資料結構

Data Structure

Lab 10

姓名：曾致嘉

學號：113AB0014

Lab10-Q1

Q1: Please write a function that receives the number of layers and returns the sum of all nodes at that layer. If the input exceeds the actual height of the tree, show a warning message.

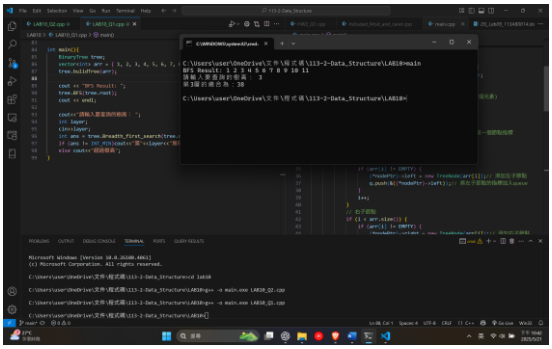
Code

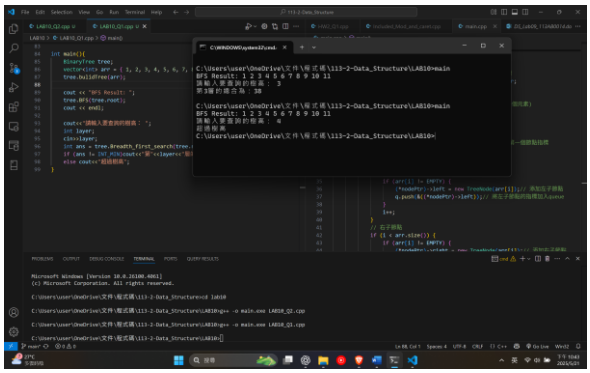
```
#include <iostream>
#include <queue>
#include <vector>
using namespace std;
const int EMPTY=INT_MIN;
class TreeNode{
public:
    int value;
    TreeNode* left;
    TreeNode* right;

    TreeNode(int val):value(val), left(nullptr), right(nullptr){}
};

class BinaryTree{
public:
    TreeNode* root;
    BinaryTree():root(nullptr){}

    TreeNode* bulidTree(const vector<int>& arr){
        if(arr.empty() || arr[0]==EMPTY)return nullptr;

        queue<TreeNode**> q;
        root = new TreeNode(arr[0]);
        q.push(&root);// 刻意塞入，彌補之後每一次要 pop

        size_t i =1;
        while (!q.empty() && i< arr.size()){
            TreeNode** nodePtr = q.front();
            q.pop();

            if(i<arr.size()){
```

```
                if (arr[i] != EMPTY){
                    (*nodePtr)->left = new TreeNode(arr[i]);
                    q.push(&((*nodePtr)->left));
                }
                i++;
            }
            if(i<arr.size()){
                if (arr[i] != EMPTY){
                    (*nodePtr)->right = new TreeNode(arr[i]);
                    q.push(&((*nodePtr)->right));
                }
                i++;
            }
        }
        return root;
    }
    void BFS(TreeNode* root) {
        if (root == nullptr) return;
        queue<TreeNode*> q;//建立 queue 儲存待處理的節點指標
        q.push(root); // 將根節點的指標加入 queue

        while (!q.empty()) {
            TreeNode* current = q.front();// 取出 queue 的第一個節點指標
            q.pop();// 將該節點從 queue 中刪除
            cout << current->value << " ";
            if (current->left) q.push(current->left);   // 將左子節點的指標加入 queue
            if (current->right) q.push(current->right);// 將左子節點的指標加入 queue
        }
    }
    int Breadth_first_search(TreeNode* root, int target_layer) {
        if (!root) return 0;// 拒絕空樹
        queue<TreeNode*> q;
        q.push(root);
        int current_layer = 0;

        while (!q.empty()) {
            int level_size = q.size(); //判斷父層有多少兄弟
            int level_sum = 0;
```

```
                for (int i=0;i<level_size;i++){
                    TreeNode* node = q.front(); q.pop();//依序取出兄弟們
                    if (current_layer == target_layer)level_sum+=node ->value;// 目標等級
                    if (node ->left) q.push(node->left); // 預處理下一層左兒子
                    if (node->right) q.push(node->right);// 預處理下一層右兒子
                }
                if (current_layer==target_layer)return level_sum; //算完目標層跳過後面
                current_layer++;
            }
            return INT_MIN; //找不到
        }
};

int main(){
    BinaryTree tree;
    vector<int> arr = { 1, 2, 3, 4, 5, 6, 7, 8, 9, EMPTY, EMPTY, 10, 11, EMPTY, EMPTY};
    tree.bulidTree(arr);

    cout << "BFS Result: ";
    tree.BFS(tree.root);
    cout << endl;

    cout<<"請輸入要查詢的樹高： ";
    int layer;
    cin>>layer;
    int ans = tree.Breadth_first_search(tree.root, layer);
    if (ans != INT_MIN)cout<<"第"<<layer<<"層的總合為："<< ans<<endl;
    else cout<<"超過樹高";
}
```

