```c
#include <stdio.h>
#define MAX 100
struct Node {
    int data;
    int left;
    int right;
};
struct Node tree[MAX];
int nodeCount = 0;
int createNode(int data) {
    tree[nodeCount].data = data;
    tree[nodeCount].left = -1;
    tree[nodeCount].right = -1;
    return nodeCount++;
}
int search(int inorder[], int start, int end, int value) {
    for (int i = start; i <= end; i++)
        if (inorder[i] == value)
            return i;
    return -1;
}
```

```c
int buildTree(int preorder[], int inorder[], int start, int end, int* preIndex) {
    if (start > end) return -1;
    int current = createNode(preorder[*preIndex]);
    (*preIndex)++;
    if (start == end)
        return current;
    int inIndex = search(inorder, start, end, tree[current].data);
    tree[current].left = buildTree(preorder, inorder, start, inIndex - 1, preIndex);
    tree[current].right = buildTree(preorder, inorder, inIndex + 1, end, preIndex);
    return current;
}
void printLevelOrder(int rootIndex) {
    int queue[MAX];
    int front = 0, rear = 0;
    queue[rear++] = rootIndex;
    while (front < rear) {
        int curr = queue[front++];
        if (curr == -1) {
            printf("null ");
            continue;
        }
        printf("%d ", tree[curr].data);
        queue[rear++] = tree[curr].left;
        queue[rear++] = tree[curr].right;
    }
}
```

```c
int main() {
    int preorder[] = {3, 9, 20, 15, 7};
    int inorder[]  = {9, 3, 15, 20, 7};
    int n = sizeof(preorder) / sizeof(preorder[0]);
    int preIndex = 0;
    int rootIndex = buildTree(preorder, inorder, 0, n - 1, &preIndex);
    printf("Level order (null shown): ");
    printLevelOrder(rootIndex);
    return 0;
}
```

```
Level order (null shown): 3 9 20 null null 15 7
null null null null
```