Problem 1: Prove that for any prime \( p \) and integer \( k \geq 1 \) it holds that \( \phi(p^k) = (p - 1)p^{k-1} \).

Solution: The integers among \( \mathbb{Z}_{p^k} = \{0, 1, 2, \ldots, p^k - 1\} \), that are not relatively prime with \( p^k \) are precisely those integers that have \( p \) in their prime decompositions, ie that are the multiples of \( p \), which are enumerated \( 0 \cdot p, 1 \cdot p, 2 \cdot p, \ldots, (p^{k-1} - 1) \cdot p \); thus there are \( p^{k-1} \) such nonunits. In particular, \( \phi(p^k) = p^k - p^{k-1} = (p - 1)p^{k-1} \).

Problem 2: Write a MATLAB function that performs fast exponentiation. The input consists of positive integer \( n \), \( a \in \mathbb{Z}_n \), and a positive integer power \( k \). The output is \( a^k \mod n \). Be sure to provide nontrivial examples to illustrate that your code is working properly.

Problem 3: Suppose \( p \) and \( q \) are different prime numbers, and let \( n := p \cdot q \). Give a very simple formula for finding \( p \) and \( q \) using only \( n \) and \( \phi(n) \). Explain why this illustrates that there may not be an efficient algorithm for computing \( \phi(n) \) (if you are only given the number \( n \) and nothing more).

Solution: We have here that \( \phi(n) = (p - 1)(q - 1) = pq - p - q + 1 = n - p - q + 1 = n - p - \frac{n}{p} + 1 \), which we multiply by \( p \) to obtain \( \phi(n)p = np - p^2 - n + p \), which simplifies to the quadratic equation \( p^2 + (\phi(n) - n - 1)p + n = 0 \). By the quadratic formula, \( p \) is either \( \frac{-(\phi(n) - n - 1) + \sqrt{(\phi(n) - n - 1)^2 - 4n}}{2} \) or \( \frac{-(\phi(n) - n - 1) - \sqrt{(\phi(n) - n - 1)^2 - 4n}}{2} \), and \( q \) is the other of these two. Thus, If Eve has an efficient algorithm for computing \( \phi(n) \) from \( n \) then she can efficiently factor \( n \), and thus there probably isn’t an efficient algorithm to compute \( \phi(n) \) from \( n \), because there probably isn’t an efficient algorithm for factoring \( n \).

Problem 4: Suppose Bob has public RSA key \((n, e) = (8439833, 5711029)\) and Alice sends him the ciphertext \( c = 62472 \) encrypted with Bob’s key. Find the corresponding plaintext \( m \).

Solution: Because \( n = 8439833 \) isn’t too big, MATLAB factors it as \( n = pq \) for \( p = 2803 \) and \( q = 3011 \), where \( p \) and \( q \) are prime. Now, since \( e = 5711029 \) is public, Eve can use the
Extended Euclid Algorithm to find \( d = e^{-1} \mod (p-1)(q-1) \); indeed, she would quickly obtain \( d = 9769 \). With this, Even can obtain the plaintext \( m \) by computing \( m = c^d \mod n \), which is \( m = 2345678 \).

**Problem 5:** Write a MATLAB program whose input consists of two primes \( p, q \equiv 3 \mod 4 \) and \( c \in \mathbb{Z}_{pq} \). The output should be the four square roots of \( c \mod pq \), if they exist (in other words, your program does Rabin decryption). Use your code to find the four square roots of \( 6245706 \mod 9353881 \). (Hint: The number 9353881 isn’t too big for MATLAB to factor; use the command “factor.” Also, the command “format rat” will ensure that all data is displayed and stored as integers.)

**Solution:** MATLAB can factor \( n = 9353881 \) into \( n = pq \) where \( p = 2999 \) and \( q = 3119 \). Then, using the square root script that you write, we obtain the four square roots \( 1443540, 8119314, 1234567, \) and \( 7910341 \).

**Problem 6:** Let \( p \) and \( q \) be primes such that \( q = 2p + 1 \), and consider any \( a \in \mathbb{Z}_q \) such that \( a \not\equiv 0, 1, -1 \mod q \). Prove that \( a \) is primitive mod \( q \) if and only if \( a^p = -1 \mod q \).

**Solution:** Note that \( \phi(q) = q-1 = 2p \) only has positive divisors 1, 2, \( p \), or 2\( p \) by Lagrange’s Theorem. What it would mean that \( a \) has order 1 is that \( a^1 = 1 \mod q \), ie that \( a = 1 \), and what it would mean that \( a \) has order 2 is that \( a^2 = 1 \mod q \), but since there can be only two square roots of any number mod any prime, the square roots of 1 mod \( q \) are \( 1 \mod q \) and \( -1 \mod q \), hence \( a = -1 \mod q \). Thus, excluding \( a \) being 0, 1 or \( -1 \mod q \), either the order of \( a \) is \( p \), in which case \( a^p = 1 \mod q \), or else the order of \( a \) is \( 2p \) (ie \( a \) is a primitive root), in which case \( a^p \mod q \) is the non-one square root of \( a^{2p} = 1 \mod q \), ie \( a^p = -1 \mod q \). Thus, indeed, if \( a \) is not 0, 1, \( -1 \mod q \) then \( a^p \mod q \) is \( -1 \) or 1 according as \( a \) is a primitive root or not. [Important note: This gives us an efficient way to test whether or not a given \( a \in \mathbb{Z}_q^* \) is primitive, since fast exponentiation is efficient. A brute force test for primitivity which checks all of the powers of \( a \) would not be efficient.]