```cpp
#include <iostream>
#include <vector>
#include <stack>
#include <omp.h>

using namespace std;

const int MAX = 100000;

vector<int> graph[MAX];

bool visited[MAX];

/// @brief
/// @param node

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;
            if (visited[curr_node]) {
                cout << curr_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]) {
                    s.push(adj_node);
                }
            }
        }
    }
}


int main() {
    int n, m, start_node;
    cout << "Enter No of Node,Edges,and start node:" ;
    cin >> n >> m >> start_node;
    //n: node,m:edges
    cout << "Enter Pair of edges:" ;
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        //u and v: Pair of edges
```

```
        graph[u].push_back(v);
        graph[v].push_back(u);
    }
    #pragma omp parallel for
    for (int i = 0; i < n; i++) {
        visited[i] = false;
    }
    dfs(start_node);
    for (int i = 0; i < n; i++) {
        if (visited[i]) {
            cout << i << " ";
        }
    }
    return 0;
}
```