Question 1 :

Rehashing in hashmap ?

**Rehashing is a fundamental operation in hash table data structures, including hashmaps. It involves resizing the underlying array (or bucket array) of the hashmap to maintain a balance between the number of elements stored and the capacity of the hashmap.**

Here's how rehashing typically works in a hashmap implementation:

1. **Capacity Threshold**: Hashmaps typically have a capacity threshold or load factor. When the number of elements in the hashmap exceeds a certain threshold (often expressed as a ratio of the number of elements to the capacity), rehashing is triggered.
2. **New Capacity**: When rehashing is triggered, a new, larger array (or bucket array) is created. The new capacity is usually larger than the current capacity, aiming to reduce the likelihood of collisions and maintain efficiency.
3. **Rehashing Process**: Each element in the old array needs to be rehashed and inserted into the new array. This involves recalculating the hash code of each element based on the new capacity and then inserting it into the appropriate position in the new array.
4. **Transfer of Elements**: As elements are rehashed, they are transferred from the old array to the new array. Depending on the implementation, this might involve copying elements or simply updating references to them.
5. **Updating References**: Once all elements are transferred to the new array, the hashmap starts using the new array as its underlying data structure. Any references to the old array are updated to point to the new array, and the old array is typically garbage collected.

Rehashing is essential for maintaining the performance characteristics of a hashmap, such as constant-time average-case operations. It allows the hashmap to adapt dynamically to changes in the number of elements stored while keeping the time complexity of operations low.

However, it's worth noting that rehashing can be a costly operation in terms of time and memory, especially if it needs to be performed frequently due to rapid changes in the number of elements. Therefore, choosing an appropriate load factor and initial capacity is crucial for balancing memory usage and performance.