LARSON—MATH 353–CLASSROOM WORKSHEET 20
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 c20.

Perfect Numbers Investigation
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.

Our immediate goal is to find sufficient conditions for a number to be perfect? If we can prove something this would also apply to odd perfect numbers. (So is \( P \) were a sufficient condition for a number to be perfect and an odd number had property \( P \) then it would be perfect!)

As we go we are storing all tested definitions and properties in perfect_numbers.sage. Several others have been added there, as well as the command load("conjecturing.py")—so that program will be loaded every time perfect_numbers.sage is loaded.

2. So evaluate: load("perfect_numbers.sage") to load what we have so far (my current copy is in the Handouts folder. Your copy may have different functions. It should be in your Root/Home directory. Try is_perfect(n) for a few values of \( n \) to make sure your code is loaded.

The Conjecturing program
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.

3. The 8th Run. The last conjecture we generated is true for all positive integers:

\(((\neg\text{has_divisor_deficit})\neg\text{has_divisor_surplus})\rightarrow\text{is_perfect})\)

Explain how to interpret this conjecture and why it is true. (Recall, the caret symbol represents exclusive or or XOR).
4. **Ore** Ore defines $\nu(n)$ to be the number of divisors of integer $n$, $\sigma(n)$ to be the sum of all divisors of $n$ and $H(n) = \frac{\nu(n)}{\sigma(n)}$ 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. Check that we get the 4 perfect numbers below 100,000.

5. **A new Run.** Evaluate:

```python
# 9th run, perfect numbers 496, 8128 added, integral harmonic number 140 added
#properties has_integral.harmonic.mean, is_odd, is_even added
Integers = [6, 28, 496, 8128, 8, 15, 7, 25, 12, 140]

Properties = [is_perfect, Integer.is_prime, Integer.is_square, Integer.is_squarefree, has_divisor_surplus, has_integral.harmonic.mean, is_odd, is_even]

Prop_of_interest = Properties.index(is_perfect)

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

If a conjecture is true, the only way to be certain is to *prove* it. If it is false, the only way to be certain of that is to find an example that demonstrates falsity (a *counterexample*).

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. 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`)? See the new *invariants* added there—these might be useful for inventing useful new properties.

8. **Necessary Conditions.** We can also investigate *necessary* conditions for a number to be perfect. Try:

```python
#Necessary Condition Conjectures. 1st Run.
Integers = [6, 28, 496, 8128, 8, 15, 7, 25, 12, 140]

Properties = [is_perfect, Integer.is_prime, Integer.is_square, Integer.is_squarefree, has_divisor_surplus, has_integral.harmonic.mean, is_odd, is_even]

Prop_of_interest = Properties.index(is_perfect)

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

9. **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—c20 worksheet attached” (so that it will be properly recorded).