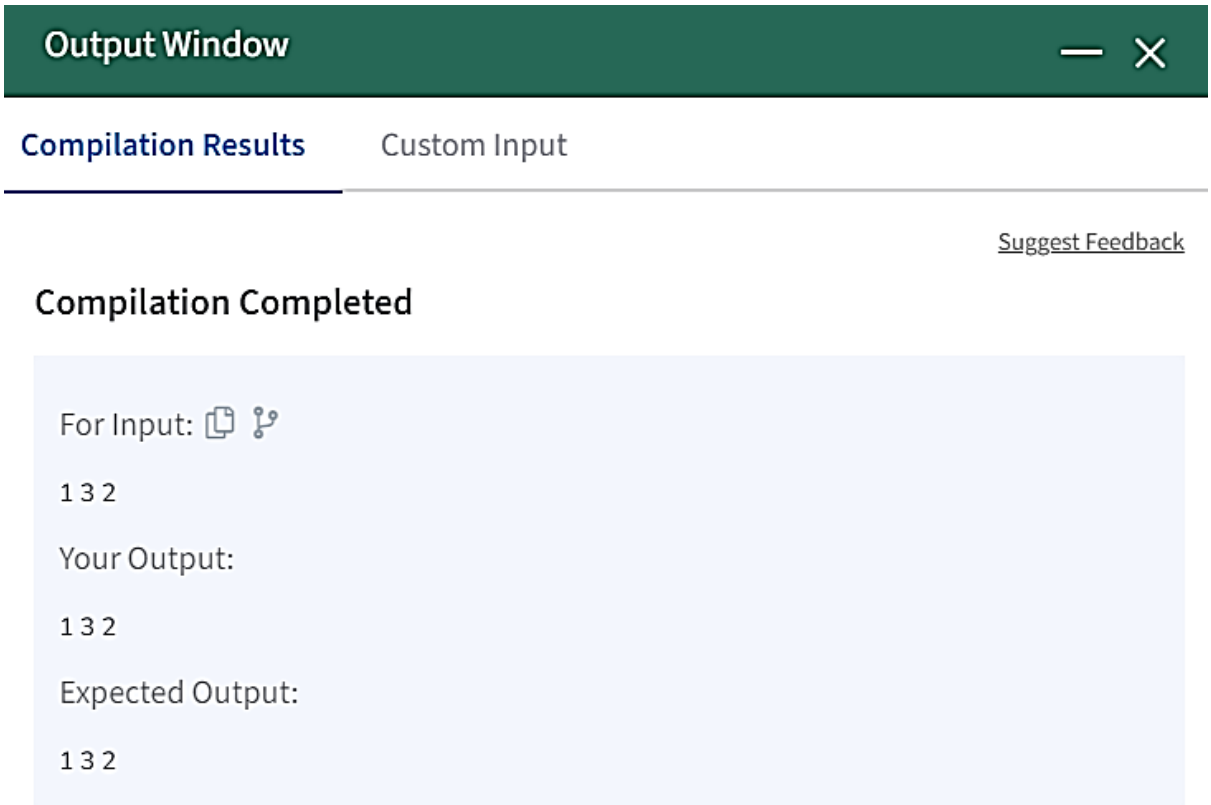
CODE

```
/*
class Node
{
   int data;
   Node left, right;

   Node(int item)
   {
      data = item;
      left = right = null;
   }
```

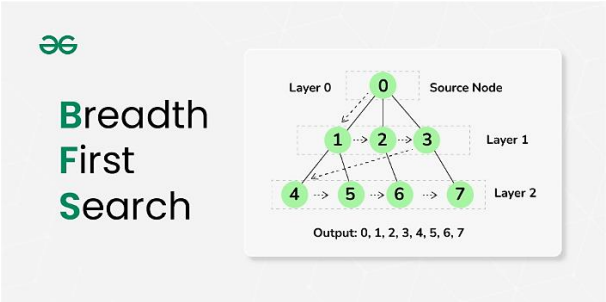
```
}
*/
class Solution
{
   //Function to return the level order traversal of a tree.
   static ArrayList <Integer> levelOrder(Node root)
   {
     // Write Your code here
     ArrayList <Integer> ln = new ArrayList<>();
     Queue<Node> q = new LinkedList <>();
     q.add(root);
     while(!q.isEmpty())
     {
        Node n = q.remove();
        if(n.left!=null)q.add(n.left);
        if(n.right!=null)q.add(n.right);
        ln.add(n.data);
     }
     return ln;
   }
}
```

OUTPUT



EXPLANATION

Level Order Traversal (Breadth First Search or BFS) of Binary Tree: Level Order Traversal technique is defined as a method to traverse a Tree such that all nodes present in the same level are traversed completely before traversing the next level.



Node Class: This is like a blueprint for each element in our tree. Each node holds a number (data), and it can have a left and a right child.

Solution Class and levelOrder Method:

- This Solution class contains a method called levelOrder, which helps us traverse through the tree.
- It starts at the top of the tree (the root) and visits each level one by one, from left to right.

ArrayList and Queue:

- An ArrayList is like a flexible array that can grow or shrink as needed. Here, it's used to store the numbers we find in the tree in the order we encounter them.
- A Queue is like a line of people waiting their turn. In this case, it's a line of nodes waiting to be visited in the tree.

Traversal:

We start by putting the root of the tree into the queue.

Then, we repeat these steps:

- Take out the first node from the queue.
- If this node has a left child, put it into the queue.
- If this node has a right child, put it into the queue.
- Remember the number in this node.
- We keep doing this until there are no more nodes in the queue.

Result:

- As we visit each node, we remember the number in it and store it in our ArrayList.

- After we've visited all the nodes, we return this list of numbers. This list is our level order traversal of the tree, meaning it shows the numbers in the order we encountered them, level by level, from left to right.

In simple terms, this code helps us explore each "floor" of the tree one by one, from top to bottom, left to right, and write down the numbers we find on each "floor."