}

Assignment Questions 4

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.

Solution:-

```
class Solution {
    public List<Integer> arraysIntersection(int[] arr1, int[] arr2, int[] arr3) {
        List<Integer> rst = new LinkedList<>();
        int i = arr1.length - 1, j = arr2.length - 1, k = arr3.length - 1;

        while (i >= 0 && j >= 0 && k >= 0) {
            if (arr1[i] == arr2[j] && arr2[j] == arr3[k]) {
                if (rst.isEmpty() || arr1[i] != rst.get(rst.size() - 1)) rst.add(0, arr1[i]);
                i--;
                j--;
                k--;
            } else if (arr2[j] < arr3[k]) k--;
            else if (arr1[i] < arr2[j]) j--;
            else i--;
        }

        return rst;
    }
}
```

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:-

```
class Solution {
    public List<List<Integer>> findDifference(int[] nums1, int[] nums2) {

     HashSet<Integer> set1=new HashSet<Integer>();
      HashSet<Integer> set2=new HashSet<Integer>();

     for(int ele: nums1){
         set1.add(ele);
     }

     for(int ele:nums2){
         set2.add(ele);
     }


     List<List<Integer>> list=new ArrayList<>();

      ArrayList<Integer> l1=new ArrayList<>();

      ArrayList<Integer> l2=new ArrayList<>();
     for(int ele:set2){

         if(set1.contains(ele)==false){
           l1.add(ele);
         }
     }


       for(int ele:set1){

         if(set2.contains(ele)==false){
           l2.add(ele);
         }
     }


       list.add(l2);
       list.add(l1);
       return list;
    }
}
```

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:-

```
class Solution {
    public int[][] transpose(int[][] matrix) {
        int[][] ans = new int[matrix[0].length][matrix.length];
        int row = 0;
        int col = 0;

        for(int i = 0; i < matrix.length; i++) {
            for(int j = 0; j < matrix[0].length; j++) {
                ans[row][col] = matrix[i][j];

                row++;
                if(row % ans.length == 0) {
                    row = 0;
                    col++;
                }
            }
        }
        return ans;
    }
}
```

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:-

```
class Solution {
    public int arrayPairSum(int[] nums)
    {
        Arrays.sort(nums);
     int sum=0;
     for(int i=0;i<nums.length;i=i+2)
     {
         sum=sum+nums[i];
     }
     return 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:

nput: n = 5

Output: 2

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

Solution:-

```
class Solution {
    public int arrangeCoins(int n) {
        long i=0;
        for(i=1;i*(i+1)/2<=n;i++);
        return (int)i-1;
    }
}
```

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:-

```
class Solution {
    public int[] sortedSquares(int[] A) {
        int n = A.length;
        int[] result = new int[n];
        int i = 0, j = n - 1;
        for (int p = n - 1; p >= 0; p--) {
            if (Math.abs(A[i]) > Math.abs(A[j])) {
                result[p] = A[i] * A[i];
                i++;
            } else {
                result[p] = A[j] * A[j];
                j--;
            }
        }
        return result;
    }
}
```

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:-

```
class Solution {
    public int maxCount(int m, int n, int[][] ops) {
int minM = m;
int minN = n;
for( int i = 0; i< ops.length; i++)
{
if(ops[i][0]!=0&&ops[i][0]<minM){
minM = ops[i][0];
}
if(ops[i][1]!=0&&ops[i][1]<minN){
minN = ops[i][1];
}
}
return minM*minN;
}
}
```

Question 8

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

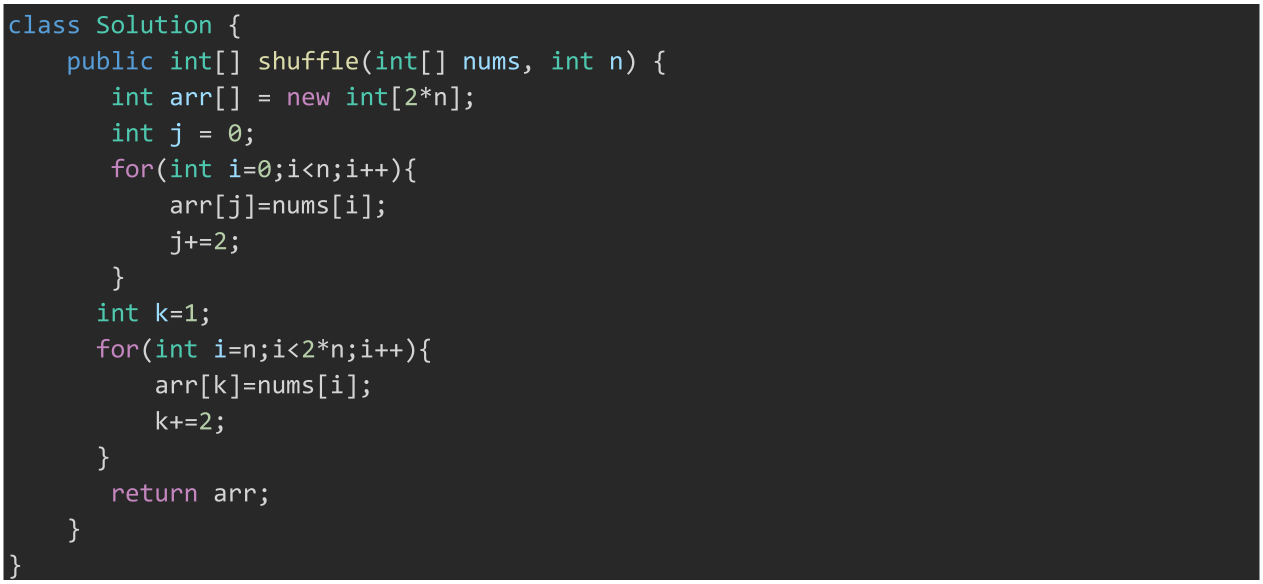
Return the array in the form [x1,y1,x2,y2,...,xn,yn].

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]



```
class Solution {
    public int[] shuffle(int[] nums, int n) {
       int arr[] = new int[2*n];
       int j = 0;
       for(int i=0;i<n;i++){
           arr[j]=nums[i];
           j+=2;
       }
      int k=1;
      for(int i=n;i<2*n;i++){
          arr[k]=nums[i];
          k+=2;
      }
       return arr;
    }
}
```