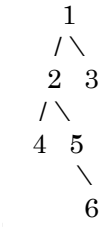
Morris traversal in C++

```
#include <iostream>
#include <vector>
using namespace std;
// TreeNode structure definition
struct TreeNode {
    int key;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) {
        key = x;
        left = nullptr;
        right = nullptr;
    }
};

// Function to perform Morris preorder traversal
vector<int> preorderTraversal(TreeNode* root) {
    vector<int> preorder;
    TreeNode* cur = root;

    while (cur != nullptr) {
        if (cur->left == nullptr) {
            preorder.push_back(cur->key);
            cur = cur->right;
        } else {
            TreeNode* prev = cur->left;
            while (prev->right != nullptr && prev->right !
= cur) {
                prev = prev->right;
            }

            if (prev->right == nullptr) {
                prev->right = cur;
                preorder.push_back(cur->key);
                cur = cur->left;
            } else {
                prev->right = nullptr;
                cur = cur->right;
            }
        }
    }

    return preorder;
}

int main() {
    // Constructing the binary tree
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->left = new TreeNode(4);
    root->left->right = new TreeNode(5);
    root->left->right->right = new TreeNode(6);

    // Performing Morris preorder traversal
    vector<int> preorder = preorderTraversal(root);

    // Printing the result
    cout << "The Preorder Traversal is: ";
    for (int i = 0; i < preorder.size(); i++) {
```

Tree Structure



🧠 Morris Preorder Key Idea

- Use the **rightmost node** in the left subtree to **thread** back to the current node.
- When revisiting via the thread, remove the link and move right.

□ Dry Run Table

We'll walk through the preorderTraversal function.

Step	cur	Action	preorder	Thread Created?
1	1	Left exists → find predecessor (5)	[1]	✅ prev->right = 1
2	2	Left exists → find predecessor (4)	[1, 2]	✅ prev->right = 2
3	4	No left child → visit, move right (nullptr)	[1, 2, 4]	❌
4	2	Thread exists → remove, move right to 5		🔁
5	5	No left child → visit, move right to 6	[1, 2, 4, 5]	❌
6	6	No left child → visit, move right (nullptr)	[1, 2, 4, 5, 6]	❌
7	1	Thread exists → remove, move right to 3		🔁
8	3	No left child → visit, move right (nullptr)	[1, 2, 4, 5, 6, 3]	❌

✅ Final Output:

The Preorder Traversal is: 1 2 4 5 6 3

```
        cout << preorder[i] << " ";
    }
    cout << endl;

    // Deallocating memory
    delete root->left->right->right;
    delete root->left->right;
    delete root->left;
    delete root->right;
    delete root;

    return 0;
}
```

The Preorder Traversal is: 1 2 4 5 6 3