Boundary Traversal of binary tree

Medium Accuracy: 26.78% Submissions: 100k+ Points: 4

Given a Binary Tree, find its Boundary Traversal. The traversal should be in the following order:

1. **Left boundary nodes:** defined as the path from the root to the left-most node ie- the leaf node you could reach when you always travel preferring the left subtree over the right subtree.
2. **Leaf nodes:** All the leaf nodes except for the ones that are part of left or right boundary.
3. **Reverse right boundary nodes:** defined as the path from the right-most node to the root. The right-most node is the leaf node you could reach when you always travel preferring the right subtree over the left subtree. Exclude the root from this as it was already included in the traversal of left boundary nodes.

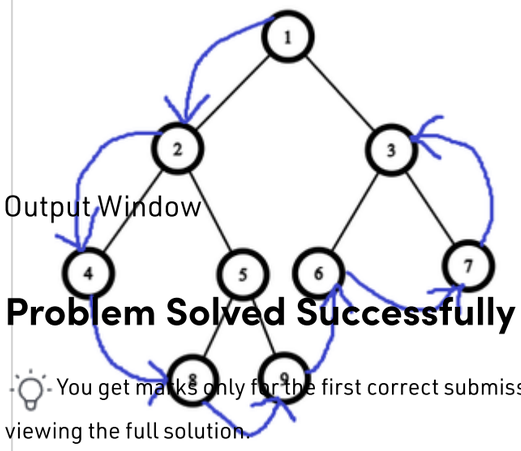
Note: If the root doesn't have a left subtree or right subtree, then the root itself is the left or right boundary.

Example 1:

Input:

```
        1
      /   \
     2     3
    / \   / \
   4   5 6   7
      / \
     8   9
```

Output: 1 2 4 8 9 6 7 3
Explanation:



Output Window

Problem Solved Successfully

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed: **93**/93

Total Time Taken: **0.15**/1.26

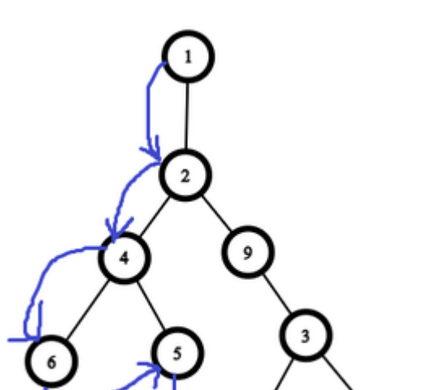
Example 93

Input:

```
            1
          /
        2
      /   \
    6     5     3
              /   \
             7     8
```

Output: 1 2 4 6 5 7 8
Explanation:

As you can see we have not taken right subtree. See **Note** (https://contribute.geek



Correct Submission Count: **2**

Attempts No.: **4**

Your Task:
This is a function problem. You don't have to take input. Just complete the **function boundary()** that takes the root node as input and returns an array containing the boundary values in anti-clockwise.

```
126         if(root != originalRoot and root->left == NULL and root->right == NULL)
127             ans.push_back(root->data);
128         leafTraverse(root->right, ans, originalRoot);
129     }
130     void rightTraverse(Node* root, vector<int>& right){
131         if(root == NULL)
132             return;
133         while(root->left or root->right){
134             right.push_back(root->data);
135             if(root->right)
136                 root = root->right;
137             else root = root->left;
138         }
139
140
```

C++ (g++ 5.4) Test against custom input

Average Time: 35m
Your Time: 79m 20s

Compile & Run Submit