Last name ______________________

First name ______________________

LARSON—OPER 731—HOMEWORK WORKSHEET 10
König-Egervary Theorem in Sage.

1. Log in to your Sage Cloud account.
   (a) Start Chrome browser.
   (b) Go to `http://cocalc.com`
   (c) Click "Sign In".
   (d) Click the project for our course.
   (e) Click "New", call it **h10**, then click "Sage Worksheet".

2. The König-Egervary Theorem is a theorem for bipartite graphs. First we'll make a random bipartite graph g with 9 vertices, relabel it, and view/show it.

```
g=graphs.RandomBipartite(4,5,0.5)
g.relabel()
g.show()
```

3. Define a list V of vertices and E of edges for your graph g:

```
V = g.vertices()
E = g.edges(labels = False)
```

4. Evaluate V and E to see what you have.

5. Find a maximum matching of g (by hand) for your graph g (it will be different for each student).

6. Find a minimum vertex cover of graph g.

7. *Prove* that your matching is maximum and your vertex cover is minimum.

8. Enter and run this Integer Programming *model* of the maximum matching problem:

```
IP = MixedIntegerLinearProgram(maximization=True)
x = IP.new_variable(integer=True,nonnegative=True)

IP.set_objective(sum(x[e] for e in E))
#sets a decision variable corresponding to each edge

for v in V: #vertex constraints (one for each vertex with an incident edge)
    IncidentEdges = [e for e in E if e[0]==v or e[1]==v]
    if len(IncidentEdges) > 0:
        IP.add_constraint(sum(x[e] for e in IncidentEdges) <= 1)

IP.solve()
IP.get_values(x)
```

10. Enter and run this Linear Programming *relaxation* of the maximum matching IP:

```
LP = MixedIntegerLinearProgram(maximization=True)
x = LP.new_variable(nonnegative=True)
LP.set_objective(sum(x[e] for e in E))

for v in V:
    IncidentEdges = [e for e in E if e[0]==v or e[1]==v]
    if len(IncidentEdges) > 0:
        LP.add_constraint(sum(x[e] for e in IncidentEdges) <= 1)

LP.solve()
LP.get_values(x)
```

11. We're solving an LP with real-valued variables. But what do you notice about the decision variables in the solution dictionary? Why shouldn't this surprise you (based on what we learned about this situation in class)?

12. Now let's set-up and solve the Dual IP. We showed in class that this models the minimum vertex cover problem. Enter and run:

```
DualIP = MixedIntegerLinearProgram(maximization=False)
y = DualIP.new_variable(integer=True, nonnegative=True)
DualIP.set_objective(sum(y[v] for v in V))
#sets a decision variable corresponding to each vertex

for e in E: #one constraint for each edge
    DualIP.add_constraint(y[e[0]]+y[e[1]] >= 1)

DualIP.solve()
DualIP.get_values(y)
```

13. Explain Sage's output. How many vertices are in a minimum vertex cover? What minimum vertex cover does it produce?

14. Now let's *relax* this:

```
DualLP = MixedIntegerLinearProgram(maximization=False)
y = DualLP.new_variable(nonnegative=True)
DualLP.set_objective(sum(y[v] for v in V))

for e in E:
    DualLP.add_constraint(y[e[0]]+y[e[1]] >= 1)

DualLP.solve()
DualLP.get_values(y)
```

15. Again we're solving an LP with real-valued variables. But what do you notice about the decision variables in the solution dictionary? Why shouldn't this surprise you?