Problem 1.
Let’s try out RSA! There is a complete description of the algorithm at the bottom of the page. You’ll probably need extra paper. \textbf{Check your work carefully!}

(a) As a team, go through the \textbf{beforehand} steps.

- Choose primes $p$ and $q$ to be relatively small, say in the range 10-40. In practice, $p$ and $q$ might contain several hundred digits, but small numbers are easier to handle with pencil and paper.
- Try $e = 3, 5, 7, \ldots$ until you find something that works. Use Euclid’s algorithm to compute the gcd.
- Find $d$ (using the Pulverizer —see appendix for a reminder on how the Pulverizer works —or Euler’s Theorem).

When you’re done, put your public key on the board. This lets another team send you a message.

(b) Now send an encrypted message to another team using their public key. Select your message $m$ from the codebook below:

- 2 = Greetings and salutations!
- 3 = Yo, wassup?
- 4 = You guys are slow!
- 5 = All your base are belong to us.
- 6 = Someone on our team thinks someone on your team is kinda cute.
- 7 = You are the weakest link. Goodbye.

(c) Decrypt the message sent to you and verify that you received what the other team sent!
Beforehand The receiver creates a public key and a secret key as follows.

1. Generate two distinct primes, \( p \) and \( q \).
2. Let \( n = pq \).
3. Select an integer \( e \) such that \( \gcd(e, (p - 1)(q - 1)) = 1 \).
   The public key is the pair \((e, n)\). This should be distributed widely.
4. Compute \( d \) such that \( de \equiv 1 \pmod{(p - 1)(q - 1)} \).
   The secret key is the pair \((d, n)\). This should be kept hidden!

Encoding The sender encrypts message \( m \), where \( 0 \leq m < n \), to produce \( m' \) using the public key:

\[ m' = \text{rem}(m^e, n). \]

Decoding The receiver decrypts message \( m' \) back to message \( m \) using the secret key:

\[ m = \text{rem}((m')^d, n). \]

Problem 2.

A critical fact about RSA is, of course, that decrypting an encrypted message always gives back the original message! That is, that \( \text{rem}((m^e)^d, pq) = m \). This will follow from something slightly more general:

**Lemma 2.1.** Let \( n \) be a product of distinct primes and \( a \equiv 1 \pmod{\phi(n)} \) for some nonnegative integer, \( a \). Then

\[ m^a \equiv m \pmod{n}. \]

(a) Explain why Lemma 2.1 implies that \( k \) and \( k^5 \) have the same last digit. For example:

\[ 2^5 = 32 \quad 79^5 = 3077056392 \]

*Hint: What is \( \phi(10) \)?*

(b) Explain why Lemma 2.1 implies that the original message, \( m \), equals \( \text{rem}((m^e)^d, pq) \).

(c) Prove that if \( p \) is prime, then

\[ m^a \equiv m \pmod{p} \]

for all nonnegative integers \( a \equiv 1 \pmod{p - 1} \).

(d) Prove that if \( n \) is a product of distinct primes, and \( a \equiv b \pmod{p} \) for all prime factors, \( p \), of \( n \), then \( a \equiv b \pmod{n} \).

(e) Combine the previous parts to complete the proof of Lemma 2.1.
Appendix

Inverses, Fermat, Euler

Lemma (Inverses mod n). If k and n are relatively prime, then there is integer k' called the modulo n inverse of k, such that
\[ k \cdot k' \equiv 1 \pmod{n}. \]

Remark: If gcd(k, n) = 1, then sk + tn = 1 for some s, t, so we can choose k' := s in the previous Lemma. So given k and n, an inverse k' can be found efficiently using the Pulverizer.

Theorem (Fermat’s (Little) Theorem). If p is prime and k is not a multiple of p, then
\[ k^{p-1} \equiv 1 \pmod{p} \]

Definition. The value of Euler’s totient function, \( \phi(n) \), is defined to be the number of positive integers less than n that are relatively prime to n.

Lemma (Euler Totient Function Equations).
\[
\begin{align*}
\phi(p^k) &= p^k - p^{k-1} & \text{for prime, } p, \text{ and } k > 0, \\
\phi(mn) &= \phi(m) \cdot \phi(n) & \text{when } \gcd(m, n) = 1.
\end{align*}
\]

Theorem (Euler’s Theorem). If k and n are relatively prime, then
\[ k^{\phi(n)} \equiv 1 \pmod{n} \]

Corollary. If k and n are relatively prime, then \( k^{\phi(n)-1} \) is an inverse modulo n of k.

Remark: Using fast exponentiation to compute \( k^{\phi(n)-1} \) is another efficient way to compute an inverse modulo n of k.

The Pulverizer

Euclid’s algorithm for finding the GCD of two numbers relies on repeated application of the equation:
\[ \gcd(a, b) = \gcd(b, \text{rem}(a, b)) \]

For example, we can compute the GCD of 259 and 70 as follows:
\[
\begin{align*}
\gcd(259, 70) &= \gcd(70, 49) & \text{since } \text{rem}(259, 70) = 49 \\
&= \gcd(49, 21) & \text{since } \text{rem}(70, 49) = 21 \\
&= \gcd(21, 7) & \text{since } \text{rem}(49, 21) = 7 \\
&= \gcd(7, 0) & \text{since } \text{rem}(21, 7) = 0 \\
&= 7.
\end{align*}
\]

The Pulverizer goes through the same steps, but requires some extra bookkeeping along the way: as we compute \( \gcd(a, b) \), we keep track of how to write each of the remainders (49, 21, and 7, in the example) as a linear combination of a and b (this is worthwhile, because our objective is to write
the last nonzero remainder, which is the GCD, as such a linear combination). For our example, here is this extra bookkeeping:

<table>
<thead>
<tr>
<th>259</th>
<th>70</th>
<th>49</th>
<th>$\text{rem}(x, y) = x - q \cdot y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>49</td>
<td>21</td>
<td>$= 259 - 3 \cdot 70$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$= 70 - 1 \cdot (259 - 3 \cdot 70)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$= -1 \cdot 259 + 4 \cdot 70$</td>
</tr>
<tr>
<td>49</td>
<td>21</td>
<td>7</td>
<td>$= 49 - 2 \cdot 21$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$= (259 - 3 \cdot 70) - 2 \cdot (-1 \cdot 259 + 4 \cdot 70)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$= 3 \cdot 259 - 11 \cdot 70$</td>
</tr>
</tbody>
</table>

We began by initializing two variables, $x = a$ and $y = b$. In the first two columns above, we carried out Euclid’s algorithm. At each step, we computed $\text{rem}(x, y)$, which can be written in the form $x - q \cdot y$. (Remember that the Division Algorithm says $x = q \cdot y + r$, where $r$ is the remainder. We get $r = x - q \cdot y$ by rearranging terms.) Then we replaced $x$ and $y$ in this equation with equivalent linear combinations of $a$ and $b$, which we already had computed. After simplifying, we were left with a linear combination of $a$ and $b$ that was equal to the remainder as desired. The final solution is boxed.