Q1.

sol.class TwoSum:

def \_\_init\_\_(self, list1, target):

self.list1 = list1

self.target = target

def solution(self):

length = len(list1)

for i in range(length-1):

for j in range(i+1, length):

if list1[i]+list1[j] == self.target:

new\_list = i, j

return list(new\_list)

return -1

list1 = [2,7,11,15]

target = 9

obj = TwoSum(list1, target)

print(obj.solution())

Q2.

Sol.def remove\_element(nums, val):

k = 0

for num in nums:

if num != val:

nums[k] = num

k += 1

return k

nums = [3, 2, 2, 3]

val = 3

result = remove\_element(nums, val)

print(f"Modified nums: {nums[:result]}")

print(f"Count of elements not equal to val: {result}")

Q3.

Sol.def search\_insert(nums, target):

left = 0

right = len(nums) - 1

while left <= right:

mid = (left + right) // 2

if nums[mid] == target:

return mid

elif nums[mid] < target:

left = mid + 1

else:

right = mid - 1

return left

nums = [1, 3, 5, 6]

target = 5

index = search\_insert(nums, target)

print(f"Target {target} should be inserted at index {index}")

Q4.

Sol. def incrementLargeInteger(digits):

carry = 1

for i in range(len(digits) - 1, -1, -1):

digits[i] += carry

carry = digits[i] // 10

digits[i] %= 10

if carry:

digits.insert(0, carry)

return digits

digits = [1, 2, 3] result = incrementLargeInteger(digits)

print(result) # Output: [1, 3, 0]

Q5.

Sol. class Solution(object):

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

i = 0

j = 0

end = len(nums1)-1

while end>=0 and not nums1[end]:

end-=1

while j<len(nums2) :

if i>end and not nums1[i]:

nums1[i] = nums2[j]

j+=1

elif nums1[i]>nums2[j]:

self.shift(nums1,i)

nums1[i] = nums2[j]

end+=1

j+=1

i+=1

return nums1

def shift(self,num,i):

j = len(num)-1

while not num[j]:

j-=1

while j>=i:

num[j+1] = num[j]

j-=1

ob = Solution()

print(ob.merge([1,2,3,0,0,0],3,[2,5,6],3))

Q6.

Sol. def containsDuplicate(nums):

seen = set()

for num in nums:

if num in seen:

return True

seen.add(num)

return False

nums = [1, 2, 3, 1]

print(containsDuplicate(nums))

Q7.

Sol. def moveZeroes(nums):

slow = 0

for fast in range(len(nums)):

if nums[fast] != 0:

nums[slow] = nums[fast]

slow += 1

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

nums[i] = 0

nums = [0, 1, 0, 3, 12]

moveZeroes(nums)

print(nums)

Q8.

Sol.def findErrorNums(nums):

n = len(nums)

seen = set()

duplicate = 0

for num in nums:

if num in seen:

duplicate = num

else:

seen.add(num)

missing = 0

for num in range(1, n+1):

if num not in seen:

missing = num

return [duplicate, missing]

nums = [1, 2, 2, 4]

result = findErrorNums(nums)

print(result)