DSA ASSIGNMENT-5(2D-ARRAY)

Solution 1:

```
class Solution {
    public int[][] construct2DArray(int[] original, int m, int n) {
    int [][] arr=new int [m][n];
    if(original.length!=m*n)return new int[0][0];
    int ix=0;
         for(int i=0;i<arr.length;i++) {
         for(int j=0;j<arr[0].length;j++){
         arr[i][j]=original[ix++];

         }

     }
     return arr;
    }
}
```

Solution 2:

```
class Solution {
    public int arrangeCoins(int n) {
         int ans = 1;
    while(n > 0){
      ans++;
      n = n-ans;
    }
    return ans-1;
    }
}
```

Solution 3:

```
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;
    }
}
```

Solution 4:

```
class Solution {
    public List<List<Integer>> findDifference(int[] nums1, int[] nums2) {
        List<Integer> list1=toListValue(nums1);
        List<Integer> list2=toListValue(nums2);
        Set<Integer> list3=new HashSet<Integer>();
        for(int a:list1){
            if(!list2.contains(a)) list3.add(a);
        }
        Set<Integer> list4=new HashSet<Integer>();
        for(int b:list2){
            if(!list1.contains(b)) list4.add(b);
        }
        List<List<Integer>> arrayList=new ArrayList<List<Integer>>();
        arrayList.add(new ArrayList<Integer>(list3));
        arrayList.add(new ArrayList<Integer>(list4));
        return arrayList;
    }
    public List<Integer> toListValue(int[] nums1){
        return Arrays.stream(nums1).boxed().collect(Collectors.toList());
    }
}
```

Solution 5:

```
class Solution {
    public int findTheDistanceValue(int[] arr1, int[] arr2, int d) {
        int count=0;
        int x=0;
        for(int i=0;i<arr1.length;i++){
            x=0;
            for(int j=0;j<arr2.length;j++){
                int diff=Math.abs(arr1[i]-arr2[j]);
                if(diff<=d){
                    j=arr2.length;
                }
                else{
                    x++;
                }
            }
            if(x==arr2.length){
                count++;
            }
        }
        return count;
    }
}
```

Solution 6:

```
lass Solution {
    public List<Integer> findDuplicates(int[] nums) {
        HashSet<Integer> states = new HashSet<Integer>();
        List<Integer> result = new ArrayList<>();


        for (int num : nums) {
            if (states.contains(num)) {
                result.add(num);
            } else {
                states.add(num);
            }
        }

        return result;
    }
}
```

Solution 7:

```
class Solution {
    public int findMin(int[] nums) {
        int n=nums.length;
        int st=0;
        int end=n-1;
        int ans=-1;
        while(st<=end)
        {
            int mid=st+(end-st)/2;
            if(nums[mid]<=nums[n-1])
            {
                end=mid-1;
                ans=nums[mid];
            }else{
                st=mid+1;
            }
        }
        return ans;
    }
}
```

Solution 8:

```
class Solution {
    public int[] findOriginalArray(int[] nums) {
       int[] vacarr = new int[0];
        // when we need to return vacant array
        int n= nums.length;
            // size of the array
        if(n%2!=0)
        {
            return vacarr;
            // when we will have odd number of integer in our input(double array
can't be in odd number)

        }
        HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();
             // for storing the frequencies of each input
        int[] ans = new int[(nums.length/2)];
        // answer storing array

        for(int i=0;i<n;i++)
        {
            hm.put(nums[i], hm.getOrDefault(nums[i],0)+1);
            // storing the frequencies
        }
        int temp = 0;

        Arrays.sort(nums);
        // sorting in increasing order
        for(int i: nums)
        {

            if(hm.get(i)<=0)
            {
              // if we have already decreased it's value when we were checking y/2
value, like 2,4 we will remove 4 also when we will check 2 but our iteration will
come again on 4.

                continue;
            }

            if(hm.getOrDefault(2*i,0)<=0)
            {   // if we have y but not y*2 return vacant array
```

```
                return vacarr;
            }
            ans[temp++] = i;
             // if we have both y and y*2, store in our ans array
           // decrease the frequency of y and y*2
            hm.put(i, hm.get(i)-1);
            hm.put(2*i, hm.get(2*i)-1);
        }

        return ans;
    }
}
```
