Written problems

1. Textbook exercise 4.5.

   Solution.

   (a) The public verification key is $A \equiv g^a \equiv 437^{6104} \pmod{6961}$, which reduces to 2065.

   (b) The signature consists of $S_1 \equiv g^k \equiv 437^{4451} \equiv 3534 \pmod{6961}$, so $S_1 = 3534$, and $S_2 \equiv k^{-1}(D - aS_1) \pmod{p-1}$, which reduces to $4451^{-1}(5584 - 6104 \cdot 3534) \equiv 5888 \pmod{6960}$. So the signature is $(3534, 5888)$.

   Indeed, the signature is verified by checking that $437^{3534}3534^{5888} \equiv 437^{5584} \pmod{6961}$.

2. Textbook exercise 4.6.

   Solution.

   Here is a bit of code in the Python terminal to check these signatures.

```
>>> p = 6961
>>> g = 437
>>> A = 4250
>>> D = [1521,1837,1614]
>>> S1 = [4129,3145,2709]
>>> S2 = [5575,1871,2994]
>>> for i in range(3):
...     lhs = pow(A,S1[i],p)*pow(S1[i],S2[i],p) % p
...     rhs = pow(g,D[i],p)
...     print lhs,rhs
...
231 231
6208 2081
2243 2243
>>>
```

   So we see that the two numbers match for D and D'', but do not match for D'. So the first and third signatures are valid, while the second is not.

3. Textbook exercise 4.9.

   Solution.

   (a) The verification key is $A \equiv 4488^{674} \equiv 4940 \pmod{22531}$, so $A = 4940$.

   (b) The first part is found as follows: $g^k = 4488^{574} \equiv 7954 \pmod{22541}$, so therefore $S_1 = 7954\%751 = 444$. The second part is $S_2 \equiv k^{-1}(D + aS_1) \equiv 297 \cdot (244 + 674 \cdot 444) \equiv 56 \pmod{q}$. So the signature is $(444, 56)$.

4. Textbook exercise 4.10.

   Solution.

   You can check these signatures in the Python terminal as follows. For this code to work, you must first provide a working definition of `inv_mod(a,m)`, a function which computes the inverse of a modulo m, which can be done using the extended Euclidean algorithm.

```
>>> p = 22531
>>> q = 751
>>> g = 4488
>>> A = 22476
>>> S1 = [183,211]
>>> S2 = [260,97]
>>> D = [329,432]
>>> for i in range(2):
...     V1 = inv_mod(S2[i],q)*D[i] %q
...     V2 = inv_mod(S2[i],q)*S1[i] %q
...     lhs = pow(g,V1,p) * pow(A,V2,p) %p %q
...     rhs = S1[i]
...     print lhs,rhs
...
183 183
224 211
>>>
```

   The output shows that the first signatures (part (a)) is valid, while the second one (part (b)) is not.

5. Textbook exercise 4.8.

   Solution.

   (a) If Eve observes that $S_1 = S_1'$, then she can conclude that Samantha has made this mistake.
   Strictly speaking, this inference requires the assumption that g is a primitive root (which is assumed in the textbook's description of ElGamal signatures), since $S_1 = S_1'$ guarantees only that $g^k \equiv g^{k'} \pmod{p}$. However, it turns out that if $k \equiv k' \pmod{\mathrm{ord}_p(g)}$ is a strong enough condition to guarantee that swapping k with k' would not change the signature created, so Eve can safely assume that $k = k'$ if she finds that $S_1 = S_1'$. You may simply replace $p-1$ with $\mathrm{ord}_p(g)$ in the analysis that follows to obtain an argument that would be valid for any g (it would only reconstruct a modulo $\mathrm{ord}_p(g)$, but that is all that Eve needs, since a is always used in an exponent for a).

