1. [12 points] Suppose that Eve has succeeded in solving the discrete logarithm problem; she has written an efficient function `dlp(g,h,p)` that finds a solution x to the congruence $g^x \equiv h \pmod{p}$, assuming that such a solution exists.

   Show that Eve can use her function to break the Elgamal cryptosystem, by writing a function `analyzeElgamal(p,g,A,c1,c2)` that returns the plaintext m given the parameters p, g, Alice's public key A, and a ciphertext (c_1, c_2) sent to Alice. The notation is as in Table 2.3 at the back of the exam packet (which you may detach for convenience). Your function may call Eve's hypothetical function `dlp`, and you may also use Python's built-in function `pow` for modular powers and assume that you have already written a function `modinv` computing modular inverses.

2. [12 points] Let E be the elliptic curve over $\mathbb{Z}/5\mathbb{Z}$ defined by the equation

$$y^2 = x^3 - x + 1.$$

   Let P be the point $(1,1)$. Determine the point $(-3) \cdot P$. Do the arithmetic by hand, and show your computations.

3. [12 points] Alice is using the Elgamal cryptosystem (as in Table 2.3 at the back of the packet), with parameters $p = 41, g = 11$, private key $a = 7$, and public key $A = 28$. She receives the ciphertext $(c_1, c_2) = (10, 10)$ from Bob. Determine the plaintext m.

   Note. There is a multiplication table modulo 41 at the back of the exam packet, which you can detach for convenience. Using an efficient procedure, it should not be necessary to do more than six or so modular multiplications. You should clearly show your steps, and to receive full credit it should be clear that your method would scale well to larger parameters.

4. [12 points] Eve is using Shanks's Babystep-Giantstep algorithm to try to solve some discrete logarithm problems. Define $p = 929$ and $g = 347$. The order of g modulo p is

$$\mathrm{ord}_{929}(347) = 29$$

   (you may assume this in your answers). Eve chooses stepsize $N = 6$.

   Eve wishes to solve the following two problems.

$$347^x \equiv 20 \pmod{929}$$
$$347^x \equiv 3 \pmod{929}$$

   She therefore contructs three lists, which are shown in the table below.

n	$347^n \mod 929$	$20 \cdot 347^{-6n} \mod 929$	$3 \cdot 347^{-6n} \mod 929$
0	1	20	3
1	347	673	519
2	568	304	603
3	148	568	271
4	261	719	433
5	454	830	589

   (a) Briefly explain why stepsize $N = 6$ is sufficient for these discrete logarithm problems.

(b) One of these two discrete logarithm problems has a solution. Determine which one, and determine a solution x to it.

(c) One of these two discrete logarithm problems does *not* have a solution. Determine which one, and *prove* that it has no solution.

5. Let p, q be two prime numbers.

(a) [2 points] Suppose that $g \in (\mathbb{Z}/p\mathbb{Z})^\times$. State the definition of the *order of g modulo p*.

(b) [4 points] Prove that if $\text{ord}_p(g) = q$, then $p \equiv 1 \pmod{q}$.

(c) [6 points] Suppose that $p = 1 + kq$, for some integer k. Prove that if $g, h \in (\mathbb{Z}/p\mathbb{Z})^\times$ and $h \equiv g^k \not\equiv 1 \pmod{p}$, then $\text{ord}_p(h) = q$.

6. (a) [6 points] Suppose that p is a prime number. Prove that if $b \in (\mathbb{Z}/p\mathbb{Z})^\times$ satisfes $b^2 \equiv 1 \pmod{p}$, then either $b \equiv 1 \pmod{p}$ or $b \equiv -1 \pmod{p}$.

(b) [6 points] Suppose that n is an odd number, and an integer $a \in \{1, 2, \cdots, n-1\}$ satisifes

$$a^{(n-1)/2} \not\equiv 1 \pmod{n} \quad \text{and}$$
$$a^{(n-1)/2} \not\equiv -1 \pmod{n}.$$

Prove that n is *not* prime.

7. Samantha uses ECDSA for digital signatures. This problem follows the notation of Table 6.7 (see the tables at the back of the packet). She is using a point G of order $q = 41$ (of course this is too small to be secure in practice, but it is chosen here to make the computations feasible by hand). The specific elliptic curve used, and the points G and V, are not provided because they are not necessary to solve this problem.

