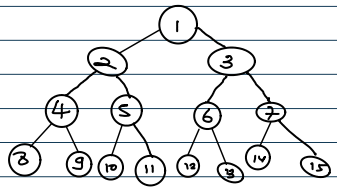
Infix, Prefix, Postfix display of a Binary Tree :-

Prefix Representation :-



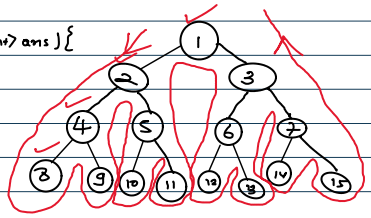
Now In order to print this prefix order our code would be :-

Work : Calling left subtree, Calling right subtree

Root, Left, Right

Code :-

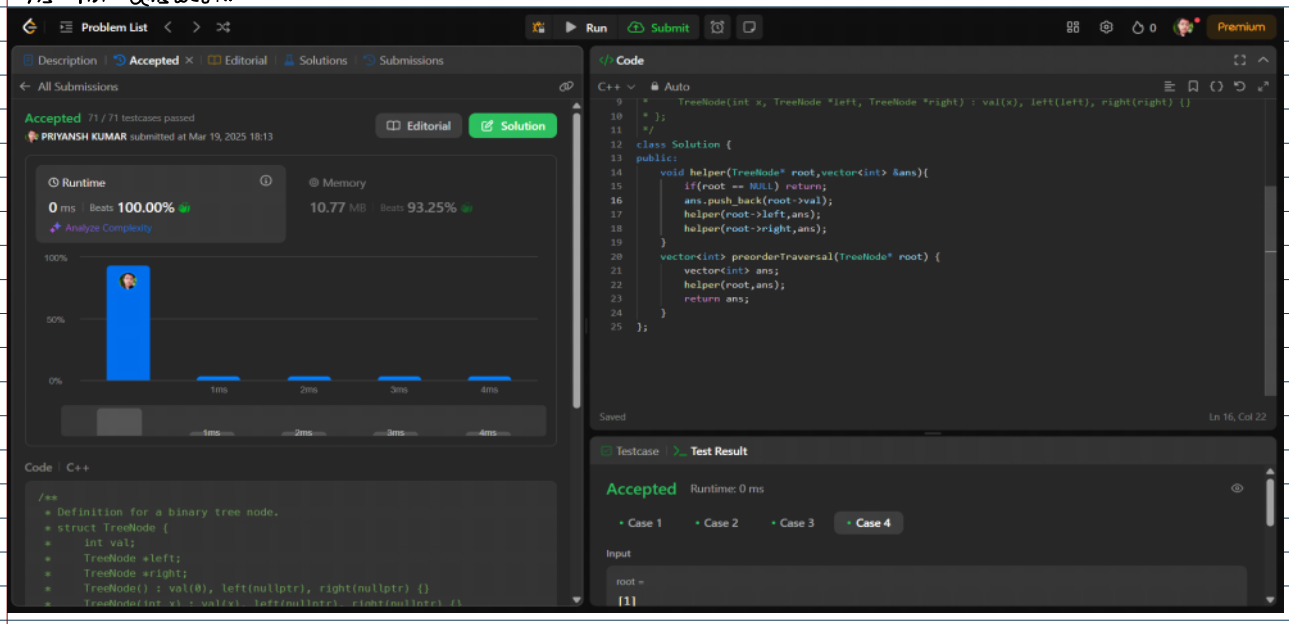
```
Void Prefix(TreeNode* root, Vector<int> ans){
    if(root == NULL) return;
    ans.push_back(root->val);
    Prefix(root->left, ans);
    Prefix(root->Right, ans);
}
```

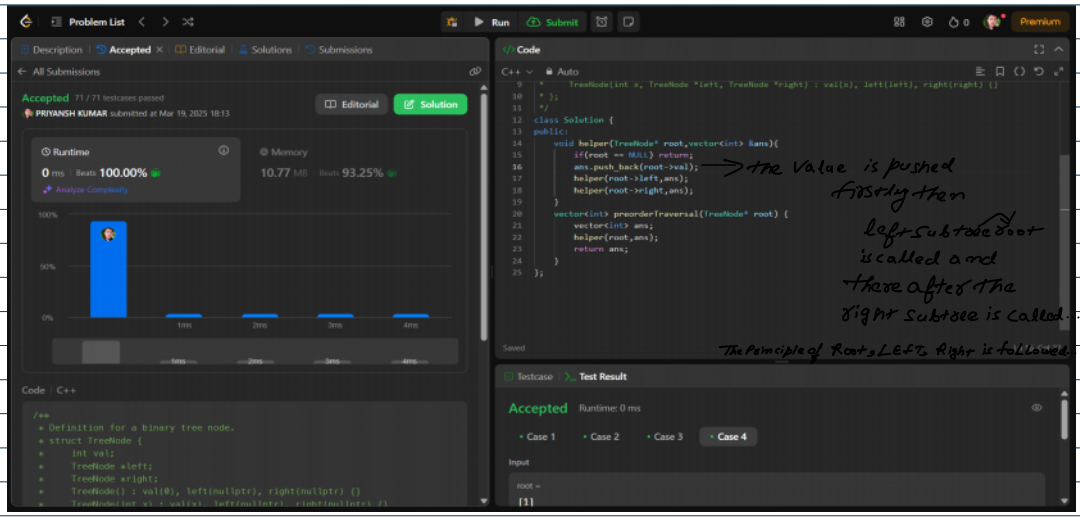


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	2	4	8	9	5	10	11	3	6	12	13	7	14	15

This is how the vector would be filled and returned

As An answer...





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...