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.

```
def arraysIntersection(arr1, arr2, arr3):
    p1 = p2 = p3 = 0
    result = []

    while p1 < len(arr1) and p2 < len(arr2) and p3 < len(arr3):
        if arr1[p1] == arr2[p2] == arr3[p3]:
            result.append(arr1[p1])
            p1 += 1
            p2 += 1
            p3 += 1
        elif arr1[p1] <= arr2[p2] and arr1[p1] <= arr3[p3]:
            p1 += 1
        elif arr2[p2] <= arr1[p1] and arr2[p2] <= arr3[p3]:
            p2 += 1
        else:
            p3 += 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.

```
def findDisappearedNumbers(nums1, nums2):
    set1 = set(nums1)
    set2 = set(nums2)
    diff1 = list(set1 - set2)
    diff2 = list(set2 - set1)
    return [diff1, diff2]
```

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]]

```
def transpose(matrix):
    rows = len(matrix)
    columns = len(matrix[0])
    transpose = [[0] * rows for _ in range(columns)]

    for i in range(rows):
        for j in range(columns):
            transpose[j][i] = matrix[i][j]
    return transpose
```

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*.

```
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*.

```
def arrangeCoins(n):
    left = 0
    right = n


    while left <= right:
        mid = (left + right) // 2
        total_coins = (mid * (mid + 1)) // 2
        if total_coins <= 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*.

```
def sortedSquares(nums):
    squared_nums = []
    for num in nums:
        squared_nums.append(num * num)
    squared_nums.sort()
    return squared_nums
```

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*.

```
def maxCount(m, n, ops):
    min_row = float('inf')
    min_col = float('inf')
    for op in ops:
        min_row = min(min_row, op[0])
        min_col = min(min_col, op[1])
    max_count = min_row * min_col
    return max_count
```

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].

```
def shuffle(nums):
    n = len(nums) // 2
    result = []
    for i in range(n):
        result.append(nums[i])
        result.append(nums[i+n])
    return result
```