**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] Explanation: Only 1 and 5 appeared in the three arrays.**

**Program: def arraysIntersection(arr1, arr2, arr3):**

**i, j, k = 0, 0, 0**

**p, q, r = len(arr1), len(arr2), len(arr3)**

**res = []**

**while i < p and j < q and k < r**

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

**res.append(arr1[i])**

**i += 1**

**j += 1**

**k += 1**

**Continue**

**min\_next\_possible = max(arr1[i], arr2[j], arr3[k])**

**if arr1[i] < min\_next\_possible**

**i += 1**

**if arr2[j] < min\_next\_possible:**

**j += 1**

**if arr3[k] < min\_next\_possible**

**k += 1**

**return res**

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

**Output: [1,5]**

**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].**

**Program:   
class Solution:**

**def findDifference(self, nums1: List[int], nums2: List[int]) -> List[List[int]]:**

**set1 = set(nums1)**

**set2 = set (nums2)**

**return [set1 - set2, set2 - set1]**

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

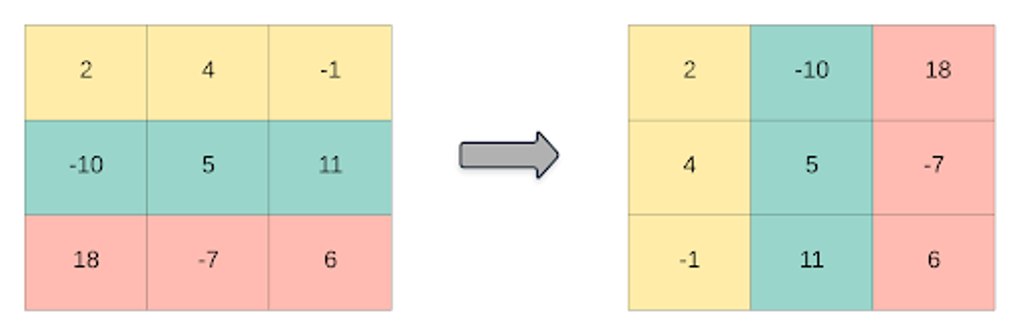
**OUTPUT: [[1,3],[4,6]]**

**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]]**

**Program:**

**class Solution:**

**def transpose(self, matrix: List[List[int]]) -> List[List[int]]:**

**row, col = len(matrix), len(matrix[0])**

**output = [[0]\*row for i in range(col)]**

**for r in range(row) :**

**for c in range(col) :**

**output[c][r] = matrix[r][c]**

**return output**

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

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

**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.**

**Program:**

**class Solution:**

**def arrayPairSum(self, nums: List[int]) -> int:**

**nums.sort()**

**sums = 0**

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

**sums+=nums[i]**

**return sums**

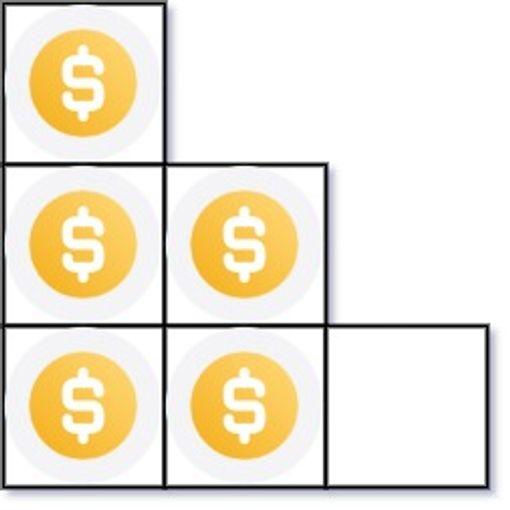
**INPUT: nums = [1,4,3,2]**

**OUTPUT : 4**

**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.**

**Program:**

**class Solution:**

**def arrangeCoins(self, n: int) -> int:**

**k = (int)(sqrt(2\*n))**

**sum = (int)(k\*(k+1)/2)**

**if sum > n: return k-1**

**return k**

**INPUT: n = 5**

**OUTPUT : 2**

**Question6. 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]**

**Program:   
 class Solution:**

**def sortedSquares(self, nums: List[int]) -> List[int]:**

**res = []**

**left = 0**

**right = len(nums) - 1**

**while left <= right:**

**if abs(nums[left]) > abs(nums[right]):**

**res.append(nums[left] \*\* 2)**

**left += 1**

**else:**

**res.append(nums[right] \*\* 2)**

**right -= 1**

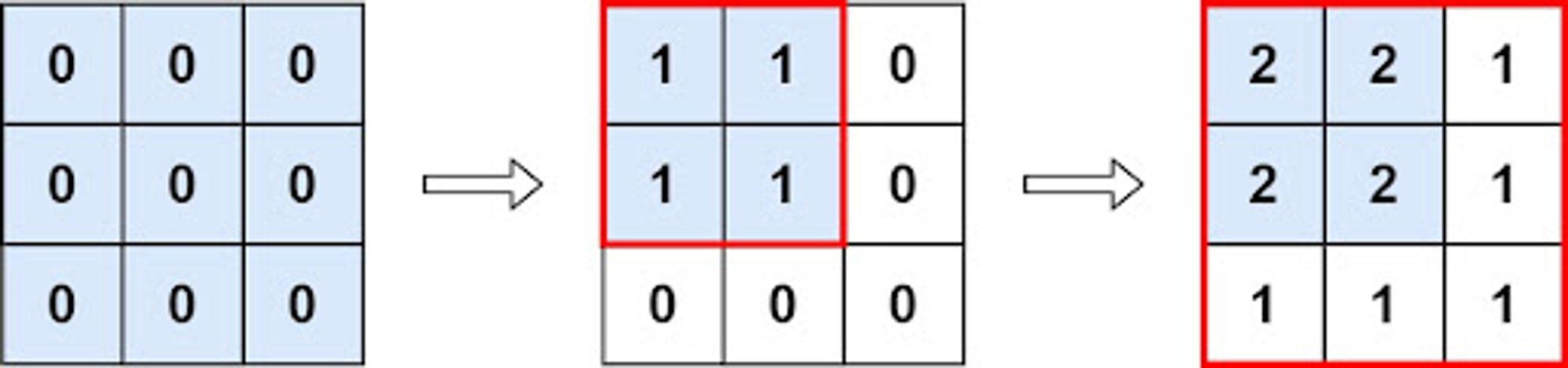
**return res[::-1]**

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

**OUTPUT: [0,1,9,16,100]**

**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.**

**program :   
 class Solution:**

**def maxCount(self, m: int, n: int, ops: List[List[int]]) -> int:**

**mr = m**

**mc = n**

**for i in range(len(ops)):**

**mr = min(mr, ops[i][0])**

**mc = min (mc, ops[i][1])**

**return mr\*mc**

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

**OUTPUT : 4**

**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].

**Program:**

class Solution:

def shuffle(self, nums: List[int], n: int) -> List[int]:

ans = []

for i in range(n):

ans.append(nums[i])

ans.append(nums[i+n])

return ans

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

Output = [2,3,5,4,1,7]