**Question 1** Given three integer arrays arr1, arr2 and arr3 **sorted** in **strictly increasing** order, return a sorted array of **only** the integers that appeared in **all** three arrays.

**Example 1:**

Input: arr1 = [1,2,3,4,5], arr2 = [1,2,5,7,9], arr3 = [1,3,4,5,8]

Output: [1,5]

**Solution :**

function findCommonElements(arr1, arr2, arr3):

result = []

i = j = k = 0

while i < length(arr1) and j < length(arr2) and k < length(arr3):

if arr1[i] == arr2[j] == arr3[k]:

result.append(arr1[i])

i += 1

j += 1

k += 1

else if arr1[i] <= arr2[j] and arr1[i] <= arr3[k]:

i += 1

else if arr2[j] <= arr1[i] and arr2[j] <= arr3[k]:

j += 1

else:

k += 1

return result

**Question 2**

Given two **0-indexed** integer arrays nums1 and nums2, return *a list* answer *of size* 2 *where:*

* answer[0] *is a list of all* ***distinct*** *integers in* nums1 *which are* ***not*** *present in* nums2\*.\*
* answer[1] *is a list of all* ***distinct*** *integers in* nums2 *which are* ***not*** *present in* nums1.

**Note** that the integers in the lists may be returned in **any** order.

**Example 1:**

**Input:** nums1 = [1,2,3], nums2 = [2,4,6]

**Output:** [[1,3],[4,6]]

**Explanation:**

For nums1, nums1[1] = 2 is present at index 0 of nums2, whereas nums1[0] = 1 and nums1[2] = 3 are not present in nums2. Therefore, answer[0] = [1,3].

For nums2, nums2[0] = 2 is present at index 1 of nums1, whereas nums2[1] = 4 and nums2[2] = 6 are not present in nums2. Therefore, answer[1] = [4,6].

**Solution :**

def findDistinctElements(nums1, nums2):

distinctNums1 = set(nums1)

distinctNums2 = set(nums2)

notInNums2 = list(distinctNums1 - distinctNums2)

notInNums1 = list(distinctNums2 - distinctNums1)

return [notInNums1, notInNums2]

**Question 3** Given a 2D integer array matrix, return *the* ***transpose*** *of* matrix.

The **transpose** of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.

**Example 1:**

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [[1,4,7],[2,5,8],[3,6,9]]

**Solution :**

def transpose(matrix):

rows = len(matrix)

cols = len(matrix[0])

# Create a new matrix with swapped dimensions

transposed = [[0 for \_ in range(rows)] for \_ in range(cols)]

# Populate the transposed matrix

for i in range(rows):

for j in range(cols):

transposed[j][i] = matrix[i][j]

return transposed

**Question 4** 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.

**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.

**Solution:**

def arrayPairSum(nums):

nums.sort()

max\_sum = 0

for i in range(0, len(nums), 2):

max\_sum += nums[i]

return max\_sum

**Question 5** You have n coins and you want to build a staircase with these coins. The staircase consists of k rows where the ith row has exactly i coins. The last row of the staircase **may be** incomplete.

Given the integer n, return *the number of* ***complete rows*** *of the staircase you will build*.

**Example 1:**

**Input:** n = 5

**Output:** 2

**Explanation:** Because the 3rd row is incomplete, we return 2.

**Solution :**

def arrangeCoins(n):

left = 0

right = n

while left <= right:

mid = left + (right - left) // 2

curr = (mid \* (mid + 1)) // 2

if curr == n:

return mid

if curr < n:

left = mid + 1

else:

right = mid - 1

return right

**Question 6** Given an integer array nums sorted in **non-decreasing** order, return *an array of* ***the squares of each number*** *sorted in non-decreasing order*.

**Example 1:**

Input: nums = [-4,-1,0,3,10]

Output: [0,1,9,16,100]

**Explanation:** After squaring, the array becomes [16,1,0,9,100]. After sorting, it becomes [0,1,9,16,100]

**Solution :**

def sortedSquares(nums):

squared = [num\*\*2 for num in nums]

squared.sort()

return squared

**Question 7** You are given an m x n matrix M initialized with all 0's and an array of operations ops, where ops[i] = [ai, bi] means M[x][y] should be incremented by one for all 0 <= x < ai and 0 <= y < bi.

Count and return *the number of maximum integers in the matrix after performing all the operations*

**Example 1:**

**Input:** m = 3, n = 3, ops = [[2,2],[3,3]]

**Output:** 4

**Explanation:** The maximum integer in M is 2, and there are four of it in M. So return 4

**Solution :**

def maxCount(m, n, ops):

minRow = m

minCol = n

for op in ops:

minRow = min(minRow, op[0])

minCol = min(minCol, op[1])

return minRow \* minCol

**Question 8**

Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn].

*Return the array in the form* [x1,y1,x2,y2,...,xn,yn].

**Example 1:**

**Input:** nums = [2,5,1,3,4,7], n = 3

**Output:** [2,3,5,4,1,7]

**Explanation:** Since x1=2, x2=5, x3=1, y1=3, y2=4, y3=7 then the answer is [2,3,5,4,1,7].

**Solution :**

def shuffle(nums, n):

result = []

for i in range(n):

result.append(nums[i])

result.append(nums[i + n])

return result