```cpp
#include <iostream>
#include <vector>
#include <limits>
#include <fstream>
#include <sstream>

using namespace std;

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

vector<Edge>
readGraphFromFile(const
string& filename) {
    vector<Edge> edges;
    ifstream
file(filename);
    if (file.is_open()) {
        string line;
        while
(getline(file, line)) {
            stringstream
ss(line);
            Edge edge;
            ss >> edge.src
>> edge.dest >>
edge.weight;

edges.push_back(edge);
        }
        file.close();
    }
    return edges;
}

void printPath(const
vector<int>& prev, const
vector<int>& dist, int
vertex) {
    if (prev[vertex] != -
1) {
        printPath(prev,
dist, prev[vertex]);
        cout << " -> ";
    }
    cout << vertex << "
(Weight: " << dist[vertex]
<< ")";
}

void bellmanFord(const
vector<Edge>& edges, int
numVertices, int source) {
    vector<int>
dist(numVertices,
numeric_limits<int>::max()
);
    vector<int>
prev(numVertices, -1);

    dist[source] = 0;

    for (int i = 1; i <
numVertices; i++) {
        for (const auto&
edge : edges) {
            int u =
edge.src;
            int v =
edge.dest;
            int weight =
edge.weight;
            if (dist[u] !=
numeric_limits<int>::max()
```

```
&& dist[u] + weight <
dist[v]) {
                dist[v] =
dist[u] + weight;
                prev[v] =
u;
            }
        }
    }

    for (const auto& edge
: edges) {
        int u = edge.src;
        int v = edge.dest;
        int weight =
edge.weight;
        if (dist[u] !=
numeric_limits<int>::max()
&& dist[u] + weight <
dist[v]) {
            cout << "Graph
contains a negative
cycle!" << endl;
            return;
        }
    }

    for (int i = 0; i <
numVertices; i++) {
        if (i != source) {
            cout <<
"Shortest path from " <<
source << " to " << i <<
": ";
printPath(prev, dist, i);
            cout << endl;
        }
    }
}

int main() {
    string filename =
"graph1.txt";
    int numVertices = 0;
    int source;

    vector<Edge> edges =
readGraphFromFile(filename
);
    for (const auto& edge
: edges) {
        numVertices =
max(numVertices,
max(edge.src, edge.dest) +
1);
    }

    cout << "Enter the
source vertex (u): ";
    cin >> source;

    bellmanFord(edges,
numVertices, source);

    return 0;
}
```

```
Enter the source vertex (u): 1
Shortest path from 1 to 0: 1 (Weight: 0) -> 3 (Weight: 2) -> 0 (Weight: 6)
Shortest path from 1 to 2: 1 (Weight: 0) -> 3 (Weight: 2) -> 0 (Weight: 6) -> 2 (Weight: 9)
Shortest path from 1 to 3: 1 (Weight: 0) -> 3 (Weight: 2)
```

```
0 1 5
0 2 3
1 3 2
2 1 1
2 3 6
3 0 4
```