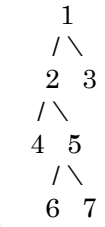
Root 2 Node path in C++

```
#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 get the path from root to a node with
value x
bool getPath(TreeNode* root, vector<int>& arr, int x)
{
    // If root is NULL, there is no path
    if (root == nullptr)
        return false;

    // Push the node's value into 'arr'
    arr.push_back(root->key);

    // If it is the required node, return true
    if (root->key == x)
        return true;

    // Check in the left subtree and right subtree
    if (getPath(root->left, arr, x) || getPath(root-
>right, arr, x))
        return true;

    // If the required node does not lie in either subtree,
    // remove current node's value from 'arr' and return
false
    arr.pop_back();
    return false;
}

int main() {
    // Constructing the binary tree
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->left->left = new TreeNode(4);
    root->left->right = new TreeNode(5);
    root->left->right->left = new TreeNode(6);
    root->left->right->right = new TreeNode(7);
    root->right = new TreeNode(3);

    vector<int> arr;

    bool res = getPath(root, arr, 7);

    if (res) {
        cout << "The path is: ";
        for (int it : arr) {
            cout << it << " ";
```

Tree Structure



🧭 **Target: 7**

We'll step through getPath(root, arr, 7).

Step	Current Node	arr Content	Found?
1	1	[1]	❌
2	2	[1, 2]	❌
3	4	[1, 2, 4]	❌ → backtrack
4	Backtrack	[1, 2]	
5	5	[1, 2, 5]	❌
6	6	[1, 2, 5, 6]	❌ → backtrack
7	Backtrack	[1, 2, 5]	
8	7	[1, 2, 5, 7]	✅ Found!

✅ **Final Output:**

The path is: 1 2 5 7

```
        }
        cout << endl;
    } else {
        cout << "Node not found in the tree." << endl;
    }

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

    return 0;
}
```

The path is: 1 2 5 7