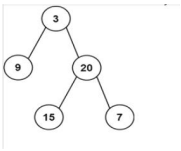
12.

Given two integer arrays preorder and inorder where preorder is the preorder traversal of a binary tree and inorder is the inorder traversal of the same tree, construct and return the binary tree.



Input: preorder = [3,9,20,15,7], inorder = [9,3,15,20,7]

Output: [3,9,20,null,null,15,7]

CODE:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct TreeNode {
    int val;
    struct TreeNode* left;
    struct TreeNode* right;
} TreeNode;

TreeNode* buildTree(int* preorder, int preorderSize, int* inorder, int inorderSize) {
    if (preorderSize == 0) return NULL;

    TreeNode* root = (TreeNode*)malloc(sizeof(TreeNode));
    root->val = preorder[0];
    root->left = root->right = NULL;

    int idx = -1;
    for (int i = 0; i < inorderSize; i++) {
```

```
            if (inorder[i] == preorder[0]) {
                idx = i;
                break;
            }
        }
        root->left = buildTree(preorder + 1, idx, inorder, idx);
        root->right = buildTree(preorder + idx + 1, preorderSize - idx - 1, inorder + idx + 1,
inorderSize - idx - 1);

        return root;
}

void printTree(TreeNode* root) {
        if (root == NULL) return;
        printf("%d ", root->val);
        printTree(root->left);
        printTree(root->right);
}

int main() {
        int preorder[] = {3, 9, 20, 15, 7};
        int inorder[] = {9, 3, 15, 20, 7};
        int preorderSize = sizeof(preorder) / sizeof(preorder[0]);
        int inorderSize = sizeof(inorder) / sizeof(inorder[0]);

        TreeNode* root = buildTree(preorder, preorderSize, inorder, inorderSize);
        printTree(root);
        printf("\n");
```

```
    return 0;
}
```

output:

3 9 20 15 7