1. Prof. Namlleh observed that in the Diffie-Hellman key exchange protocol, user $U$ publishes $Y_U = g^{X_U} \mod p$, where $p$ is a known large prime number, and $g \in \mathbb{Z}_p^*$ is a generator of $\mathbb{Z}_p^*$. Note that in this question we assume that $g$ is a generator.

(a) Given $p, g, Y_U = g^{X_U} \mod p$, show how to reveal the least significant bit of $x_U$.

His assistant, Dr. Eiffid, proposed the following algorithm to compute DLOGs, given the least significant bit of $X_U$:

- Compute the least significant bit of $X_U$.
- If the LSB is 1, let $Y'_{U} = Y_U \cdot g^{-1}$, otherwise let $Y'_{U} = Y_U$.
- Compute the square root of $Y'_{U}$, and repeat the algorithm on the square root until $Y'_{U} = 1$. (As we will see in class, there exists an efficient algorithm that computes the square roots of a QR modulo a prime number.)
- By collecting the LSBs one can reconstruct $X_U$.

(b) Show that each $Y'_{U}$ is a quadratic residue modulo $p$.

(c) Let $X'$ be the number composed of the LSBs computed by the algorithm. Show that $X'$ satisfies $g^{X'} \equiv Y_U \pmod{p}$. Conclude that the secret key of $U$ is $X_U \equiv X' \pmod{p-1}$.

(d) Explain why the above algorithm cannot compute the DLOG $X_U$ in spite of the above.

2. (a) Let $p > 3$ be a prime. Prove that $-3$ is a quadratic residue modulo $p$ if and only if $p \equiv 1 \pmod{3}$.

(b) A Mersenne prime is a prime number of the form $2^n - 1$ (it is easy to show that $n$ must be prime, but we will not prove it here). Prove that 3 is a QNR modulo any Mersenne prime greater than 3. This property is used in the Lucas-Lehmer test, which is a deterministic, efficient algorithm for testing the primality of numbers of the form $2^n - 1$.

3. For a cipher with a key $K$ a message $M$ is called a fixpoint if $E_K(M) = M$.

Prove that in an RSA system with $n = pq$ and a public key $(n, e)$ the number of fixpoints is $(\gcd(p - 1, e - 1) + 1) \cdot (\gcd(q - 1, e - 1) + 1)$. 
4. In the Merkle-Damgard scheme a message of arbitrary length is hashed by dividing it to blocks of fixed length: $M_1, M_2, ..., M_l$. The compression function $h_i = C(h_{i-1}, M_i)$ compresses the previous chaining value $h_{i-1}$ and the current message block $M_i$, and generates a new chaining value $h_i$. The value $h_0 = IV$ is **fixed and known to everyone**, and the resulting hash value is the last chaining value $h_l$, as seen in the figure:

In this question assume that the length of all the messages is a multiple of 512 bits, and that the hash does not add padding nor append the message length to the last block of the message. Assume also that the $IV$ value, the chaining values $h_i$, and the hash value are all 160-bit long, and the message blocks $M_i$ are 512-bit long.

(a) Given a specific IV value, describe how we can use the Birthday Paradox in order to find a one-block message $M = m_1$, and a two-block message $M' = m'_1m'_2$, with the same hash value. What is the complexity of the attack you described?
   - Given a specific IV value, describe how we can find a one-block message, and a three-block message with the same hash value?

(b) How can we use both attacks described in (a) in order to find 4 messages of 4 different lengths, that have the same hash value?

(c) Extend the attack from (b) in order to get collision of $2^n$ different messages of $2^n$ different lengths. What is the complexity of the attack?

(d) Now we want to get $2^n$ different messages, all having the same length, and the same hash value. Describe how we can do that. Does your attack work if we append the message length to the last block of each message?