In [1]:

```
'''**Question 1**
Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1),

**Example 1:**
Input: nums = [1,4,3,2]
Output: 4

**Explanation:** All possible pairings (ignoring the ordering of elements) are:

1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4
So the maximum possible sum is 4'''


def arrayPairSum(nums):
    nums.sort()
    max_sum=0
    for i in range(0,len(nums),2):
        max_sum += min(nums[i],nums[i+1])
    return max_sum

nums = [1,4,3,2]
x = arrayPairSum(nums)
print(x)
```

```
4
```

In [1]:

```
'''Question 2
Alice has n candies, where the ith candy is of type candyType[i]. Alice noticed that she

The doctor advised Alice to only eat n / 2 of the candies she has (n is always even). Al

Given the integer array candyType of length n, return the maximum number of different ty

Example 1:
Input: candyType = [1,1,2,2,3,3]
Output: 3

Explanation: Alice can only eat 6 / 2 = 3 candies. Since there are only 3 types, she can


def distributeCandies(candyType):
        # Create a set/Hashset to find unique candy
        unique_candies = len(set(candyType))
        return min(unique_candies, len(candyType) // 2)

candyType = [1,1,2,2,3,3]
x = distributeCandies(candyType)
print(x)
```

```
3
```

In [2]:

```
'''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 s

Example 1:
Input: nums = [1,3,2,2,5,2,3,7]
Output: 5

Explanation: The longest harmonious subsequence is [3,2,2,2,3].'''

def findLHS(nums):
        freq = {}
        for num in nums:
            freq[num] = freq.get(num, 0) + 1
        max_length = 0
        for num in freq:
            if num + 1 in freq:
                max_length = max(max_length, freq[num] + freq[num + 1])
        return max_length

nums = [1,3,2,2,5,2,3,7]
x = findLHS(nums)
print(x)
```

```
5
```

In [3]:

```
'''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

Example 1:
Input: flowerbed = [1,0,0,0,1], n = 1
Output: true'''

def canPlaceFlowers(flowerbed, n):
        for i in range(len(flowerbed)):
            if flowerbed[i] == 0 and (i == 0 or flowerbed[i-1] == 0) and (i == len(flowe
                flowerbed[i] = 1
                n -= 1
        if n > 0:
            return False
        return True

flowerbed = [1,0,0,0,1]
n = 1
x = canPlaceFlowers(flowerbed , n)
print(x)
```

```
True
```

In [4]:

```
'''Question 5
Given an integer array nums, find three numbers whose product is maximum and return the

Example 1:
Input: nums = [1,2,3]
Output: 6'''

def maximumProduct(nums):
        nums.sort()
        return max(nums[0]*nums[1]*nums[-1],nums[-1]*nums[-2]*nums[-3])
        # here (nums[0]*nums[1]*nums[-1]) is used because without it for some example
        # code given wronge output. for example:- nums = [-100,-98,-1,2,3,4] in this cas
        # actual output is 39200 but the code give the result 24.

nums = [1,2,3]
x = maximumProduct(nums)
print(x)
```

```
6
```

In [5]:

```
'''Question 6
Given an array of integers nums which is sorted in ascending order, and an integer targe
write a function to search target in nums. If target exists, then return its index. Othe
return -1.

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

Input: nums = [-1,0,3,5,9,12], target = 9
Output: 4

Explanation: 9 exists in nums and its index is 4'''

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

        while left <= right:
            mid = (left + right) // 2
            if nums[mid] == target:
                return mid
            elif nums[mid] < target:
                left = mid + 1
            else:
                right = mid - 1
        return -1

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

```
4
```

In [6]:

```
'''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 num
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 other

Example 1:
Input: nums = [1,2,2,3]
Output: true'''

def isMonotonic(nums):
        increasing = decreasing = True

        for i in range(len(nums) - 1):
            if nums[i] > nums[i+1]:
                increasing = False
            if nums[i] < nums[i+1]:
                decreasing = False

        return increasing or decreasing

nums = [1,2,2,3]
x = isMonotonic(nums)
print(x)
```

```
True
```

In [7]:

```
'''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[

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

Example 1:
Input: nums = [1], k = 0
Output: 0

Explanation: The score is max(nums) - min(nums) = 1 - 1 = 0.'''

def smallestRangeI(nums , k):
        if len(nums)==1:
            return 0
        else:
            return max(0,max(nums)-min(nums)-2*k)

nums = [1]
k = 0
x = smallestRangeI(nums , k)
print(x)
```

```
0
```

In [ ]: