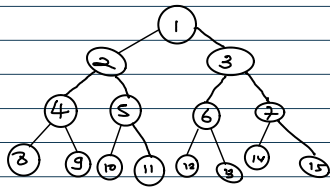


Infix, Prefix, Postfix Display of a Binary Tree:-

Prefix Representation:-



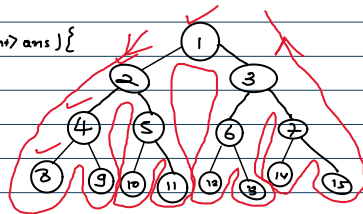
Now in order to print this prefix order our code would be:-

Work, Calling left subtree, Calling right subtree

Root, Left, Right

Code:-

```
void Prefix(TreeNode* root, Vector<int> ans) {
    if (root == NULL) return;
    ans.push_back(root->val);
    Prefix(root->left, ans);
    Prefix(root->right, ans);
}
```



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	4	8	9	5	10	12	13	6	14	7	15			

This is how the vector would be filled and returned

As An answer...

Code

```

1  * TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
2  * };
3  *
4  *
5  *
6  *
7  *
8  *
9  *
10 *
11 *
12 class Solution {
13 public:
14     void helper(TreeNode* root, vector<int> &ans) {
15         if (root == NULL) return;
16         ans.push_back(root->val);
17         helper(root->left, ans);
18         helper(root->right, ans);
19     }
20     vector<int> preorderTraversal(TreeNode* root) {
21         vector<int> ans;
22         helper(root, ans);
23         return ans;
24     }
25 };

```

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4

Input

root =

[1]

Accepted 71/71 testcases passed
PRIVANSH KUMAR submitted at Mar 19, 2025 18:13

Runtime: 0 ms Beats 100.00%
Memory: 10.77 MB Beats 93.25%

```

// Definition for a binary tree node.
// struct TreeNode {
//     int val;
//     TreeNode *left;
//     TreeNode *right;
//     TreeNode() : val(0), left(nullptr), right(nullptr) {}
//     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
//     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
// };

class Solution {
public:
    void helper(TreeNode* root, vector<int> &ans) {
        if (root == NULL) return;
        ans.push_back(root->val);
        helper(root->left, ans);
        helper(root->right, ans);
    }

    vector<int> preorderTraversal(TreeNode* root) {
        vector<int> ans;
        helper(root, ans);
        return ans;
    }
};

```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4

Input: root = [1]

Handwritten notes:
 - the value is pushed firstly then left subtree root is called and there after the right subtree is called.
 - The Principle of Root, Left, Right is followed.

Now for postfix and Infix pushing of Value in the Ans Vector just the order of pushing Would Be Changed...

for Infix pushing of Value firstly Left Subtree would Be Called then the pushing of Value Would take Place and then Right subtree would Be called..

And Now for the Postfix expression filling of the data in the Ans Vector would Be Done at last firstly the Right Subtree, and left Subtree Would Be Called...

Below is the code Submission for infix and Postfix expressions...

Accepted 71/71 testcases passed
PRIVANSH KUMAR submitted at Mar 19, 2025 18:13

Runtime: 0 ms Beats 100.00%
Memory: 11.00 MB Beats 43.72%

```

// Definition for a binary tree node.
// struct TreeNode {
//     int val;
//     TreeNode *left;
//     TreeNode *right;
//     TreeNode() : val(0), left(nullptr), right(nullptr) {}
//     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
//     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
// };

class Solution {
public:
    void helper(TreeNode* root, vector<int> &ans) {
        if (root == NULL) return;
        helper(root->left, ans);
        ans.push_back(root->val);
        helper(root->right, ans);
    }

    vector<int> inorderTraversal(TreeNode* root) {
        vector<int> ans;
        helper(root, ans);
        return ans;
    }
};

```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4

Input: root = [1,null,2,3]

Accepted 71/71 testcases passed
PRIVANSH KUMAR submitted at Mar 19, 2025 18:13

Runtime: 0 ms Beats 100.00%
Memory: 11.00 MB Beats 43.72%

```

// Definition for a binary tree node.
// struct TreeNode {
//     int val;
//     TreeNode *left;
//     TreeNode *right;
//     TreeNode() : val(0), left(nullptr), right(nullptr) {}
//     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
//     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
// };

class Solution {
public:
    void helper(TreeNode* root, vector<int> &ans) {
        if (root == NULL) return;
        helper(root->left, ans);
        helper(root->right, ans);
        ans.push_back(root->val);
    }

    vector<int> postorderTraversal(TreeNode* root) {
        vector<int> ans;
        helper(root, ans);
        return ans;
    }
};

```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4

Input: root = [1,null,2,3]