Assignment Questions 6

Question 1

A permutation perm of n + 1 integers of all the integers in the range [0, n] can be represented as a string s of length n where:

- s[i] == 'I' if perm[i] < perm[i + 1], and
- s[i] == 'D' if perm[i] > perm[i + 1].

Given a string s, reconstruct the permutation perm and return it. If there are multiple valid permutations perm, return **any of them**.

Example 1:

Input: s = "IDID"

Output:

[0,4,1,3,2]

Solution:-

```java
class Solution {
    public int[] diStringMatch(String s) {
        int low = 0;
        int size = s.length();
        int high = size;
        int[] output = new int[size+1];
        char[] arr = s.toCharArray();
        for(int i=0;i<size;i++){
            if(arr[i] == 'I'){
                output[i] = low;
                low++;
            }
            else{
                output[i] = high;
                high--;
            }
        }
        output[size] = high;
        return output;
    }
}
```

Question 2

You are given an m x n integer matrix matrix with the following two properties:

- Each row is sorted in non-decreasing order.
- The first integer of each row is greater than the last integer of the previous row.

Given an integer target, return true *if* target *is in* matrix *or* false *otherwise*.

You must write a solution in O(log(m * n)) time complexity.

Example 1:

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3

Output: true

Solution:-
```
class Solution {

    public boolean searchMatrix(int[][] matrix, int target) {
       int i = 0 , j = matrix[0].length - 1;
        while(i < matrix.length && j >= 0){
            if(matrix[i][j] > target){
                j--;
            }else if(matrix[i][j] < target){
                i++;
            }else{
                return true;
            }
        }
        return false;
    }
}
```

Question 3

Given an array of integers arr, return *true if and only if it is a valid mountain array*.

Recall that arr is a mountain array if and only if:

- arr.length >= 3
- There exists some i with 0 < i < arr.length - 1 such that:
  - arr[0] < arr[1] < ... < arr[i - 1] < arr[i]
  - arr[i] > arr[i + 1] > ... > arr[arr.length - 1] </aside>
- **Example 1:**
- **Input:** arr = [2,1]
- **Output:**

- false

Solution:-

```
class Solution {
    public boolean validMountainArray(int[] arr) {
        int f1=0,f2=0,f3=0,false_Flag=0;
        for(int i=0;i<arr.length-1 && arr.length >= 3;i++){
            //at first strictly incre then equal or decre
            if(arr[i] < arr[i+1]){
                if(f2==1 ||  f3==1) false_Flag=1;
                f1=1;
            }
            else if(arr[i] == arr[i+1]){
                if(f1==0 || f3==1) false_Flag=1;
                f2=1;

            }
            else if(arr[i] > arr[i+1]){
                if(f1==0) false_Flag=1;
                f3=1;
            }
        }
        if( f1==1 && f3==1 && false_Flag==0) return true;
        return false;
        // return false;
    }
}
```

Question 4

Given a binary array nums, return *the maximum length of a contiguous subarray with an equal number of* 0 *and* 1.

Example 1:

Input: nums = [0,1]

Output: 2

Explanation:

[0, 1] is the longest contiguous subarray with an equal number of 0 and 1.

Solution:-

```
class Solution {
    public int findMaxLength(int[] nums) {
        int count = 0;
        for (int i = 0; i < nums.length; i++) {
            int zeros = 0, ones = 0;
            for (int j = i; j < nums.length; j++) {
                if (nums[j] == 0) {
                    zeros++;
                } else {
                    ones++;
                }
                if (zeros == ones) {
                    count = Math.max(count, j - i + 1);
                }
            }
        }
        return count;
    }
}
```

Question 5

The **product sum** of two equal-length arrays a and b is equal to the sum of a[i] * b[i] for all 0 <= i < a.length (**0-indexed**).

- For example, if a = [1,2,3,4] and b = [5,2,3,1], the **product sum** would be 15 + 22 + 33 + 41 = 22.

Given two arrays nums1 and nums2 of length n, return *the **minimum product sum** if you are allowed to **rearrange** the **order** of the elements in* nums1.

Example 1:

Input: nums1 = [5,3,4,2], nums2 = [4,2,2,5]

Output: 40

Explanation:

We can rearrange nums1 to become [3,5,4,2]. The product sum of [3,5,4,2] and [4,2,2,5] is 34 + 52 + 42 + 25 = 40.

