Written problems:

1. Textbook exercise 3.7 (RSA example)
2. Textbook exercise 3.8 (Cracking RSA by factoring)
3. Textbook exercise 3.11 (a proposed, but ultimately insecure, alternative to RSA)
4. Textbook exercise 3.13 (Danger of repeating the same modulus with different encrypting exponents)
5. Textbook exercise 3.10 (finding a deciphering exponent can help factor a modulus)
6. Textbook exercise 4.2 (RSA signature examples)

Programming problems:

1. In written problem 4, 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. $0 \leq m < N$). 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. We've discussed in class the need for choosing primes p such that $p-1$ has a large prime factor. It is also considered a good idea to ensure that $p+1$ also has a large prime factor (for reasons we won't discuss). In this problem, you will write a function `strongPrime(qbits,pbits)` to construct such a prime. You will be given integers `qbits` and `pbits`, and should return 3 prime numbers q_1, q_2, p such that both q_1 and q_2 are at least `qbits` bits long, p is exactly `pbits` bits long, and such that $q_1 \mid (p-1)$ and $q_2 \mid (p+1)$. As with last week's `makeQP` problem, I recommend choosing the subordinate primes q_1, q_2 first, and using these to narrow the search for the last prime p.

3. This problem concerns a modular arithmetic problem that we have not yet considered, but which we may need in a future programming problem. You will be given integers m, b, and N, and your goal is to solve the congruence $mx \equiv b \pmod N$ for x. When m is relatively prime to N, this is accomplished by multiplying by the inverse of m; you should figure out how to solve such a congruence in cases where m may have common factors with N. It is possible that no solutions exist. If solutions exist, they can all be described in a single congruence $x \equiv r \pmod M$, where r, M are integers and M is not necessarily the same as N. Write a function `linearCong(m,b,N)` that either returns `None` if no solutions exist, or returns a pair `(r,M)` describing the general solution if solutions do exist.

Hint Re-write the original congruence as an equation with one more variable, and try to convert it to a congruence (possibly with a different modulus) in which the coefficient of x is invertible.