Balanced in C++

```
#include <iostream>
#include <algorithm>
using namespace std;

// Node structure for the binary tree
struct Node {
    int key;
    Node* left;
    Node* right;

    Node(int item) {
        key = item;
        left = right = nullptr;
    }
};

// Function to calculate the height of the tree and check balance
pair<bool, int> isBalancedHelper(Node* root) {
    if (root == nullptr)
        return {true, 0};

    // Recursively get heights of left and right subtrees
    auto left = isBalancedHelper(root->left);
    auto right = isBalancedHelper(root->right);

    // If either subtree is unbalanced, the whole tree is unbalanced
    if (!left.first || !right.first)
        return {false, -1};

    // Check if the current subtree is balanced
    if (abs(left.second - right.second) > 1)
        return {false, -1};

    // Return balanced status and height of the current subtree
    return {true, max(left.second, right.second) + 1};
}

// Function to check if the binary tree is balanced
bool isBalanced(Node* root) {
    return isBalancedHelper(root).first;
}

int main() {
    Node* root = new Node(1);
    root->left = new Node(2);
    root->right = new Node(3);
    root->left->left = new Node(4);
    root->left->right = new Node(5);
    root->left->left->left = new
```

Binary Tree Structure

```
      1
     / \
    2   3
   / \
  4   5
 /
6
```

🧮 Dry Run Table: isBalancedHelper

We'll do a **postorder traversal** (left → right → root) and track the balance and height of each subtree.

Node	Left Subtree (Balanced, Height)	Right Subtree (Balanced, Height)	Height Difference	Is Current Balanced?	Current Height
6	(true, 0)	(true, 0)	0	✅ Yes	1
4	(true, 1)	(true, 0)	1	✅ Yes	2
5	(true, 0)	(true, 0)	0	✅ Yes	1
2	(true, 2)	(true, 1)	1	✅ Yes	3
3	(true, 0)	(true, 0)	0	✅ Yes	1
1	(true, 3)	(true, 1)	**2**	❌ No	—

❌ **Final Result:**

- Node 1 is **not balanced** because its left and right subtrees have a height difference of **2**, which is more than 1.
- Hence, isBalanced(root) returns false.

✅ **Output:**

Is the tree balanced? No

```
Node(6);

    bool balanced = isBalanced(root);
    cout << "Is the tree balanced? " <<
(balanced ? "Yes" : "No") << endl;

    return 0;
}
```

Is the tree balanced? No