

Inorder traversal using stack

```
class Solution {
public:
    // Function to return a list containing the inorder traversal of the tree.
    vector<int> inOrder(Node* root) {
        vector<int> result;
        stack<Node*> st;
        Node* curr = root;

        while (curr != nullptr || !st.empty()) {
            while (curr != nullptr) {
                st.push(curr);
                curr = curr->left;
            }
            curr = st.top();
            st.pop();
            result.push_back(curr->data);
            curr = curr->right;
        }

        return result;
    }
};
```

The screenshot shows the GeeksforGeeks online compiler interface. The URL is <https://www.geeksforgeeks.org/problems/inorder-traversal/1>. The code is written in C++ (g++ 5.4) and is titled "Problem". The code is as follows:

```
24 public:
25     int data;
26     Node* left;
27     Node* right;
28
29     // Constructor to initialize a new node
30     Node(int val) {
31         data = val;
32         left = NULL;
33         right = NULL;
34     }
35 };
36
37
38 class Solution {
39 public:
40     // Function to return a list containing the inorder traversal of the tree.
41     vector<int> inOrder(Node* root) {
42         vector<int> result;
43         stack<Node*> st;
44         Node* curr = root;
45
46         while (curr != nullptr || !st.empty()) {
47             while (curr != nullptr) {
48                 st.push(curr);
49                 curr = curr->left;
50             }
51             curr = st.top();
52             st.pop();
53             result.push_back(curr->data);
54             curr = curr->right;
55         }
56
57         return result;
58     }
59 };
```

The "Output Window" shows the "Compilation Results" for "Case 1". The input is "1 2 3 4 5" and the output is "4 2 5 1 3". The "Expected Output" is also "4 2 5 1 3".