Discussion Section





```
C:\Users\user\OneDrive\文件\程式碼\11
BFS Result: 1 2 3 4 5 6 7 8 9 10 11
請輸入要查詢的樹高： 3
第3層的總合為：38
```

```
C:\Users\user\OneDrive\文件\程式碼\11
BFS Result: 1 2 3 4 5 6 7 8 9 10 11
請輸入要查詢的樹高： 4
超過樹高
```

Lab10-Q2

Q2: Please write a function that takes a node as input, returns the values in its left and right subtrees, and indicates which one is larger. If the node is a leaf or not in the tree, show a warning message.

Code

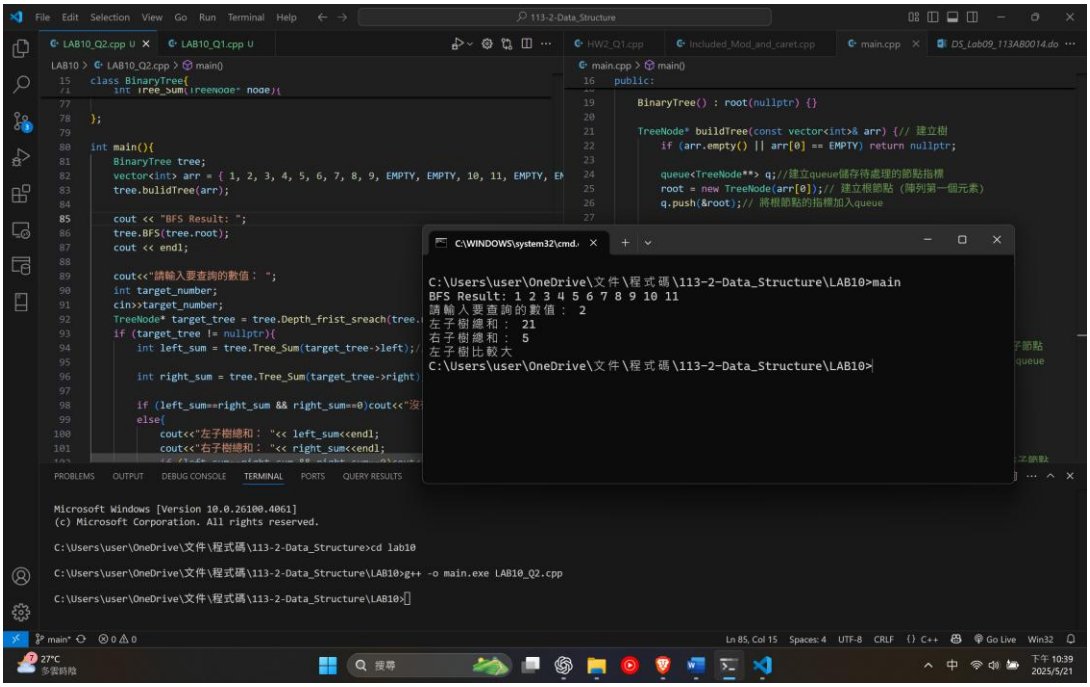
```
#include <iostream>
#include <queue>
#include <vector>
using namespace std;
const int EMPTY=INT_MIN;
class TreeNode{
public:
    int value;
    TreeNode* left;
    TreeNode* right;

    TreeNode(int val):value(val), left(nullptr), right(nullptr){}
};

class BinaryTree{
public:
    TreeNode* root;
    BinaryTree():root(nullptr){}
    TreeNode* bulidTree(const vector<int>& arr){
        if(arr.empty() || arr[0]==EMPTY)return nullptr;
        queue<TreeNode**> q;
        root = new TreeNode(arr[0]);
        q.push(&root);// 刻意塞入，彌補之後每一次要 pop

        size_t i =1;
        while (!q.empty() && i< arr.size()){
            TreeNode** nodePtr = q.front();
            q.pop();

            if(i<arr.size()){
                if (arr[i] != EMPTY){
                    (*nodePtr)->left = new TreeNode(arr[i]);
```

