Refer to the second page of the Course Survey for instructions on submitting written work on Gradescope, and to the instructions on Problem Set 1 for developing and submitting programming problems.

Written problems

1. Suppose that p is a prime number, and $g \in \mathbb{Z}/p\mathbb{Z}$.

   (a) Suppose that n is any integer such that $g^n \equiv 1 \pmod{p}$, and a, b are two integers such that $a \equiv b \pmod{n}$. Prove that $g^a \equiv g^b \pmod{p}$.

   (b) Deduce from part (a) that if k, ℓ are two integers such that $k\ell \equiv 1 \pmod{p-1}$, then for all $x, y \in \mathbb{Z}/p\mathbb{Z}$, $x^k \equiv y \pmod{p}$ if and only if $x \equiv y^\ell \pmod{p}$. (This means that it is possibly to compute "kth roots modulo p." This idea is crucial to the RSA cryptosystem. See Programming Problem 3.)

   (c) Suppose now that n is the **order of** $g \pmod{p}$, and assume that g is a unit modulo p[1]. Prove that for any two integers a, b,

   $$a \equiv b \pmod{n} \text{ if and only if } g^a \equiv g^b \pmod{p}.$$

   (For one direction of the "if and only if," you simply need to cite part (a). The other direction requires a separate argument, using specific properties of the order.)

   Note: this fact will be very important when we study the "digital signature algorithm" (DSA) in Chapter 4. Roughly speaking, arithmetic mod p (involving powers of g) will happen "in public," while arithmetic mod n will happen "in private," with the difficulty of discrete logarithms providing the veil between the two. The most important thing to remember is that a different modulus is used when working with the exponents.

2. Textbook exercise 1.33 (1.31 in the 1st edition), part (a). Also read part (b) and think about it (this is relevant to Programming Problem 2).

3. Textbook exercise 1.34 (1.32 in the 1st edition), parts (a), (b), and (c).

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

Programming problems

(~~due on the same date as the written problems, Wednesday 2/20 by 10pm~~ due Friday 2/22, due to last week's tech issues)

The test banks and Gradescope autograders are in preparation, and will be available soon. Once the notebooks with samples cases are ready, they will be available in the Dropbox folder linked below:

`https://www.dropbox.com/sh/jrp6f9mk08ewqnf/AABFGBvYxGr_PWkAMkEnpWNHa?dl=0`

1. Write a function `dh(g,p,B)` that performs Alice's role in Diffie-Hellman key exchange, using a randomly chosen secret number a. More precisely: you are given a prime number p, an element $g \in \mathbb{Z}/p\mathbb{Z}$, and Bob's transmission B. Your function should return a pair of two

[1] This clause was added after the fact, due to my error. The statement is not true if $g \equiv 0 \pmod{p}$, so it's necessary to add the assumption that $g \not\equiv 0 \pmod{p}$. I apologize for the error.

numbers A, S, where A is the number you transmit to Bob, and S is the shared secret. You should look up how to generate random numbers in Python (don't worry about finding a "cryptographically secure" random number generator, just use the standard one in Python). The prime p will be between 16 and 256 bits long, and the autograder will run your code several times on each input to make sure that it appears to be choosing the secret number a randomly.

2. 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 2 for a way to do this. If it is *n*ot possible to find such an element g, your function should `return None`.

3. Write a function `kthroot(k,y,p)`, that takes an integer y, a prime p, and an element $y \in \mathbb{Z}/p\mathbb{Z}$, and determines an element $x \in \mathbb{Z}/p\mathbb{Z}$ such that $x^k \equiv y \pmod{p}$. You may assume that $\gcd(k, p-1) = 1$ in all test cases. In the largest test cases, p will be 256 bits long, but a naive approach will receive partial credit.

   Hint: make use of Written Problem 1, part (b).

4. In the ElGamal cryptosystem (to be discussed in class on Friday 2/15), it is crucial that Bob generates a new random element (ephemeral key) k for every transmission, for reasons we 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.