The exam is *double-sided*. Make sure to read both sides of each page.
- The time limit is 50 minutes.
- No calculators are permitted.
- You are permitted one page of notes, front and back.
- The textbook's summary tables for the systems we have studied are provided on the last sheet. You may detach this sheet for easier reference.
- For any problem asking you to write a program, you may write in a language of your choice or in pseudocode, as long as your answer is sufficiently specific to tell the runtime of the program.
This page intentionally left blank.
(1) Alice and Bob are using Elgamal encryption, with public parameters $p = 29, g = 19$. The summary table for Elgamal is on the back page of this packet. There is also a multiplication table for $\mathbb{Z}/29$, so that you do not need to do the arithmetic by hand.

(a) Alice chooses the private key $a = 9$. What is her public key? Express your answer as an integer in $\{0, 1, 2, \cdots, 28\}$.

\[
A \equiv g^a \mod p \\
\equiv 19^9 \mod 29
\]

Fast-powering:

$19^9 \equiv 19 \cdot 19^8 \equiv 19 \cdot 25 \equiv 11 \mod 29$

$19^8 = (19^4)^2 \equiv 24^2 \equiv 25 \mod 29$

$19^4 = (19^2)^2 \equiv 13^2 \equiv 24 \mod 29$

$19^2 = 19 \cdot 19 \equiv 13 \mod 29$

$A = 11$

Part (b) on reverse side. (3 points)
(b) Bob sends ciphertext (8, 25) to Alice. What is the plaintext? Express your answer as an integer in \{0, 1, 2, \ldots, 28\}.

\[ C_2 \equiv A^k \cdot m \mod p \]
\[ = C_1^a \cdot m \mod p \]

\[ \Rightarrow \quad 25 \equiv 8^a \cdot m \mod 29 \]

Fast-powerning to find \(8^a\): (all congruences \(\mod 29\))

\[ 8^1 \equiv 8 \equiv 20 \equiv 15 \]
\[ 8^2 \equiv (8^1)^2 = 7 \equiv 20 \]
\[ 8^3 \equiv (8^2)^2 = 6 \equiv 7 \]
\[ 8^4 \equiv 8 \cdot 8 \equiv 6 \]

\[ \text{So} \quad 25 \equiv 15 \cdot m \mod 29. \]

We see from the table that \(2 \cdot 15 \equiv 1 \mod 29\), i.e. \(2 \equiv 15^{-1} \mod 29\).

\[ \text{So} \quad m \equiv 2 \cdot 25 \equiv 21 \mod 29. \]

\[ m = 21 \]

(3 points)
(2) (a) Suppose that $a \mid m$. Prove that the congruence $ax \equiv ab \pmod{m}$ holds if and only if the congruence $x \equiv b \pmod{\frac{m}{a}}$ holds (all variables are integers).

\[
ax \equiv ab \pmod{m}
\]

\[
(\Longleftrightarrow) \exists k \in \mathbb{Z} \text{ s.t. } ax = ab + km
\]

\[
(\Longleftrightarrow) \exists k \in \mathbb{Z} \text{ s.t. } x = b + k \cdot \left(\frac{m}{a}\right) \quad \text{(divide by } a\text{)}
\]

\[
(\Rightarrow) x \equiv b \pmod{\left(\frac{m}{a}\right)}.
\]

Part (b) on reverse side. (3 points)
(b) Suppose that $\gcd(a, m) = 1$. Prove that the congruence $ax \equiv ab \pmod{m}$ holds if and only if the congruence $x \equiv b \pmod{m}$ holds.

$$\gcd(a, m) = 1 \iff \exists u, v \in \mathbb{Z} \text{ st. } au + bv = 1.$$  
  (ext. euclidean alg.)

For this $u \in \mathbb{Z}$, $au \equiv 1 \pmod{m}$.

So

$$ax \equiv ab \pmod{m}$$

$$\Rightarrow (ua)x \equiv (ua)b \pmod{m} \quad \text{(mul. both sides by } u)$$

$$\Rightarrow 1 \cdot x \equiv 1 \cdot b \pmod{m}.$$  

Conversely,

$$x \equiv b \pmod{m}$$

$$\Rightarrow ax \equiv ab \pmod{m} \quad \text{(mul. both sides by } a)$$

Therefore

$$ax \equiv ab \pmod{m} \iff x \equiv b \pmod{m}.$$  

(3 points)
(3) Write a program that reduces breaking Diffie-Hellman key exchange to breaking Elgamal encryption.

More precisely: suppose that Eve has written a function \texttt{break.elg}(p, g, A, c1, c2) with the following behavior: if the arguments are as in Table 2.3 (back of the packet), then this function will return \( m \). Make use of this function to write a function \texttt{break.dh}(p, g, A, B), which accepts arguments as labeled in Table 2.2 and returns the corresponding shared secret.