Samantha makes the error of publishing signatures on two different documents, using the same ephemeral key e. The two documents, and the corresponding signatures, are shown below.

d	s_1	s_2
11	26	10
21	26	11

(a) [2 points] Suppose Eve examines these two signatures. What might she notice that would make her suspect that Samantha has repeated an ephemeral key?

(b) [10 points] Determine Samantha's signing key s. You may wish to use the multiplication table modulo 41 at the back of the exam packet for calculations.

8. [12 points] Eve is attempting to solve the Elliptic Curve discrete logarithm problem, and she has managed the following breakthrough. She has written a function `equalComb(P, Q, A, B, p)` with the following behavior. Given two points P, Q on the curve defined by $y^2 \equiv x^3 + A\,x + B \pmod{p}$, the function returns, if possible, four integers i_1, j_1, i_2, j_2 satisfying the following conditions: $i_1 P \neq i_2 P$, $\quad j_1 Q \neq j_2 Q$, and

$$i_1 P \oplus j_1 Q = i_2 P \oplus j_2 Q.$$

You may assume in this problem that, for any P, Q given as input, integers satisfying these conditions exist.

Show that Eve can make use of this function to solve the ECDLP, in cases where the number of points on the elliptic curve is a known prime number q. That is, **write an efficient function** `ecdlp(P,Q,A,B,p,q)` which takes two points P, Q as above, numbers A, B, p defining the elliptic curve, and a prime number q equal to the number of points on the curve, and returns an integer n such that $nP = Q$. Your function may call Eve's hypothetical function `equalComb` described above, and you may also assume that you have already implemented functions for modular powers, modular inverses, elliptic curve addition, and elliptic curve multiplication.

Also **prove that your function works**, i.e. that the returned number n does indeed solve $nP = Q$. As noted above, you may assume in your proof that integers i_1, j_1, i_2, j_2 as described above exist.

You may assume that all of these functions require at most $\mathcal{O}(\log p)$ arithmetic operations. For full points, your function should be efficient enough that it too requires at most $\mathcal{O}(\log p)$ arithmetic operations (but you do not need to prove that this is so).

9. [12 points] Consider the following variant of DSA, considered on Problem Set 10.

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.)

Now suppose that Eve wishes to produce a *blind forgery* for this system. That is, she will create a document D and valid signature (S_1, S_2) (but she will not be able to choose the document in advance; this is why it is called a "blind" forgery). She begins by creating the number S_1 as follows: she chooses two random integers i, j with $1 \le i, j \le q-1$, and then she computes

$$S_1 = (A_1^i A_2^j)\%p\%q.$$

Describe a procedure Eve could now follow to *efficiently* find integers D and S_2 such that (S_1, S_2) satisfies the verification equation for document D.

Also **prove** that the procedure you describe will indeed satisfy the verification equation. You need not write your solution as code, but it should be clear that this can be done in an efficient way.

Reference tables from textbook:

Public parameter creation	
A trusted party chooses and publishes a (large) prime p and an integer g having large prime order in $\mathbb{F}_p^*$.	
Private computations	
Alice	**Bob**
Choose a secret integer a. Compute $A \equiv g^a \pmod{p}$.	Choose a secret integer b. Compute $B \equiv g^b \pmod{p}$.
Public exchange of values	
Alice sends A to Bob $\longrightarrow$	A
B $\longleftarrow$	Bob sends B to Alice
Further private computations	
Alice	**Bob**
Compute the number $B^a \pmod{p}$.	Compute the number $A^b \pmod{p}$.
The shared secret value is $B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \pmod{p}$.	

Table 2.2: Diffie–Hellman key exchange

Public parameter creation	
A trusted party chooses and publishes a large prime p and an element g modulo p of large (prime) order.	
Alice	**Bob**
Key creation	
Choose private key $1 \le a \le p-1$. Compute $A = g^a \pmod{p}$. Publish the public key A.	
Encryption	
	Choose plaintext m. Choose random element k. Use Alice's public key A to compute $c_1 = g^k \pmod{p}$ and $c_2 = mA^k \pmod{p}$. Send ciphertext (c_1, c_2) to Alice.
Decryption	
Compute $(c_1^a)^{-1} \cdot c_2 \pmod{p}$. This quantity is equal to m.	

