LEET CODE – Merge Two Binary Trees

You are given two binary trees root1 and root2.

Imagine that when you put one of them to cover the other, some nodes of the two trees are overlapped while the others are not. You need to merge the two trees into a new binary tree. The merge rule is that if two nodes overlap, then sum node values up as the new value of the merged node. Otherwise, the NOT null node will be used as the node of the new tree.

Return the merged tree.

Note: The merging process must start from the root nodes of both trees.

CODE:

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
struct TreeNode* mergeTrees(struct TreeNode* root1, struct TreeNode*
root2) {
if (root1 == NULL) return root2;
if (root2 == NULL) return root1;

struct TreeNode* merged = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
merged->val = root1->val + root2->val;
merged->left = mergeTrees(root1->left, root2->left);
merged->right = mergeTrees(root1->right, root2->right);

return merged;
}
```

OUTPUT :

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

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]

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

root1 =

[1]

root2 =

[1,2]

Output

[2,2]

Expected

[2,2]