#include <iostream>

#include <vector>

#include <queue>

#include <unordered\_map>

#include <unordered\_set>

using namespace std;

class Graph {

private:

int vertices;

unordered\_map<int, vector<int>> adjList;

public:

Graph(int v){

vertices = v;

}

void addEdge(int u, int v) {

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

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

}

void dfsUtil(int v, vector<bool>& visited) {

visited[v] = true;

cout << v << " ";

for (int i : adjList[v]) {

if (!visited[i]) {

dfsUtil(i, visited);

}

}

}

void dfs(int start) {

vector<bool> visited(vertices, false);

dfsUtil(start, visited);

}

void bfs(int start) {

vector<bool> visited(vertices, false);

queue<int> q;

visited[start] = true;

q.push(start);

while (!q.empty()) {

int v = q.front();

q.pop();

cout << v << " ";

for (int i : adjList[v]) {

if (!visited[i]) {

visited[i] = true;

q.push(i);

}

}

}

}

};

int main() {

unordered\_map<string, int> landmarks = {

{"College", 0},

{"Park", 1},

{"Library", 2},

{"Mall", 3},

{"Theater", 4}

};

Graph graph(5);

graph.addEdge(landmarks["College"], landmarks["Park"]);

graph.addEdge(landmarks["College"], landmarks["Library"]);

graph.addEdge(landmarks["Park"], landmarks["Mall"]);

graph.addEdge(landmarks["Library"], landmarks["Theater"]);

graph.addEdge(landmarks["Mall"], landmarks["Theater"]);

cout << "DFS traversal: ";

graph.dfs(landmarks["College"]);

cout << "\nBFS traversal: ";

graph.bfs(landmarks["College"]);

return 0;

}