Q=>1

```
def closest_sum(nums, target):
  min_diff = float("inf")
  sum = 0
  for i in range(len(nums)):
    for j in range(i + 1, len(nums)):
      for k in range(j + 1, len(nums)):
        curr_sum = nums[i] + nums[j] + nums[k]
        diff = abs(target - curr_sum)
        if diff < min_diff:
          min_diff = diff
          sum = curr_sum


  return sum
if __name__ == "__main__":
  nums = [-1, 2, 1, -4]
  target = 1
  print(closest_sum(nums, target))
```

Q=>2

```
def four_sum(nums, target):
  quadruplets = []
  nums.sort()
  for i in range(len(nums)):
    if i > 0 and nums[i] == nums[i - 1]:
      continue
    for j in range(i + 1, len(nums)):
      if j > i + 1 and nums[j] == nums[j - 1]:
        continue
      target_sum = target - nums[i] - nums[j]
      left = j + 1
      right = len(nums) - 1
      while left <= right:
        if nums[left] + nums[right] == target_sum:
          quadruplets.append([nums[i], nums[j], nums[left], nums[right]])
```

```
     left += 1

     right -= 1

    elif nums[left] + nums[right] < target_sum:

     left += 1

    else:

     right -= 1


 return quadruplets



if __name__ == "__main__":

 nums = [1, 0, -1, 0, -2, 2]

 target = 0

 print(four_sum(nums, target))
```

Q=>3

```
def next_permutation(nums):

 i = len(nums) - 1

 while i > 0 and nums[i - 1] >= nums[i]:

  i -= 1


 if i == 0:

  nums.reverse()

  return nums


 j = len(nums) - 1

 while nums[j] <= nums[i - 1]:

  j -= 1


 nums[i - 1], nums[j] = nums[j], nums[i - 1]

 nums[i:] = nums[i:][::-1]


 return nums

if __name__ == "__main__":
 nums = [1, 2, 3]
 print(next_permutation(nums))
```

Q=>4

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

  return low
if __name__ == "__main__":
  nums = [1, 3, 5, 6]
  target = 5
  print(search_insert(nums, target))
```

Q=>5

```
def array_plus_one(digits):
   carry = 1
  for i in range(len(digits) - 1, -1, -1):
    digits[i] += carry
    if digits[i] == 10:
      digits[i] = 0
      carry = 1
    else:
      carry = 0

  if carry == 1:
    digits.append(1)
  return digits



if __name__ == "__main__":
  digits = [1, 2, 3]
  print(array_plus_one(digits))
```

Q=>6

```
def find_single_one(nums):
  seen = {}
  for num in nums:
    if num in seen:
      seen[num] += 1
    else:
      seen[num] = 1

  for num, count in seen.items():
    if count == 1:
      return num

  raise ValueError("No single one found")



if __name__ == "__main__":
  nums = [2, 2, 1]
  print(find_single_one(nums))
```

Q=>7

```
def find_missing_ranges(nums, lower, upper):
  ranges = []
  current_range = [lower, lower]
  for num in nums:
    if num < current_range[1]:
      current_range[1] = num
    else:
      if current_range[0] != current_range[1]:
        ranges.append(current_range)
      current_range = [num, num]

  if current_range[0] != current_range[1]:
    ranges.append(current_range)

  ranges.extend([(n, upper) for n in range(current_range[1] + 1, upper + 1)])

  return sorted(ranges)



if __name__ == "__main__":
  nums = [0, 1, 3, 50, 75]
  lower = 0
  upper = 99
  print(find_missing_ranges(nums, lower, upper))
```

Q=>8

```
def can_attend_all_meetings(intervals):
  intervals.sort()
  for i in range(1, len(intervals)):
    if intervals[i][0] < intervals[i - 1][1]:
      return False

  return True
if __name__ == "__main__":
  intervals = [[0, 30], [5, 10], [15, 20]]
  print(can_attend_all_meetings(intervals))
```