Written problems:

1. Textbook exercise 4.10 (DSA verification examples)

2. We've seen previously that Elgamal signatures can leak the private key if the signer accidentally repeats an ephemeral key, and will see in class soon that the same is true of DSA. This problem considers the related question: what if she chooses two(or more) ephemeral keys that are closely related? **In both parts, you may freely assume that numbers you need to be invertible are invertible, for simplicity.**

   (a) Suppose that Samantha releases a document D with signature (S_1, S_2), and also a document D' with signature (S_1', S_2'). Assume she created these signatures with ephemeral keys k, k', respectively. Suppose that she makes the error of choosing $k' = 2k + 1$, and that Eve realizes this fact in some way. Find a formula for the private key a that Eve could use to extract it, i.e. stated only in terms of numbers that she knows and using only operations that can be done very efficiently.

   (b) Suppose now that Samantha released *three signatures* on documents D, D', D''. Eve manages to learn that the ephemeral keys satisfy $k + k' + k'' \equiv 0 \pmod{q}$. Find a formula for the private key a that Eve could use to extract it.

   > **Note** These two relationships-between-keys are somewhat artificial, but they generalize substantially with a bit more notation. The point is that Samantha should not allow Eve to discover any relationship among any collection of her ephemeral keys. The best way to do this is to make them truly random.

3. Suppose that Samantha and Victor use the following variation on DSA. The system uses the same public parameters p, q, g as DSA (notation as in Table 4.3 of the textbook). Rather that publishing a single verification key A, Samantha chooses *two* secret signing keys a_1, a_2, and publishes two verification keys A_1, A_2 such that

$$\begin{aligned} A_1 &\equiv g^{a_1} \pmod{p}, \text{ and} \\ A_2 &\equiv g^{a_2} \pmod{p}. \end{aligned}$$

   A signature on a document D consists of a pair of integers (S_1, S_2). When he receives a document D with signature (S_1, S_2), Victor will use the following verification procedure.

   - Compute $V_1 \equiv DS_2^{-1} \pmod{q}$ and $V_2 \equiv S_1 S_2^{-1} \pmod{q}$ (as in DSA).
   - Verify that
$$\left((A_1^{V_1} A_2^{V_2}) \% p\right) \% q = S_1.$$
     (If this equation is false, the signature is considered invalid.)

   Devise an (efficient) *signing procedure* that Samantha could follow to produce valid signatures. Write out your procedure as a Python function (receiving a document D and the parameters and private keys as input), and prove that your program returns a valid signature according to Victor's procedure above.

Programming problems:

1. Write a function `verifyDSA(p,q,g,A,d,s1,s2)` that verifies DSA signatures. Here, p, q, g are public parameters, A is the public (verification) key, d is the document, and (s_1, s_2) is the signature. The function should return `True` or `False`.

2. Devise a method to create "blind forgeries" for a given DSA public key. That is, write a function `dsaBlind(p,q,g,A)` given p, g and A as in DSA, generate integers (d, s_1, s_2) such that (s_1, s_2) is a valid signature for d for the verification key A. You will likely want to adapt the strategy from one of last weeek's problems from Elgamal to DSA. Your method should be non-deterministic; the grading script will give the same test case multiple times to check that the same answer is not returned each time.