Assignment #2

Implement **Bellman-Ford** algorithm. Output should contain the shortest path, path cost, and the table containing the costs and parent of each node after each iteration.

Date of Performance: 12/06/2023	Student ID: 20210104108
Date of Submission: 19/06/2023	Name: Sharjil Shabab Khan
	Group: C1

```
#include <bits/stdc++.h>
using namespace std;

struct Edge
{
   int u, v, w;
};

vector<Edge> E;
int dist[1000];
int parent[1000];
int n,itr=0;

void printTable()
{
   cout << "Iteration\tVertex\tCost\tParent\n";
   cout << "Iteration " << itr << ":\n";
   itr++;
   for (int j = 1; j <= n; j++)
   {
      cout << j << "\t\t" << dist[j] << "\t\t" << parent[j] << "\n";
   }

}

void BellmanFord(int s)
{
   for (int i = 1; i <= n; i++)
   {
      dist[i] = 100000000;
      parent[i] = -1;
   }
   dist[s] = 0;

   for (int i = 1; i < n; i++)
   {
     for (Edge e : E)
        {
                 if (dist[e.v] > dist[e.u] + e.w)

                        {
                                printTable();
                                dist[e.v] = dist[e.u] + e.w;
                                parent[e.v] = e.u;
                        }
        }
```

```
    }
}

void printPath(int node)
{
    if (parent[node] == -1)
    {
        cout << node;
        return;
    }
    printPath(parent[node]);
    cout << " -> " << node;
}

int main()
{
    int node, e;
    cout << "Enter the number of vertices: ";
    cin >> node;
    n = node;

    cout << "Enter the number of edges: ";
    cin >> e;

    cout << "Enter the edges (source, destination, weight):\n";
    while (e--)
    {
        Edge edge;
        cin >> edge.u >> edge.v >> edge.w;
        E.push_back(edge);
    }

    int s;
    cout << "Enter the source vertex: ";
    cin >> s;
    BellmanFord(s);

    cout << "Shortest paths from source " << s << ":\n";
    for (int i = 1; i <= node; i++)
    {
        cout << "Path to vertex " << i << ": ";
        printPath(i);
        cout << "\nPath cost: " << dist[i] << "\n\n";
    }
    return 0;
}
```