Question 1

1st solution (brute force )

def twoSum(nums, target):

    result = []

    n = len(nums)

    for i in range(n - 1):

        for j in range(i + 1, n):

            if nums[i] + nums[j] == target:

                pair = [i, j]

                if pair not in result:

                    result.append(pair)

    return result

2nd Solution (Improved approach)

def twoSum(nums, target):

    nums.sort()

    left = 0

    right = len(nums) - 1

    while left < right:

        temp = nums[left] + nums[right]

        if temp == target:

            return [left, right]

        elif temp < target:

            left += 1

        else:

            right -= 1

    return 0

using two pointer approach

Question 2

1st approach   
encounter target and remove immediately and move forward

def removeOccurences(nums, val):

    n = len(nums)

    i = 0

    while i < n:

        if nums[i] == val:

            nums.remove(nums[i])

            n -= 1

        else:

            i += 1

    res = len(nums)

    return res , nums

Question 3

1st approach

Take advantage of sorted array, find the position of target element if we get same elemnt in the array, otherwise if get the element greater than target return that index of greatest element

def FindPerfectPlace(nums, target):

    for i in range(len(nums)):

        if nums[i] == target:

            return i

        elif nums[i] > target:

            return i

    return len(nums)

Question 4

Left ( due to some confusion, Sorry )

Question 5

1st approach

def merge(nums1, m, nums2, n):

    for i in range(n):

        nums1[m + i] = nums2[i]

    nums1.sort()

return nums1

simply merge two array and apply sort function

2nd Approach (using three pointer )

def mergeND(nums1, m, nums2, n):

    nums1dup = nums1[0:m]

    p1 = 0

    p2 = 0

    pm = 0

    while p1 < m and p2 < n:

        if nums1dup [p1] <= nums2[p2]:

            nums1[pm] = nums1dup[p1]

            p1 += 1

        else:

            nums1[p] = nums2[p2]

            p2 += 1

        pm += 1

    while p1 < m:

        nums1[p] = nums1dup[p1]

        p1 += 1

        pm += 1

    while p2 < n:

        nums1[pm] = nums2[p2]

        p2 += 1

        pm += 1

return nums1

6th question

1st approach

Constantly adding element in set , why set because set only store unique element, during forward movement any element encounters occurrence in set also , Jackpot , here is our catch , we find a duplicate .

def Duplicatefinder(nums):

    temp = set()

    for num in nums:

        if num in temp:

            return True

        temp.add(num)

    return False

7th question

1st approach

def moveZeroes(nums):

    pointer = 0

    for i in nums:

        if i != 0:

            nums[pointer] = i

            pointer += 1

    for i in range(pointer, len(nums)):

        nums[i] = 0

return nums

8th question

1st approach

def twiceOrMissing(nums):

    duplicate = None

    mising = None

    for i in range(len(nums)):

        for j in range(i + 1, len(nums)):

            if nums[i] == nums[j]:

                duplicate = nums[i]

                break

    for i in range(1, len(nums) + 1):

        if i not in nums:

            mising = i

            break

    return [duplicate, mising]