Last name _____

First name _____

LARSON—OPER 635–SAGE WORKSHEET 15
Degree Relaxation TSP Tour with Subtour Elimination

1. Log in to your Sage Cloud account.

   (a) Start Firefox or Chrome browser.

   (b) Go to `http://cloud.sagemath.com`

   (c) Click "Sign In".

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

   (e) Click "New", call it **s14**, then click "Sage Worksheet".

   The degree relaxation TSP produced a "tour" that had only some of the features of a tour. Among other things it included sub-tours (or islands). This provided a lower bound on the length of an optimal tour.

   Now we will find a tour that satisfies the degree relaxation constraints **and** successively eliminates subtours.

2. Cut-and-paste your code from the previous worksheet to re-create the graph DFJ (of the 42 US cities and their distances).

3. To observe the improved tours as we add subtour constraints, start by re-initializing the degree relaxation LP (named `LPd` with decision variables named `xd`) and solving. What is the current length of an optimal "tour"?

```
LPd = MixedIntegerLinearProgram(maximization=False)
xd = LPd.new_variable(nonnegative=True)
LPd.set_objective(sum([DFJ.edge_label(i,j)*xd[(i,j)] for i in range(42) \
    for j in range(42) if j>i]))
for v in DFJ.vertices():
    LPd.add_constraint(sum([xd[u] for u in [(i,j) for i in range(42) \
        for j in range(42) if (i==v and j>i) or (j==v and i<j)]]) == 2)
for e in DFJ.edges(labels=False):
    LPd.add_constraint(x[e], max=1)
LPd.solve()
```

4. The "solution" (which is not actually a tour) consists of arcs whose corresponding decision variables are 0, 1 or $\frac{1}{2}$ (adding up to 2 for each node). Let's look at the pieces of this graph consisting of arcs whose corresponding decision variables are 1. In a solution there will be exactly one component (consisting of all the nodes). Evaluate:

```
tour_graph = Graph(42)
D = LPd.get_values(xd)
tour_graph.add_edges([(i,j) for (i,j) in D if D[(i,j)]==1])
tour_graph.show()
```

5. It looks like there are 4 components. To confirm, evaluate:
`len(tour_graph.connected_components())`

6. For each of these 4 subtours, we'll add a constraint that says that the sum of the decision variables for the arcs in this tour must sum to no more than one less than the number of nodes in this component (currently they add up to the number of nodes, which is too much). We can do this in a loop, and without hand-typing the arc labels ourselves.

```
for i in range(len(tour_graph.connected_components())):
    subtour = tour_graph.connected_components()[i]
    LPd.add_constraint(sum([xd[u] for u in [(i,j) \
        for i in subtour for j in subtour if j>i]]) <= len(subtour)-1)
```

7. So now `LPd` consists of all the original constraints plus the 4 new subtour constraints. Let's solve and see if we get an improvement. Evaluate: `LPd.solve()`. What is the current length of an optimal "tour"?

8. Again let's look at the pieces of this graph consisting of arcs whose corresponding decision variables are 1. In a solution there will be exactly one component (consisting of all the nodes). Evaluate:

```
tour_graph = Graph(42)
D = LPd.get_values(xd)
tour_graph.add_edges([(i,j) for (i,j) in D if D[(i,j)]==1])
tour_graph.show()
```

9. Again add constraint for each subtour. Evaluate:

```
for i in range(len(tour_graph.connected_components())):
    subtour = tour_graph.connected_components()[i]
    LPd.add_constraint(sum([xd[u] for u in [(i,j) \
        for i in subtour for j in subtour if j>i]]) <= len(subtour)-1)
```

10. Again let's solve and see if we get an improvement. Evaluate: `LPd.solve()`. What is the current length of an optimal "tour"?

11. Repeat this procedure one more time: identify subtours, add constraints that will be violated by our current tour and report the current optimum. Is this result a genuine tour? You can make a picture using the values of the keys in the LP solution. Evaluate:

```
for key in D:
    if D[key] > 0:
        print key, D[key]
```