Q1

class Solution:

def mergeKLists(self, head: List[Optional[ListNode]]) -> Optional[ListNode]:

new=[]

for i in head:

while(i):

new.append(i.val)

i = i.next

a=sorted(new,reverse=True)

final=None

for i in a:

final=ListNode(i,final)

return final

Q3

from queue import PriorityQueue as pq

class Solution:

def sortArray(self, nums: List[int]) -> List[int]:

q = pq()

for i in nums: q.put(i)

for i in range(len(nums)): nums[i] = q.get()

return nums

Q4

class Solution:

def moveZeroes(self, nums: List[int]) -> None:

zero\_pos = 0

for i in range(len(nums)):

if nums[i] != 0:

nums[i], nums[zero\_pos] = nums[zero\_pos], nums[i]

zero\_pos += 1

return zero\_pos

Q5

class Solution:

def rearrangeArray(self, nums: List[int]) -> List[int]:

positives = [num for num in nums if num > 0]

negatives = [num for num in nums if num < 0]

res = zip(positives, negatives)

return chain(\*res)

Q6

class Solution:

def merge(self, nums1: List[int], m: int, nums2: List[int], n: int) -> None:

last = m + n -1

while m > 0 and n > 0:

if nums1[m-1] > nums2[n-1]:

nums1[last] = nums1[m-1]

m -= 1

else:

nums1[last] = nums2[n-1]

n -= 1

last -= 1

while n > 0:

nums1[last] = nums2[n-1]

n,last = n-1, last-1

Q7

class Solution:

def intersection(self, nums1: List[int], nums2: List[int]) -> List[int]:

if len(nums1) < len(nums2):

nums1,nums2 = nums2,nums1

res = []

nums1 = sorted(nums1)

nums2 = set(nums2)

for i in nums2:

l,r = 0,len(nums1)-1

while l <=r:

m = (l+r)>>1

if nums1[m] == i:

res.append(i)

break

else:

if nums1[m] < i:

l = m + 1

else:

r = m - 1

return res

Q8

class Solution:

def intersect(self, nums1: List[int], nums2: List[int]) -> List[int]:

count1 = {}

count2 = {}

for num in nums1:

count1[num] = count1.get(num, 0) + 1

for num in nums2:

count2[num] = count2.get(num, 0) + 1

arr = []

for num in count1:

if num in count2:

count = min(count1[num], count2[num])

arr.extend([num] \* count)

return arr