Math 458 Assignment 3 - Due Sunday May 3, 11:59pm

This assignment is all about using your computer to do all the stuff we've been talking about, so there's no proofs on it! Just code. Sorry this took me so long to put together. It's important to me that everyone participate, and that *everyone understand everything on the assignment*, so if you notice someone starting to fall behind, either in the math or the python, please bring the entire group to me to ask for help by early next week.

1. Write a Python function which, given n, returns a list of all the primes less than n. There should be 25 primes less than 100, for instance. How many prime numbers are there which are less than 367400?

2. Implement the fast powering algorithm in Python. Use it to compute the last five digits of the number $2^{10^{15}}$. You could in principle do this by computing

   ```
   2**(10**15) % 100000
   ```

   in python, but that calculation would take about 300 terabytes of memory, so maybe don't do that. Test it on some smaller numbers first.

3. Implement the Extended Euclidean Algorithm in Python. Use it to find the inverse of 197189 (mod 999979), and to compute the numbers $\gcd(2^{1000} - 229, 1000! + 98)$ and $\gcd(887519, 146744, 388025, 880464, 189074)$.

4. Implement the Miller-Rabin primality test in Python, using the fast algorithms above. Use it to find a (probable) prime which is over 1000 bits long (≥ 302 digits).

5. Implement the Chinese Remainder theorem in Python. That is, given a list of numbers $x_1, \ldots, x_k$ and a list of moduli $m_1, \ldots, m_k$, solve the system of congruences

   $$\begin{aligned} x &\equiv x_1 \pmod{m_1} \\ &\vdots \\ x &\equiv x_k \pmod{m_k} \end{aligned}$$

   Make sure you're using the fast algorithms you implemented in exercises 2 and 3 to do this, so that you can handle big numbers. You can do this the way I explained in class, but it's probably easier if you do what it says on the wikipedia entry for the Chinese remainder theorem, in the "Existence (direct construction)" section.

6. Now we put it all together! Let $p = 37055228012567588205472561716198899109643$.

   (a) Convince me that p is probably prime.

   (b) Find the smallest primitive root $g \pmod{p}$.

   (c) Why is it practical to compute $\log_g(100) \pmod{p}$?

   (d) **CHALLENGE PROBLEM:** compute $\log_g(100) \pmod{p}$.