#include <iostream>

#include <vector>

#include <queue>

#include <unordered\_map>

#include <unordered\_set>

#include <stack>

#include <string>

using namespace std;

// Structure to represent a node in the graph

struct Node {

string name;

vector<string> neighbors;

};

// Graph class representing the map of the area around the college

class Graph {

private:

unordered\_map<string, Node> adjacencyList;

public:

// Function to add a node to the graph

void addNode(const string& nodeName, const vector<string>& neighbors) {

adjacencyList[nodeName] = { nodeName, neighbors };

}

// Depth First Search traversal

void DFS(const string& startNode) {

unordered\_set<string> visited;

stack<string> stack;

stack.push(startNode);

while (!stack.empty()) {

string currentNode = stack.top();

stack.pop();

if (visited.find(currentNode) == visited.end()) {

cout << currentNode << " ";

visited.insert(currentNode);

for (const string& neighbor : adjacencyList[currentNode].neighbors) {

stack.push(neighbor);

}

}

}

}

// Breadth First Search traversal

void BFS(const string& startNode) {

unordered\_set<string> visited;

queue<string> queue;

queue.push(startNode);

while (!queue.empty()) {

string currentNode = queue.front();

queue.pop();

if (visited.find(currentNode) == visited.end()) {

cout << currentNode << " ";

visited.insert(currentNode);

for (const string& neighbor : adjacencyList[currentNode].neighbors) {

queue.push(neighbor);

}

}

}

}

};

int main() {

// Create a graph representing the map of the area around the college

Graph graph;

// Add nodes and their neighbors to the graph

graph.addNode("College", {"Library", "Cafeteria"});

graph.addNode("Library", {"Park", "Gym"});

graph.addNode("Cafeteria", {"Park", "Gym"});

graph.addNode("Park", {"Hospital"});

graph.addNode("Gym", {"Hospital"});

graph.addNode("Hospital", {});

// Perform DFS starting from the College node

cout << "DFS traversal starting from College: ";

graph.DFS("College");

cout << endl;

// Perform BFS starting from the College node

cout << "BFS traversal starting from College: ";

graph.BFS("College");

cout << endl;

return 0;

}