Last name _____

First name _____

LARSON—MATH 356–SAGE WORKSHEET 08
Spanning Trees!

Reminders

1. Read ahead in our textbook. We're into Chp. 2 and trees

Coding Algorithms

1. Log in to your Sage/CoCalc account.
   (a) Start the Chrome browser.
   (b) Go to `http://cocalc.com` and sign in.
   (c) You should see an existing Project for our class. Click on that.
   (d) Click "New", call it **s08**, then click "Sage Worksheet".
   (e) For each problem number, label it in the Sage cell where the work is. So for Problem 1, the first line of the cell should be `#Problem 1`.
   (f) When you are finished with the worksheet, click "make pdf", email me the pdf (at `clarson@vcu.edu`, with a header that says **Math 356 s08 worksheet attached**).

Saving and Re-using Code

We've coded several graphs now, and have added code for functions of graph invariants and auxiliary functions and stored them in "graphs.sage". I pushed my updated version to your Handouts folder. Either copy that file to your Home directory—or add the new stuff to your own "graphs.sage" file. We'll need those functions.

2. I've updated the copy of "graphs.sage" in your Handouts folder to include what we've added in class. *Copy* the current version from Handouts to your Home directory.

3. *Load* your copy of "graphs.sage". Run: `load('graphs.sage')`.

4. Run: `my_graphs` to see what graphs we have so far. (This is partly a test to make sure your file is loaded.)

Concepts from Our Text
These include: *size, order, complete graphs, bipartite-ness, isomorphic graphs, subgraph, complement, incidence matrix, adjacency matrix, degrees, minimum degree, maximum degree, identical graphs, connectedness, number of components, tree-ness, distances.* **These are all built-in to Sage/Cocalc!**

5. Use the built-in functions to test for bipartite-ness, tree-ness, numbers of components, and finding distances.

Trees

How can we test if a graph is a tree? There are many ways. In class we proved that a graph is a tree if and only if it is connected and every edge is a cut-edge. We used this idea to code our `is_tree(g)` function.

A *spanning tree* of a connected graph G is a spanning subgraph of G which is a tree. (Is it true that every connected graph has a spanning tree?)

6. How can we find a spanning tree of a connected graph? Write a function that inputs a connected graph and returns a spanning tree of that graph.

7. How can we find a minimum weight spanning tree of a connected weighted graph? Write a function that inputs a connected weighted graph and returns a minimum weight spanning tree of that graph.

8. Let K_n be the complete graph on n vertices. Give the edges random weights in $[0, 1]$. What is the *average* (expected) weight of a minimum weight spanning tree?