Question2.cpp

```cpp
//<----Lab 13 - Graph Traversal---->

// Q2. Implement graph traversal methods using adjacency matrix instead of adjacency list.

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

class Graph
{

private:
    int vertices;
    vector<vector<int>> adjacencyMatrix;

public:
    Graph(int V) : vertices(V), adjacencyMatrix(V,vector<int>(V, 0)) {}
    // Add an edge to the graph
    void addEdge(int v, int w)
    {
        adjacencyMatrix[v][w] = 1;
        adjacencyMatrix[w][v] = 1; // Assuming an undirected graph
    }
    // Depth-First Search starting from a given vertex
    void DFS(int startVertex)
    {
        vector<bool> visited(vertices, false);
        stack<int> stack;
        visited[startVertex] = true;
        stack.push(startVertex);
        cout << "Depth-First Search starting from vertex " << startVertex << ":\n";
        while (!stack.empty())
        {
            int currentVertex = stack.top();
            stack.pop();
            cout << currentVertex << " ";
            // Visit all adjacent vertices
            for (int neighbor = 0; neighbor < vertices; ++neighbor)
            {
                if (adjacencyMatrix[currentVertex][neighbor] == 1 && !visited[neighbor])
                {
                    visited[neighbor] = true;
                    stack.push(neighbor);
                }
            }
        }
        cout << endl;
    }
};
int main()
{
    // Create a graph with 7 vertices

    Graph graph(7);
    // Add edges to the graph
    graph.addEdge(0, 1);
    graph.addEdge(0, 2);
    graph.addEdge(1, 3);
    graph.addEdge(1, 4);
    graph.addEdge(2, 5);
    graph.addEdge(2, 6);
    // Perform DFS starting from vertex 0
    graph.DFS(0);
    return 0;
}
```