Assignment 4 (2D Arrays)

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.
Solution:

```
void findCommon(int ar1[], int ar2[], int ar3[])
   {

      int i = 0, j = 0, k = 0;


      while (i < ar1.length && j < ar2.length
           && k < ar3.length) {

         if (ar1[i] == ar2[j] && ar2[j] == ar3[k]) {
            System.out.print(ar1[i] + " ");
            i++;
            j++;
            k++;
         }


         else if (ar1[i] < ar2[j])
            i++;


         else if (ar2[j] < ar3[k])
            j++;


         else
            k++;
      }
   }
```

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.

Solution:

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



    HashSet<Integer> set1=new HashSet();

     HashSet<Integer> set2=new HashSet();


    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.

Solution:

```
 public int[][] transpose(int[][] matrix) {

        int[][] answer = new int[matrix[0].length][matrix.length];

        for(int i=0; i < matrix.length; i++){

            for (int j = 0; j < matrix[0].length; j++){
```

```
            answer[j][i] = matrix[i][j];
          }
        }
        return answer;
    }
```

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*.

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*.

Solution:

```
 public int arrangeCoins(int n) {

        long s=1,e=n,mid,ans=0;

        while(s<=e){

            mid = s +(e-s)/2;

            if((mid*(mid+1))/2<=n){

                ans=mid;

                s=mid+1;

            }else{

                e=mid-1;

            }

        }

        return (int)ans;

    }
```

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*.

Solution:

```
public int[] sortedSquares(int[] nums) {

        int n = nums.length;

        int[] result = new int[n];

        int left = 0;

        int right = n - 1;

        int i = n - 1;


        while (left <= right) {
            int leftSquare = nums[left] * nums[left];

            int rightSquare = nums[right] * nums[right];


            if (leftSquare > rightSquare) {

                result[i] = leftSquare;

                left++;

            } else {

                result[i] = rightSquare;

                right--;

            }


            i--;

        }


        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*

Solution:

```
public int maxCount(int m, int n, vector<vector<int>>& ops) {

        for (auto op : ops){

            m = min(m, op[0]);

            n = min(n, op[1]);
        }
        return m * n;
    }
```

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].

Solution:

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