Last name _____

First name _____

LARSON—OPER 731—SAGE WORKSHEET (h13)
Vertex Packing IPs and LPs in Sage.

1. **Log in to VCU's Athena cluster**.

   The following directions assume you have an Athena account, that you have set up Sage, and that you have set up (using `make`) the CONJECTURING program.

   (a) Start the Chrome browser.

   (b) If you are off-campus, you'll need to connect to the VPN first.

   (c) Then go to `https://athena3.hprc.vcu.edu`

   (d) Login using your VCU EID as your username, and your corresponding VCU password.

   (e) Click the Apps button and a `Sage` session. The default options are fine. This will take a couple of minutes.

   (f) Click the Apps button and start an "athena shell access" session (this will give you a terminal window, where we can issue commands).

   (g) Your Sage session will first say "Queued", then "Starting". When it is ready you will see a button that says, "Connect to Sage". Click that.

   (h) You should then get an "untitled" interactive-Python notebook (ipynb), or the last file you had open the previous time you used Athena.

   (i) When your notebook opens look on the upper-right to make sure the SageMath kernel is running (if it isn't you can change the *kernel*).

2. First make a random graph with 100 vertices. We will use edge probabilities equal to $\frac{1}{2}$. So this is like flipping a coin to decide whether to put an edge between any given pair of vertices. Evaluate `g=graphs.RandomGNP(100,0.5)`.

3. You should have approximately $\frac{1}{2}$ of the possible edges, that is, $\frac{1}{2}\binom{100}{2} = 2500$. Your graph will have more or less than this. Use $g.size()$ to find out how many edges your graph g has.

4. Now let's find the cardinality of a largest vertex packing in g. We will model this as an integer program:

```
p = MixedIntegerLinearProgram(maximization=True)
x = p.new_variable(integer=True, nonnegative=True)
p.set_objective(sum(x[v] for v in g.vertices()))
for (u,v) in g.edge_iterator(labels=False):
    p.add_constraint(x[u] + x[v] <= 1)
p.solve()
```

5. 100 vertices is small enough that the vertex packing number (independence number) can be computed by Sage's algorithm (`cliquer`). Run: `g.independent_set(value_only=Tru` to see if you get the same value.

6. If g were large enough, the IP solver would likely not finish. But we can *relax* our problem by replacing the integer requirement with a non-negative real number requirement. Run:

```
p = MixedIntegerLinearProgram(maximization=True)
x = p.new_variable(nonnegative=True)
p.set_objective(sum(x[v] for v in g.vertices()))
for (u,v) in g.edge_iterator(labels=False):
    p.add_constraint(x[u] + x[v] <= 1)
p.solve()
```

7. What did you get? Can you explain this result?

8. To find out how many constraints your LP has, evaluate: `p.number_of_constraints()`. Can you explain this number?

9. Now let's make a larger random graph with say 1000 vertices. Evaluate `g=graphs.RandomGNP(` How many edges does your graph have?

10. I doubt the IP solver would solve the vertex packing IP for this graph in a reasonable time. Find the optimum value of the corresponding vertex packing LP. Can you explain this number?

11. How many constraints does your LP have? (Its probably close to $250,000$. You would want to solve this LP by hand!)