

```

#include <stdio.h>

#include <stdlib.h>

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

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

int main() {
    int preorder[] = {3, 9, 20, 15, 7};
    int inorder[] = {9, 3, 15, 20, 7};
    int n = sizeof(preorder) / sizeof(preorder[0]);
    struct TreeNode* stack[100];
    int top = -1;
    struct TreeNode* root = malloc(sizeof(struct TreeNode));
    root->val = preorder[0];
    root->left = root->right = NULL;
    stack[++top] = root;
    int i,j;
    for (i = 1, j = 0; i < n; i++) {
        struct TreeNode* temp = NULL;
        struct TreeNode* node = malloc(sizeof(struct TreeNode));
        node->val = preorder[i];
        node->left = node->right = NULL;
        while (top != -1 && stack[top]->val == inorder[j]) {

```

```

temp = stack[top--];

j++;

}

if (temp != NULL) {
temp->right = node;
} else {
stack[top]->left = node;
}

stack[++top] = node;
}

printf("Inorder Traversal of the Constructed Tree: ");

inorderTraversal(root);

printf("\n");

return 0;
}

```

The screenshot shows the Programiz Online C Compiler interface. The code editor on the left contains a C program that constructs a binary tree from a preorder traversal and then performs an inorder traversal. The output window on the right displays the result of the traversal.

```

main.c
21 struct TreeNode* root = malloc(sizeof(struct TreeNode));
22 root->val = preorder[0];
23 root->left = root->right = NULL;
24 stack[++top] = root;
25 int i,j;
26 for (i = 1, j = 0; i < n; i++) {
27     struct TreeNode* temp = NULL;
28     struct TreeNode* node = malloc(sizeof(struct TreeNode));
29     node->val = preorder[i];
30     node->left = node->right = NULL;
31     while (top != -1 && stack[top]->val == inorder[j]) {
32         temp = stack[top--];
33         j++;
34     }
35     if (temp != NULL) {
36         temp->right = node;
37     } else {
38         stack[top]->left = node;
39     }
40     stack[++top] = node;
41 }
42 printf("Inorder Traversal of the Constructed Tree: ");
43 inorderTraversal(root);
44 printf("\n");
45 return 0;
46 }

```

Output

```

/tmp/TGD7no6y8T.o
Inorder Traversal of the Constructed Tree: 9 3 15 20 7

```