**1a.)** To check if the list has any duplication, use two for-loops by hashing.

1. Read the list from left to right.
2. If the current read element is present in the hashtable and no duplicates of that element are found, return true.
3. Else, return false.

Since the algorithm has two for-loop, with each loop using a linked list implementation to sort in linear time, the time complexity of the algorithm is O; we are individually checking if each element is duplicated or not *twice*.

A better algorithm to do this job would be one that uses one for-loop and having a temporary array that keeps count of duplicated elements. This is a better algorithm because it has a smalled time complexity whilst completing the same task as the first algorithm. The time complexity for this algorithm would be O(n).

**1b.)** To find all the duplicate numbers in the list, we can use an algorithm similar to that of part a.’s the algorithm with one for-loop and a temporary array to stores duplicated elements.

1. Read the list from left to right.

2. If the current read element is sorted into the temporary array, print(“Number [value] duplicated”).

3. Else, return true.

Because this algorithm uses one for-loop, the time complexity is O(n).