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. Textbook exercise 1.10, parts (a) and (b) (use a calculator/computer for the arithmetic, but show the steps).
2. Textbook exercise 1.16 (1.15 in 1st edition), parts (a), (b), and (c).
3. Textbook exercise 1.19 (1.18 in 1st edition).
4. Textbook exercise 1.24 (1.23 in 1st edition), part (a).
5. Textbook exercise 2.4 parts (a) and (b) (use a calculator/computer for the arithmetic).

Programming problems

1. Write a function `modinv(a,m)` which takes integers a and m as input (where $0 \leq a < m$) and returns the inverse $a^{-1} \pmod{m}$ if it exist. If the inverse does not exist, your function should `return None`. The test cases will range in size, with the largest being 256-bit integers; a naive algorithm will be able to solve about a third of the cases.
2. Write a function `modpow(a,n,m)` that takes integers a, n, m and returns $a^n \pmod{m}$. The exponent n may be positive or negative (it will be positive in at least half of the test cases), but you may assume that if $n < 0$, then a is a unit modulo m (you'll need to use your code from the previous problem to handle the $n < 0$ case). **For this problem, please do not use Python's built-in function for modular powers. You should implement the fast-powering algorithm yourself to see how it works. However, you may look up and use the built-in function on all programming assignments after this one.** The test cases will range in size, with a and n 256-bit integers in the largest case.
3. Write a function `disclog(g,h,p)` that solves the discrete logarithm problem in a naive way (quickly enough to work in less than 1 second if p is a 20-bit prime). Multiple answers are possible (we'll discuss this later); an answer n will be marked correct as long as $g^n \equiv h \pmod{p}$. To allow you to see exactly where the naive approach becomes too slow (or, if you're up for it, to allow you to try to implement better methods), the test bank will include cases where p ranges up to 40 bits, but **you will receive full credit as long as your code solves the test cases up to 20-bit primes**. (Later, we'll discuss and implement an algorithm that can solve the entire test bank).