Question 1 (non-programming question): Involves answering theoretical questions about hash tables, such as the importance of immutable keys, the inevitability of collisions, the effectiveness of different hash functions, and problems when keys are deleted.

* Why should hash table keys in a program be of an immutable type such as integers or immutable strings?
  + Hash table keys need to be immutable because the hash table's functionality relies on the consistency of a key's hash value. If the value of a key can change, its hash value might change too, leading to the inability to locate the original key-value pair in the hash table. Immutable keys ensure that once a key-value pair is added to the hash table, it can always be retrieved using the same hash value.
  + Example: Imagine a library book labeled with a unique number (immutable key). If the number changes, finding the book using the old number becomes impossible.
* Assuming there are more possible keys than the capacity of the hash table, why is some amount of collisions inevitable?
  + When the total number of possible keys exceeds the hash table's capacity, according to the pigeonhole principle, at least one hash slot will be mapped to by multiple keys. In this situation, collisions are inevitable because the hash table's capacity cannot accommodate every possible key.
  + Example: Consider a parking lot with 100 spaces (hash table capacity) and 200 cars (keys). Since there are more cars than spaces, some cars will have to share a space, leading to a 'collision'.
* If you are given a hash table with a capacity of 100, why is the function h(k) = k mod 10 not a good hash function?
  + In this case, the hash function h(k) = k mod 10 only uses the remainder of the key k divided by 10 as the hash value. This means there are only 10 possible hash values (0 to 9), leading to a high number of collisions since the hash table has 100 slots. A better hash function should distribute hash values more evenly to reduce collisions.
  + Example: Imagine assigning students (keys) to 10 classrooms (hash values) based on their roll numbers mod 10, but you have 100 students. Each classroom will have 10 students, overfilling them, while 90 other classrooms remain unused.
* In open-addressing collision resolution, why can't a key be deleted from a table by replacing it with NULL?
  + In open addressing, if a key is deleted and its position is set to NULL, subsequent searches that encounter this NULL will stop, even if there are valid keys that have been moved to later positions due to collisions. Thus, simply setting a position to NULL disrupts the probing process of a chain, preventing correct retrieval of keys that were moved due to collisions.
  + Example: Think of a line of people waiting for a movie. If one person leaves (key deleted) and their spot is left empty (set to NULL), anyone joining the line later will think the line ends at the empty spot, not realizing there are more people ahead.