1. **Polynomial-Time Functions**

   (a) No
   (b) Yes
   (c) Yes
   (d) No
   (e) Yes

2. **Perfect Secrecy with Decimal Key**

   Yes, this is a perfect secrecy scheme. For any cipher text c, the possibilities of the original message being m_0 or m_1 are the same.

   Suppose both the cipher text c and the message m are of length l. According to the algorithm, the nth bit of c (denoted as $c[n-1]$) only depends on the nth bits of m and k (denoted as $m[n-1]$ and $k[n-1]$, respectively). For $c[n-1] \in 0$ to 9, there always exists a value for $k[n-1]$ such that

   $$(m[n-1] + k[n-1]) mod 10 = c[n-1].$$

   The probability of $m[n-1]$ being any value from 0 to 9 is equal. The property holds true for every bit throughout the length l. For any combination of c and m, the value of k is a uniform discrete distribution over the key space with each key having a probability of 10^{-l}. Thus,

   $$Pr(C = c \mid m_0) = Pr(C = c \mid m_1) = 10^{-l},$$

   which is by definition perfect secrecy.

3. **Perfect Secrecy with Decimal Message**

   No, this is NOT a perfect secrecy scheme. For any cipher text c, the possibilities of the original message being m_0 or m_1 can be different.

   Using the same notations as in Question 2, we denote the nth bit of the cipher text as $c[n-1]$. According to the algorithm, $k[n-1]$ is either 0 or 1. Therefore, for a given value of $c[n-1]$, there exists only two possible values for $m[n-1]$. For example, if $c[n-1] = 0$, then $m[n-1]$ has to be 0 or 9. The fact that the l-bit string k is binary limits m to a subset of the whole message space. The property discussed in Question 2 no longer exists in this question.

   In other word, there exists such c and m that no k can encrypt m to c. $Pr(C = c \mid m_0)$ and $Pr(C = c \mid m_1)$ are not equal for some m_0 and m_1, so the algorithm is not perfect secrecy.

   Reflecting on the two problems together, regardless of encryption method, a key of length l in a finite symbol alphabet A can be used to encrypt a message m of l in a finite symbol alphabet B with perfect secrecy if and only if $\mid A \mid \geq \mid B \mid$.

   If it is with perfect secrecy then $\mid A \mid \geq \mid B \mid$, which can be proved by contradiction. Suppose $\mid A \mid < \mid B \mid$ and that at least one message keypair encrypts to c. By the set-up, the message space is at most $(\mid A \mid)^l$ while the key space $(\mid A \mid)^l$. Then there exists m^* such that no k can encrypt m^* to c, which results in different probability for $Pr(C = c \mid m_0)$ and $Pr(C = c \mid m_1)$.

   Trivially, if $\mid A \mid \geq \mid B \mid$ then there must exist at least one way of mapping from m to c according to the pigeonhole principle.

4. **Double Encryption**

(a) If $(Setup_1, Encrypt_1, Decrypt_1)$ is secure, then $(Setup^*, Encrypt^*, Decrypt^*)$ is secure.

Suppose there exists an algorithm B that can break $(Setup^*, Encrypt^*, Decrypt^*)$ in polynomial time. B can first send the challenger (m_0, m_1) for $Encrypt_1(K_1, m_b)$, and then send the attacker $Encrypt_2(K_2, Encrypt_1(K_1, m_b))$. The two steps of encryption is exactly $Encrypt^*$. By assumption, the attacker can break $Encrypt^*$ and send back b' to B, which is then sent to the challenger. From the challenger's perspective, sending $Encrypt_1(K_1, m_b)$ results in guessing the right b'. This indicates that $(Setup_1, Encrypt_1, Decrypt_1)$ is not secure, which contradicts our assumption.

(b) If $(Setup_2, Encrypt_2, Decrypt_2)$ is secure, then $(Setup^*, Encrypt^*, Decrypt^*)$ is secure.

Suppose there exists an algorithm B that can break $(Setup^*, Encrypt^*, Decrypt^*)$ in polynomial time. B can first send the challenger $(Encrypt_1(K_1, m_0), Encrypt_1(K_1, m_0))$ in exchange for $Encrypt_2(K_2, Encrypt_1(K_1, m_b))$, and then send the attacker $Encrypt_2(K_2, Encrypt_1(K_1, m_b))$. The two steps is equivalent to $Encrypt^*$. By assumption, the attacker can break $Encrypt^*$ and send back b' to B, which is then sent to the challenger. From the challenger's perspective, sending $Encrypt_2$ of two messages results in guessing the right b'. This indicates that $(Setup_2, Encrypt_2, Decrypt_2)$ is not secure, which contradicts our assumption.

(c) When we encrypt using XOR of message and key, both $(Setup_1, Encrypt_1, Decrypt_1)$ and $(Setup_2, Encrypt_2,$ are individually secure but this combination is not.

Formally, we define

$$Encrypt_1(K, m) = m \oplus K$$

and

$$Encrypt_2(K, m) = m \oplus K.$$

Each encryption is secure just like one-time pad. However, the result of double encryption,

$$Encrypt_2(K, Encrypt_1(K, m)) = (m \oplus K) \oplus K = m,$$

is the message itself, which is by no means secure.