Below are some extra examples for Math 301. These examples are to augment the lectures and help you do some of the assigned homework problems.

1. A good piece of software everyone can get and use is Pari/GP. It is completely free. It is available from the Pari/GP website pari.math.u-bordeaux.fr . Pari/GP is included in Sage, so Sage can do anything you can do in GP, but I find GP simpler to install and use for small calculations. I just run in on the command-line.

   For example, in our textbook Harold mentions that Goldbach's conjecture has been verified to 100000 by Pipping. This happened in 1938, and was a very labor-intensive calculation. To duplicate this calculation using GP, just type

   ```
   forstep(i=4,100000,2,j=1;while(isprime(i-prime(j))<1,j=j+1))
   ```

   This will run and return nothing in a mere few seconds (or less!). Had there been any even integer between 4 and 100000 which was not the sum of two primes, the loop never would have ended, so this verifies Goldbach's conjecture to 100000.

   You can change the 100000 to much higher values and let it run and verify Goldbach's well beyond Pipping's result. It would take a rather long time to duplicate the current best result though, which has verified Goldbach's to over 10^{18}.

   This version's more fun: it will spit out one representation of each even number from 4 to 100000 as a sum of two primes:

   ```
   forstep(i=4,100000,2,j=1;while(isprime(i-prime(j))<1,j=j+1);
   print(i," = ",prime(j)," + ",2*i-prime(j)))
   ```

2. **Eulcidean Algorithm Worst-Case Scenario!** When we use the Euclidean algorithm to find the GCD of two positive integers, a and b, with $a > b$, it tends to go pretty quickly. However, some applications take more time than others. For example, if $b \mid a$, then it only takes one step: b divides a and leaves zero remainder, so one application of the division algorithm and we're done.

   On the other hand, sometimes things like this happen:

$$\begin{aligned}
89 &= (1)(55) + 34 \\
55 &= (1)(34) + 21 \\
34 &= (1)(21) + 13 \\
21 &= (1)(13) + 8 \\
13 &= (1)(8) + 5 \\
8 &= (1)(5) + 3 \\
5 &= (1)(3) + 2 \\
3 &= (1)(2) + 1 \\
2 &= (2)(1) + 0
\end{aligned}$$

(In this example, each remainder is more than half the size of the previous remainder (except for the last two steps). If you try other starting pairs, you will see this is not generally the case.)

Notice the pattern here: $89 = 55 + 34, 55 = 34 + 21, 34 = 21 + 13, 13 = 8 + 5, 8 = 5 + 3$, $5 = 3 + 2$. This sequence of numbers, $1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$, is called the **Fibonacci sequence**. Formally, we define the Fibonacci sequence F_i by $F_1 = 1, F_2 = 1$ and $F_i = F_{i-1} + F_{i-2}$ for $i > 2$.

It turns out that consecutive numbers in the Fibonacci sequence are the worst-case scenario for the Euclidean Algorithm.

That is, suppose the Euclidean algorithm takes n steps to finish for a pair of positive integers a and b, $a > b$. Then the smallest a and b can be are F_{n+2} and F_{n+1}.

We can prove this using induction.

Suppose for a pair of integers a and b, $a > b$, it takes n steps for the algorithm to finish. Suppose $n = 1$. Then $b \mid a$, and the smallest positive values that a and b could be are $a = 2$ and $b = 1$, i.e. $a = F_2$, $b = F_1$. So, the $n = 1$ case is true.

Suppose that the statement is true for all values of n up to some $k - 1$. Suppose it takes k steps for a pair a, b. Then we'd have, applying just one step of the algorithm,

$$a = q_0 b + r_0.$$

Then, the problem would be to find the gcd of b and r_0, which will take $k - 1$ steps. By, the induction hypothesis, the smallest values that b and r_0 could be are

$$b = F_{k+1} \text{ and } r = F_k.$$

Plugging these into $a = q_0 b + r_0$, assuming the minimum $q_0 = 1$ yields

$$a = b + r_0 = F_{k+1} + F_k = F_{k+2}.$$

Hence, if the algorithm takes k steps, then a is at least as large as F_{k+2}.

One can show (we might do this later in the course) that

$$F_k \sim \frac{\phi^k}{\sqrt{5}} \text{ where } \phi = \frac{1 + \sqrt{5}}{2}$$

and one can use this to prove that the number of steps needed in the Euclidean algorithm is never more than five times the number of digits of a.

3. We know, thanks to Euclid, that there are infinitely many primes. In 1837, Johann Peter Gustav Lejeune Dirichlet proved that there are infinitely many primes of the form $ak + b$, for any relatively prime integers a and b. So, for instance, there are infinitely many primes of the form $4k + 1$, $4k + 3$, $5k + 1$, $5k + 2$, etc. Dirichlet's proof is rather elaborate; it would take a few weeks in a course to give a nice development of it.

   However, certain special cases are much simpler to prove. Here is one such case.

   Theorem There are infinitely many primes of the form $4k + 3$.

   Proof: Suppose there are finitely many primes of the form $4k + 3$ greater than 3.

Say $m > 0$ and $p_1, p_2, \ldots, p_m$ are all of the primes greater than 3 that are of the form $4k + 3$.

Let $x = 4(p_1 p_2 \cdots p_m) + 3$.

Note that 3 does not divide x.

Since $p_i < x$ for all i and x is of the form $4k + 3$, x is composite.

Write x as $x = q_1 q_2 \cdots q_r$ where $r > 0$ and q_i are (not necessarily distinct) primes.

Since x is odd, $q_i \neq 2$ for any i.

Hence, all q_i are odd, and so are of the form $4k + 1$ or $4k + 3$.

Suppose all q_i are of the form $4k + 1$. That is, suppose all $q_i \equiv 1 \pmod 4$. Then

$$x \equiv 1 \cdot 1 \cdots 1 \equiv 1 \pmod 4.$$

That is, x is of the form $4k + 1$. This is a contradiction, since x is of the form $4k + 3$.

Hence, at least one of the q_i is of the form $4k + 3$; let us suppose q_j is of the form $4k + 3$.

Then $q_j = p_n$ for some n. Then p_n divides x, but since $x = 4(p_1 p_2 \cdots p_m) + 3$,

$$x \equiv 3 \pmod{p_n}$$

and since $p_n \neq 3$,

$$x \not\equiv 0 \pmod{p_n}.$$

In other words, p_n *does not* divide x, and so we have another contradiction.

Thus, it is not possible that the list $p_1, p_2, \ldots, p_m$ contains all the primes of the form $4k + 3$, and hence there are infinitely many such primes. ■.

(Note: A slightly different proof is possible by setting $x = 4(p_1 p_2 \cdots p_m) - 1$ instead.)