Problem Statement: Design and implement Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for BFS and DFS.

Output

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

const int MAX = 100;
vector<int> graph[MAX];
bool visited[MAX];

void dfs(int node) {
   stack<int> s;
   s.push(node);

   while (!s.empty()) {
      int curr_node = s.top();
      s.pop();

      if (!visited[curr_node]) {
         visited[curr_node] = true;

         // Push unvisited neighbors onto the stack
         for (int i = 0; i < graph[curr_node].size(); i++) {
            int adj_node = graph[curr_node][i];
            if (!visited[adj_node]) {
               s.push(adj_node);
            }
         }
      }
   }
}

int main() {
   int n, m, start_node;

   cout << "Enter number of nodes, edges and starting node: " << flush;
   cin >> n >> m >> start_node;

   cout << "Enter " << m << " edges (u v):" << endl;
```

```
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        graph[u].push_back(v);
        graph[v].push_back(u); // For undirected graph
    }

    for (int i = 0; i < n; i++) {
        visited[i] = false;
    }

    dfs(start_node);

    cout << "\nNodes visited in DFS order starting from " << start_node << ":" << endl;
    for (int i = 0; i < n; i++) {
        if (visited[i]) {
            cout << i << " ";
        }
    }
    cout << endl;

    return 0;
}
```

```
 *  Executing task: .\dfs.exe

Enter number of nodes, edges and starting node: 5 4 0
Enter 4 edges (u v):
0 1
0 2
1 3
3 4

Nodes visited in DFS order starting from 0:
0 1 2 3 4
 *  Terminal will be reused by tasks, press any key to close it.
```