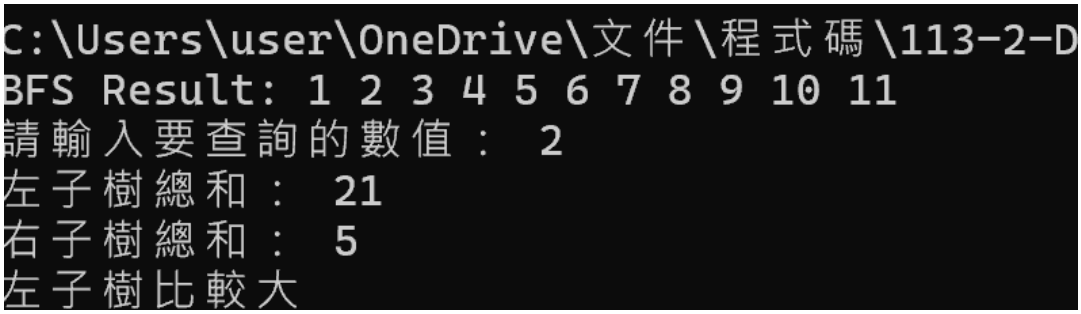
```
                q.push(&((*nodePtr)->left));
            }
            i++;
        }
        if(i<arr.size()){
            if (arr[i] != EMPTY){
                (*nodePtr)->right = new TreeNode(arr[i]);
                q.push(&((*nodePtr)->right));
            }
            i++;
        }
    }
    return root;
}
void BFS(TreeNode* root) {
    if (root == nullptr) return;
    queue<TreeNode*> q;//建立 queue 儲存待處理的節點指標
    q.push(root); // 將根節點的指標加入 queue

    while (!q.empty()) {
        TreeNode* current = q.front();// 取出 queue 的第一個節點指標
        q.pop();// 將該節點從 queue 中刪除
        cout << current->value << " ";
        if (current->left) q.push(current->left);   // 將左子節點的指標加入 queue
        if (current->right) q.push(current->right);// 將左子節點的指標加入 queue
    }
}
TreeNode* Depth_frist_sreach(TreeNode* node ,int target_value){
    if (node == nullptr) return nullptr;// 走到底了或本身就是空指標
    if (node->value == target_value)return node;// 找到目標數字

    TreeNode* LeftNode = Depth_frist_sreach(node->left, target_value);// 從左邊開始找，沒找到回傳 nullptr
    if (LeftNode!= nullptr)return LeftNode;// 在左邊找到了

    TreeNode* RightNode = Depth_frist_sreach(node->right, target_value);// 從右邊找
    return RightNode;// 在右邊找到回傳指標，沒找到就回傳空指標
```