Solution:-

```
class Solution {

   public int minProductSum(int[] nums1, int[] nums2) {

       Arrays.sort(nums1);

       Arrays.sort(nums2);

       int sum = 0;

       int length = nums1.length;

       for (int i = 0; i < length; i++)

           sum += nums1[i] * nums2[length - 1 - i];

       return sum;

   }

}
```

Question 6

An integer array original is transformed into a **doubled** array changed by appending **twice the value** of every element in original, and then randomly **shuffling** the resulting array.

Given an array changed, return original *if* changed *is a **doubled** array. If* changed *is not a **doubled** array, return an empty array. The elements in* original *may be returned in **any** order.*

Example 1:

Input: changed = [1,3,4,2,6,8]

Output: [1,3,4]

Explanation: One possible original array could be [1,3,4]:

- Twice the value of 1 is 1 * 2 = 2.
- Twice the value of 3 is 3 * 2 = 6.
- Twice the value of 4 is 4 * 2 = 8.

Other original arrays could be [4,3,1] or [3,1,4]

Solution:-

```
class Solution {
    public int[] findOriginalArray(int[] changed) {

        int len = changed.length;
        if((len&1) != 0) return new int[0];

        // Sorting the array
        Arrays.sort(changed);

        // Store frequencies in map
        Map<Integer,Integer> map = new HashMap<>();
        for(int e : changed) map.put(e,map.getOrDefault(e,0)+1);

        int[] res = new int[len/2];
        int k = 0;
        for(int i=0; i<len; i++){
            int ele = changed[i];

            // if map contains 'ele'
            if(map.containsKey(ele)){

                // if map contains 'ele*2'
                if(map.containsKey(ele*2)){
                    res[k++] = ele;

                    // reduce frequency of 'ele' and 'ele*2'
                    map.put(ele,map.get(ele)-1);
                    map.put(ele*2,map.get(ele*2)-1);

                    // if freq of any key becomes <=0, remove it from map
                    if(map.get(ele)<=0) map.remove(ele);
                    if(map.containsKey(ele*2) && map.get(ele*2)<=0)
map.remove(ele*2);
                }
                else return new int[0];
            }

        }
        return res;
    }
}
```

Question 7

Given a positive integer n, generate an n x n matrix filled with elements from 1 to n2 in spiral order.

Example 1:

Input: n = 3

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

Solution:-

```
class Solution {
    public int[][] generateMatrix(int n) {


        int [][] arr = new int[n][n];

        int i = 0;      //starting row index
        int j = 0;      //starting col index

        int k = n;    //ending row index
        int l = n;    //ending col index

        int count = 0;

        boolean flag = true;

        while(flag) {

            flag = false;


            //--------------LEFT --->> RIGHT------------//
            while(j < l)
            {
              arr[i][j] = ++count;
              j++;
                flag = true;
            }
            j--;
            i++;


            //-------------TOP --->> BOTTOM-------------//
            while (i < k) {
                arr[i][j] = ++count;
                i++;
                flag = true;
            }
            j--;
            i--;
            k--;

            //-----------RIGHT --->> LEFT---------//
            while (j >= n - l) {
                arr[i][j] = ++count;
                j--;
                flag = true;
            }
            j++;
            l--;
            i--;

            //----------BOTTOM --->> TOP-------------//
            while (i >= n - k) {
                arr[i][j] = ++count;
                i--;
                flag = true;
            }
            i++;
            j++;
        }


        return arr;
    }
}
```

Question 8

Given two sparse matrices mat1 of size m x k and mat2 of size k x n, return the result of mat1 x mat2. You may assume that multiplication is always possible.

Example 1:

Input: mat1 = [[1,0,0],[-1,0,3]], mat2 = [[7,0,0],[0,0,0],[0,0,1]]

Output:

[[7,0,0],[-7,0,3]]

Solution:-

```
class Solution {

    public int[][] multiply(int[][] mat1, int[][] mat2) {

        int r1 = mat1.length, c1 = mat1[0].length, c2 = mat2[0].length;

        int[][] res = new int[r1][c2];

        Map<Integer, List<Integer>> mp = new HashMap<>();

        for (int i = 0; i < r1; ++i) {

            for (int j = 0; j < c1; ++j) {

                if (mat1[i][j] != 0) {

                    mp.computeIfAbsent(i, k -> new ArrayList<>()).add(j);

                }

            }

        }

        for (int i = 0; i < r1; ++i) {

            for (int j = 0; j < c2; ++j) {

                if (mp.containsKey(i)) {

                    for (int k : mp.get(i)) {

                        res[i][j] += mat1[i][k] * mat2[k][j];

                    }

                }

            }

        }

        return res;

    }}
```