#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 << " ";

// Push adjacent nodes to stack (in reverse order for correct DFS)

for (int i = graph[curr\_node].size() - 1; i >= 0; 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 of the graph:\n";

cin >> n >> m >> start\_node;

cout << "Enter pairs of nodes representing edges:\n";

for (int i = 0; i < m; i++) {

int u, v;

cin >> u >> v;

graph[u].push\_back(v);

graph[v].push\_back(u);

}

// Parallel initialization of the visited array

#pragma omp parallel for

for (int i = 0; i < n; i++) {

visited[i] = false;

}

dfs(start\_node);

return 0;

}