Floor and Ceil in C++

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

class BSTFloorCeil
{
public:
   struct Node
   {
      int data;
      Node *left;
      Node *right;

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

   Node *root;

   Node *Floor(Node *node, int x)
   {
      Node *res = nullptr;
      while (node != nullptr)
      {
         if (node->data == x)
         {
            return node;
         }
         if (node->data > x)
         {
            node = node->left;
         }
         else
         {
            res = node;
            node = node->right;
         }
      }
      return res;
   }

   int Ceil(Node *node, int x)
   {
      if (node == nullptr)
      {
         return -1;
      }
      if (node->data == x)
      {
         return node->data;
      }
      if (node->data < x)
      {
         return Ceil(node->right, x);
      }
      int ceil = Ceil(node->left, x);
      return (ceil >= x) ? ceil : node->data;
   }
```

BST Structure

Let's first visualize the tree:

```
      8
    /   \
   4     12
  / \   /  \
 2   6 10  14
```

You're querying for:

- **Floor of 7**
- **Ceiling of 7**

▾ Floor Function Walkthrough (tree.Floor(tree.root, 7))

Node* Floor(Node* node, int x)

We need the **largest value ≤ 7**.

Step	Current Node	Comparison (data vs 7)	Action	Floor Candidate
1	8	8 > 7	Go left	nullptr
2	4	4 < 7	Save 4, go right	4
3	6	6 < 7	Save 6, go right	6
4	null	-	Exit loop	6

✓ **Result:** Floor of 7 is **6**

▴ Ceil Function Walkthrough (tree.Ceil(tree.root, 7))

We need the **smallest value ≥ 7**.

int Ceil(Node* node, int x)

It's a recursive function.

Step	Node	Comparison (data vs 7)	Action	Result
1	8	8 > 7	Check left subtree	Left = 4
2	4	4 < 7	Recurse right → 6	

```
};

int main()
{
    BSTFloorCeil tree;

    // Construct the BST
    tree.root = new BSTFloorCeil::Node(8);
    tree.root->left = new BSTFloorCeil::Node(4);
    tree.root->right = new BSTFloorCeil::Node(12);
    tree.root->left->left = new BSTFloorCeil::Node(2);
    tree.root->left->right = new BSTFloorCeil::Node(6);
    tree.root->right->left = new
BSTFloorCeil::Node(10);
    tree.root->right->right = new
BSTFloorCeil::Node(14);

    // Find floor and ceiling
    BSTFloorCeil::Node *floorNode =
tree.Floor(tree.root, 7);
    int floorValue = (floorNode != nullptr) ? floorNode-
>data : -1;
    cout << "The floor is: " << floorValue << endl;

    int ceilValue = tree.Ceil(tree.root, 7);
    cout << "The ceiling is: " << ceilValue << endl;

    return 0;
}
```

Step	Node	Comparison (data vs 7)	Action	Result
3	6	6 < 7	Recurse right → null	Return -1
Back	4	ceil = -1, node.data=4	return node.data = 4	Not >= 7 → fail
Back	8	ceil = 4	4 < 7 → return 8	✓ Match

✓ **Result:** Ceiling of 7 is **8**

📄 Final Output
The floor is: 6
The ceiling is: 8

```
The floor is: 6
The ceiling is: 8
```