// Online C++ compiler to run C++ program online

// Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP

#include <iostream>

#include <queue>

#include <vector>

#include <omp.h>

using namespace std;

class Graph {

private:

int V;

vector<int>\* adj;

public:

Graph(int V) {

this->V = V;

adj = new vector<int>[V];

}

void addEdge(int v, int w) {

adj[v].push\_back(w);

adj[w].push\_back(v);

}

void bfs(int start) {

vector<bool> visited(V, false);

queue<int> q;

visited[start] = true;

q.push(start);

while (!q.empty()) {

int u = q.front();

q.pop();

cout << u << " ";

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

if (!visited[v]) {

visited[v] = true;

q.push(v);

}

}

}

}

void dfs(int start) {

vector<bool> visited(V, false);

dfs\_helper(start, visited);

}

private:

void dfs\_helper(int u, vector<bool>& visited) {

visited[u] = true;

cout << u << " ";

for (int i = 0; i < adj[u].size(); i++) {

int v = adj[u][i];

if (!visited[v]) {

dfs\_helper(v, visited);

}

}

}

};

int main() {

Graph g(6);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 3);

g.addEdge(2, 4);

g.addEdge(3, 4);

g.addEdge(3, 5);

cout << "BFS starting from vertex 3: ";

g.bfs(3);

cout << endl;

cout << "DFS starting from vertex 5: ";

g.dfs(5);

cout << endl;

return 0;

}

// BFS starting from vertex 3: 3 1 4 5 0 2

// DFS starting from vertex 5: 5 3 1 0 2 4