#include <iostream>

#include <vector>

#include <queue>

#include <omp.h>

void parallelBFS(std::vector<std::vector<int>>& graph, int startNode, std::vector<bool>& visited) {

std::queue<int> bfsQueue;

bfsQueue.push(startNode);

visited[startNode] = true;

#pragma omp parallel num\_threads(MAX\_THREADS)

{while (!bfsQueue.empty()) {

int currentNode;

#pragma omp critical

{ currentNode = bfsQueue.front();

bfsQueue.pop(); }

#pragma omp for

for (int neighbor : graph[currentNode]) {

#pragma omp critical

{if (!visited[neighbor]) {

visited[neighbor] = true;

bfsQueue.push(neighbor);}}}}}}

int main() {

std::vector<std::vector<int>> graph = {

{1, 2}, // Node 0

{0, 2, 3, 4}, // Node 1

{0, 1, 4}, // Node 2

{1}, // Node 3

{1, 2} // Node 4

};

int startNode = 0;

std::vector<bool> visited(graph.size(), false);

parallelBFS(graph, startNode, visited);

for (int i = 0; i < visited.size(); i++) {

if (visited[i]) {

std::cout << "Node " << i << " was visited." << std::endl; }}

return 0;}