**Last name** _____

**First name** _____

**LARSON—MATH 353–CLASSROOM WORKSHEET 19**
**$n^2 + 1$ Primes Investigation.**

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-c19**.
   (g) Make sure you have SAGE as the *kernel*.
   (h) Look in your `Home` directory. You should see a `conjecturing.py` file and an `expressions` file **AND** today's Jupyter notebook.

**Review**

2. The **question** is: are the infinitely many primes of the form $n^2 + 1$?
3. **Open Conjectures**.

```
count_prime_divisors(x) <= digits10(x)
count_prime_divisors(x) <= minimum(digits10(x), half_count_divisor
count_prime_divisors(x) <= ceil(log(count_divisors(x)))
```

   Are any of these resolved?

4. Here's an idea for new invariants. For any $n = x^2 + 1$ integer, we can also grab onto the $x$ and define any invariants for *that* integer. These are also invariants for $x$. There are no wrong invariants—and also you can't tell ahead of time what invariants will show up in useful conjectures.

   Some of these are now defined and added to `number_theory.sage`. Let's test these.

**Methodology**

   (a) We need `conjecturing.py` loaded for our investigation. I added the command to the `number_theory.sage` file. Every time that file is loaded, the `Conjecturing` program will get loaded too.
   (b) We need *invariants*.

To use the `Conjecturing` program, we'll need some *invariants*. The ones we've coded in class are in the *number_theory.sage* file in your Handouts folder. We don't want to keep re-coding those. We can use this file as a permanent record of everything we've coded for this research. Copy or move this file to your Home directory. We'll do this every class as I update them. If you have/use your own invariants, please name your file something else.

These invariants code some of our built-up knowledge about integers. The more of these we code up, the more interesting, useful, simple and possibly true conjectures we'll get. We can cook these up ourselves, or search books and papers for more.

(c) We need *theorems*.

We will include these in the "lab report" section of `number_theory.sage` and maintain a list, "theorems".

(d) We need *example objects*. We started with a few prime and non-prime integers of the form $n^2 + 1$. and we don't want too many (so we get simpler conjectures. More examples can lead to *overfitting*).

One idea is to just use counterexamples to previously falsified conjectures. This keeps the conjecturing program from ever re-making a previously falsified conjecture. We will include any counterexamples we find in our Lab Report and maintain a list, "objects" which includes all of these.

(e) When we find interesting conjectures we need to either prove them or find counterexamples. This iterative process necessarily leads to new (mathematical) *knowledge*.

(f) The basic form of our conjecture-making calls for an *upper bound* investigation of an "invariant of interest" (`inv_of_interest` for the `invariants` list) is:

```
conjs = conjecture(objects, invariants, invariant_of_interest,
    upperBound = True, theory = theorems, debug=True)
for conj in conjs:
    print(conj)
```

This also gives the name `conjs` to the list of conjectures. We can then use this name to create names for single conjectures (for instance, `conj0 = conjs[0]` for the 0-index conjecture in this list). Individual conjectures (like `conj0`) have an evaluation method we can use for testing.

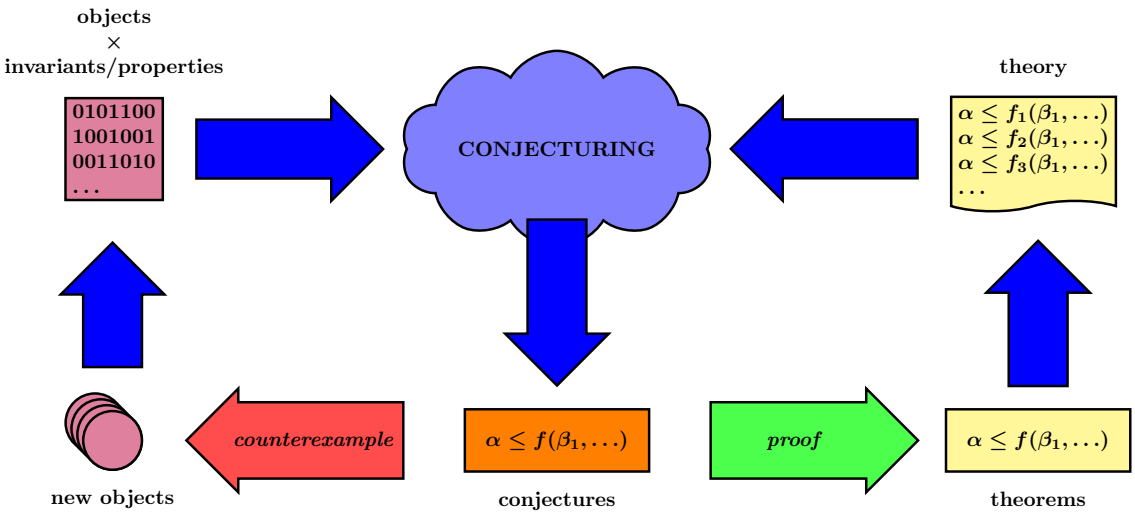
(g) We can push our investigation forward by any of the following:

i. Finding a counterexample to a conjecture and adding it to the examples/objects list.

ii. Proving a conjecture and adding it to the theorems list.

iii. Coding/adding new invariants.

5. Now let's **experiment**!




**Getting your classwork recorded**
When you are done, before you leave class...

1. Click the "Print" menu choice (under "File") and make a pdf of this worksheet (html is OK too).
2. Send me an email (`clarson@vcu.edu`) with an informative header like "Math 353 - c19 worksheet attached" (so that it will be properly recorded).
3. Remember to attach today's classroom worksheet!