Due May 6, 5pm

This project will use IMDB's movie/actor dataset to compute the Bacon Number of an actor, which refers to the shortest path through the movie/actor graph that connects two actors – the so called 'Six Degrees of Separation' problem. You will use BRIDGES to display the path that leads from the actor to Kevin Bacon(or another chosen actor). ***Currently, linkes in BRIDGES cannot have their properties altered, so only the nodes in the path will be highlighted.***

This will be a team project, with 2 students per team. I will consider extra credit for those who prefer or would like to do the project by themselves.

A description of this problem can be found at

`http://introcs.cs.princeton.edu/java/45graph/`

Dataset:

We will use the same dataset of IMDB's actor/movie dataset as in project 1.

The graph representation utilizes the following classes in BRIDGES:

- **HashMap<String, GraphList<** E **>>:** This is the main representation of the graph on BRIDGES, mapping vertex names (in our case, actors and movies) to GraphList, which is a helper class that represents a singly linked list of edges and the source vertex.
- **GraphList<** E **>.** This is a helper class, and contains the source vertex, and an SLelement¡Edge¿, the adjacency list containing the terminating vertex of each edge emanating from the source vertex. Has methods to add to the list. See the API for details.
- **Edge.** Used by GraphList, and is the generic variable that will contain the vertex name and weight of the edge(similar to book implementation).

Tasks.

1. Build the full graph, using the above classes and visualize the graph(use bridges.setDataStructure(graph, "graphl"), where graph is the graph you have built, containing the actor/movie graph. Note that actors and movies should only occur once, i.e., you have a unique list of actors and movies. Each actor/movie pair results in two edges, from actor$\Longrightarrow$ movie and from movie$\Longrightarrow$ actor. The graph you are building is effectively an undirected graph.
2. Implement the BFS traversal algorithm on graphs(see text for details). You will need to use a queue(use the book's implementation). The BFS traversal algorithm is sufficient to compute the Bacon number since the graph is not weighted(assume the weight of all edges of the graph is 1). Thus the shortest path between two nodes in the graph will be always determined by the BFS traversal (which is not true for a weighted graph). As before you will maintain a distance value at each node of the graph. When a child is reached, its distance value is then updated. Get a good understanding of the Dijkstra's shortest path algorithm, and modify the BFS algorithm to produce the Bacon number value. You can use a prev[v] array to keep track of the parent of the visited nodes(v) in the graph; this will aid in determining the Bacon number path. **Implementation.** You extend the SLelement class to hold a distance

value and the prev node identifier. The distance value is continually updated by the BFS traversal(to reflect the shortest path, as in the Dijkstra algorithm; the main difference is that when you visit a child, you add 1 (path length of the edge to the parent's distance value and compare that to the child's current distance value. If a shorter path is found, then the distance value is replaced and the parent node identifier is replaced. **When the target node is found, then we can use the parent identifiers to identify the nodes and highlight them and visualize the path and the Bacon number (path length).**

3. **Bacon Number and Path:** Allow the user to input an actor name. Your program will compute the Bacon number(path distance to 'Kevin Bacon' node) of that actor and display the path (use a unique color to display the nodes). Color the nodes that are not in the path but were visited by another color. On your console(within Eclipse), also print the Bacon number and the path of actors/movies that lead to the Kevin Bacon node.

Evaluation:

You will do an interactive demo of your implementation. This should be a fun project.

Submission Requirements.

Turn in your source code to Moodle; ensure it is well documented.