Q=>1

```
def convert_1d_to_2d(array, rows, columns):
  array_2d = []
  for i in range(rows):
    row = array[i * columns:(i + 1) * columns]
    array_2d.append(row)
  return array_2d
```

Q=>2

```
def staircase(n):
  rows = 0
  while n > 0:
    if n >= rows + 1:
      n -= rows + 1
      rows += 1
    else:
      break
  return rows
```

Q=>3

```
def squares_of_sorted_array(nums):
  squares = []
  for num in nums:
    squares.append(num * num)
  squares.sort()
  return squares
```

Q=>4

```
def distinct_integers_in_two_arrays(nums1, nums2):
  answer = [[], []]
  seen = set()
  for num in nums1:
    if num not in seen:
      answer[0].append(num)
      seen.add(num)
  seen = set()
  for num in nums2:
    if num not in seen:
      answer[1].append(num)
      seen.add(num)
  return answer
```

Q=>5

```
def distance_value(arr1, arr2, d):
  count = 0
  for num in arr1:
    found = False
    for other_num in arr2:
      if abs(num - other_num) <= d:
        found = True
        break
    if not found:
      count += 1
  return count
```

Q=>6

```
def find_duplicates(nums):
  seen = set()
  duplicates = []
  for num in nums:
    if num in seen:
      duplicates.append(num)
    else:
      seen.add(num)
  return duplicates
```

Q=>7

```
def find_minimum_element(nums):
  low = 0
  high = len(nums) - 1
  while low <= high:
    mid = (low + high) // 2
    if nums[mid] < nums[high]:
      return nums[mid]
    elif nums[mid] > nums[low]:
      high = mid - 1
    else:
      high -= 1
  return nums[0]
```

Q=>8

```
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
```