**Due Oct. 26, 11.59pm**

*This project will use IMDB's movie/actor dataset to demonstrate a number of graph algorithms, involving traversing the graph and computing metrics based on the graph structure. This project will be in 2 parts and will use BRIDGES for visualizing the required output of each task.*

In part 1(this project), you will read in the given dataset and display the graph using an adjacency list representation. You will use the BRIDGES system to display and demonstrate some basic operations on the graphs.

**Dataset:**

We will use a small version of IMDB's actor/movie dataset for this project. This will be provided as a text file, containing a set of actor-movie pairs.

**Tasks.**

*Here you will do a set of preliminary tasks that will be useful to complete the full project. Your application should have the following components ready and tested(all using BRIDGES).*

1. **Load Input Dataset.** Read in the actor-movie dataset. Since the data comes in as actor-movie pairs, actors and movies can occur multiple times in the data, thus, we need to **ensure that the graph only contains one instance of an actor or movie.** Actors and movies are names (with no intervening spaces, so can be read in as a string.

   - Use the **Java Hashmap**(`https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html` to avoid duplicates. Use the actor and movie name as the key to insert or find if an actor or movie exists in the hash map. If the actor or movie does not already exist, then a new graph node (vertex) is created, else the existing node is used to create the link between the actor and movie. For example

     ```
     GraphAdjList gr;
     String actorName; // actorName is an actor read in from the input
     if (!gr.getVertices().containsKey(actorName)
             "add a new vertex to graph";
     ```

     *Note that the BRIDGES graph implementation uses Java Hashmaps to index into the vertices list. Each vertex is of type SLelement< E >, so a linked list represents the set of edge terminating vertices.*
   - Use the graph's addVertex() and addEdge() methods to build the graph, as illustrated in the example worked out in class.
   - Your graph should contain a unique set of actors and movies, once all of the data has been read into memory.
   - Visualize the graph using BRIDGES.

2. You will implement the BFS traversal algorithm (a queue implementation will be provided) **Review section 3.5, text**. BFS does a level order traversal of the graph.

- Run the BFS traversal from a user specified (can be based on keyboard input from the user) on the graph and highlight all the nodes (in some color) that are visited. The entire graph should be highlighted.
- Modify your traversal so that the the opacities of the nodes and links are reduced as the level of the graph increases. Start with an opacity of 1.0 for the start node (level 0) and then 0.95 for the next level, and so on. You should be able to visualize the traversal progression. Play with the opacity variation to see a nice gradual variation of the node colors by level.

3. **[Extra Credit.]**Implement the DFS traversal (no need for a queue, as it is recursive and repeat what you did with the BFS traversal. Note that levels increase with each recursive call.

**Evaluation:**

By interactive demo; a schedule will be set up to test your program by the teaching assistant.

**Submission Requirements.**

Turn in your source code to Canvas by the deadline; ensure it is well documented.