```
#include "Graph.h"

Graph::Graph(const char* file)
{
	ifstream ifs(file);
	// citesc din prima linie a fisierului nr de varfuri si numarul de arce
	ifs >> V >> E;

	edges = new Edge[E];

	for (int i = 0; i < E; i++) {
		ifs >> edges[i].src;
		ifs >> edges[i].dest;
		ifs >> edges[i].weight;
	}


}
// tiparim drumurile minme de la sursa la fiecare nod
void printPath(int* p, int dest) {
	if (dest != p[dest])
		printPath(p, p[dest]);
	cout << dest << "->";

}

void Graph::print()
{
	for (int i = 0; i < E; i++)
		cout << edges[i].src << "  " << edges[i].dest << "   " << edges[i].weight<<endl;
}

void Graph::BellManFord(int start)
{
	int* dist = new int[V];
	int* parent = new int[V];
	// pas 1 initializam vectorul dist
	for (int i = 0; i < V; i++) {
		dist[i] = INT_MAX;
		parent[i] = i;
	}
	dist[start] = 0;
```

```
	//pas2 relaxam muchiile de V-1 ori
	for (int i = 1; i <= V - 1; i++) {
		for (int j = 0; j < E; j++) {
			int u = edges[j].src;
			int v = edges[j].dest;
			int weight = edges[j].weight;
			if (dist[u] != INT_MAX
				&& dist[u] + weight < dist[v]) {
				dist[v] = dist[u] + weight;
				parent[v] = u;
			}
		}
	}
	// se verifica daca exista cicluri negative

	for (int i = 0; i < E; i++) {
		int u = edges[i].src;
		int v = edges[i].dest;
		int weight = edges[i].weight;
		if (dist[u] != INT_MAX
			&& dist[u] + weight < dist[v]) {
			printf("Graph contains negative weight cycle");
			return; // If negative cycle is detected, simply
					// return
		}
	}
	for (int i = 0; i < V; i++)
		cout << dist[i] << "	";

	for (int i = 0; i < V; i++) {
		printPath(parent, i);
		cout << endl;
	}


}
```