```
/*
    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>
using namespace std;

void singleComponentBFS(bool** graph, int v, bool* visited, int vertex) {
    queue<int> pendingVertices;
    pendingVertices.push(vertex);
    visited[vertex] = true;

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

        cout << frontVertex << " ";

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

void BFS(bool** graph, int v) {
    bool* visited = new bool[v]();

    for (int i = 0; i < v; ++i) {
        if (!visited[i]) {
            singleComponentBFS(graph, v, visited, i);
        }
    }

    delete[] visited;
}

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;
    }

    BFS(graph, v);

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

    delete[] graph;
}
```