LARSON—MATH 353–CLASSROOM WORKSHEET 18
Perfect Numbers—Conjecturing—Ore.

1. Log in to CoCalc.
   (a) Start the Chrome browser.
   (b) Go to `https://cocalc.com`
   (c) Login (**your VCU email address** is probably your username).
   (d) You should see an existing Project for our class. Click on that.
   (e) Click "New", then "Worksheets", then call it **c18**.

Perfect Numbers & Files

A number (integer) n is *perfect* if the sum of its proper divisors (the divisors less than n) equals n (or equivalently that the sum of all divisors equals $2n$). 6 is the smallest perfect number.

Now it is the case on any larger program that you will want to use functions you have previously defined. These are called *tools*. Instead of copying and pasting from your old code. You can save them as *files* and load them as needed. We are saving all our relevant number theory invariants and properties in the file `perfect_numbers.sage`.

2. Now evaluate: `load("perfect_numbers.sage")`.
3. Now try `is_perfect(n)` for a few values of n to make sure your code is loaded.

The Conjecturing program

We will be using the CONJECTURING program to make conjectures of the form: **if and integer has property P then the integer is perfect**. Here's we say: **An integer having Property P is a sufficient condition for the integer to be perfect**. Finding sufficient conditions for being perfect adds to the *theory* of integer perfect-ness. In general adding to or extending existing theory can give researchers new tools for answering unsolved or open questions, such as: Do odd perfect numbers exist?

By the design of the CONJECTURING program, all produced conjectures are guaranteed to be true for all *input* integers. We interpret the conjectures as being true for *all* integers. This claim may be true or may be false. If it is true, we must *prove* it; and if it is false, we must find an example that shows or demonstrates that the conjecture is false. Such an example is called a **counterexample**. Every time the CONJECTURING program produces a conjecture that is false and we find a counterexample we will add that counterexample to the list of input integers for the next run of the program.

4. Switch back to your $c18$ Sage worksheet and test your installation. First `load` the CONJECTURING program by running the command: `load("conjecturing.py")`.
5. Now let's see if we can get a conjecture with the "properties" version of the CONJECTURING program. Type the following lines into a $c18$ worksheet cell and then run it:

```
Integers = [6,28,8]

Properties = [is_perfect, Integer.is_prime, Integer.is_square,
Integer.is_squarefree]

Prop_of_interest = Properties.index(is_perfect)

propertyBasedConjecture(Integers,Properties,Prop_of_interest,sufficient=True)
```

6. What conjectures do you get? (Are they true? If not find a counterexample and add it to Sage. Then re-run to get new conjectures.)

7. To be maximally useful, we need *lots* of integer properties (these will go in the Properties list). What else can we add? (And at some point CONJECTURING will stop producing interesting conjectures: either the program times out with no conjectures or the conjectures become too complicated to think about. At this point the best way to move forward is to add new properties. In general the more "knowledge" we have of a subject—including knowledge of what properties the "objects" of the subjects have—-the more things we can say about those objects.)

8. Two related properties would be having a divisor sum which is more than $2n$ and having a divisor sum which is less than $2n$. Call these `has_divisor_surplus(n)` and `has_divisor_deficiency(n)`. Test them.

9. Find all the integers up to 100 with divisor sum less than $2n$.

10. If both functions work, add them to your file `perfect_numbers.sage`.

11. Now add the names of your two new properties to the Properties list and re-run CONJECTURING.

12. What other integer properties do we already know—or that we can cook up—to add to our Properties list (and, after suitable testing, to `perfect_numbers.sage`)?

Ore

13. Ore defines $\nu(n)$ to be the number of divisors of integer n and $H(n) = \frac{\nu(n)}{2}$ to be the *harmonic mean* of n. He argues that for perfect numbers this is an integer. How can we define this property? Call it `has_integral_harmonic_mean`. Find all the *Ore harmonic numbers* up to 100,000. Make sure we get the 4 perfect numbers below 100,000.

14. Now add `has_integral_harmonic_mean` to your `.sage` file, your Properties list, rerun `Conjecturing`, and see if you get any interesting new conjectures.

15. **Getting your classwork recorded**

    When you are done, before you leave class...

    (a) Click the "Make pdf" (Adobe symbol) icon and make a pdf of this worksheet. (If Cocalc hangs, click the printer icon, then "Open", then print or make a pdf using your browser).

    (b) Send me an email with an informative header like "Math 353—c18 worksheet attached" (so that it will be properly recorded).