You may use any functions that are built into Python (or any language you have written your homework in), plus the hypothetical function \texttt{break.elg}. You may also assume that you have already written a function \texttt{ext.euclid}(a, b), with the following behavior: given two positive integers \( a, b \), this function returns a list of three integers \([u, v, d]\), where \( d = \gcd(a, b) \) and \( au + bv = d \).

For full points, your program must require at most \( \mathcal{O}(\log p) \) arithmetic operations (not counting any operations needed to compute \texttt{break.elg}).

Suppose that \( A \equiv g^a \mod p \) and \( B \equiv g^b \mod p \). Observe that for any \( C_2 \), if Eve computes
\[
    m = \texttt{break.elg}(p, g, A, B, C_2),
\]
then \( m \) will satisfy
\[
    m \equiv C_2 \cdot B^{-a} \mod p
    = C_2 \cdot g^{-ab} \mod p.
\]

In particular, \( \texttt{break.elg}(p, g, A, B, 1) = (g^{ab})^{-1} \mod p \), the inverse of the shared secret. So Eve can break DH as follows:

```python
def break_dh(p, q, A, B):
    m = break_elg(p, q, A, B, 1)
    [u, v, d] = ext_euclid(m, p)
    return u * q ** p  # this is m^r \mod p.
```

More space for work on reverse side. (6 points)
Additional space for problem 3.
(4) Let $p$ be a prime number, and $[g]_p$ an element of $(\mathbb{Z}/p)\ast$.

(a) Prove that if $\text{ord}[g]_p = 17$, then $p \equiv 1 \pmod{17}$.

From a result in class (Prop. 1.29 in text),

$$[g]_p^n = [1]_p \Rightarrow \text{ord}[g]_p \mid n.$$  

By Fermat's little theorem,

$$[g]_p^{p-1} = [1]_p.$$  

Therefore

$$\text{ord}[g]_p \mid (p-1)$$

$$\Rightarrow \quad 17 \mid (p-1)$$

$$\Rightarrow \quad p \equiv 1 \pmod{17}.$$  

Part (b) on reverse side.  

(3 points)
(b) Prove conversely that if \( p \equiv 1 \pmod{17} \), then there exists some element \([g]_p\) of order 17.

By the primitive root theorem, \( \exists \) a primitive root \([h]_p\) (an element of order \( p-1 \)).

Let \( [g]_p = [h]_p^{(p-1)/17} \). (makes sense since \( 17 \mid (p-1) \), since \( p \equiv 1 \pmod{17} \)).

Now, \( [g]_p^{17} = [h]_p^{0-1} = [1]_p \), so \( \text{ord} [g]_p \mid 17 \).

17 is prime, so \( \text{ord} [g]_p \in \{1,17\} \).

But \( \text{ord} [g]_p \neq 1 \), since \( [g]_p \neq [1]_p \) (since \( [h]_p \) is primitive, \( [h]_p^{(p-1)/17} = [1]_p \), since \((p-1) \div 17\) & \( \text{ord} [h]_p = (p-1) \)).

So \( \text{ord} [g]_p = 17 \). Thus indeed exists an element of order 17.

(3 points)
(5) Suppose that Alice and Bob use the following variant of Elgamal. The parameters 
p, g are as in table 2.3, and Alice chooses a secret key \( a \) and public key \( A \) in the same 
manner as in table 2.3. However, instead of following table 2.3, Bob computes his 
ciphertext as followings: he chooses a random element \( k \), and computes 
\[
\begin{align*}
  c_1 &\equiv A^k \pmod{p} \\
  c_2 &\equiv m \cdot g^k \pmod{p},
\end{align*}
\]
then sends \((c_1, c_2)\) to Alice.

Explain how Alice can efficiently decipher messages, i.e. determine \( m \) from \((c_1, c_2)\). 
You will need to place a restriction on Alice’s original choice of private key \( a \) in order 
for decryption to be possible; clearly state this restriction.

We know that
\[
e \equiv f \Rightarrow g^e \equiv g^f \pmod{p}.
\]

So if \( a \) has an inv\( a^{-1} \) \( \pmod{p} \), we have
\[
(c_1)^u \equiv A^{uk} \equiv g^{auk} \equiv g^{1-k} \pmod{p}
\]
and hence
\[
m \equiv c_2 \cdot c_1^{-u} \pmod{p},
\]
which Alice can compute easily:

1) Find \( u \equiv a^{-1} \pmod{p} \) w/ ext. eucl. alg.

2) Find \( c_1^u \) w/ fast-powering \( \pmod{p} \)

3) Find \( (c_1^u)^{-1} \pmod{p} \) w/ ext. eucl. alg.

Alice must choose here private key such that \( \gcd(a, p-1) = 1 \)
for this to work.

Remark \( \gcd(a, \text{ord}[g]_p) = 1 \) is enough; this work as long as \( u \equiv a^{-1} \pmod{\text{ord}[g]_p} \). But if a \& \( \text{ord}[g]_p \) have a common 
factor, then it’s possible to show that \((c_1, c_i)\) do not uniquely 
determine \( m \).

\( \text{More space for work on reverse side.} \)  

(6 points)
Additional space for problem 5.