   (b) Eve knows the following two congruences (I have replaced k' with k and S_1 with S_1' since they are known to be equal).

$$\begin{aligned} S_2 &\equiv k^{-1}(D - aS_1) \pmod{p-1} \\ S_2' &\equiv k^{-1}(D - aS_1) \pmod{p-1} \end{aligned}$$

Multiplying by k on both sides and rearranging, these are equivalent to the following system of two linear congruences.

$$\begin{aligned} kS_2 + aS_1 &\equiv D \pmod{p-1} \\ kS_2' + aS_1 &\equiv D' \pmod{p-1} \end{aligned}$$

In these congruences, only k and a are the only unknowns (to Eve). She may now extract Samantha's private signing key by solving this system using a method of her choice (for example: Gaussian elimination or substitution). It is possible that this process won't recover a uniquely, but only up to a smaller modulus (we will see this possibility in the next part), but this will reduce the problem of determining a to a checking a short list of candidates.

(c) We wish to solve the following system.

$$\begin{aligned} 209176k + 208913a &\equiv 153405 \pmod{348148} \\ 217800k + 208913a &\equiv 127561 \pmod{348148} \end{aligned}$$

We can start by solving for a in terms of k.

$$\begin{aligned} a &\equiv (208913)^{-1}(153405 - 209176k) \pmod{348148} \\ a &\equiv 212917(153405 - 209176k) \pmod{348148} \\ a &\equiv 331469 - 293492k \pmod{348148} \end{aligned}$$

Now we can attempt to solve for k by substituting this expression into the second congruence.

$$\begin{aligned} 217800k + 208913(331469 - 293492k) &\equiv 127561 \pmod{348148} \\ 217800k + 153405 - 209176k &\equiv 127561 \pmod{348148} \\ (217800 - 209176)k &\equiv 127561 - 153405 \pmod{348148} \\ 8624k &\equiv 322304 \pmod{348148} \end{aligned}$$

At this point, we appear to be in trouble: $\gcd(8624, 348148) = 4$, so you cannot invert the coefficient of k. The best we can do is to first reduce the modulus by first dividing through (on both sides of the congruence, and in the modulus) by 4. After doing this, however, the coefficient will be invertible, and we can solve for k.

$$\begin{aligned} 2156k &\equiv 80576 \pmod{87037} \\ k &\equiv (2156)^{-1}80576 \pmod{87037} \\ &\equiv 8518 \cdot 80576 \pmod{87037} \\ k &\equiv 59623 \pmod{87037} \end{aligned}$$

We have now learend something about the ephemeral key k: it may not be 59623 on the nose, but it is congruent to it modulo $87037 = \frac{p-1}{4}$. So there are only four possible values of k:

$$k \quad \text{is one of} \quad 59623, 59623 + 87037, 59623 + 2 \cdot 87037, 59623 + 3 \cdot 87037$$

There are only four options (since 4 was the factor by which we had to reduce $p - 1$ to solve for k), so we can just compute

$$a \equiv 33149 - 293492k$$

for each of those four options. This means that a must be one of:

$$72729, 159766, 246803, 333840.$$

Now Eve must merely try each possibility. In fact,

$$g^{72729} \equiv 1185149 \pmod{p},$$

so Samantha's private signing key must be, in fact, $a = 72729$.

6. (a) Let $f(n)$ denote the probability that two numbers chosen uniformly at random from the first n positive integers are relatively prime. Calculate $f(100k)$ for $k = 1, 2, \cdots, 10$. What appears to be the long-term behavior of the function $f(n)$?

   (b) Calculate the product of $(1 - \frac{1}{p^2})$, where p ranges over all primes up to 100. Give an informal explanation why the result appears to resemble the numbers you computed in part (a).

Solution.

(a) We can use the following function to count the number of pairs (a, b) with gcd 1.

```
def gcd(a,b):
        while b != 0:
                a,b = b,a%b
        return a

def num_pairs(n):
        num = 0
        for i in range(1,n+1):
                for j in range(1,n+1):
                        if gcd(i,j) == 1:
                                num += 1
        return num
```

We can now calculate the probabilities in question as follows.

```
>>> for k in range(1,11):
...     num = num_pairs(100*k)
...     den = (100*k)**2
...     print 'k=',k,':',float(num)/den
...
k= 1 : 0.6087
k= 2 : 0.611575
k= 3 : 0.608833333333
k= 4 : 0.60846875
k= 5 : 0.608924
k= 6 : 0.608330555556
k= 7 : 0.608234693878
k= 8 : 0.6085921875
k= 9 : 0.608211111111
k= 10 : 0.608383
>>>
```

These probabilities appear to be converging to some value, approximately 0.608.

