1. **Sort an Array [1 point]**

Given an array of integers nums, sort the array in ascending order and return it.

You must solve the problem **without using any built-in** functions in O(nlog(n)) time complexity and with the smallest space complexity possible.

**Example 1:**

**Input:** nums = [5,2,3,1]

**Output:** [1,2,3,5]

**Explanation:** After sorting the array, the positions of some numbers are not changed (for example, 2 and 3), while the positions of other numbers are changed (for example, 1 and 5).

**Example 2:**

**Input:** nums = [5,1,1,2,0,0]

**Output:** [0,0,1,1,2,5]

**Explanation:** Note that the values of nums are not necessairly unique.

**Constraints:**

* 1 <= nums.length <= 5 \* 10000
* -5 \* 10000 <= nums[i] <= 5 \* 10000

1. **Sort List [1 point]**

Given the head of a linked list, return the list after sorting it in ***ascending order***.

**Example 1:**

A diagram of a diagram

Description automatically generated

**Input:** head = [4,2,1,3]

**Output:** [1,2,3,4]

**Example 2:**

A diagram of a diagram

Description automatically generated

**Input:** head = [-1,5,3,4,0]

**Output:** [-1,0,3,4,5]

**Example 3:**

**Input:** head = []

**Output:** []

**Constraints:**

* The number of nodes in the list is in the range [0, 5 \* 10000].
* -100000 <= Node.val <= 100000

1. **Remove Duplicates from Sorted List [1 point]**

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:**

A diagram of a flowchart

Description automatically generated

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

**Output:** [1,2]

**Example 2:**

A diagram of a diagram

Description automatically generated

**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.

1. **Remove Linked List Elements [1 point]**

Given the head of a linked list and an integer val, remove all the nodes of the linked list that has Node.val == val, and return *the new head*.

**Example 1:**

A diagram of a diagram

Description automatically generated

**Input:** head = [1,2,6,3,4,5,6], val = 6

**Output:** [1,2,3,4,5]

**Example 2:**

**Input:** head = [], val = 1

**Output:** []

**Example 3:**

**Input:** head = [7,7,7,7], val = 7

**Output:** []

**Constraints:**

* The number of nodes in the list is in the range [0, 10000].
* 1 <= Node.val <= 50
* 0 <= val <= 50

1. **Reverse Linked List [1 point]**

Given the head of a singly linked list, reverse the list, and return *the reversed list*.

**Example 1:**



**Input:** head = [1,2,3,4,5]

**Output:** [5,4,3,2,1]

**Example 2:**

A diagram of a diagram

Description automatically generated

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

**Output:** [2,1]

**Example 3:**

**Input:** head = []

**Output:** []

**Constraints:**

* The number of nodes in the list is the range [0, 5000].
* -5000 <= Node.val <= 5000

**Note:** A linked list can be reversed either iteratively or recursively. You should implement both.