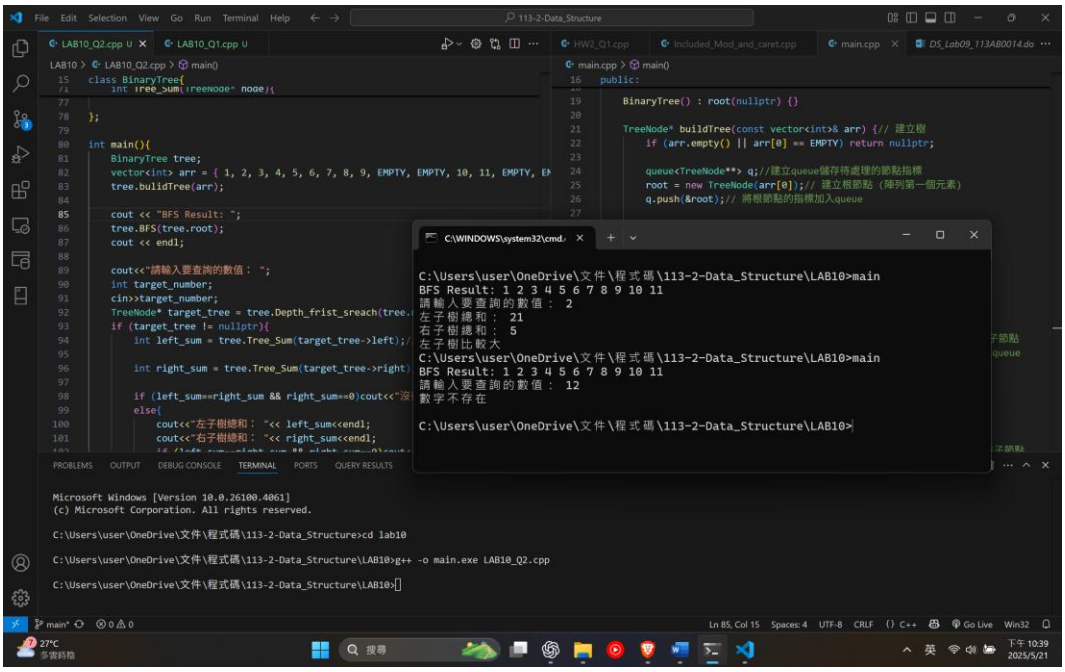
```
    }

    int Tree_Sum(TreeNode* node){
        if (node == nullptr)return 0;
        int left = Tree_Sum(node->left);//左邊的總和
        int right = Tree_Sum(node->right);//右邊得總和
        return left + right + node->value;//左邊+右邊+自己
    }

};

int main(){
    BinaryTree tree;
    vector<int> arr = { 1, 2, 3, 4, 5, 6, 7, 8, 9, EMPTY, EMPTY, 10, 11, EMPTY, EMPTY};
    tree.bulidTree(arr);

    cout << "BFS Result: ";
    tree.BFS(tree.root);
    cout << endl;

    cout<<"請輸入要查詢的數值： ";
    int target_number;
    cin>>target_number;
    TreeNode* target_tree = tree.Depth_frist_sreach(tree.root, target_number);//找到目標
節點的位址
    if (target_tree != nullptr){
        int left_sum = tree.Tree_Sum(target_tree->left);//算左子樹

        int right_sum = tree.Tree_Sum(target_tree->right);//算右子樹

        if (left_sum==right_sum && right_sum==0)cout<<"沒有子樹"<<endl;
        else{
            cout<<"左子樹總和： "<< left_sum<<endl;
            cout<<"右子樹總和： "<< right_sum<<endl;
            if (left_sum==right_sum && right_sum==0)cout<<"一樣大"<<endl;
            else{
                if (left_sum>right_sum)cout<<"左";
                else cout<<"右";
```

