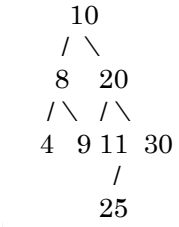
in C++

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

class Node {
public:
    int key;
    Node* left;
    Node* right;

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

class PairWithGivenSum {
public:
    static vector<int> treeToList(Node* root,
vector<int>& list) {
        if (root == nullptr)
            return list;

        treeToList(root->left, list);
        list.push_back(root->key);
        treeToList(root->right, list);

        return list;
    }

    static bool isPairPresent(Node* root, int target) {
        vector<int> nodeList;
        vector<int> sortedList = treeToList(root,
nodeList);

        int start = 0;
        int end = sortedList.size() - 1;

        while (start < end) {
            if (sortedList[start] + sortedList[end] ==
target) {
                cout << "Pair Found: " << sortedList[start]
<< " + " << sortedList[end] << " = " << target << endl;
                return true;
            } else if (sortedList[start] + sortedList[end] <
target) {
                start++;
            } else {
                end--;
            }
        }

        cout << "No such values are found!" << endl;
        return false;
    }
};

int main() {
    Node* root = new Node(10);
    root->left = new Node(8);
```

BST Structure



🧠 Step 1: Inorder Traversal

This step creates a **sorted array** of all node values.

Node Visited	List After Visit
4	[4]
8	[4, 8]
9	[4, 8, 9]
10	[4, 8, 9, 10]
11	[4, 8, 9, 10, 11]
20	[4, 8, 9, 10, 11, 20]
25	[4, 8, 9, 10, 11, 20, 25]
30	[4, 8, 9, 10, 11, 20, 25, 30]

Final Sorted List:
[4, 8, 9, 10, 11, 20, 25, 30]

🧠 Step 2: Two-Pointer Search

We now search for a pair that sums to 33.

Start Index	End Index	Pair Checked	Sum	Action
0 (4)	7 (30)	4 + 30	34	Too big → end--
0 (4)	6 (25)	4 + 25	29	Too small → start++
1 (8)	6 (25)	8 + 25	33	✅ Found! Return true

✅ Output:

Pair Found: 8 + 25 = 33

```
    root->right = new Node(20);
    root->left->left = new Node(4);
    root->left->right = new Node(9);
    root->right->left = new Node(11);
    root->right->right = new Node(30);
    root->right->right->left = new Node(25);

    int sum = 33;

    PairWithGivenSum::isPairPresent(root, sum);

    return 0;
}
```

```
Pair Found: 8 + 25 = 33
```