

Last name \_\_\_\_\_

First name \_\_\_\_\_

**LARSON—MATH 353—CLASSROOM WORKSHEET 11**  
**Conjecturing in CoCalc/SAGE.**

1. Sign in to your CoCalc account.

- (a) Start the Chrome browser.
- (b) Go to `https://cocalc.com`
- (c) Log in to your account.
- (d) You should see an existing Project for our class. Click on that.
- (e) Make sure you are in your Home directory (if you put files in the Handouts directory they could be overwritten.)
- (f) Click “New”, then “Jupyter Notebook”, then call it **353-c11**.
- (g) Make sure you have SAGE as the *kernel*.

**Review: setting up Conjecturing**

- (a) We set up the conjecturing program by following the steps at:  
`http://nvcleemp.github.io/conjecturing/`
- (b) The Conjecturing program .zip file is in your 353-handouts directory.
- (c) *Move* the program to your Home directory.
- (d) Click “New” and then “Terminal” to get a Sage terminal window.
- (e) 

```
~$ unzip conjecturing-0.13.zip
~$ cd conjecturing-0.13
~/conjecturing-0.13$ make
```
- (f) If you were successful, you have a `conjecturing.py` file and an `expressions` file in your Home directory.
- (g) Look in your Home directory. You should see a `conjectures.py` file and an `expressions` file.

**Facts about the Produced Conjectures**

- (1) **Truth.** They are TRUE for every input object.
- (2) **Significance.** Each conjecture, when added to the list of conjectures, was “better” for at least one input object than any previously stored conjecture.

2. Code and test each of the following procedures.

```
1 def digits10(n):
2     return len(n.digits(10))
3
4 def digits2(n):
5     return len(n.digits(2))
```

```

1 def count_divisors(n):
2     return len(divisors(n))
3
4 def count_prime_divisors(n):
5     return len(factor(n))
6
7 def number(n):
8     return n

```

3. Here is a minimal test for generating upper-bound conjectures for an integer  $n$ .

```

1 invariants = [number, digits10, digits2]
2 objects = [2, 3, 10, 147]
3 invariant_of_interest = invariants.index(number)
4 conjectures = conjecture(objects, invariants, invariant_of_interest,
    upperBound = True, debug = 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*).

4. 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.)
5. Now generate lower-bound conjectures for an integer  $n$ . Are they true?
6. Now let's expand the included invariants.

```

1 invariants = [number, digits10, digits2, count_divisors,
    count_prime_divisors]
2 objects = [2, 3, 10]
3 invariant_of_interest = invariants.index(number)
4 conjectures = conjecture(objects, invariants, invariant_of_interest,
    upperBound = True, debug = True)

```

7. 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.)
8. We can get better conjectures by adding more integer invariants. What can we add?

## Property Conjectures

For integers (or any other object-type) we can conjectures necessary or sufficient conditions for an integer to have that property.

```

1 properties = [is_prime, is_even]
2 property_of_interest = properties.index(is_prime)
3 objects = [3]
4 propertyBasedConjecture(objects, properties, property_of_interest)

```

9. 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.)

### **Getting your classwork recorded**

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

- (a) Click the “Print” menu choice (under “File”) and make a pdf of this worksheet (html is OK too).
- (b) Send me an email ([clarson@vcu.edu](mailto:clarson@vcu.edu)) with an informative header like “Math 353 - c11 worksheet attached” (so that it will be properly recorded).
- (c) Remember to attach today’s classroom worksheet!