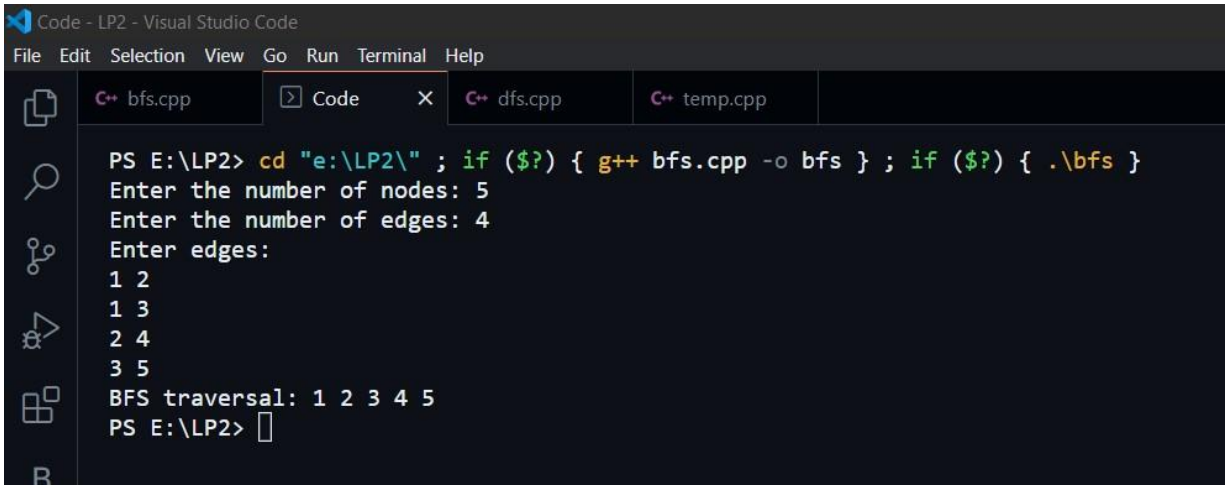
# Assignment No 1

**Breadth-First Search (BFS)**

```
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

// TreeNode class definition
class TreeNode {
public:
    int val;
    vector<TreeNode*> children;

    TreeNode(int val) {
        this->val = val;
    }

    void addChild(TreeNode* child) {
        children.push_back(child);
    }
};

// BFS traversal of a tree
void bfs(TreeNode* root) {
    if (!root) return;

    queue<TreeNode*> q;
    q.push(root);

    while (!q.empty()) {
        TreeNode* curr = q.front();
        q.pop();
        cout << curr->val << " ";

        for (auto child : curr->children) {
            q.push(child);
        }
    }
}

// Build a tree from user input
TreeNode* buildTree(int n, int m) {
    cout<<"Enter edges: "<<endl;
    vector<TreeNode*> nodes(n + 1);

    for (int i = 1; i <= n; i++) {
        nodes[i] = new TreeNode(i);
    }

    for (int i = 1; i <= m; i++) {
        int parent, child;
        cin >> parent >> child;
        nodes[parent]->addChild(nodes[child]);
```

```
    }

    return nodes[1];
}

int main() {
    int n, m;
    cout << "Enter the number of nodes: ";
    cin >> n;
    cout << "Enter the number of edges: ";
    cin >> m;

    TreeNode* root = buildTree(n, m);

    cout << "BFS traversal: ";
    bfs(root);

    return 0;
}
```




## Depth-First Search (DFS)

```
#include <iostream>
#include <vector>

using namespace std;

// TreeNode class definition
class TreeNode {
public:
    int val;
    vector<TreeNode*> children;

    TreeNode(int val) {
        this->val = val;
    }

    void addChild(TreeNode* child) {
        children.push_back(child);
    }
};

// DFS traversal of a tree
void dfs(TreeNode* node) {
```

```
    if (!node) return;

    cout << node->val << " ";

    for (auto child : node->children) {
        dfs(child);
    }
}

// Build a tree from user input
TreeNode* buildTree(int n, int m) {
    cout<<"Enter edges: "<<endl;
    vector<TreeNode*> nodes(n + 1);

    for (int i = 1; i <= n; i++) {
        nodes[i] = new TreeNode(i);
    }

    for (int i = 1; i <= m; i++) {
        int parent, child;
        cin >> parent >> child;
        nodes[parent]->addChild(nodes[child]);
    }

    return nodes[1];
}

int main() {
    int n, m;
    cout << "Enter the number of nodes: ";
    cin >> n;
    cout << "Enter the number of edges: ";
    cin >> m;

    TreeNode* root = buildTree(n, m);

    cout << "DFS traversal: ";
    dfs(root);

    return 0;
}
```