USS: Introduction to mathematical cryptography
Tuesday July 19 problems

1. (a) Let $n = 100$. What is the size of n? What is the size of n in bits?
   (b) Let $n = 10,000$. What is the size of n? What is the size of n in bits?
   (c) Let $n = 10^{24}$. What is the size of n? What is the size of n in bits?
   (d) What is
   $$\lim_{n\to\infty} \frac{\lfloor \log_{10} n \rfloor + 1}{\lfloor \log_2 n \rfloor + 1}?$$
   (e) Can you explain the remark in the notes that says that for our purposes we can use either notion of size (in digits or in bits) interchangeably?

2. (a) Let $f(k) = k^3 + 5k^2 - k + 10$. Show that $f \sim k^3$.
   (b) Let $f(k) = k^3 + 5k^2 - k + 10$. Show that $f \ll k^4$.
   (c) Let $f(k) = k^3 + 5k^2 - k + 10$. Show that $f \gg k$.
   (d) Show that $\log k \ll k^a$ for any positive number a.
   (e) Let $f(k) = \sqrt{2^k + k^2 + k}$. Give a simple function g such that $f \sim g$.

3. Let $G = \langle g \rangle$ be a cyclic group and consider the following algorithm to compute g^n: First compute $g^2 = g \cdot g$, then $g^3 = g^2 \cdot g$, then $g^4 = g^3 \cdot g$, and so on, until one computes g^n. In this manner, each operation is a group multiplication, and we count one group multiplication as one step.

   (a) How many steps does it take to compute g^n?
   (b) Is this algorithm fast (the number of steps grows polynomially) or slow (the number of steps grows exponentially)? Remember that to determine if an algorithm is fast or slow, we must consider the number of steps as a function of the **size** of n.

4. Let n be a positive integer, and consider the following algorithm to find a factor of n: First check if n is divisible by 2, then check if n is divisible by 3, then check if n is divisible by 5, and so on, checking each prime number in turn, ending when a prime factor is found. (Assume you have the complete list of prime numbers handy.) In this manner, each operation is a division, and we will count one division as one step.

   If n is even, this algorithm always takes exactly one step, so this algorithm can be pretty fast, there's no doubt about that! Because of this, in this problem we will be interested in getting an upper bound on how many steps it takes to factor n with this algorithm in the **worst-case scenario**. This is like when we were computing the number of steps for schoolbook multiplication and assumed that there were carries in absolutely every position possible.

(a) We will first give an upper bound on the worst-case scenario for this algorithm by considering an algorithm that always takes even more steps to find a factor for n: First check if n is divisible by 2, then check if n is divisible by 3, then check if n is divisible by 4, and so on, checking each integer in turn, and not skipping ahead to the next prime number. What is the worst-case scenario for the number of steps for this algorithm? In other words, after how many divisions are you **guaranteed** to find a factor for n?

(b) Now suppose that you know that there are usually about $\frac{x}{\log x}$ prime numbers that are less than any given integer x. What is the worst-case scenario for the number of steps for our original algorithm, which only attempts division by prime numbers?

(c) Is this algorithm fast (the number of steps grows polynomially) or slow (the number of steps grows exponentially)? Remember that to determine if an algorithm is fast or slow, we must consider the number of steps as a function of the **size** of n.

5. A function $f\colon \mathbb{N} \to \mathbb{N}$ is **negligible** if $f \ll \frac{1}{p(k)}$ for every polynomial $p(k)$.

   (a) Is $f(k) = e^{-k}$ negligible?

   (b) Is $f(k) = e^{-\sqrt{k}}$ negligible?

   (c) Is $f(k) = \frac{1}{k^3+1}$ negligible?