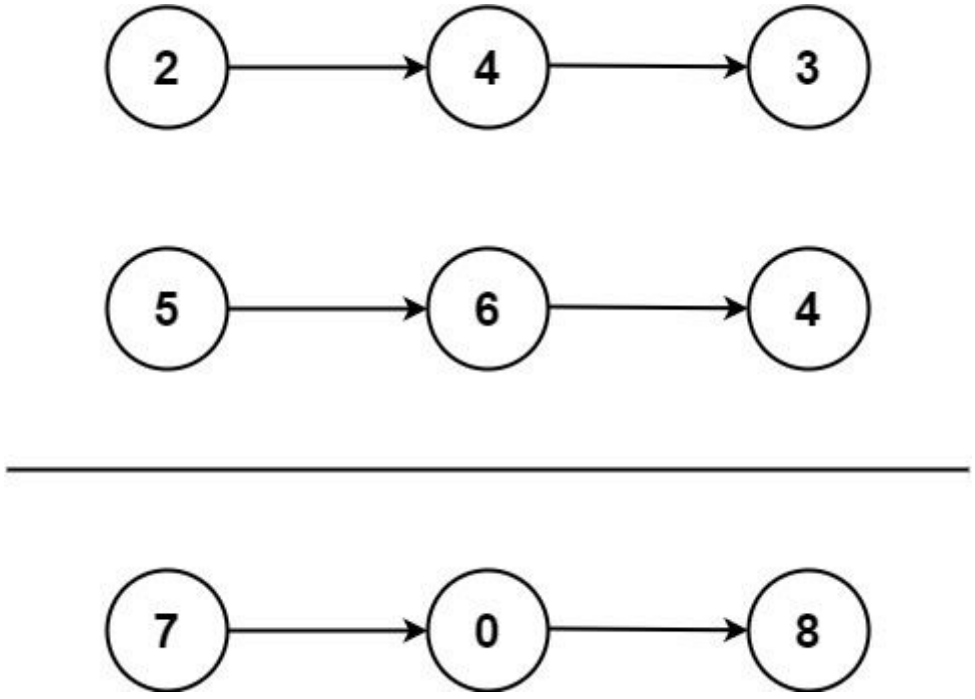
Exercise 2

Question 1:

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:



1. Input: l1 = [2,4,3], l2 = [5,6,4]
2. Output: [7,0,8]
3. Explanation: 342 + 465 = 807.

Example 2:

4. Input: l1 = [0], l2 = [0]
5. Output: [0]

Example 3:

6. Input: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]
7. Output: [8,9,9,9,0,0,0,1]

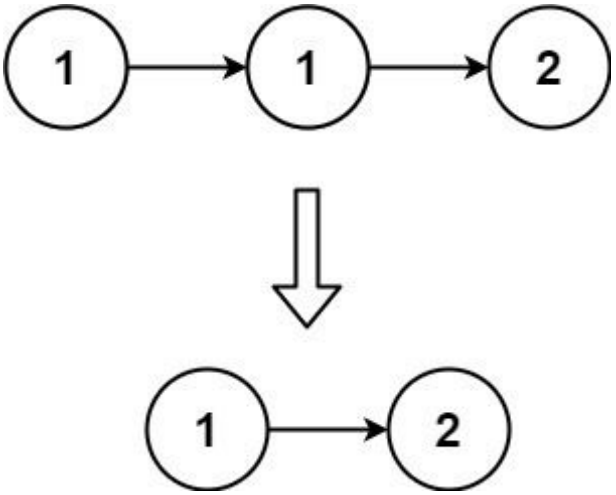
Constraints:

- The number of nodes in each linked list is in the range [1, 100].
- 0 <= Node.val <= 9
- It is guaranteed that the list represents a number that does not have leading zeros.

Question 2:

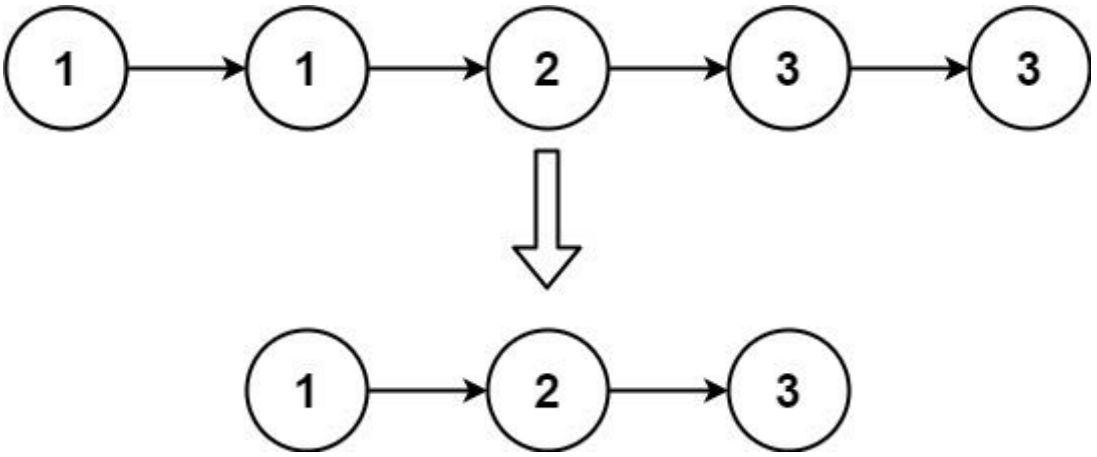
Given the head of a sorted linked list, *delete all duplicates such that each element appears only once*. Return *the linked list sorted as well*.

Example 1:



Input: head = [1,1,2]
Output: [1,2]

Example 2:



Input: head = [1,1,2,3,3]
Output: [1,2,3]

Constraints:

- The number of nodes in the list is in the range [0, 300].
- -100 <= Node.val <= 100
- The list is guaranteed to be sorted in ascending order.

Question 3:

Given a non-empty array of integers, return the *k* most frequent elements.
Example 1:

Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]

Example 2:

Input: nums = [1], k = 1
Output: [1]

Note:

- You may assume k is always valid, $1 \leq k \leq$ number of unique elements.
- Your algorithm's time complexity must be better than O(n log n), where n is the array's size.
- It's guaranteed that the answer is unique, in other words the set of the top k frequent elements is unique.
- You can return the answer in any order.

Question 4:

Design a class to find the kth largest element in a stream. Note that it is the kth largest element in the sorted order, not the kth distinct element.
Implement KthLargest class:

- KthLargest(int k, int[] nums) Initializes the object with the integer k and the stream of integers nums.
- int add(int val) Returns the element representing the kth largest element in the stream.

  Example 1:

  Input

  ["KthLargest", "add", "add", "add", "add", "add"]
  [[3, [4, 5, 8, 2]], [3], [5], [10], [9], [4]]
  Output
  [null, 4, 5, 5, 8, 8]

  Explanation
  KthLargest kthLargest = new KthLargest(3, [4, 5, 8, 2]);
  kthLargest.add(3); // return 4
  kthLargest.add(5); // return 5
  kthLargest.add(10); // return 5
  kthLargest.add(9); // return 8
  kthLargest.add(4); // return 8

Constraints:

- 1 <= k <= 104
- 0 <= nums.length <= 104
- -104 <= nums[i] <= 104
- -104 <= val <= 104
- At most 104 calls will be made to add.
- It is guaranteed that there will be at least k elements in the array when you search for the kth element.