```
#include<bits/stdc++.h>
using namespace std;

vector<int> getPathBFS(vector<vector<bool>>& edges, int sv, int ev) {
    vector<int> ans;
    queue<int> q;
    vector<bool> visited(edges.size(), false);
    q.push(sv);
    visited[sv] = true;
    unordered_map<int, int> m;

    while(!q.empty()) {
        int front = q.front();
        q.pop();
        for(int i = 0; i < edges.size(); i++) {
            if(!visited[i] && edges[front][i]) {
                m[i] = front;
                q.push(i);
                visited[i] = true;

                if(i == ev) {
                    int curr = ev;
                    ans.push_back(ev);
                    while(curr != sv) {
                        curr = m[curr];
                        ans.push_back(curr);
                    }
                    return ans;
                }
            }
        }
    }
    return {};
}

/*
vector<int> getPathBFS(vector<vector<bool>>& gp,int v, int sv, int ev) {
    vector<int> ans;
    vector<bool> visited(v, false);
    queue<int> q;
    q.push(sv);
    visited[sv] = true;
    unordered_map<int, int> mp;
    bool fine = false;

    while( !q.empty() && !fine) {
        int front = q.front();
        q.pop();
```

```
        for(int i = 0; i < v; i++) {
            if(gp[front][i] && !visited[i]) {
                mp[i] = front;
                q.push(i);
                visited[i] = true;

                if(i == ev) {
                    fine = true;
                    break;
                }
            }
        }
    }
    if( fine) {
        int cur = ev;
        ans.push_back(ev);
        while( cur != sv) {
            cur = mp[cur];
            ans.push_back(cur);
        }
        return ans;
    }
}
*/

int main() {

    int v, e;
    cin >> v >> e;
    vector<vector<bool>> graph(v, vector<bool>(v, false));

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

    int sv, ev;
    cin >> sv >> ev;

    vector<int> ans = getPathBFS(graph, sv, ev);
    for(auto i : ans) {
        cout << i << " ";
    }

    // No Need to deallocate memory , we created it statically;
```

```
    return 0;
}

/*
#include <bits/stdc++.h>
using namespace std;

vector<int> getPath(vector<vector<int>>& edges, int n, vector<bool>& visited, int s, int e) {
    vector<int> ans;
    if(s == e) {
        ans.push_back(e);
        return ans;
    }

    queue<int> q;
    unordered_map<int, int> map;
    q.push(s);
    visited[s] = true;

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

        for(int i = 0; i < n; i++) {
            if(!visited[i] && edges[front][i]) {
                q.push(i);
                map[i] = front;
                visited[i] = true;
                if(i == e) {
                    ans.push_back(e);
                    int x = e;
                    while(map[x] != s) {
                        ans.push_back(map[x]);
                        x = map[x];
                    }
                    ans.push_back(s);
                    return ans;
                }
            }

        }
    }
    return ans;
}

int main() {
        int n, e;
    cin >> n >> e;
    vector<vector<int>>edges(n, vector<int>(n, 0));
```

```
    vector<bool> visited(n, false);

    for(int i = 0; i < e; i++) {
        int f, s;
        cin >> f >> s;
        edges[f][s] = 1;
        edges[s][f] = 1;
    }

    int a, b;
    cin >> a >> b;
    vector<int> ans = getPath(edges, n, visited, a, b);
    for(auto &i : ans) {
        cout << i << " ";
    }

}
*/
```