```
/*
    Time complexity: O(V + E)
    Space complexity: O(V^2)

    where V is the number of vertices in the input graph and
    E is the number of edges in the input graph
*/

#include <iostream>
#include <queue>
#include <vector>
using namespace std;

vector<int>* getBFSPath(bool** graph, int v, int start, int end) {
    queue<int> pendingVertices;
    bool* visited = new bool[v]();

    pendingVertices.push(start);
    visited[start] = true;

    bool pathFound = false;

    vector<int> parent(v, -1);

    while (!pendingVertices.empty() && !pathFound) {
        int front = pendingVertices.front();
        pendingVertices.pop();

        for (int i = 0; i < v; i++) {
            if (graph[front][i] && !visited[i]) {
                parent[i] = front;
                pendingVertices.push(i);
                visited[i] = true;
                if (i == end) {
                    pathFound = true;
                    break;
                }
            }
        }
    }

    delete[] visited;

    if (!pathFound) {
        return NULL;
    }

    vector<int>* output = new vector<int>();
```

```
    int current = end;
    output->push_back(end);

    while (current != start) {
        current = parent[current];
        output->push_back(current);
    }

    return output;
}

int main() {
    int v, e;
    cin >> v >> e;

    bool** graph = new bool*[v];

    for (int i = 0; i < v; ++i) {
        graph[i] = new bool[v]();
    }

    for (int i = 0, a, b; i < e; ++i) {
        cin >> a >> b;
        graph[a][b] = true;
        graph[b][a] = true;
    }

    int start, end;
    cin >> start >> end;

    vector<int>* output = getBFSPath(graph, v, start, end);

    if (output != NULL) {
        for (int i = 0; i < output->size(); ++i) {
            cout << output->at(i) << " ";
        }
        delete output;
    }

    for (int i = 0; i < v; ++i) {
        delete[] graph[i];
    }

    delete[] graph;
}
```