Assignment-3
Submitted By: Yamin Hossain
ID: 011192117

Question-1:

```
#include <iostream>
#include <vector>
#include <climits>
using namespace std;
// Structure to represent an edge in the graph
struct Edge {
    int src, dest, weight;
};
// Function to find the shortest paths from a given
source node
void bellmanFord(vector<Edge>& edges, int numVertices,
int src) {
    // Initialize distances from the source to all
other vertices as INFINITY
    vector<int> dist(numVertices, INT_MAX);
    dist[src] = 0; // Distance from source to itself is
0
    // Relax all edges (numVertices - 1) times
    for (int i = 1; i <= numVertices - 1; i++) {
        for (const auto& edge : edges) {
            int u = edge.src;
            int v = edge.dest;
            int w = edge.weight;
            if(dist[u]!=INT_MAX&&dist[u]+w< dist[v]) {
                dist[v] = dist[u] + w;
            }
        }
    }
```

```
    // Check for negative-weight cycles
    for (const auto& edge : edges) {
        int u = edge.src;
        int v = edge.dest;
        int w = edge.weight;
       if (dist[u] != INT_MAX && dist[u] + w < dist[v])
{
      cout << "Graph contains a negative-weight cycle!"
<< endl;
            return;
        }
    }

    // Print the shortest distances
    cout << "Shortest distances from node " << src << "
to all other nodes:" << endl;
    for (int i = 0; i < numVertices; i++) {
        cout << "Node " << i << ": " << dist[i] <<
endl;
    }
}

int main() {
    // Define the number of vertices in the graph
    int numVertices = 5;

    // Define the edges of the graph
    vector<Edge> edges = {
        {0, 1, 5},
        {1, 3, 2},
        {4, 3, -1},
        {2, 4, 1},
        {1, 2, 1}
    };
```

```
    // Specify the source node
    int src = 0;

    // Run the Bellman-Ford algorithm
    bellmanFord(edges, numVertices, src);

    return 0;
}
```

Output:

Since the graph contains a negative value, Dijkstra's algorithm will not work here. So I have used the Bellman-Ford algorithm for solving the question. The output is given below:

```
Shortest distances from node 0 to all other nodes:
Node 0: 0
Node 1: 5
Node 2: 6
Node 3: 6
Node 4: 7
```

Question-2:

```
#include <iostream>
#include <vector>
#include <limits.h>

using namespace std;

struct Edge {
    int src, dest, weight;
};

void bellmanFord(int vertices, vector<Edge> &edges, int
start) {
    vector<int> distance(vertices, INT_MAX);
    distance[start] = 0;
```

```
    // Relax all edges V-1 times
    for (int i = 0; i < vertices - 1; i++) {
        for (const auto &edge : edges) {
            if (distance[edge.src] != INT_MAX &&
distance[edge.src] + edge.weight < distance[edge.dest])
{
                distance[edge.dest] =
distance[edge.src] + edge.weight;
            }
            if (distance[edge.dest] != INT_MAX &&
distance[edge.dest] + edge.weight < distance[edge.src])
{
                distance[edge.src] =
distance[edge.dest] + edge.weight;
            }
        }
    }

    // Check for negative weight cycles
    for (const auto &edge : edges) {
        if (distance[edge.src] != INT_MAX &&
distance[edge.src] + edge.weight < distance[edge.dest])
{
            cout << "Graph contains a negative weight
cycle!" << endl;
            return;
        }
    }

    // Print the shortest distances
    cout << "Shortest distances from node " << start <<
":\n";
    for (int i = 0; i < vertices; i++) {
```

```
        if (distance[i] == INT_MAX)
            cout << "Node " << i << ": INF" << endl;
        else
            cout << "Node " << i << ": " << distance[i]
<< endl;
    }
}

int main() {
    int vertices = 5;
    vector<Edge> edges = {
        {0, 1, 2},
        {1, 2, 1},
        {2, 4, 3},
        {2, 3, -4},
        {3, 1, 2}
    };

    bellmanFord(vertices, edges, 0);
    return 0;
}
```

Output:
The graph contains a negative weight cycle.