Module-2 Homework

- Give the code for each question
- Give me a clear expalanation of your solution way for each question
- Submit the notebook as well as its pdf version

Question 1: Leetcode Question #189. Easy. "Rotate Array" Given an integer array nums, rotate the array to the right by k steps, where k is non-negative. Example 1: Input: nums = [1,2,3,4,5,6,7], k = 3 Output: [5,6,7,1,2,3,4] Explanation: rotate 1 steps to the right: [7,1,2,3,4,5,6] rotate 2 steps to the right: [6,7,1,2,3,4,5] rotate 3 steps to the right: [5,6,7,1,2,3,4] Example 2: Input: nums = [-1,-100,3,99], k = 2 Output: [3,99,-1,-100] Explanation: rotate 1 steps to the right: [99,-1,-100,3] rotate 2 steps to the right: [3,99,-1,-100]

In [16]:
```
def rotate_array(nums, k):
    l = len(nums)                       # take the length of nums
    temp = None                         # take temp -- to store current-store
    last = None                         # take last -- to get the last-value

    for rotation in range(k):           # iterate through rotations - k
        for i in range(l):                # iterte through the len(l)
            if i == 0:                      # if the element is 1st in the list:
                last = nums[l-1]                # store last -- a last element of [nums]
            else:                           # else
                last = temp                     # store last -- the latest temp (element from the last-it

            temp = nums[i]              # store the element to temp
            nums[i] = last              # update the element to last

    return nums

nums = [1,2,3,4,5,6,7]
k = 3
print('nums:', rotate_array(nums, k))
```

```
nums: [5, 6, 7, 1, 2, 3, 4]
```

In [17]:
```python
def rotate_array(nums, k):
    l = len(nums)                           # take the length of nums
    temp = None                             # take temp -- to store current-store
    last = None                             # take last -- to get the last-value

    for rotation in range(k):               # iterate through rotations - k
        for i in range(l):                    # iterte through the len(l)
            if i == 0:                          # if the element is 1st in the list:
                last = nums[l-1]                    # store last -- a last element of [nums]
            else:                               # else
                last = temp                         # store last -- the latest temp (element from the last-it

            temp = nums[i]                  # store the element to temp
            nums[i] = last                  # update the element to last

    return nums

nums = [-1,-100,3,99]
k = 2
print('nums:', rotate_array(nums, k))
```

```
nums: [3, 99, -1, -100]
```

Question 2: Leetcode Question #665. Medium. "Non-decreasing Array" Given an array nums with n integers, your task is to check if it could become non-decreasing by modifying at most one element. We define an array is non-decreasing if nums[i] <= nums[i + 1] holds for every i (0-based) such that (0 <= i <= n - 2). Example 1: Input: nums = [4,2,3] Output: true Explanation: You could modify the first 4 to 1 to get a non-decreasing array. Example 2: Input: nums = [4,2,1] Output: false Explanation: You cannot get a non-decreasing array by modifying at most one element. Constraints: n == nums.length

In [20]:
```
def can_non_decreasing(nums):
    modify_count = 0                                        # count of changes -- make list to non-decreasing
    n = len(nums)                                           # take the length of nums

    for i in range(n):                                      # iterate theough len(nums)
        if (i != n-1) and (nums[i] > nums[i+1]):            # if (i is not the last)
                                                            # and (current element is bigger than the next elemen
                modify_count += 1                               # increase modify_count to + 1
                if modify_count > 1:                            # if modify_count greater than 1
                    return False                                    # then, return false

    if modify_count == 1:                                   # if modify_count is 1
        return True                                             # then, return true

nums = [4,2,3]
print(can_non_decreasing(nums))
```

```
True
```

In [19]:
```
def can_non_decreasing(nums):
    modify_count = 0                                        # count of changes -- make list to non-decreasing
    n = len(nums)                                           # take the length of nums

    for i in range(n):                                      # iterate theough len(nums)
        if (i != n-1) and (nums[i] > nums[i+1]):            # if (i is not the last)
                                                            # and (current element bigger than the next element)
                modify_count += 1                               # increase modify_count to + 1
                if modify_count > 1:                            # if modify_count greater than 1
                    return False                                    # then, return false

    if modify_count == 1:                                   # if modify_count is 1
        return True                                             # then, return true

nums = [4,2,1]
print(can_non_decreasing(nums))
```

