Project 4: Graphs
Due May 20, 11-59pm

Your job is to write a class that can load an (undirected) graph from a file. Then, your class should support a number of options, all accessed through a text-driven menu. Let's list the things you're going to need to be able to do.

1. **Loading the Graph**: You will need to open the file specified by the user and read in a representation of an undirected graph.
2. **Display a menu**: Display a menu the user can make choices from. Each choice will either give information about the currently loaded graph or transform it in some way.
3. **Is Connected**: You will need to determine whether or not the graph is connected.
4. **Minimum Spanning Tree**: If the graph is connected, you will need to find a minimum spanning tree for the graph and print an error otherwise.
5. **Shortest Path**: Prompt the user for a node. Then, print the lengths of the shortest paths from that node to all other nodes and the actual paths as well.

Loading the Graph

When your program starts, you must prompt the user for a graph file. This graph file will start with a number saying the number of nodes in the graph. Then, each line after that will correspond to a particular node. Each such line will start with a number specifying the number of edges connected to a node. The rest of the line will give a series of pairs of numbers. The first number in the pair gives the number of node that the current node is connected to. The second gives the length of the edge connecting them.

We will use the following graph for many of our examples.

The input file for this graph is as follows

```
6
2 1 5 5 6
2 0 5 2 3
3 1 3 3 4 5 2
2 2 4 4 5
2 3 5 5 7
3 0 6 2 2 4 7
```

Display a Menu

After the graph has been loaded, display a menu the user can make choices from. Each choice will either give information about the currently loaded graph or transform it in some way.

The choices for the menu are:

1. Is Connected
2. Minimum Spanning Tree
3. Shortest Path
4. Quit

The user should be able to pick any sequence of choices repeatedly until finally selecting Quit (4). Sample output for the menu is as follows.

```
1. Is Connected
2. Minimum Spanning Tree
3. Shortest Path
4. Quit

Make your choice (1 - 4):
```

Is Connected

If the user selects choice 1, you will need to determine whether or not the graph is connected. If the graph is connected, print Graph is connected. Otherwise print Graph is not connected.
Output for the sample graph is as follows.

```
Graph is connected.
```

Minimum Spanning Tree

If the graph is connected, you will need to find a minimum spanning tree for the graph. Otherwise, print Error: Graph is not connected. Print the MST in the same general format that is used for the graph input file.

The MST for the sample graph is as follows.

Output for the sample graph is as follows.

```
6
1 1 5
2 0 5 2 3
3 1 3 3 4 5 2
2 2 4 4 5
1 3 5
1 2 2
```

Shortest Path

Prompt the user for a node. Then, print the lengths of the shortest paths from that node to all other nodes in parentheses. For the starting node, print (0). For unreachable nodes, print (Infinity). After the length of each shortest path, print a tab and then the nodes visited in the path itself, separated by arrows. Note that you will need to keep a back pointer giving the previous node in the path as well as the best distance found so far in order to produce each shortest path.

If a user entered node 0 for this choice, the output for the sample graph would be as follows.

```
From which node would you like to find the shortest paths (0 - 5): 0
0: (0) 0
1: (5) 0 -> 1
2: (8) 0 -> 1 -> 2
3: (12) 0 -> 1 -> 2 -> 3
4: (13) 0 -> 5 -> 4
5: (6) 0 -> 5
```

Hints

- Start early. Try to work on the pieces in the order they are given to you.
- You are allowed to implement the graph with an adjacency matrix or with adjacency lists. It's up to you.
- Think before you code. 1 minute of design == 10 minutes of coding == 100 minutes of debugging.
- You may use the Java Collection Framework if you need lists, queues, or stacks. Very little there is likely to help you much. You are (naturally) forbidden from using graph libraries such as JGraph and JGraphT.