Written problems:

> **Note** Most problems below rely on some notions we will not discuss until Friday 2/22. You may want to wait until then to begin the problem set, or to read ahead.

1. (How to take kth roots modulo a prime) Let p be prime, $y \in (\mathbb{Z}/p\mathbb{Z})^\times$, and let k be a positive integer such that $\gcd(k, p-1) = 1$.

   (a) Let $\ell \equiv k^{-1} \pmod{p-1}$. Prove that if $x \in \mathbb{Z}/p\mathbb{Z}$ is defined by $x \equiv y^\ell \pmod{p}$, then $x^k \equiv y \pmod{p}$.

   (b) Prove conversely that if $x^k \equiv y \pmod{p}$, then $x \equiv y^\ell \pmod{p}$, and conclude that there is a *unique* $x \in \mathbb{Z}/p\mathbb{Z}$ such that $x^k \equiv y \pmod{p}$.

2. Suppose that m and n are integers such that $\gcd(m, n) = 1$.

   (a) Prove that if $a \in \mathbb{Z}$ is divisible by both m and n, then $mn \mid a$. (*Hint:* use Euclid's lemma, which states: if $a \mid bc$ and $\gcd(a, b) = 1$, then $a \mid c$).

   (b) Suppose that $a, b \in \mathbb{Z}$ satisfy the two congruences

   $$\begin{aligned} a &\equiv b \pmod{m} \\ a &\equiv b \pmod{n}. \end{aligned}$$

   Prove that $a \equiv b \pmod{mn}$ as well.

3. Textbook exercise 1.33, part (a). (a way to find order-q elements)
   Also read part (b) and think about it; you don't need to write up a solution, but the statement may help you think about how to solve Programming Problem 1).

> **Note** The statement of this problem uses the notation $\mathbf{F}_p$ as an alternative to $\mathbb{Z}/p\mathbb{Z}$; see Remark 1.23 in the text. When using the notation $\mathbf{F}_p$, the text write $=$ instead of $\equiv \pmod{p}$, and always leaves "take remainders" implicit. For example, they write $a^{(p-1)/q}$ rather than $a^{(p-1)/q} \% p$, but this remainder is always taken implicitly. You are free to do this when writing your solutions as well, as long as you are clear about the fact that you are doing all arithmetic in $\mathbf{F}_p$.

4. Textbook exercise 2.8. Use a computer for the arithmetic. For part (d), I suggest using your naive discrete logarithm function from Problem Set 2. (Elgamal examples)

Programming problems:

1. Write a function `findOrderQ(q,p)` that takes two prime numbers q and p and returns an element $g \in \mathbb{Z}/p\mathbb{Z}$ of order q if possible. See Written Problem 3 for a way to do this. If it is *n*ot possible to find such an element g, your function should `return None`.

2. Write a program which deciphers a message sent to you with Elgamal encryption, given the public parameters p, g, your private key a, and the ciphertext (c_1, c_2). More specifically, write a function `decipherElgamal(p,g,a,c1,c2)` that returns the plaintext `m` (notation as in the table on p. 72). The length of the prime will vary, up to as large as 256 bits.

3. In the ElGamal cryptosystem, it is crucial that Bob generates a new random element (ephemeral key) k for every transmission, for reasons we will discuss in class. In this problem, you will implement some code that Eve might use to break Bob's encryption if he is not careful. Specifically, suppose that Bob sends two messages using the same value of k, and that Eve somehow learns the contents of the first message (we discuss in class some reasons why this is not implausible).

   Specifically, suppose that all parties know the paramters g and p and Alice's public key A, and Eve intercepts two transmissions from Bob to Alice. The first is a ciphertext `(c11, c12)`, which Eve determines corresponds to a plaintext `m1` (perhaps a standard greeting, or a weather report). The second is a ciphertext `c21,c22` corresponding to a plantext `m2` that Eve hopes to extract. Write a function `analyzeElgamal(g,p,A,c11,c12,m1,c21,c22)` that returns `m2`, assuming that Bob has foolishly used the same value of k for both transmissions.

   The largest test cases will use 256-bit primes p, but a naive appraoche will earn partial credit.

4. Alice and Bob use ElGamal encryption on a regular basis, using public parameters p, g. Alice's public key is A. Eve has intercepted two ciphertexts `(c11,c12), (c21,c22)` from Bob to Alice, and has determined in some way that that the plaintext of the first transmission is a specific number `m1` (e.g. it is a standard greeting).

   Furthermore, Eve has a hunch that Bob is not generating his ephemeral keys very well. After last week's problem set, he knows better than to use the same ephemeral key twice, but the keys are still related. In particular, if `k1` was Bob's ephemeral key for the first transmission, his ephemeral key for the second was computed in this way:

   ```
   k2 = u * k1 + v
   ```

   where `u,v` are two integers between 1 and 100 that Eve does not know.

   Write a function `relatedkeys(g,p,A,c11,c12,m1,c21,c22)` that Eve could use to compute the second plaintext `m2` under these assumptions. The function should return a single integer, the value `m2`. The largest test cases will use 256-bit primes p.