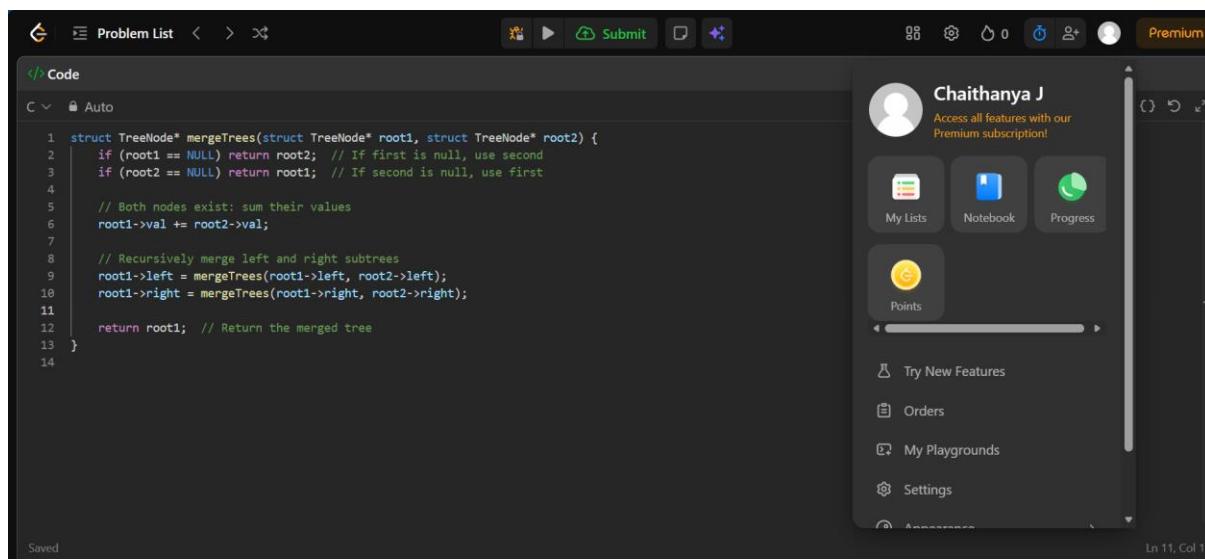


# Leetcode:

```
struct TreeNode* mergeTrees(struct TreeNode* root1, struct TreeNode* root2) {  
    if (root1 == NULL) return root2; // If first is null, use second  
    if (root2 == NULL) return root1; // If second is null, use first  
  
    // Both nodes exist: sum their values  
    root1->val += root2->val;  
  
    // Recursively merge left and right subtrees  
    root1->left = mergeTrees(root1->left, root2->left);  
    root1->right = mergeTrees(root1->right, root2->right);  
  
    return root1; // Return the merged tree  
}
```



# OUTPUT:

The screenshot shows a dark-themed interface for an online judge. At the top, there are navigation icons and a "Premium" badge. Below that, the title "Test Result" is displayed, followed by the status "Accepted" and runtime "0 ms". Two test cases are listed: "Case 1" and "Case 2", both of which have passed. The "Input" section contains two lines of code defining binary tree nodes. The "Output" section shows the expected result of the program's execution. The "Expected" section also displays the same result. A "Contribute a testcase" button is located at the bottom right.

```
root1 =  
[1,3,2,5]  
  
root2 =  
[2,1,3,null,4,null,7]  
  
Output  
[3,4,5,5,4,null,7]  
  
Expected  
[3,4,5,5,4,null,7]
```

This screenshot shows another instance of the online judge interface. It has a similar layout with a "Premium" badge at the top right. The title "Test Result" is present, along with the status "Accepted" and runtime "0 ms". Two test cases are shown: "Case 1" and "Case 2", both of which have passed. The "Input" section contains two lines of code defining binary tree nodes. The "Output" section shows the expected result of the program's execution. The "Expected" section also displays the same result. A "Contribute a testcase" button is located at the bottom right.

```
root1 =  
[1]  
  
root2 =  
[1,2]  
  
Output  
[2,2]  
  
Expected  
[2,2]
```

# OBSERVATION:

21/12/25

Pg - 8 b.

Page No:

Date:

Leetcode :-

Merge two Binary Trees.

Code:-

```
struct TreeNode* mergeTrees(struct TreeNode* root1, struct TreeNode* root2) {
    if (root1 == NULL) return root2;
    if (root2 == NULL) return root1;

    root1->val += root2->val;
    root1->left = mergeTrees(root1->left, root2->left);
    root1->right = mergeTrees(root1->right, root2->right);

    return root1;
}
```

Output of Case 1 :-

Input  $\Rightarrow$  root1 = [1, 3, 2, 5]  
root2 = [0, 1, 3, null, 4, null, 7]

Output  $\Rightarrow$  [3, 4, 5, 5, 4, null, 7]

Case 2 :-

Input :- root1 = [1]      root2 = [1, 2]  
Output :- [2, 2]