(b) Here's a bit of code to do this computation. Of course we could also have generated the list of primes automatically, but it is short enough that it isn't hard to type them in by hand.

```
>>> primes = [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97]
>>> prod = 1.
>>> for p in primes:
...     prod *= (1 - 1./(p*p))
...
>>> prod
0.60903372539951639
>>>
```

So the product in question is 0.609, which is rather close to the numbers near 0.608 found in part (a).

Informally, we might expect this for the following reason: a is relatively prime to b if and only if there is no prime p such that p divides both a and b. The probability that p divides a is roughly $1/p$, and the probability that p divides b is roughly $1/p$ as well. These events are independent, so the probability that p divides both a and b is roughly $1/p^2$, so $1 - \frac{1}{p^2}$ is roughly the probability that p doesn't divide both a and b.

If we are chosing our integers a and b from a large enough set, the various events

$$E_p = \{\text{choices } (a, b) \text{ such that } p \text{ doesn't divide both a and b}\}$$

ought to be independent of each other, due to the Chinese Remainder theorem. So it stands to reason that the probability that all of these events are true should be roughly equal to the (infinite) product $\prod_p \left(1 - \frac{1}{p^2}\right)$, where p ranges over all primes.

In fact, this informal reasoning can be made precise. The limit, as n goes to ∞, of $f(n)$ is equal to the infinite product described above. They are both equal to $\frac{6}{\pi^2} = \frac{1}{\sum_{n=1}^{\infty} \frac{1}{n^2}}$,

although I will not show the details of this fact. If the n^2 is replaced by n^k, one obtains the probability (rather, the limit of probabilities) such k different integers have no common divisor.

7. Textbook exercise 5.5.

Solution.

(a) There are $100 \cdot 5 = 500$ possible outcomes: one for each pair of a student and a prize.

(b) There are 100 choices of the first prize winner, then 99 choices for the second, then 98 for the third. When choosing prizes, there are first 5 choices, then 4, then 3.
So the total number of outcomes is $100 \cdot 99 \cdot 98 \cdot 5 \cdot 4 \cdot 3$, or $58,212,000$.

(c) The only difference here is that the number of ways to assign prizes is $5 \cdot 5 \cdot 5$, since there is the same slate of options for each winner. This will be larger than the answer to (b). It comes out to $100 \cdot 99 \cdot 98 \cdot 5^3 = 121,275,000$.

(d) The number of ways to choose the winners is $\binom{100}{3} = 161700$. Once the winners are selected, each can choose prize, and each has five choices, so the overall number of outcomes is $\binom{100}{3} \cdot 5^3 = 20,212,500$. This is exactly one sixth of the answer in (c), since each outcome in this problem gives rise to $3! = 6$ outcomes in (c), by arranging the winners in all possible orders.

8. Textbook exercise 5.6.

Solution.

(a)

$$\begin{aligned}(5z+2)^3 &= \binom{3}{0}(5z)^3 + \binom{3}{1}(5z)^2 \cdot 2 + \binom{3}{2}(5z) \cdot 2^2 + \binom{3}{3}2^3 \\ &= 125z^3 + 150z^2 + 60z + 8\end{aligned}$$

(b)

$$\begin{aligned}(2a-3b)^4 &= \binom{4}{0}(2a)^4 + \binom{4}{1}(2a)^3(-3b) + \binom{4}{2}(2a)^2(-3b)^2 + \binom{4}{3}(2a)(-3b)^3 + \binom{4}{4}(-3b)^4 \\ &= 16a^4 - 96a^3b + 216a^2b^2 - 216ab^3 + 81b^4\end{aligned}$$

(c)

$$\begin{aligned}(x-2)^5 &= \binom{5}{0}x^5 + \binom{5}{1}x^4(-2) + \binom{5}{2}x^3(-2)^2 + \binom{5}{3}x^2(-2)^3 + \binom{5}{4}x(-2)^4 + \binom{5}{5}(-2)^5 \\ &= x^5 - 10x^4 + 40x^3 - 80x^2 + 80x - 32\end{aligned}$$

9. Textbook exercise 5.7.

Solution.

