Last name _____

First name _____

LARSON—MATH 556–SAGE WORKSHEET 09

Finding α-critical graphs

1. Log in to your Sage/Cocalc account.

   (a) Start Firefox or Chrome browser.

   (b) Go to `http://cocalc.com` and "Sign In".

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

   (d) Click "New", call it **s09**, then click "Sage Worksheet".

   We can use Sage to find graphs with certain properties. The first thing to learn is a way to name graphs and have Sage return graphs with certain names.

2. Evaluate `pete = graphs.PetersenGraph()`. A useable name for the Petersen Graph, called a "graph6 string" can be found by evaluating `pete.graph6_string()`. What did you get?

3. A graph6 string is just a way to code the adjacency matrix of a graph. Since the adjacency matrix is not unique, neither is this code. Given a graph6 string Sage knows how to de-code it, and construct the graph. Evaluate `g = Graph('IheA@GUAo')`. This creates a graph named g from the graph6 string 'IheA@GUAo'. Now evaluate `g.show()` to see what g looks like and draw it.

4. Let `k_3_4 = graphs.CompleteBipartiteGraph(3,4)`. Find a graph6 string for this graph.

5. Here is code to generate *all* graphs with 3 points, test them to see if they are connected and bipartite, and to print out the graph6 string for any that are. Evaluate and write the result.

```
for g in graphs(3):
    if g.is_connected() and g.is_bipartite():
        print g.graph6_string()
```

6. You should have gotten a graph6 string for a single graph. Now use that string to generate and show a graph. What graph was it? Draw it.

7. Imitate the preceding code to find graph6 strings for all connected bipartite graphs with 4 points.

8. Now use *show* to see what they look like and draw them here.

Earlier we proved an interesting result about the structure of τ-critical bipartite graphs. α-critical graphs are also interesting. Define a graph to be α-critical if the deletion of any edge increases its independence number.

9. Prove that the **triangle** C_3 (the cycle on 3 points) is α-critical.

In the last worksheet we used the following code to define an *independence number* function. Here is that code again together with a test for whether a graph g has the property of being connected and α-critical.

```
def independence_number(g):
    return g.independent_set(value_only=True)

def is_alpha_critical(g):
    if not g.is_connected():
        return False
    alpha = independence_number(g)
    for e in g.edges():
        gc = copy(g)
        gc.delete_edge(e)
        alpha_prime = independence_number(gc)
        if alpha_prime <= alpha:
            return False
    return True
```

10. Here is code to print graph6 strings of all connected graphs with 5 points that are α-critical. What are they?

```
for g in graphs(5):
    if is_alpha_critical(g):
        print g.graph6_string()
        g.show()
```