Q=>1

```
def reconstruct_permutation(s):
  n = len(s)
  perm = [0] * (n + 1)
  for i in range(n):
    if s[i] == 'I':
      perm[i + 1] = perm[i] + 1
    else:
      perm[i + 1] = perm[i] - 1
  for i in range(1, n + 1):
    perm[perm[i]] = i
  return perm
```

Q=>2

```
def find_integer_in_matrix(matrix, target):
  row = 0
  col = n - 1
  while row < m and col >= 0:
    if matrix[row][col] == target:
      return True
    elif matrix[row][col] < target:
      row += 1
    else:
      col -= 1
  return False
```

Q=>3

```
def is_mountain_array(arr):
  if len(arr) < 3:
    return False
  increasing = True
  decreasing = True
  for i in range(1, len(arr)):
    if increasing and arr[i] <= arr[i - 1]:
      increasing = False
    elif decreasing and arr[i] >= arr[i - 1]:
      decreasing = False
    elif not increasing and not decreasing:
      return False
  return increasing and decreasing
```

Q=>4

```
def max_length_of_subarray_with_equal_0_and_1(nums):
  count_0 = 0
  count_1 = 0
  max_len = 0
  for num in nums:
    if num == 0:
      count_0 += 1
    else:
      count_1 += 1
    if count_0 == count_1:
      max_len = max(max_len, count_0 + count_1)
    elif count_0 > count_1:
      count_0 = 0
    else:
      count_1 = 0
  return max_len
```

Q=>5

```
def min_product_sum(nums1, nums2):
```

```
  min_sum = float("inf")
  for permutation in itertools.permutations(nums1):
    sum = 0
    for i in range(len(nums1)):
      sum += permutation[i] * nums2[i]
    min_sum = min(min_sum, sum)
  return min_sum
```

Q=>6

```
def find_original_array(changed):
  seen = set()
  original = []
  for num in changed:
    if num in seen:
      original.append(num // 2)
    else:
      seen.add(num * 2)
  return original
```

Q=>7

```
def generate_spiral_matrix(n):
  matrix = [[0 for _ in range(n)] for _ in range(n)]
  i, j = 0, 0
  counter = 1
  while counter <= n * n:
    for k in range(4):
      if k == 0:
        while j < n and matrix[i][j] == 0:
          matrix[i][j] = counter
          counter += 1
          j += 1
      elif k == 1:
        while i < n and matrix[i][j] == 0:
          matrix[i][j] = counter
          counter += 1
          i += 1
      elif k == 2:
        while j >= 0 and matrix[i][j] == 0:
          matrix[i][j] = counter
          counter += 1
          j -= 1
      elif k == 3:
        while i >= 0 and matrix[i][j] == 0:
          matrix[i][j] = counter
          counter += 1
          i -= 1
  return matrix
```

Q=>8

```
def multiply_sparse_matrices(mat1, mat2):
  product = [[0 for _ in range(len(mat2[0]))] for _ in range(len(mat1))]
  for i in range(len(mat1)):
    for j in range(len(mat2[0])):
      for k in range(len(mat2)):
        if mat1[i][k] != 0 and mat2[k][j] != 0:
          product[i][j] += mat1[i][k] * mat2[k][j]
  return product
```