Project5-part1 and part2: DEMO

Run1: Testing invalid menu-choice handling.

Run *DiGraphTest* program for the following data and requests.
- at the prompt for number of vertices, enter **4**
- enter the following data **exactly** as given below (keep line breakings as shown)

a
1 2 //after this line it should give a feedback about adding the edge (1,2)
x //should get an error message (this is an invalid menu choice)
yyy //should get an error message (this is an invalid menu choice)
a
2 3 //after this line it should give a feedback about adding the edge (2,3)
a 3 4 //should get an error message (***this is an invalid menu choice***) – menu choice should be one letter
//and it should be the ***only value on the line***
p // the graph is output: it should look like this

```
The graph is the following:
1 is connected to:  2
2 is connected to:  3
3 is connected to:
4 is connected to:
```

q

Run2: Testing *DiGraphTest* program with PIPED input

- Create a file containing the following data **exactly** as given below, keeping the line breakings as shown.

```
5
a
1 2
a
4 5
a
4 3
v
e
p
q
```

- Compile and Run *DiGraphTest* from the command line, ***piping*** input data from the prepared file (suppose the name of the file is *data*). You should NOT change your program (your code stays as is – **NO** file input).

```
javac DiGraphTest.java
java   DiGraphTest < data
```

You should not input anything else: your program should now output feedback on all requests:

```
Enter number of vertices   //This is the prompt your program outputs to get the N
:  output of the menu       //The menu will be printed here (to save space, I did not include it here)
(1,2) edge is now added to the graph
(4,5) edge is now added to the graph
(4,3) edge is now added to the graph
Number of vertices is 5
Number of edges is 3
The graph is the following:
1 is connected to:  2
2 is connected to:
3 is connected to:
4 is connected to:   5, 3
5 is connected to:
Good bye.
```

Run3: Testing program's general functionality:

Run *DiGraphTest* program for the following data and requests.

- Provide **6** as the number of vertices in the graph.
- Add one by one the following edges: (2,1), (2,3), (3,4), (3,5), (3,6), (4,1), (4,6), (5,1).
- Vertex count: //should be **6**
- Edge count: //should be **8**
- Print: //the output will look like this:

The graph is the following:
1 is connected to:
2 is connected to: 1, 3
3 is connected to: 4, 5, 6
4 is connected to: 1, 6
5 is connected to: 1
6 is connected to:

- Topological Sort: //with the order of neighbors as listed above, your output should be: **2, 3, 4, 5, 6, 1**
- Delete the edge (2,3)
- Edge count: //should be **7**
- Add the edge (3,2)
- Vertex count: //should be **6**
- Edge count: //should be **8**
- Print: //the output will look like this:

The graph is the following:
1 is connected to:
2 is connected to: 1
3 is connected to: 4, 5, 6, 2
4 is connected to: 1, 6
5 is connected to: 1
6 is connected to:

- Topological Sort: //with the order of neighbors as listed above, your output should be: **3, 4, 5, 2, 6, 1**
- Add the edge (1,5)
- Topological Sort: //should output **error message** indicating that the **graph is cyclic** (should **NOT** crash)
- Delete one by one the following edges: (1,5), (2,1), (5,1), (4,1)
- Edge count: //should be **5**
- Print: //the output will look like this:

The graph is the following:
1 is connected to:
2 is connected to:
3 is connected to: 4, 5, 6, 2
4 is connected to: 6
5 is connected to:
6 is connected to:

- Topological Sort: //with the order of neighbors as listed above, your output should be: **1, 3, 4, 5, 2, 6**
- Add two edges (4,6), (3,2) //Note: that both edges exist in the graph, so nothing should change
- Edge count: //should be **5**
- Print: //the output will look like this:

The graph is the following:
1 is connected to:
2 is connected to:
3 is connected to: 4, 5, 6, 2
4 is connected to: 6
5 is connected to:
6 is connected to:

- Delete one by one all edges: (3,4), (3,5), (3,6), (3,2), (4,6)
- Vertex count: //should be **6**
- Edge count: //should be **0**

//Continued on the next page

- Print: //the output will look like this:

The graph is the following:
1 is connected to:
2 is connected to:
3 is connected to:
4 is connected to:
5 is connected to:
6 is connected to:

- Add two edges (1,2), (3,1)
- Edge count: //should be **2**
- Print: //the output will look like this:

The graph is the following:
1 is connected to: 2
2 is connected to:
3 is connected to: 1
4 is connected to:
5 is connected to:
6 is connected to:

- Quit