Last name _____

First name _____

LARSON—OPER 731—SAGE WORKSHEET (h07)
Linear Programming 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*).

   Here's the linear program we'd like to solve:

   maximize: $x_1 + x_2 + x_3$

$$\begin{array}{lrcrcrcl} & x_1 & + & x_2 & & & \leq & 1 \\ \text{subject to:} & x_1 & & & + & x_3 & \leq & 1 \\ & & & x_2 & + & x_3 & \leq & 1 \end{array}$$

   We will let `LP` be the name of our linear program. Of course, the name can be *anything*; it can be `Dantzig`, `D` or `Mathzilla`, anything that you are not already using for something else.

   We will also tell Sage that our objectiive is to find the **maximum** value of the objective function.

2. Evaluate `LP = MixedIntegerLinearProgram(maximization=True)`

   Now we will let x be the name of the variable vector, and also require that the vector entries be non-negative.

3. Evaluate `x = LP.new_variable(nonnegative=True)`.

4. Evaluate `LP.set_objective(x[1] + x[2] + x[3]))`.

   Now lets add our constraints.

5. Evaluate:

```
LP.add_constraint(x[1] + x[2], max = 1)
LP.add_constraint(x[1] + x[3], max = 1)
LP.add_constraint(x[2] + x[3], max = 1)
```

6. Now evaluate `LP.solve()` to solve the linear program. What do you get?

   This should print the maximum possible value of the objective function. (And there is probably a teeny error - all LP solvers are subject to some numerical instability.) All the work was done in the last step. If the LP is big this could take some time. Now let's see a feasible solution that attains the optimal value.

7. Evaluate `LP.get_values(x)`. What do you get?

   If you want only integer solutions for x—turning our problem into an Integer Programming problem. We can set x to be integer.

8. Evaluate `LP.set_integer(x)`. Then we can resolve. Evaluate `LP.solve()` and `LP.get_values(x)` again. What do you get? What does it mean?

   Now lets set up the dual of the last linear program. We found that the dual is:

   minimize: $y_1 + y_2 + y_3$

$$\begin{array}{lccccccc} & y_1 & + & y_2 & & & \geq & 1 \\ \text{subject to:} & y_1 & & & + & y_3 & \geq & 1 \\ & & & y_2 & + & y_3 & \geq & 1 \end{array}$$

   We'll call this system LPdual. Here are all the steps.

9. Evaluate:

```
LPdual = MixedIntegerLinearProgram(maximization=False)
y = LPdual.new_variable(nonnegative=True)
LPdual.set_objective(y[1] + y[2] + y[3])
LPdual.add_constraint(y[1] + y[2], min = 1)
LPdual.add_constraint(y[1] + y[3], min = 1)
LPdual.add_constraint(y[2] + y[3], min = 1)
LPdual.solve()
LPdual.get_values(y)
```

   What did you get? What does it mean?