Last name ______________________

First name ______________________

LARSON—OPER 731—CLASSROOM WORKSHEET 08
Linear Programming 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 **c08**, then click "Sage Worksheet".

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

   maximize: $x_1 + x_2 + x_3$

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

   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 objective 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)`.

   Now we will tell Sage what the objective function is. Notice that we are also implicitly giving the components of vector x the names $x[1]$, $x[2]$ and $x[3]$.

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] <= 1)
LP.add_constraint(x[1] + x[3] <= 1)
LP.add_constraint(x[2] + x[3] <= 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?

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

   minimize: $y_1 + y_2 + y_3$

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

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

8. 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] >=1)
LPdual.add_constraint(y[1] + y[3] >= 1)
LPdual.add_constraint(y[2] + y[3] >= 1)
LPdual.solve()
LPdual.get_values(y)
```

9. We can also find the *polyhedron* corresponding to an LP. For our example above, evaluate: `p=LP.polyhedron()`. The polyhedron object has methods to produce relevant information. Try: `p.vertices_list()`.

10. We can also get inequalities with the same feasible region. Try: `inequalities()`.

11. Now find the polyhedron defined by the feasible region of the dual LP, its vertices, and its defining inequalities.

12. To set up an integer programming problem we only need to switch a parameter to make our variable be integer. Re-write our LP definition: change the name to IP and use `x = IP.new_variable(integer=True, nonnegative=True)`. Solve it to get an optimum and values that obtain that optimum.

13. Do the same for the LPdual: restrict the variables to be integer. Call it IPdual. Use `y = IPdual.new_variable(integer=True, nonnegative=True)`. Solve it to get an optimum and values that obtain that optimum.