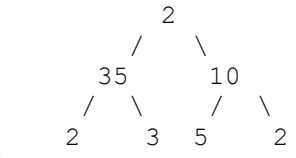
Children Sum in C++

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

// Definition of the Node class
class Node {
public:
    int key;
    Node* left;
    Node* right;

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

// Function to reorder the binary tree based on
Children Sum Property
void reorder(Node* root) {
    if (root == nullptr) return;

    int child = 0;
    if (root->left != nullptr) {
        child += root->left->key;
    }
    if (root->right != nullptr) {
        child += root->right->key;
    }

    if (child < root->key) {
        if (root->left != nullptr) root->left->key = root-
>key;
        else if (root->right != nullptr) root->right->key =
root->key;
    }

    reorder(root->left);
    reorder(root->right);

    int tot = 0;
    if (root->left != nullptr) tot += root->left->key;
    if (root->right != nullptr) tot += root->right->key;
    if (root->left != nullptr || root->right != nullptr)
root->key = tot;
}

// Function to change the tree based on Children Sum
Property
void changeTree(Node* root) {
    reorder(root);
}

int main() {
    Node* root = new Node(2);
    root->left = new Node(35);
    root->left->left = new Node(2);
    root->left->right = new Node(3);
    root->right = new Node(10);
    root->right->left = new Node(5);
    root->right->right = new Node(2);
```

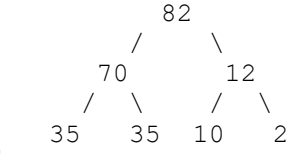
Initial Tree Structure



🔄 Dry Run: Step-by-Step Execution

Node Visited	Children Before	Action Taken	Node Key After
2 (root)	35 + 10 = 45	Children > root → No update to children	—
35	2 + 3 = 5	Children < 35 → Set both children to 35	—
2 (left)	null	Leaf node	35
3 (right)	null	Leaf node	35
Back to 35	35 + 35 = 70	Set node key = sum of children	70
10	5 + 2 = 7	Children < 10 → Set left to 10 (since left exists)	—
5 (left)	null	Leaf node	10
2 (right)	null	Leaf node	2
Back to 10	10 + 2 = 12	Set node key = sum of children	12
Back to root	70 + 12 = 82	Set root = sum of its updated children	82

🌳 Final Tree Structure



✅ Output

```
Modified Tree:
Root: 82
Left: 70, Left Left: 35, Left Right: 35
Right: 12, Right Left: 10, Right Right:
2
```

```
    changeTree(root);

    // Display the modified tree
    cout << "Modified Tree:" << endl;
    cout << "Root: " << root->key << endl;
    cout << "Left: " << root->left->key << ", Left Left: "
<< root->left->left->key << ", Left Right: " << root-
>left->right->key << endl;
    cout << "Right: " << root->right->key << ", Right
Left: " << root->right->left->key << ", Right Right: "
<< root->right->right->key << endl;

    return 0;
}
```

🔁 Summary of Key Logic in `reorder()`:

1. **Preorder Phase**:
   - Push parent's value down to children if sum of children < parent.
2. **Postorder Phase**:
   - After children updated, update parent's value as sum of updated children.

```
Modified Tree:
Root: 50
Left: 38, Left Left: 35, Left Right: 3
Right: 12, Right Left: 10, Right Right: 2
```