(a) Using the definition of $\binom{n}{j}$ in terms of factorials, we can verify the identity as follows.

$$\begin{aligned}
\binom{n-1}{j-1} + \binom{n-1}{j} &= \frac{(n-1)!}{(j-1)!(n-j)!} + \frac{(n-1)!}{j!(n-j-1)!} \\
&= \frac{1}{n-j} \cdot \frac{(n-1)!}{(j-1)!(n-j-1)!} + \frac{1}{j} \frac{(n-1)!}{(j-1)!(n-j-1)!} \\
&= \left(\frac{1}{n-j} + \frac{1}{j}\right) \frac{(n-1)!}{(j-1)!(n-j-1)!} \\
&= \frac{n}{j(n-j)} \cdot \frac{(n-1)!}{(j-1)!(n-j-1)!} \\
&= \frac{n!}{j!(n-j)!} \\
&= \binom{n}{j}
\end{aligned}$$

(b) By the binomial theorem (which could equally well have been the definition of the numbers $\binom{n}{i}$),

$$\begin{aligned}
(x+y)^n &= \sum_{i=0}^{n} \binom{n}{i} x^{n-i} y^i \\
(x+y)^{n-1} &= \sum_{i=0}^{n-1} \binom{n-1}{i} x^{n-i-1} y^i
\end{aligned}$$

Now, consider what happens when the product $(x+y)\left(\sum_{i=0}^{n-1} \binom{n-1}{i} x^{n-i-1} y^i\right)$ is expanded, *without regrouping and simplifying the terms.* Since $(x+y)$ has two terms and $\sum_{i=0}^{n-1} \binom{n-1}{i} x^{n-i-1} y^i$ has n terms, there will be $2n$ terms in this expansion. There are two ways that the monomial $x^{n-i} y^i$ can occur: either as $x \cdot x^{n-1-i} y^i$, or as $y \cdot x^{n-i} y^{i-1}$. Now, observe that the former occurs with coefficient $\binom{n-1}{i}$, and the latter occurs with coefficient $\binom{n-1}{i-1}$. Therefore, upon regrouping terms, the cofficient of $x^{n-i} y^i$ in this product is equal to $\binom{n-1}{i} + \binom{n-1}{i-1}$.

But of course this product is also equal to $(x+y)^n$, hence this coefficient must also equal $\binom{n}{i}$. Therefore $\binom{n-1}{i} + \binom{n-1}{i-1} = \binom{n}{i}$.

(c) Assume that our objects are labeled $\{1, 2, \cdots, n\}$. The number $\binom{n}{j}$ is the number of j-element subsets of $\{1, 2, \cdots, n\}$. We can divide these subsets into two classes: those that include n, and those that do not.

The subsets that *do* include n are in one-to-one correspondence with the $(j-1)$-element subsets of $\{1, 2, \cdots, n-1\}$, by removing the element n. Therefore there are $\binom{n-1}{j-1}$ such subsets.

The subsets that *do not* include n are precisely the j-element subsets of $\{1, 2, \cdots, n-1\}$. Therefore there are $\binom{n-1}{j}$ of these subsets.

Therefore the total number of j-element subsets is $\binom{n-1}{j-1} + \binom{n-1}{j}$, which must therefore be equal to $\binom{n}{j}$.

Programming problems

10. Write a program that verifies whether or not a given DSA signature is valid. You will be given the public parameters and public key, and a document with a purported signature.

