CSCI 2270
HW6: Graph homework in Java (Last homework!)
Due Saturday, 12/14, at 8 am, by Moodle.

Reading: Graph chapter in Carrano, lecture notes from Wednesday 12/4.

Please begin by downloading Graph.java (the new version posted here) and FestivalOfGraphs.java (which contains stubs for the functions you will write) and GraphTester.java (which is similar to the test code we'll use). Also, download graph.txt, graph2.txt, and graph3.txt, which are files defining particular graphs to play with. The test code loads one of these automatically.

Graph.java is like the file from lab, except that there's a function to display information about the edges to you, called printAdjacencyMatrix(). Use this for sanity checking your graphs; the edge matrix should always be symmetric for the graphs in this assignment, because these graphs are undirected.

The code you'll write will complete the stubs in FestivalOfGraphs.java. The file already contains an init() method that lets you load in a graph from a file. Leave that alone. Write code to complete DepthFirstList, BreadthFirstList, DepthFirstSpanningTree, and BreadthFirstSpanningTree. All of these algorithms are well described in the text, and they are all closely related; get one of them to work and you are close to getting all 4. You will make them able to work from any starting vertex, not just vertex 0.

If you choose to go for extra credit, you can complete Prim's minimum spanning tree algorithm (for 10% extra credit) or Dijkstra's shortest path algorithm (for 15% extra credit). But be aware that extra credit work needs to be done without so much help from the TAs and LAs; they will have their hands full with the basics. These two algorithms are discussed in the text, but not very clearly. If you find the book's explanation of these somewhat ugly, the web may have some better sources. For Prim's, return a graph containing the edges for the minimum spanning tree. For Dijkstra, return an ArrayList (an array) of strings, which are either numbers (the path weight, if the nodes are connected) or the letter '-' if there is no path between the two nodes.

Towards the end of the week, I will post some graphs and solutions so you can test your code against mine. We will test your code using the same test file, but we'll use different graphs than graph.txt. graph2.txt, and graph3.txt. Later this week, I'll post solutions for the graphs in graph.txt, graph2.txt, and graph3.txt so you can check your work.