Last name ______________________

First name ______________________

LARSON—MATH 750–SAGE WORKSHEET 02
Getting Started with Sage/Colcalc.

1. Create a Sage/Cocalc account.
   (a) Start the Chrome browser.
   (b) Go to `http://cocalc.com`
   (c) Login. You should see an existing Project for our class. Click on that.
   (d) Click "New", then "Worksheets", then call it **s02**.

   Sets in Sage

2. First define the set of integers $S = [10]$. Run: `S = Set([1..10])`.
3. Evaluate (run): `S`, to see what S is.
4. Let's generate the power set of S and call it X. Run: `X=Subsets(S)`. Then evaluate: `X`.
5. How big is this collection? Run: `len(X)`.
6. We can use *list comprehension* to define new collections—very similarly to the way we do it in class (and books, etc). Lets generate the collection of sets from X which contain between 2 and 5 elements. Run: `X2 = [s for s in X if 2 <= len(s) <= 5]`.
7. $X2$ is no longer a generator-item but a genuine list. Run: `X2`. Ugh. Well, how many sets are in this collection?
8. Last week we wrote a function to check is a family of sets is reflexive and another to check if it is anti-symmetric. Can you write a function to check if a family of sets X is transitive? Make sure it works on X and $X2$.

   Integers in Sage

   Sage was originally developed by number theorists and includes state-of-the-art number theory functionality. Integers in Sage are Python *objects* and thus like sets or any other objects come with *methods*, including of course the "divides" relation we used for our divisibility poset.

9. Run: `5.divides(10)`, and `7.divides(0)`.
10. Can you write code to check if the "divides" relation is reflexive for our set S?

12. Can you write code to check if the "divides" relation is transitive for our set S?

13. Can you write a function `reflexive(T)` and checks if an arbitrary set of integers T is reflexive with respect to divisibility?
Test your function with S.

14. Can you write a function `reflexive(T, f)` and checks if an arbitrary set of integers T is reflexive with respect to a relation f (f will have to itself be a function which takes two integers as inputs)?

Graphs in Sage

Sage includes the `graphs` class which contains a number of *methods*. Some of these include constructors for making well-known graphs. "pete" is an arbitrarily chosen name in what follows. We could use G or anything else instead!

15. Evaluate:

```
pete=graphs.PetersenGraph()
pete.show()
```

Here is a list of common graphs that have pre-built constructors: `http://doc.sagemath.org/html/en/reference/graphs/sage/graphs/graph_generators.html`

Find another one of these that interests you and `show` it.

We ultimately want to estimate the independence number of a graph (or the value of any other graph invariant) by developing an approximation function (mobius inversion) that equals the independence number of the graph in the limit. (By the way, whatever we do here can be imitated for *any* combinatorial structure and *any* invariant of that structure.

16. To find a maximum independent set in a graph g, run `g.independent_set()`. We haven't defined any graph named g yet, but we have pete, so try it for pete.

17. To find the independence number for a graph g, run `g.independent_set(value_only = True)`. We haven't defined any graph named g yet, but we have pete, so again try it for pete.

18. We'll want to construct a variety of subgraphs of a given graph. To get the subgraph *induced* by a subset S of vertices, use: `g.subgraph(S)`.
Evaluate: `pete.subgraph([0,1,2,3,4])`.

19. If I want to see what this subgraph looks like I can give it a name and `show` it. Run `H=pete.subgraph([0,1,2,3,4])`. Then run: `H.show()`.

20. To check if a graph g is connected use `g.is_connected()`. Check if pete and H are connected.

21. We will want to generate subgraphs of various kinds. How can we generate all connected induced subgraphs of pete using the vertices $[0, 1, 2, 3, 4]$?