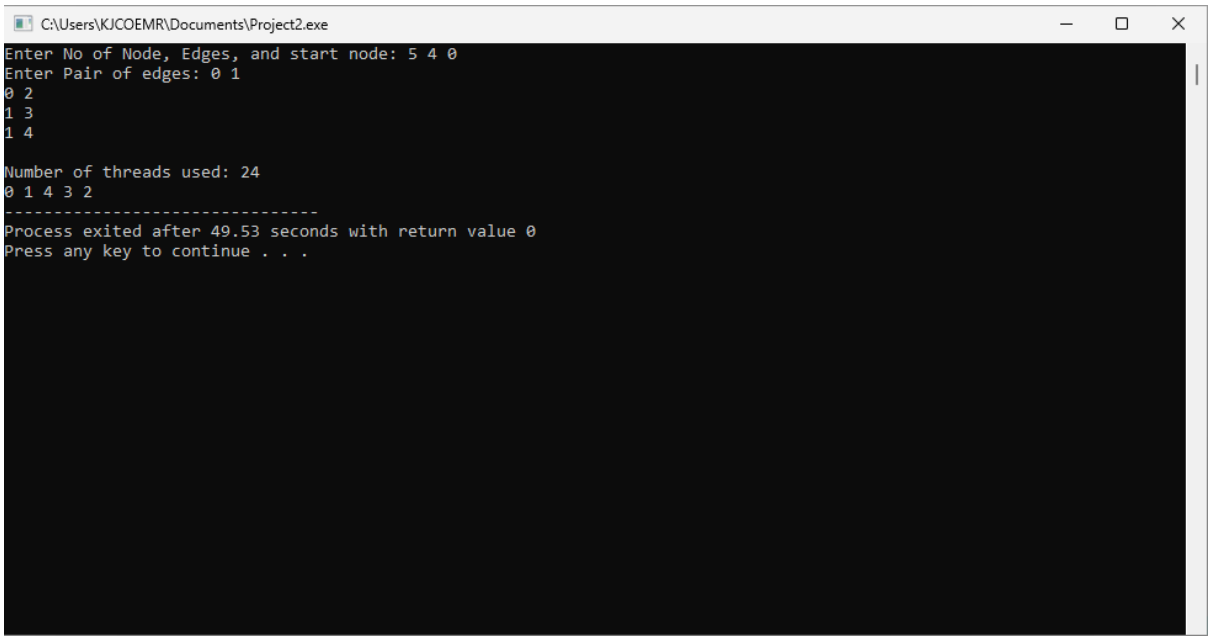
```
#include <iostream>
#include <vector>
#include <stack>
#include <omp.h>
using namespace std;
const int MAX = 100000;
vector<int> graph[MAX];
bool visited[MAX];
void dfs(int node) {
    stack<int> s;
    s.push(node);
    while (!s.empty()) {
        int curr_node = s.top();
        s.pop();
        if (!visited[curr_node]) {
            visited[curr_node] = true;
            cout << curr_node << " ";
        }
        // Parallelize the traversal of neighbors of the current node
        #pragma omp parallel for
        for (int i = 0; i < graph[curr_node].size(); i++) {
            int adj_node = graph[curr_node][i];
            if (!visited[adj_node]) {
                #pragma omp critical
                {
                    s.push(adj_node);  // Ensure thread-safety when pushing to stack
                }
            }
        }
    }
}
```

```
int main() {
    int n, m, start_node;
    cout << "Enter No of Node, Edges, and start node: ";
    cin >> n >> m >> start_node;
    cout << "Enter Pair of edges: ";
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        graph[u].push_back(v);
        graph[v].push_back(u);
    }
    // Parallelize the initialization of the visited array
    #pragma omp parallel for
    for (int i = 0; i < n; i++) {
        visited[i] = false;
    }
    // Print number of threads used for DFS
    #pragma omp parallel
    {
        if (omp_get_thread_num() == 0)  // Print number of threads once
            cout << "\nNumber of threads used: " << omp_get_num_threads() << endl;
    }
    dfs(start_node);
    return 0;
}
```

Output:


C:\Users\KJCOEMR\Documents\Project2.exe
Enter No of Node, Edges, and start node: 5 4 0
Enter Pair of edges: 0 1
0 2
1 3
1 4
Number of threads used: 24
0 1 4 3 2

Process exited after 49.53 seconds with return value 0
Press any key to continue . . .