Q1.def arraysIntersection(arr1, arr2, arr3):

result = []

i, j, k = 0, 0, 0

while i < len(arr1) and j < len(arr2) and k < len(arr3):

if arr1[i] == arr2[j] == arr3[k]:

result.append(arr1[i])

i += 1

j += 1

k += 1

elif arr1[i] < arr2[j]:

i += 1

elif arr2[j] < arr3[k]:

j += 1

else:

k += 1

return result

Q2def findDisappearedNumbers(nums1, nums2):

set1 = set(nums1)

set2 = set(nums2)

distinct\_nums1 = list(set1 - set2)

distinct\_nums2 = list(set2 - set1)

return [distinct\_nums1, distinct\_nums2]

Q3.def transpose(matrix):

rows = len(matrix)

cols = len(matrix[0])

# Create a new matrix with swapped rows and columns

transposed\_matrix = [[0] \* rows for \_ in range(cols)]

for i in range(rows):

for j in range(cols):

transposed\_matrix[j][i] = matrix[i][j]

return transposed\_matrix

Q4.def arrayPairSum(nums):

nums.sort()

pair\_sum = 0

for i in range(0, len(nums), 2):

pair\_sum += nums[i]

return pair\_sum

Q5.def arrangeCoins(n):

left = 0

right = n

while left <= right:

mid = left + (right - left) // 2

curr = mid \* (mid + 1) // 2

if curr == n:

return mid

elif curr < n:

left = mid + 1

else:

right = mid - 1

return right

Q6.def sortedSquares(nums):

n = len(nums)

result = [0] \* n

left, right = 0, n - 1

index = n - 1

while left <= right:

if abs(nums[left]) > abs(nums[right]):

result[index] = nums[left] \*\* 2

left += 1

else:

result[index] = nums[right] \*\* 2

right -= 1

index -= 1

return result

Q7.def maxCount(m: int, n: int, ops: List[List[int]]) -> int:

if not ops:

return m \* n

min\_row = float('inf')

min\_col = float('inf')

for op in ops:

min\_row = min(min\_row, op[0])

min\_col = min(min\_col, op[1])

return min\_row \* min\_col

Q8.def shuffle(nums):

n = len(nums) // 2

result = [0] \* (2 \* n)

for i in range(n):

result[2 \* i] = nums[i]

result[2 \* i + 1] = nums[n + i]

return result