ASSIGNMENT 2

Question 1

Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2),..., (an, bn) such that the sum of min(ai, bi) for all i is maximized. Return the maximized sum

Answer:

```
def arrayPairSum(nums):
    nums.sort()  # Sort the array in ascending order total_sum = 0

    for i in range(0, len(nums), 2):
        total_sum += nums[i]

    return total_sum
```

In [2]:

```
nums = [1, 4, 3, 2]  # Given result = arrayPairSum(nums)
print(result)
```

```
4
```

Question 2

Alice has n candies, where the ith candy is of type candyType[i]. Alice noticed that she started to gain weight, so she visited a doctor.

The doctor advised Alice to only eat n / 2 of the candies she has (n is always even). Alice likes her candies very much, and she wants to eat the maximum number of different types of candies while still following the doctor's advice.

Given the integer array candyType of length n, return the maximum number of different types of candies she can eat if she only eats n / 2 of them.

Answer

In [3]:

```
def distributeCandies(candyType):
    max_candies = len(candyType) // 2
    unique_candies = len(set(candyType))

    return min(unique_candies, max_candies)
```

In [4]:

```
candyType = [1, 1, 2, 2, 3, 3] # Given result = distributeCandies(candyType)
print(result)
```

```
3
```

Question 3

We define a harmonious array as an array where the difference between its maximum value and its minimum value is exactly 1.

Given an integer array nums, return the length of its longest harmonious subsequence among all its possible subsequences.

A subsequence of an array is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements.

Answer

In [5]:

```
def findLHS(nums):
    num_freq = {}  # Dictionary to store the frequency of numbers max_length = 0

    # Count the frequency of each number for num in nums:
        num_freq[num] = num_freq.get(num, 0) + 1

    # Check for each number and its complement for num in num_freq:
        if num + 1 in num_freq:
            length = num_freq[num] + num_freq[num + 1]
            max_length = max(max_length, length)

    return max_length
```

In [6]:

```
nums = [1, 3, 2, 2, 5, 2, 3, 7] # Given result = findLHS(nums)
print(result)
```

```
5
```

Question 4

You have a long flowerbed in which some of the plots are planted, and some are not. However, flowers cannot be planted in adjacent plots. Given an integer array flowerbed containing 0's and 1's, where 0 means empty and 1 means not empty, and an integer n, return true if n new flowers can be planted in the flowerbed without violating the no-adjacent-flowers rule and false otherwise

Answer

In [7]:

```
def canPlaceFlowers(flowerbed, n):
    length = len(flowerbed)
    count = 0
    i = 0

    while i < length:
        if flowerbed[i] == 0 and (i == 0 or flowerbed[i - 1] == 0) and (i == length - 1 or flowerbed[i + 1] == 0):
            flowerbed[i] = 1
            count += 1
            i += 2
        else:
            i += 1

    return count >= n
```

In [8]:

```
flowerbed = [1, 0, 0, 0, 1] # Given n = 1 # Given result = canPlaceFlowers(flowerbed, n)
print(result)
```

```
True
```

Question 5

Given an integer array nums, find three numbers whose product is maximum and return the maximum product.

Answer

In [9]:

```
def maximumProduct(nums):
    nums.sort()

# Maximum product will either be the product of three largest numbers# or the product of two smallest numbers and the largest number return max(nums[-1] * nums[-2] * nums[-3], nums[0] * nums[1] * nums[-1])
```

In [10]:

```
nums = [1, 2, 3] # Given result = maximumProduct(nums)
print(result)
```

```
6
```

Question 6

Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search the target in nums. If a target exists, then return its index. Otherwise, return -1.

You must write an algorithm with O(log n) runtime complexity.

Answer

In [11]:

```
def search(nums, target):
    left = 0
    right = len(nums) - 1

    while left <= right:
        mid = left + (right - left) // 2

        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else :
            right = mid - 1

    return -1
```

In [12]:

```
nums = [-1, 0, 3, 5, 9, 12] # Given target = 9 # Given result = search(nums, target)
print(result)
```

```
4
```

Question 7

An array is monotonic if it is either monotone increasing or monotone decreasing.

An array nums is monotone increasing if for all i <= j, nums[i] <= nums[j]. An array nums is monotone decreasing if for all i <= j, nums[i] >= nums[j].

Given an integer array nums, return true if the given array is monotonic, or false otherwise

Answer

In [13]:

```
def isMonotonic(nums):
    is_increasing = True is_decreasing = True for i in range(1, len(nums)):
        if nums[i] > nums[i - 1]:
            is_decreasing = False if nums[i] < nums[i - 1]:
            is_increasing = False return is_increasing or is_decreasing
```

In [14]:

```
nums = [1, 2, 2, 3] # Given result = isMonotonic(nums)
print(result)
```

```
True
```

Question 8

You are given an integer array nums and an integer k.

In one operation, you can choose any index i where 0 <= i < nums.length and change nums[i] to nums[i] + x where x is an integer from the range [-k, k]. You can apply this operation at most once for each index i.

The score of nums is the difference between the maximum and minimum elements in nums.

Return the minimum score of nums after applying the mentioned operation at most once for each index in it.

Answer

In [15]:

```
def minimumScore(nums, k):
    min_num = float('inf')
    max_num = float('-inf')

    for num in nums:
        min_num = min(min_num, num)
        max_num = max(max_num, num)

    if min_num + k >= max_num - k:
        return max_num - k - (min_num + k)
    else :
        return 0
```

In [16]:

```
nums = [1] # Given k = 0 # Given print(minimumScore(nums, k))
```

0