Q=>1

```
def max_pair_sum(nums):
  nums.sort()
  sum = 0
  for i in range(len(nums) // 2):
    sum += min(nums[2 * i], nums[2 * i + 1])
  return sum


if __name__ == "__main__":
  nums = [1, 4, 3, 2]
  print(max_pair_sum(nums))
```

Q=>2

```
def max_types(candyType):
  types = set()
  for candy in candyType:
    types.add(candy)
  return min(len(types), len(candyType) // 2)


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

Q=>3

```
def longest_harmonious_subsequence(nums):
  max_value = nums[0]
  min_value = nums[0]
  max_count = 1
  min_count = 1
  for num in nums:
    if num > max_value:
      max_value = num
      max_count = 1
    elif num == max_value:
      max_count += 1
    elif num < min_value:
      min_value = num
      min_count = 1
    elif num == min_value:
      min_count += 1
  return max(max_count, min_count)
```

```
if __name__ == "__main__":
  nums = [1, 3, 2, 2, 5, 2, 3, 7]
  print(longest_harmonious_subsequence(nums))
```

Q=>4

```
def can_plant_flowers(flowerbed, n):
  count = 0
  for i in range(len(flowerbed)):
    if flowerbed[i] == 0:
      if i == 0 or flowerbed[i - 1] == 0:
        count += 1
      elif i == len(flowerbed) - 1 or flowerbed[i + 1] == 0:
        count += 1
  return count >= n



if __name__ == "__main__":
  flowerbed = [1, 0, 0, 0, 1]
  n = 1
  print(can_plant_flowers(flowerbed, n))
```

Q=>5

```
def max_product_of_three(nums):
  max1 = max2 = max3 = -float("inf")
  min1 = min2 = float("inf")
  for num in nums:
    if num > max1:
      max3 = max2
      max2 = max1
      max1 = num
    elif num > max2:
      max3 = max2
      max2 = num
    elif num > max3:
      max3 = num

    if num < min1:
```

```
      min2 = min1
      min1 = num
    elif num < min2:
      min2 = num

  return max(max1 * max2 * max3, min1 * min2 * max1)


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

Q=>6

```
def binary_search(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 -1


if __name__ == "__main__":
  nums = [-1, 0, 3, 5, 9, 12]
  target = 9
  print(binary_search(nums, target))
```

Q=>8

```
def min_score(nums, k):
  min_val = nums[0]
  max_val = nums[0]
  for num in nums:
    min_val = min(min_val, num)
```

```
    max_val = max(max_val, num)

  score = max_val - min_val
  for i in range(len(nums)):
    for x in range(-k, k + 1):
      new_score = max(nums[i] + x, min_val) - min(nums[i] + x, max_val)
      if new_score < score:
        score = new_score
  return score


if __name__ == "__main__":
  nums = [1]
  k = 0
  print(min_score(nums, k))
```