61. Rotate List
We will count all nodes sum. Then we will get n = sum - k % sum. And put nth node and later ahead. TC is O(n)

```
class Solution:
    def rotateRight(self, head: ListNode, k: int) -> ListNode:
        dummy = ListNode(0)
        dummy.next = head
        dummy_mem = dummy
        count = 0
        while dummy.next:
            dummy = dummy.next
            count += 1
        last = dummy
        if count == 0 or k % count == 0:
            return head
        count = count - k % count
        dummy = dummy_mem
        for i in range(count):
            dummy = dummy.next
        last.next = dummy_mem.next
        dummy_mem.next = dummy.next
        dummy.next = None
        return dummy_mem.next
```

74. Search a 2D Matrix
We will use binary search get row first and then column second to check whether that one is the target. TC is O(logn)

```
from bisect import *
class Solution:
    def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        if not matrix or not matrix[0]:
            return False
        starts = bisect_left(list(map(lambda a: a[0], matrix)), target)
        if starts == len(matrix) or matrix[starts][0] != target:
            starts -= 1
        if starts < 0:
            return False
        i = bisect_left(matrix[starts], target)
        return i < len(matrix[0]) and matrix[starts][i] == target
```

80. Remove Duplicates from Sorted Array II
We will iterate through all elements using two pointer, if the fast one points to an element which showing for more than twice, we will ignore it and move to the next one. TC is O(n)

```
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:
        length = len(nums)
        slow, fast = 0, 0
        prev = None
        count = 0
        while fast < length:
            if fast > 0 and nums[fast] == prev and count == 2:
                fast += 1
                continue
            if fast > 0 and nums[fast] == prev:
                count += 1
            else:
                count = 1
            nums[slow] = nums[fast]
            prev = nums[fast]
            slow += 1
            fast += 1
        return slow
```

4. Search in Rotated Sorted Array II
We will check whether i is in array. TC is O(n)

```
class Solution:
    def search(self, nums: List[int], target: int) -> bool:
        return target in nums
        l, r = 0, len(nums) - 1
        if not nums:
            return False
        while l < r:
            mid = (l + r) // 2
            if nums[mid] == target:
                return True
            if nums[mid] < nums[0]:
                if target < nums[0]:
                    if nums[mid] < target:
                        l = mid + 1
                    else:
                        r = mid
                else:
                    r = mid
            else:
                if target >= nums[0]:
                    if nums[mid] < target:
```

```
                l = mid + 1
            else:
                r = mid
        else:
            l = mid + 1

    return nums[l] == target
```

82. Remove Duplicates from Sorted List II
We will collect all keys with value larger than 1, once val is in our deleted key, we will delete that node. TC is O(n)

```
from collections import defaultdict
class Solution:
    def deleteDuplicates(self, head: ListNode) -> ListNode:
        memo = defaultdict(int)
        dummy = ListNode(0)
        dummy.next = head
        dummy_mem = dummy
        while dummy.next:
            memo[dummy.next.val] += 1
            dummy = dummy.next
        dummy = dummy_mem
        deleted_keys = set(map(lambda b: b[0], filter(lambda a: a[1] > 1, memo.items())))
        while dummy and dummy.next:
            while dummy.next and dummy.next.val in deleted_keys:
                dummy.next = dummy.next.next
            dummy = dummy.next
        return dummy_mem.next
```