Objective of this assignment:

- Implement Bellman-Ford's algorithm

What you need to do:

1. **Ask questions if you have any doubt**
2. Implement Bellman-Ford's algorithm
3. Allow a user to provide a *graph* and a *source* vertex s as a text file.
4. Display the shortest path for s to every other vertex in the graph.
5. Output a file text describing the shortest path for s to every other vertex in the graph
6. **Ask questions if you have any doubt**

Objective:

The objective is to implement the Bellman-Ford's algorithm.

Input:

Your program must prompt the user to enter the name of an input file text. The input will be a text file describing the graph. For simplicity, vertices will be identified using only integers from 0 to 37,767. The weights on the edges will be also integers from -32,768 to 32,767.

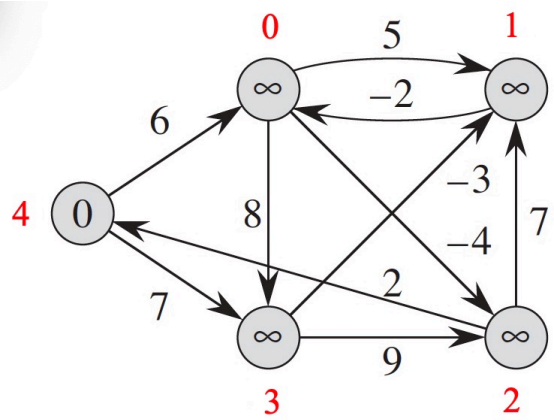
A graph and the source s are provided in a text file following this format:

```
4
0 1,5 2,-4 3,8
1 0,-2
2 1,7 4,2
3 1,-3 2,9
4 0,6 3,7
```

The first line provides the source vertex. For this example, Node `4` is the source. The remaining lines provide the adjacency list. The leftmost column is the list of vertices (here, vertices are 0, 1, 2, 3, 4). For each vertex, the edges are provided as pairs `node, weight`. For example, Vertex 4 has two edges:
- the first edge is `0,6` meaning the edge (4,0) with weight 6
- the second edge is `3,7` meaning the edge (4,3) with weight 7

A node not connected to any other vertex will be listed with no edge(s).

The above input text file `graph.txt` (available on Canvas with this assignment) represents the graph below with Vertex `4` as the source:



Output:

If the Bellman-Ford's completes **successfully**, the output will provide the shortest path from the source to each vertex in the graph other than the source. You must display the output and write it to a text file under this format:

```
0: 3 1 0
1: 3 1
2: 3 1 0 2
3: 3
```

The first column is the list of nodes in the graph (except the source) in increasing order. For each vertex, the shortest path is provided as a list of vertices describing the path. For example: the shortest path from `4` to `1` is the path `4 3 1` (see output file above)

Programming

You can implement the Bellman-Ford's algorithm in your preferred language as long as it is already available on Engineering Tux machines. Insure that your program compiles and executes correctly on Tux machines.

Grading:

- The program does not compile (0%)
- The program compiles correctly on your machine (1%)
- The program compiles correctly on a Tux machine (5%)
- The program compiles and executes correctly on your machine (20% = 2% + 2% +16%)
  - 2% for the input file
  - 2% for the output file
  - -16% for the program (working correctly with your sample `inputGraph.txt`)
- The program compiles and executes correctly on a Tux machine (100% = 10% + 10% + 80%)
  - 10% for the input file `inputGraph.txt`
  - 10% for the output file `outputShortestPaths.txt`
  - 30% for the program (working correctly with graph.txt)
  - 30% for the program (working correctly with your sample `inputGraph.txt`)
  - 20% for the program (working correctly with a grading sample file following the format provided)

Report

- Write a report. The report should not exceed one page.
- Your report must contain the following information:
  - whether the program works or not (this must be just ONE sentence).
  - If the program does not work, list the requirements not met.
  - the directions to compile and execute your program
- Good writing is expected.

What you need to turn in:

- Electronic copy of your source program (standalone/separately attached to assignment)
- Electronic copy of the report(standalone/separately attached to assignment). Submit the file as a Microsoft Word or PDF file.
- A sample graph text file (with .txt extension) having at least 8 vertices and 16 edges. Call this file `inputGraph.txt`,
- The output text file of your program. Call this file `outputShortestPaths.txt` ,

Grading

- See Points Distribution above