**Question 3**

**3.1 Designing hash Function.**

1. Size of Input and Hash Output: Make sure that the hash function, regardless of the size of the input, produces a fixed-size hash value. This limitation facilitates effective memory management.
2. Fast and deterministic: This is to maximise efficiency, the function must work quickly and reliably generate the same hash result for each input.
3. Spread of Hash Values: This is to reduce collisions, try to choose a hash function that equally distributes hash values throughout the possible range.

**Hash function Collision**

1. Chaining: Making use of a technique like chaining, in which collisions are prevented by storing several values in a data structure (linked lists, arrays, etc.) inside the hash table that hashes to the same index.
2. Open Addressing: by using an approach like open addressing, in which collisions are avoided by looking for an alternate, vacant space for the colliding key within the hash table (linear or quadratic probing).
3. Resizing: This is to ensure a better distribution of keys throughout the table, periodically enlarge the hash table or rehash the keys using a different hash function if the collision rate gets too high**.**

A simple hash function that efficiently handles collisions and takes memory limitations into account by taking these things into account can be created**.**

**3.2**