#include <iostream>

#include <vector>

#include <stack>

#include <omp.h>

using namespace std;

class ParallelDFS {

private:

vector<vector<int>> adjMatrix;

vector<int> visited;

int n;

public:

void input() {

cout << "Enter the number of vertices: ";

cin >> n;

adjMatrix.resize(n, vector<int>(n, 0));

visited.resize(n, 0);

cout << "Enter the adjacency matrix:\n";

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

for (int j = 0; j < n; j++)

cin >> adjMatrix[i][j];

}

void display() {

cout << "Adjacency Matrix:\n";

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

for (int j = 0; j < n; j++)

cout << adjMatrix[i][j] << " ";

cout << endl;

}

}

void dfs\_sequential(int start) {

stack<int> st;

fill(visited.begin(), visited.end(), 0);

st.push(start);

visited[start] = 1;

cout << "Sequential DFS Order: ";

while (!st.empty()) {

int current = st.top();

st.pop();

cout << current << " ";

for (int i = n - 1; i >= 0; i--) {

if (adjMatrix[current][i] && !visited[i]) {

st.push(i);

visited[i] = 1;

}

}

}

cout << endl;

}

void dfs\_parallel(int start) {

stack<int> st;

fill(visited.begin(), visited.end(), 0);

#pragma omp parallel

{

#pragma omp single

{

st.push(start);

visited[start] = 1;

cout << "Parallel DFS Order: ";

while (!st.empty()) {

int current = st.top();

st.pop();

cout << current << " ";

#pragma omp task firstprivate(current)

{

for (int i = n - 1; i >= 0; i--) {

if (adjMatrix[current][i] && !visited[i]) {

#pragma omp critical

{

if (!visited[i]) {

st.push(i);

visited[i] = 1;

}

}

}

}

}

}

cout << endl;

}

}

}

};

int main() {

ParallelDFS dfs;

dfs.input();

dfs.display();

int startVertex;

cout << "Enter the starting vertex for DFS: ";

cin >> startVertex;

double start, end;

start = omp\_get\_wtime();

dfs.dfs\_sequential(startVertex);

end = omp\_get\_wtime();

cout << "Time taken by Sequential DFS: " << end - start << " seconds\n";

start = omp\_get\_wtime();

dfs.dfs\_parallel(startVertex);

end = omp\_get\_wtime();

cout << "Time taken by Parallel DFS: " << end - start << " seconds\n";

return 0;

}