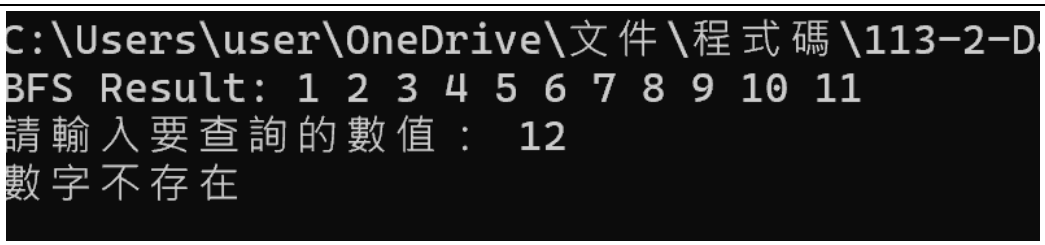
```
                    cout<<"子樹比較大";
                }
            }
        }else cout<<"數字不存在"<<endl;

}
```

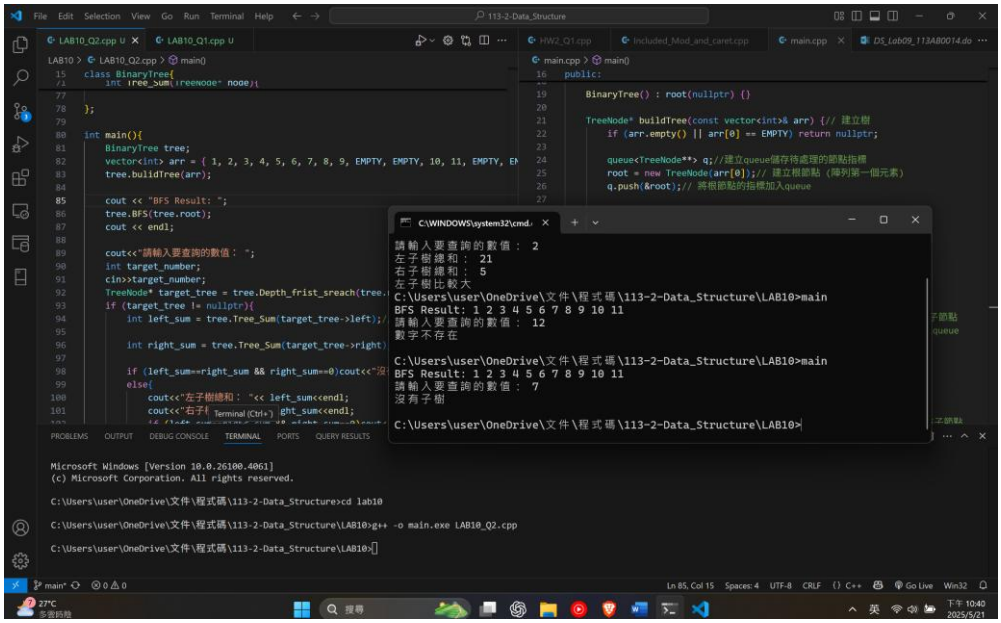
Discussion Section






C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB10>main
BFS Result: 1 2 3 4 5 6 7 8 9 10 11
請輸入要查詢的數值： 2
左子樹總和： 21
右子樹總和： 5
左子樹比較大
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB10>main
BFS Result: 1 2 3 4 5 6 7 8 9 10 11
請輸入要查詢的數值： 12
數字不存在
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB10>
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure>cd lab10
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB10>g++ -o main.exe LAB10_Q2.cpp
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB10>


C:\Users\user\OneDrive\文件\程式碼\113-2-D
BFS Result: 1 2 3 4 5 6 7 8 9 10 11
請輸入要查詢的數值： 12
數字不存在


請輸入要查詢的數值： 2
左子樹總和： 21
右子樹總和： 5
左子樹比較大
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB10>main
BFS Result: 1 2 3 4 5 6 7 8 9 10 11
請輸入要查詢的數值： 12
數字不存在
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB10>main
BFS Result: 1 2 3 4 5 6 7 8 9 10 11
請輸入要查詢的數值： 7
沒有子樹
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB10>


C:\Users\user\OneDrive\文件\程式碼\113-
BFS Result: 1 2 3 4 5 6 7 8 9 10 11
請輸入要查詢的數值： 7
沒有子樹