Identical in C++

```cpp
#include <iostream>

using namespace std;

// Definition for a binary tree node.
struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) {
        val = x;
        left = nullptr;
        right = nullptr;
    }
};

class Identical {
public:
    static bool isIdentical(TreeNode* node1, TreeNode* node2) {
        if (node1 == nullptr && node2 == nullptr)
            return true;
        else if (node1 == nullptr || node2 == nullptr)
            return false;

        return (node1->val == node2->val) &&
               isIdentical(node1->left, node2->left) &&
               isIdentical(node1->right, node2->right);
    }
};

int main() {
    TreeNode* root1 = new TreeNode(1);
    root1->left = new TreeNode(2);
    root1->right = new TreeNode(3);
    root1->right->left = new TreeNode(4);
    root1->right->right = new TreeNode(5);

    TreeNode* root2 = new TreeNode(1);
    root2->left = new TreeNode(2);
    root2->right = new TreeNode(3);
    root2->right->left = new TreeNode(4);
```

Tree Structures:

Tree 1:

```
    1
   / \
  2   3
     / \
    4   5
```

Tree 2:

```
    1
   / \
  2   3
     /
    4
```

🧮 Dry Run Table: isIdentical(root1, root2)

Call	node1 Val	node2 Val	Equal?	Recursive Calls	Final Result
isIdentical(1, 1)	1	1	✅	isIdentical(2, 2) && isIdentical(3, 3)	depends
└── isIdentical(2, 2)	2	2	✅	isIdentical(nullptr, nullptr)	✅
└── isIdentical(NULL, NULL)	NULL	NULL	✅		✅
└── isIdentical(NULL, NULL)	NULL	NULL	✅		✅
└── isIdentical(3, 3)	3	3	✅	isIdentical(4, 4) && isIdentical(5, NULL)	❌
└── isIdentical(4, 4)	4	4	✅	isIdentical(NULL, NULL)	✅
└── isIdentical(NULL,NULL)	NULL	NULL	✅		✅
└── isIdentical(NULL,NULL)	NULL	NULL	✅		✅
└── isIdentical(5, NULL)	5	NULL	❌		❌

❌ **Final Output:**

Two trees are non-identical

```
    if
(Identical::isIdentical(root1
, root2))
        cout << "Two Trees
are identical" << endl;
    else
        cout << "Two trees are
non-identical" << endl;

    return 0;
}
```

Two trees are non-identical