Last name _____

First name _____

LARSON—OPER 731—SAGE WORKSHEET 02
Graph Theory Basics

1. Log in to your Sage/Cocalc account.

   (a) Start Chrome browser.

   (b) Go to `http://cocalc.com`

   (c) Click "Sign In".

   (d) Click project **Classroom Worksheets**.

   (e) Click "New", call it **s02**, then click "Sage Worksheet".

   A **graph** is a mathematical object consisting of *points* and *lines* (also called *vertices* and *edges*). Sage includes the `graphs` class which contains a number of *methods*. Some of these include constructors for making well-known graphs. "g1" is an arbitrarily chosen name in what follows. We could use "pete" (or anything else) instead!

2. We can create our own graph using the `Graph()` constructor, and the `add_vertex()` and `add_edge()` methods. Lets make a `cycle` on 5 vertices. First initialize the graph and make the vertices:

```
g=Graph()
for i in [1..5]:
    g.add_vertex()
g.show()
```

   Notice that the vertex labels start at 0. Now make the edges:

```
for i in [0..3]:
    g.add_edge(i,i+1)
g.show()
```

   You're still missing an edge. What will you add to the code to get the missing edge?

3. Evaluate:

```
g1=graphs.PetersenGraph()
g1.show()
```

   The *order* of a graph is the number of vertices it has. The *size* of a graph is the number of edges it has. How many vertices and edges does the Petersen graph have? Evaluate `g1.order()` and `g1.size()`.

4. Find the order and size of the icosahedron graph. Use `g2=graphs.IcosahedralGraph()`, and `show` the graph.

5. Find the order and size of the dodecahedron graph. Use `g3=graphs.DodecahedralGraph()`, and `show` the graph.

6. Find the order and size of the tetrahedron graph. Use `g4=graphs.TetrahedralGraph()`, and `show` the graph.

7. Find the order and size of the octahedral graph. Use `g5=graphs.OctahedralGraph()`, and `show` the graph.

   Another way to represent a graph with order n is with an $n \times n$ *adjacency matrix* A. If the vertices of the graph are $\{v_0, v_1, \ldots, v_{n-1}\}$ (or {1,2,...,n-1} for short) then the $A_{i,j}$ is 1 if there is an edge from vertex i to vertex j, and 0 if there is not.

8. Evaluate: `g1.adjacency_matrix()` to see the adjacency matrix for the Petersen graph. Write the matrix. Make sure you understand the pattern of 0's and 1's.

9. One way to make a graph is to start with a number of vertices and then for each pair of vertices n and m, flip a coin to decide whether to put an edge between those vertices. `random()` gives a random number between 0 and 1. Try this:

```
g=Graph(10)
for i in [0..9]:
    for j in [0..9]:
        if i<j and random()<.5:
            g.add_edge(i,j)
g.size()
g.show()
```

10. The study of *random graphs* is huge and important and was initiated in a 1959 paper of Erdős and Renyi. Sage has a built in function to do this: `graphs.RandomGNP(n,p)`, where n is the number of vertices you want, and p is the probability of an edge ($0 \leq p \leq 1$). To simulate a coin flip, use $p = .5$. Run the following code a few times.

```
g=graphs.RandomGNP(10,.5)
g.size()
g.show()
```