Solution. We simply follow the instructions on page 202 of the textbook. One auxilliary method is needed, to compute inverse modulo m. I omit it here since it has come up before.

```
### Omitted: code for inv_mod(a,m)

def check_dsa(p,q,g,A,D,S1,S2):
        V1 = inv_mod(S2,q) * D % q
        V2 = inv_mod(S2,q) * S1 % q
        if (pow(g,V1,p) * pow(A,V2,p) % p % q) == S1:
                return True
        else:
                return False

### I/O
p,q,g,A = map(int,raw_input().split())
D,S1,S2 = map(int,raw_input().split())
if check_dsa(p,q,g,A,D,S1,S2):
        print 'valid'
else:
        print 'invalid'
```

11. Problem 5 showed that a pair of ElGamal signatures using the same ephemeral key can accidentally give away the signer's private key. The same is true in DSA – write a program which take public paramters and a public key for DSA, along with two signed documents that have been signed with the same ephemeral key, and computes the signer's private key from this information.

Solution.

For a valid DSA signature, the following congruence must hold.

$$\begin{aligned} S_2 &\equiv k^{-1}(D + aS_1) \pmod{q} \\ \Leftrightarrow kS_2 - aS_1 &\equiv D \pmod{q} \end{aligned}$$

Therefore, if (S_1, S_2) and (S_1, S_2') are two signatures using the same ephemeral key, for documents D, D' respectively, we have the following system of congruences.

$$\begin{pmatrix} S_2 & -S_1 \\ S_2' & -S_1 \end{pmatrix} \begin{pmatrix} k \\ a \end{pmatrix} \equiv \begin{pmatrix} D \\ D' \end{pmatrix} \pmod{q}$$

Now, the determinant of this matrix is $-S_1(S_2 - S_2') \pmod q$. The constraints of the problem specify that $S_1 \not\equiv 0 \pmod q$ and that $S_2 \not\equiv S_2' \pmod q$; together, these two conditions imply that the determinant is a unit modulo q. Therefore we can use the code from PSet 4 problem 8 to find an inverse of this matrix modulo q, which can then be used to find k and a as follows.

$$\begin{pmatrix} k \\ a \end{pmatrix} \equiv \begin{pmatrix} S_2 & -S_1 \\ S_2' & -S_1 \end{pmatrix}^{-1} \begin{pmatrix} D \\ D' \end{pmatrix} \pmod q$$

So we can find this inverse matrix $\pmod q$, then compute the dot product of the second row with the vector $\binom{D}{D'}$ (and reduce modulo q) to recover a. This is implemented in the following code, which makes use of the solution to PSet 4 problem 8.

```
### Omitted: code for inv_mod(a,m) and inv_mat(A,m)
###     (see PSet 4, #8)

p,q,g,A = map(int,raw_input().split())
D1,S11,S21 = map(int,raw_input().split())
D2,S12,S22 = map(int,raw_input().split())
assert(S12 == S11)
A = ((S21,-S11),(S22,-S12))
B = inv_mat(A,q)
a = (B[1][0]*D1 + B[1][1]*D2)%q
print a
```

12. Suppose that a fair coin is flipped n times, where $n \leq 500$. Write a program which computes the probability that the number of heads shown is between a and b inclusive, where a, b are two given integers. You should print the answer as the numerator and the denominator of a reduced fraction.

Solution.

There are 2^n possible outcomes. The number of outcomes with k heads is $\binom{n}{k}$. Hence the number of outcomes with between a and b heads is $\displaystyle\sum_{k=a}^{b} \binom{n}{k}$. To find the probability as a reduced fraction, we can take this sum as the numerator, 2^n as the denominator, then clear as many powers of 2 as possible from both the numerator and the denominator. This is done in the following code.

```
def binom(n,a):
        prod = 1
        for i in range(a):
                prod *= (n-i)
        for i in range(a):
                prod /= (i+1)
        return prod
```

```
def ways(n,a,b):
        res = 0
        for i in range(a,b+1):
                res += binom(n,i)
        return res

def prob(n,a,b):
        num = ways(n,a,b)
        denom = 2**n
        while num%2 == 0:
                num,denom = num/2,denom/2
        return num,denom

n,a,b = map(int,raw_input().split())
u,v = prob(n,a,b)
print u,v
```