Ex2a

In this exercise we will implement the Bellman-Ford algorithm.

Background (from vikipedia):

The **Bellman–Ford algorithm** is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph.

Bellman–Ford is based on the principle of relaxation, in which an approximation to the correct distance is gradually replaced by more accurate values until eventually reaching the optimum solution. The approximate distance to each vertex is always an overestimate of the true distance, and is replaced by the minimum of its old value with the length of a newly found path.

A pseudo of the algorithm for node s:

```
d[s] ← 0

for each v ∈ V – {s}

        do d[v] ← ∞

for i ← 1 to |V| – 1 do

        for each edge (u, v) ∈ E do

                if d[v] > d[u] + w(u, v) then

                        d[v] ← d[u] + w(u, v)

                        π[v] ← u

for each edge (u, v) ∈ E do

        if d[v] > d[u] + w(u, v)

                then report that a negative-weight cycle exists
```

What you need to do:

In this exercise, you need to implement the Bellman-Ford algorithm for one vertex, given as a parameter to the program.

The input is a file that contains the network topology in the following format:

a b 6

c a 1

…

Meaning, there are 2 (or actually 4) edges, one from a to b (and vice versa) with weight 6, and one from c to a (and vice versa) with weight 1.

The output of the program should be:

dst via cost

where dst is the destination node, via is the neighbor to pass a packet twards the destination and cost is the cost of the path.

Good luck!!