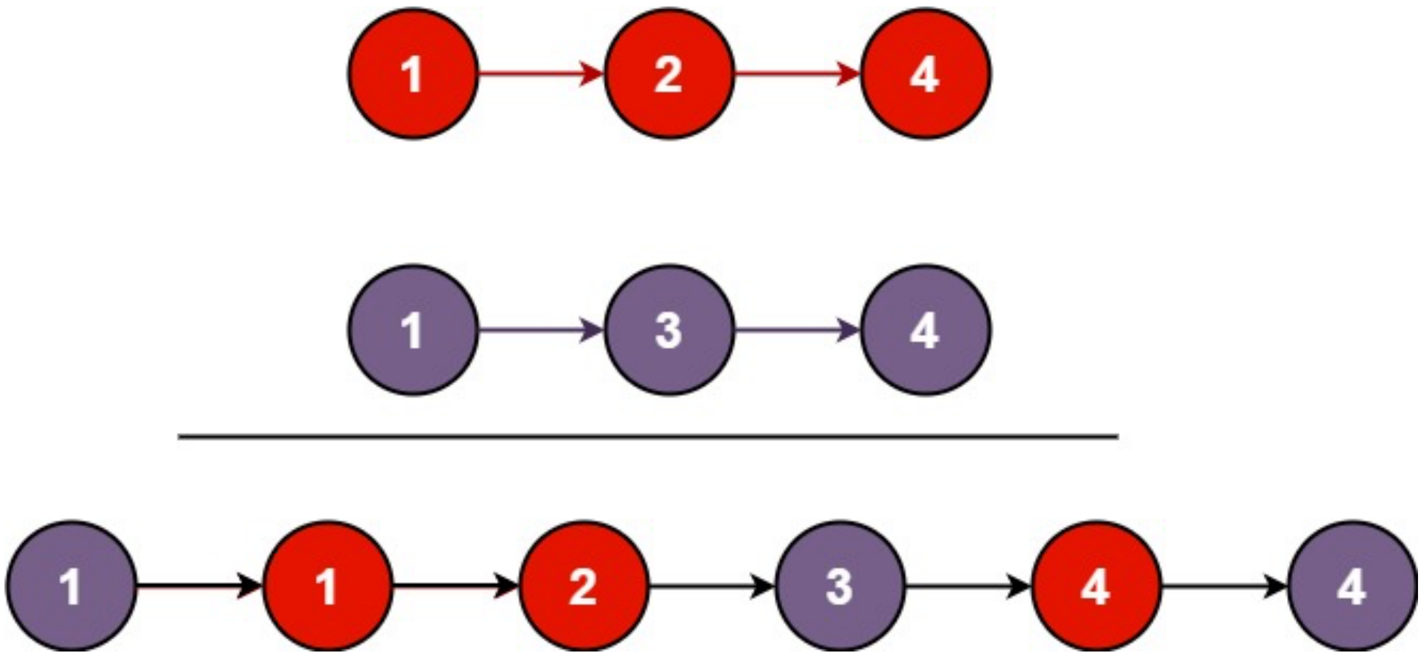
```
False
```

Question 3: Leetcode Question #21. Easy. "Merge Two Sorted List" ou are given the heads of two sorted linked lists list1 and list2.

Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

Example 1:



Input: list1 = [1,2,4], list2 = [1,3,4] Output: [1,1,2,3,4,4] Example 2:

Input: list1 = [], list2 = [] Output: [] Example 3:

Input: list1 = [], list2 = [0] Output: [0]

Constraints:

The number of nodes in both lists is in the range [0, 50]. -100 <= Node.val <= 100 Both list1 and list2 are sorted in non-decreasing order.

In [21]:
```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def mergeTwoLists(list1, list2):
    # Create a dummy node to serve as the head of the merged list
    dummy = ListNode()
    current = dummy

    # Traverse both lists simultaneously
    while list1 and list2:
        # Compare the values of the current nodes of both lists
        if list1.val < list2.val:
            current.next = list1
            list1 = list1.next
        else:
            current.next = list2
            list2 = list2.next

        # Move current pointer to the next node
        current = current.next

    # If one list is exhausted, append the remaining nodes of the other list
    if list1:
        current.next = list1
    else:
        current.next = list2

    # Return the head of the merged list (skipping the dummy node)
    return dummy.next


# Create the input lists
list1 = ListNode(1)
list1.next = ListNode(2)
```

```
list1.next.next = ListNode(4)

list2 = ListNode(1)
list2.next = ListNode(3)
list2.next.next = ListNode(4)

# Call the function and print the result
result = mergeTwoLists(list1, list2)
while result:
    print(result.val, end=" ")
    result = result.next
```

```
1 1 2 3 4 4
```