```cpp
// Eliza
// hw4

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

class Graph
{
private:
    int n;
    int **adj;
    bool *visited;
    void dfsHelper(int v);

public:
    Graph();
    ~Graph();
    void load(char *filename);
    void display();
    void displayDFS(int vertex);
    void displayBFS(int vertex);
};

int main(int argc, char *argv[])
{
    if (argc < 2)
    {
        cout << "Missing input file." << endl;
        return 1;
    }

    Graph g;
    g.load(argv[1]);
    g.display();

    cout << "DFS at vertex 0: ";
    g.displayDFS(0);

    cout << "BFS at vertex 0: ";
    g.displayBFS(0);

    return 0;
}

Graph::Graph()
{
    n = 0;
    adj = nullptr;
    visited = nullptr;
}

Graph::~Graph()
{
    if (adj != nullptr)
    {
        for (int i = 0; i < n; i++)
            delete[] adj[i];
        delete[] adj;
    }
    delete[] visited;
}

void Graph::load(char *filename)
{
    ifstream in(filename);
    in >> n;

    adj = new int *[n];
    for (int i = 0; i < n; i++)
    {
        adj[i] = new int[n];
        for (int j = 0; j < n; j++)
            adj[i][j] = 0;
    }

    int u, v;
    while (in >> u >> v)
    {
        adj[u][v] = 1;
        adj[v][u] = 1;
    }

    visited = new bool[n];
}

void Graph::display()
{
    cout << "Adjacency Matrix" << endl;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            cout << adj[i][j] << " ";
        cout << endl;
    }
}

void Graph::displayDFS(int vertex)
{
    for (int i = 0; i < n; i++)
        visited[i] = false;

    dfsHelper(vertex);
    cout << endl;
}

void Graph::dfsHelper(int v)
{
    visited[v] = true;
    cout << v << " ";

    for (int i = 0; i < n; i++)
        if (adj[v][i] == 1 && !visited[i])
            dfsHelper(i);
}

void Graph::displayBFS(int vertex)
```

```
{
    bool *seen = new bool[n];
    for (int i = 0; i < n; i++)
        seen[i] = false;

    queue<int> q;
    q.push(vertex);
    seen[vertex] = true;

    while (!q.empty())
    {
        int v = q.front();
        q.pop();
        cout << v << " ";

        for (int i = 0; i < n; i++)
        {
            if (adj[v][i] == 1 && !seen[i])
            {
                seen[i] = true;
                q.push(i);
            }
        }
    }

    cout << endl;
    delete[] seen;
}
```