Table 2.3: Elgamal key creation, encryption, and decryption

Bob	Alice
Key creation	
Choose secret primes p and q. Choose encryption exponent e with $\gcd(e, (p-1)(q-1)) = 1$. Publish $N = pq$ and e.	
Encryption	
	Choose plaintext m. Use Bob's public key (N, e) to compute $c \equiv m^e \pmod{N}$. Send ciphertext c to Bob.
Decryption	
Compute d satisfying $ed \equiv 1 \pmod{(p-1)(q-1)}$. Compute $m' \equiv c^d \pmod{N}$. Then m' equals the plaintext m.	

Table 3.1: RSA key creation, encryption, and decryption

Samantha	Victor
Key creation	
Choose secret primes p and q. Choose verification exponent e with $\gcd(e, (p-1)(q-1)) = 1$. Publish $N = pq$ and e.	
Signing	
Compute d satisfying $de \equiv 1 \pmod{(p-1)(q-1)}$. Sign document D by computing $S \equiv D^d \pmod{N}$.	
Verification	
	Compute $S^e \bmod N$ and verify that it is equal to D.

Table 4.1: RSA digital signatures

Public parameter creation	
A trusted party chooses and publishes a large prime p and primitive root g modulo p.	
Samantha	**Victor**
Key creation	
Choose secret signing key $1 \le a \le p-1$. Compute $A = g^a \pmod{p}$. Publish the verification key A.	
Signing	
Choose document D mod p. Choose random element $1 < k < p$ satisfying $\gcd(k, p-1) = 1$. Compute signature $S_1 \equiv g^k \pmod{p}$ and $S_2 \equiv (D - aS_1)k^{-1} \pmod{p-1}$.	
Verification	
	Compute $A^{S_1} S_1^{S_2} \bmod p$. Verify that it is equal to $g^D \bmod p$.

Table 4.2: The Elgamal digital signature algorithm

Public parameter creation	
A trusted party chooses and publishes large primes p and q satisfying $p \equiv 1 \pmod{q}$ and an element g of order q modulo p.	
Samantha	**Victor**
Key creation	
Choose secret signing key $1 \le a \le q-1$. Compute $A = g^a \pmod{p}$. Publish the verification key A.	
Signing	
Choose document D mod q. Choose random element $1 < k < q$. Compute signature $S_1 \equiv (g^k \bmod p) \bmod q$ and $S_2 \equiv (D + aS_1)k^{-1} \pmod{q}$.	
Verification	
	Compute $V_1 \equiv DS_2^{-1} \pmod{q}$ and $V_2 \equiv S_1 S_2^{-1} \pmod{q}$. Verify that $(g^{V_1} A^{V_2} \bmod p) \bmod q = S_1$.

Table 4.3: The digital signature algorithm (DSA)

Public parameter creation	
A trusted party chooses and publishes a (large) prime p, an elliptic curve E over $\mathbb{F}_p$, and a point P in $E(\mathbb{F}_p)$.	
Private computations	
Alice	**Bob**
Chooses a secret integer n_A. Computes the point $Q_A = n_A P$.	Chooses a secret integer n_B. Computes the point $Q_B = n_B P$.
Public exchange of values	
Alice sends Q_A to Bob $\longrightarrow$	Q_A
Q_B $\longleftarrow$	Bob sends Q_B to Alice
Further private computations	
Alice	**Bob**
Computes the point $n_A Q_B$.	Computes the point $n_B Q_A$.
The shared secret value is $n_A Q_B = n_A(n_B P) = n_B(n_A P) = n_B Q_A$.	

Table 6.5: Diffie–Hellman key exchange using elliptic curves

Public parameter creation	
A trusted party chooses a finite field $\mathbb{F}_p$, an elliptic curve $E/\mathbb{F}_p$, and a point $G \in E(\mathbb{F}_p)$ of large prime order q.	
Samantha	**Victor**
Key creation	
Choose secret signing key $1 < s < q-1$. Compute $V = sG \in E(\mathbb{F}_p)$. Publish the verification key V.	
Signing	
Choose document d mod q. Choose random element e mod q. Compute $eG \in E(\mathbb{F}_p)$ and then, $s_1 = x(eG) \bmod q$ and $s_2 \equiv (d + ss_1)e^{-1} \pmod{q}$. Publish the signature (s_1, s_2).	
Verification	
	Compute $v_1 \equiv ds_2^{-1} \pmod{q}$ and $v_2 \equiv s_1 s_2^{-1} \pmod{q}$. Compute $v_1 G + v_2 V \in E(\mathbb{F}_p)$ and verify that $x(v_1 G + v_2 V) \bmod q = s_1$.

