```
#include <iostream>
using namespace std;
// Maximum number of vertices
#define MAX 20
// Graph class
class Graph
{
private:
    int adjMatrix[MAX][MAX];
    int numVertices;

public:
    // Constructor
    Graph(int vertices)
    {
        numVertices = vertices;

        for (int i = 0; i < numVertices; i++)
        {
            for (int j = 0; j < numVertices; j++)
            {
                adjMatrix[i][j] = 0;
            }
        }
    }

    // Function to add an edge
    void addEdge(int i, int j)
    {
        if (i >= numVertices || j >= numVertices || i < 0 || j < 0)
        {
            cout << "Invalid Vertex Index!" << endl;
            return;
        }

        adjMatrix[i][j] = 1;
        adjMatrix[j][i] = 1;
    }

    // Function to remove an edge
    void removeEdge(int i, int j)
    {
        if (i >= numVertices || j >= numVertices || i < 0 || j < 0)
```

```
            return;

        adjMatrix[i][j] = 0;
        adjMatrix[j][i] = 0;
    }

    // Function to display the matrix
    void display()
    {
        cout << "\n--- Adjacency Matrix ---" << endl;
        cout << "   ";
        for (int i = 0; i < numVertices; i++)
            cout << i << " ";
        cout << endl;

        for (int i = 0; i < numVertices; i++)
        {
            cout << i << ": ";
            for (int j = 0; j < numVertices; j++)
            {
                cout << adjMatrix[i][j] << " ";
            }
            cout << endl;
        }
    }
};
// Main Function
int main()
{
    int v = 4;
    Graph g(v);

    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 3);

    g.display();

    cout << "\nRemoving edge between 1 and 2..." << endl;
    g.removeEdge(1, 2);
    g.display();
    return 0;
}
```