Due May 3, 11.59pm

This project will use the IMDB actor/movie dataset; using this dataset you will answer a number of questions on the graph representing this data and its attributes

IMDB Dataset
We will use a small curated version of the IMDB Actor/Movie dataset. Each record contains the following:

- actor name (string)
- movie name (string) (in which the actor appeared)
- movie genre (array of strings)
- movie rating (0-10.0)

To access the dataset, you will issue a BRIDGES API call as follows:

ArrayList<ActorMovieIMDB> imdb_data = bridges.getActorMovieIMDBData2()

Refer to the ActorMovieIMDB class at

`http://bridgesuncc.github.io/doc/java-api/current/html/classbridges_1_1data__src__dependent_1_1_actor_movie_i_m_d_b.html`

Tasks.

1. **Graph Construction:** Similar to project 1, you will construct a graph of the IMDB actor-movie dataset, exploring the relationships between actors, movies and the movie genres.
   - You will build a graph of the data by building edges from actor to movie, $actor \Longleftrightarrow movie$, and movie to each genre $movie \Longleftrightarrow genre_i, i \in (0..max-1)$. All these relationships go **both ways**, for instance an actor's outgoing edges point to the movies he/she acted in, as well movie nodes that connect to all of its actors.
   - **Ensure that each actor, movie and genre appears exactly once in the graph;** if a previously referenced actor is referenced again, then the stored graph vertex is retrieved from the graph to make new edges to it; and similarly for genre and movie nodes.
   - There are 3 types of nodes in the graph: actor, movie and genre. Identify them with unique labels. This can be done using the generic parameter in the addVertex() method; the second parameter can be any Java object. For instance, a String can be used. This will be needed in the project, when *iterating through the actor, movie or genre nodes*, which are all lumped together into graph's vertices.
   - **Use color and size**to distinguish between the 3 types of nodes. For instance genre nodes can be made larger, since they will be cluster centers.
   - **Output.** After the graph is constructed, **visualize the graph using BRIDGES**.
2. **Identify the Genre with the Most Movies.** The genre nodes in the graph are connected to the movie nodes; so this operation will involve counting the out-degree of the genre nodes and identifying the genre node with the largest number of outgoing edges.
   Output. Highlight the genre node, edges and the movie nodes it is connected to that has the largest count. Visualize the graph.

3. **Identify the most Versatile Actor.** We define the most versatile actor as one whose movie credits span the most number of genres. You will compute this using the graph, beginning with each actor, following the movies he/she has acted in and following those movies' genres; the actor with the most genres will be considered the most versatile; if there are ties, then identify all of them. Note that **each genre from a movie must be counted only once.** This can be done by simply maintaining a set of the genres encountered (Java Set, for instance).
   Output. You will highlight the selected actor(s), his/her connections to the movies and the related genres in the graph and visualize it using BRIDGES;

4. **Identify the most popular genre.** This operation is similar to (1) but we are identifying the number of actors involved in each genre. **Note that actors might be reached by multiple paths throught the movies but must be counted only once; follow a similar strategy as (2)**. **Output.** You will highlight the identifed genre, the connections to its movies and the actors related to them in the graph; visualize it using BRIDGES;

Evaluation:

- By interactive demo; a schedule will be set up to show your implementation by the teaching assistant.
- You will downnload your submission during the demo and run your program and demonstrate the results.
- Turn in your source code to Canvas by the deadline; **Source code must be well documented for full credit.**