Table 6.7: The elliptic curve digital signature algorithm (ECDSA)

Multiplication table modulo 41: (feel free to detach for convenience)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
3	0	3	6	9	12	15	18	21	24	27	30	33	36	39	1	4	7	10	13	16	19
4	0	4	8	12	16	20	24	28	32	36	40	3	7	11	15	19	23	27	31	35	39
5	0	5	10	15	20	25	30	35	40	4	9	14	19	24	29	34	39	3	8	13	18
6	0	6	12	18	24	30	36	1	7	13	19	25	31	37	2	8	14	20	26	32	38
7	0	7	14	21	28	35	1	8	15	22	29	36	2	9	16	23	30	37	3	10	17
8	0	8	16	24	32	40	7	15	23	31	39	6	14	22	30	38	5	13	21	29	37
9	0	9	18	27	36	4	13	22	31	40	8	17	26	35	3	12	21	30	39	7	16
10	0	10	20	30	40	9	19	29	39	8	18	28	38	7	17	27	37	6	16	26	36
11	0	11	22	33	3	14	25	36	6	17	28	39	9	20	31	1	12	23	34	4	15
12	0	12	24	36	7	19	31	2	14	26	38	9	21	33	4	16	28	40	11	23	35
13	0	13	26	39	11	24	37	9	22	35	7	20	33	5	18	31	3	16	29	1	14
14	0	14	28	1	15	29	2	16	30	3	17	31	4	18	32	5	19	33	6	20	34
15	0	15	30	4	19	34	8	23	38	12	27	1	16	31	5	20	35	9	24	39	13
16	0	16	32	7	23	39	14	30	5	21	37	12	28	3	19	35	10	26	1	17	33
17	0	17	34	10	27	3	20	37	13	30	6	23	40	16	33	9	26	2	19	36	12
18	0	18	36	13	31	8	26	3	21	39	16	34	11	29	6	24	1	19	37	14	32
19	0	19	38	16	35	13	32	10	29	7	26	4	23	1	20	39	17	36	14	33	11
20	0	20	40	19	39	18	38	17	37	16	36	15	35	14	34	13	33	12	32	11	31
21	0	21	1	22	2	23	3	24	4	25	5	26	6	27	7	28	8	29	9	30	10
22	0	22	3	25	6	28	9	31	12	34	15	37	18	40	21	2	24	5	27	8	30
23	0	23	5	28	10	33	15	38	20	2	25	7	30	12	35	17	40	22	4	27	9
24	0	24	7	31	14	38	21	4	28	11	35	18	1	25	8	32	15	39	22	5	29
25	0	25	9	34	18	2	27	11	36	20	4	29	13	38	22	6	31	15	40	24	8
26	0	26	11	37	22	7	33	18	3	29	14	40	25	10	36	21	6	32	17	2	28
27	0	27	13	40	26	12	39	25	11	38	24	10	37	23	9	36	22	8	35	21	7
28	0	28	15	2	30	17	4	32	19	6	34	21	8	36	23	10	38	25	12	40	27
29	0	29	17	5	34	22	10	39	27	15	3	32	20	8	37	25	13	1	30	18	6
30	0	30	19	8	38	27	16	5	35	24	13	2	32	21	10	40	29	18	7	37	26
31	0	31	21	11	1	32	22	12	2	33	23	13	3	34	24	14	4	35	25	15	5
32	0	32	23	14	5	37	28	19	10	1	33	24	15	6	38	29	20	11	2	34	25
33	0	33	25	17	9	1	34	26	18	10	2	35	27	19	11	3	36	28	20	12	4
34	0	34	27	20	13	6	40	33	26	19	12	5	39	32	25	18	11	4	38	31	24
35	0	35	29	23	17	11	5	40	34	28	22	16	10	4	39	33	27	21	15	9	3
36	0	36	31	26	21	16	11	6	1	37	32	27	22	17	12	7	2	38	33	28	23
37	0	37	33	29	25	21	17	13	9	5	1	38	34	30	26	22	18	14	10	6	2
38	0	38	35	32	29	26	23	20	17	14	11	8	5	2	40	37	34	31	28	25	22
39	0	39	37	35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1
40	0	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21

