Written problems:

1. Textbook exercise 3.11 (a proposed, but ultimately insecure, alternative to RSA)
2. Textbook exercise 3.13 (Danger of repeating the same modulus with different encrypting exponents)
3. This problem pins down a necessary and sufficient condition for an integer d to be a valid "deciphering exponent" for an integer e in RSA. Let p, q be distinct primes, let $N = pq$, and let L be the least common multiple of $p - 1$ and $q - 1$.
   (a) Prove that for all $a \in (\mathbb{Z}/N\mathbb{Z})^\times$, $\text{ord}_N(a)$ is the least common multiple of $\text{ord}_p(a)$ and $\text{ord}_q(a)$ (use the Chinese Remainder Theorem).
   (b) Prove that there exists an element $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ with $\text{ord}_N(a) = L$.
   (c) Prove that if d, e are positive integers such $m^{de} \equiv m \pmod N$ for all $m \in \mathbb{Z}/N\mathbb{Z}$, then $de \equiv 1 \pmod L$. (Hence it is *necessary* for d and e to be inverses modulo L in order for d to be a deciphering exponent for e.)
   (d) Prove that if d, e are positive integers such that $de \equiv 1 \pmod L$, then $m^{de} \equiv m \pmod N$ for all $m \in \mathbb{Z}/N\mathbb{Z}$. (Hence this is also a *sufficient* condition. This part was proved in class; you may follow the same proof as was given in class if you wish)
4. Textbook exercise 3.19 (making rigorous sense of the "probability $\frac{1}{\ln(n)}$" interpretation of the Prime Number Theorem; two parts)
5. Textbook exercise 3.20, parts (a) and (b) (The probability interpretation for primes in a congruence class)
6. Textbook exercise 3.10 (finding a deciphering exponent can help factor a modulus)
7. Textbook exercise 4.2 (RSA signature examples)

Programming problems:

1. In written problem 2, you saw that it is unsafe to use the same modulus N in two different RSA public keys. In this problem, you will implement the algorithm that Eve could use to exploit that situation, in a more general context.

   Suppose that you know a modulus N, two relatively prime integers e, f, and two powers $m^e \pmod N$ and $m^f \pmod N$ of an unknown integer m. You may assume that m is a unit modulo N. Write a function `mFromPowers(N,e,f,me,mf)` that computes and returns the unknown integer m (you should return m reduced modulo N, i.e. $0 \le m < N$). The integer N will be 1000 bits long in the largest test cases, but a naive approach will earn partial credit.

   Note: this algorithm has peaceful uses as well. In fact, you can think of RSA decryption as a special case: when Alice receives an RSA message, she knows $m^e \pmod N$ and $m^f \pmod N$, where $f = (p-1)(q-1)$ ($m^f \equiv 1 \pmod N$ in this case). Since $\gcd(e, (p-1)(q-1)) = 1$, this function would be able to decipher the message. Take some time to think about why only Alice can do this, and not Eve.

2. Alice decides that she wants to receive messages using a non-standard variant of RSA. Like in the usual RSA, she will choose a public key N, e, where N is a number whose factorization she knows, and $\gcd(e, \phi(N)) = 1$. In this case, she will take $N = pqr$, where p, q, r are distinct primes. To encrypt a message m for Alice ($0 \leq m < N$), Bob computes $c \equiv m^e \pmod{N}$. Write a function `rsaThreePrimes(p,q,r,e,c)` to do the following: given the three primes p, q, r, the number e, and the ciphertext c sent by Bob, recover the original plaintext m.

   Note. While this setup is perfectly functional, in practice it is more efficient to use products of two primes, hence that is the standard. I encourage you to think about why it is more efficient to use only two primes.