**Homework 10, Due Date: 10:00 am 04/24/2014,**
**Submission: Upload all source code (.java) files to Blackboard, and java programs (.class files) to EC2 instances**

Write a program to simulate the reaction at **router $V_0$** to the change in local link cost or receiving a message from its neighbor.
(**Hint**: for testing, you may want to use the network like the one given on **the slide titled "Bellman-Ford example"** instead of the simple network with only three routers x, y, and z. You may want to use the topology given on the slide to calculate the entries in Distance Vector of **router $V_0$,** Link Vector of **router $V_0$**, and Distance Vector of EACH neighbor of **router $V_0$**, which are the user inputs needed in Step 4 and Step 5)

- The routers in the network are labeled as V0, V1, V2, …, etc
- Your routing program needs to

1. Display a prompt message to ask the user to input the total number of routers, ***n***, in the network. Validate ***n*** to make sure that it is greater than or equal to 2.
2. Display two prompt messages to ask the user to input the link to a neighbor of **router $V_0$** as below:
   The index of a neighboring router:
   The cost over the link to this neighboring router:
   where the first input needs to be validated to be between 1 and ***n – 1*** and the second input needs to be validated to be positive. Keep asking for the same input for invalid cases.
3. Display a prompt message to ask the User whether to input more links to neighbors of **router $V_0$**. If yes, repeat step 2. Otherwise, DISPLAY all neighbors of router V0 as below:
   Neighbor Link Cost
   V? ???
   V? ???
   ...
   then, go to the next step. (**Hint:** you may use an array or array list to record these links to the neighbors of **router $V_0$**, and for each link, both the index of the neighboring node and the link cost need to be recorded. Here, the two-dimension cost array used for Dijsktra's algorithm is not convenient to be used.)
4. Repeat displaying the following two prompt messages ***n*** times to ask user to input ***n*** entries in the converged distance vector $\mathbf{D_0}$ and ***n*** entries in the converged link vector $\mathbf{L_0}$ in sequence, where j = 0, 1, 2, …, $n$ – 1. (Hint: you may want to use two arrays, one for $\mathbf{D_0}$ and one for $\mathbf{L_0}$ to record user inputs.)
   The least cost from router V0 to router Vj, D0(j):
   The neighboring node achieving such least cost from router V0 to router Vj, L0(j):
   After the user inputs all the entries in $\mathbf{D_0}$ and $\mathbf{L_0}$, DISPLAY them as:
   D0 = [..., ..., ... ... ,...]
   L0 = [..., ..., ... ... ,...]
5. Using the information collected in Steps 2 and 3, for **EACH neighbor** (for example, **router $V_k$**) of **router $V_0$**, repeat displaying the following prompt message ***n*** times to ask user to input ***n*** entries in the converged distance vector $\mathbf{D_k}$ in sequence, where j = 0, 1, 2, …, $n$ – 1. (**Hint**: you may want to use an array for EACH neighbor of **router $V_0$** to record user inputs.)
   The least cost from router Vk to router Vj, Dk(j):
   After the input inputs all distance vectors from the neighbors of **router $V_0$**, DISPLAY distance vector coming from EACH neighbor of router V0:
   D? = [..., ..., ... ... ,...]
   D?? = [..., ..., ... ... ,...]
   ...
   where ?, ??, ... are neighbors of router V0.
6. Ask the user to select one of the following two events to continue:
   Event 1: a change in local link cost to a neighbor of router V0
   Event 2: receiving a distance vector message from a neighbor of router V0
   6.1 If the user select Event 1, display the following two prompt messages to ask for user inputs,
   The index of this neighboring router:
   The new link cost to this neighboring router:
   6.2 If the user select Event 2, display the following first prompt message once and second prompt message ***n*** times to ask for user inputs,
   The index of the neighbor from which the distance vector message is received:
   The new least cost from this neighbor to router Vj, D?(j):
   where j = 0, 1, 2, …., $n$ – 1 in sequence and '?' is the index of this neighbor.
7. Implement the **Distance Vector algorithm** to (1) recomputed distance vector $\mathbf{D_0}$ and link vector $\mathbf{L_0}$ at **router V0** according to the user inputs in Step 6 and (2) determine whether to notify neighbors.
   7.1 If there is no need to notify neighbors, display
   There is no need to notify any neighbor!

**(more on next page!)**

7.2 If there is a need to notify neighbors, display
The list of neighbors to be notified
The list of the entries in the distance vector D0 to be sent to all the above neighbors
The list of the entries in the link vector L0 although L0 is not going to be sent out

8. Display a prompt message to ask the User whether to input a new Event of change or receiving. If yes, repeat step 6 and step 7. Otherwise, terminate this program.