Multiplication table modulo 41, continued:

	21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
2	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39
3	22	25	28	31	34	37	40	2	5	8	11	14	17	20	23	26	29	32	35	38
4	2	6	10	14	18	22	26	30	34	38	1	5	9	13	17	21	25	29	33	37
5	23	28	33	38	2	7	12	17	22	27	32	37	1	6	11	16	21	26	31	36
6	3	9	15	21	27	33	39	4	10	16	22	28	34	40	5	11	17	23	29	35
7	24	31	38	4	11	18	25	32	39	5	12	19	26	33	40	6	13	20	27	34
8	4	12	20	28	36	3	11	19	27	35	2	10	18	26	34	1	9	17	25	33
9	25	34	2	11	20	29	38	6	15	24	33	1	10	19	28	37	5	14	23	32
10	5	15	25	35	4	14	24	34	3	13	23	33	2	12	22	32	1	11	21	31
11	26	37	7	18	29	40	10	21	32	2	13	24	35	5	16	27	38	8	19	30
12	6	18	30	1	13	25	37	8	20	32	3	15	27	39	10	22	34	5	17	29
13	27	40	12	25	38	10	23	36	8	21	34	6	19	32	4	17	30	2	15	28
14	7	21	35	8	22	36	9	23	37	10	24	38	11	25	39	12	26	40	13	27
15	28	2	17	32	6	21	36	10	25	40	14	29	3	18	33	7	22	37	11	26
16	8	24	40	15	31	6	22	38	13	29	4	20	36	11	27	2	18	34	9	25
17	29	5	22	39	15	32	8	25	1	18	35	11	28	4	21	38	14	31	7	24
18	9	27	4	22	40	17	35	12	30	7	25	2	20	38	15	33	10	28	5	23
19	30	8	27	5	24	2	21	40	18	37	15	34	12	31	9	28	6	25	3	22
20	10	30	9	29	8	28	7	27	6	26	5	25	4	24	3	23	2	22	1	21
21	31	11	32	12	33	13	34	14	35	15	36	16	37	17	38	18	39	19	40	20
22	11	33	14	36	17	39	20	1	23	4	26	7	29	10	32	13	35	16	38	19
23	32	14	37	19	1	24	6	29	11	34	16	39	21	3	26	8	31	13	36	18
24	12	36	19	2	26	9	33	16	40	23	6	30	13	37	20	3	27	10	34	17
25	33	17	1	26	10	35	19	3	28	12	37	21	5	30	14	39	23	7	32	16
26	13	39	24	9	35	20	5	31	16	1	27	12	38	23	8	34	19	4	30	15
27	34	20	6	33	19	5	32	18	4	31	17	3	30	16	2	29	15	1	28	14
28	14	1	29	16	3	31	18	5	33	20	7	35	22	9	37	24	11	39	26	13
29	35	23	11	40	28	16	4	33	21	9	38	26	14	2	31	19	7	36	24	12
30	15	4	34	23	12	1	31	20	9	39	28	17	6	36	25	14	3	33	22	11
31	36	26	16	6	37	27	17	7	38	28	18	8	39	29	19	9	40	30	20	10
32	16	7	39	30	21	12	3	35	26	17	8	40	31	22	13	4	36	27	18	9
33	37	29	21	13	5	38	30	22	14	6	39	31	23	15	7	40	32	24	16	8
34	17	10	3	37	30	23	16	9	2	36	29	22	15	8	1	35	28	21	14	7
35	38	32	26	20	14	8	2	37	31	25	19	13	7	1	36	30	24	18	12	6
36	18	13	8	3	39	34	29	24	19	14	9	4	40	35	30	25	20	15	10	5
37	39	35	31	27	23	19	15	11	7	3	40	36	32	28	24	20	16	12	8	4
38	19	16	13	10	7	4	1	39	36	33	30	27	24	21	18	15	12	9	6	3
39	40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
40	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1