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]

code:-

```
public int[] diStringMatch(String s) {
        int[] result = new int[s.length() + 1];
        int end = result.length - 1, start = 0;
        int index = 0;
        while(index < s.length()){
            if(s.charAt(index) == 'I'){
                result[index] = start;
                start++;
            }else{
                result[index] = end;
                end--;
            }
            index++;
        }
        result[result.length - 1] = start;
        return result;
    }
```

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

code:-

```
public boolean searchMatrix(int[][] matrix, int target) {

     int n=matrix[0].length-1;
     for(int i=0;i<matrix.length;i++){
         if(target<=matrix[i][n]){
             int s=0;
             int e=n;
             while(s<=e){
                 int m=(s+e)/2;
                 if(matrix[i][m]==target) return true;
                 else if(matrix[i][m]<target){
                     s=m+1;
                 }else{
                     e=m-1;
                 }

             }


         }
     }
     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]

Example 1:
Input: arr = [2,1]
Output: false

code:-

```
public boolean validMountainArray(int[] arr) {
        int n = arr.length;
        if (n < 3) return false;
        int i = 0;
        while (i < n-1 && arr[i] < arr[i+1]) {
            i++;
        }
        if (i == 0 || i == n-1) return false;

        while (i < n-1 && arr[i] > arr[i+1]) {
            i++;
        }
        return i == n-1;
    }
```

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.

code:-

```
public int findMaxLength(int[] nums) {
        HashMap<Integer,Integer>  hmap = new HashMap<>();
        int maxLength = 0;
        int sum = 0;
        for(int i = 0 ; i < nums.length; i++) {
            sum += (nums[i] == 0 ? -1 : nums[i]);
            if(sum == 0)
                maxLength = i+1;
            else if(hmap.containsKey(sum)) {
                maxLength = Math.max(maxLength,i - hmap.get(sum));
            }
            else
                hmap.put(sum,i);
        }
        return maxLength;
    }
```

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 1*5 + 2*2 + 3*3 + 4*1 = 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 3*4 + 5*2 + 4*2 + 2*5 = 40.

code:-

```
public int minProductSum(int[] nums1, int[] nums2) {
        int ans = 0;
        // Sort nums1 and nums2 in ascending order.
        Arrays.sort(nums2);
        Arrays.sort(nums1);

        int i = 0;
        int j = nums2.length-1;

        while(i < nums1.length && j >= 0)
        {
            ans += nums1[i] * nums2[j];
            i++;
            j--;
        }

        return ans;
    }
```

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].

code:-

```
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]]

code:-

```
public int[][] generateMatrix(int n) {
        int [][] a = new int[n][n];
        int l=0,t=0,b=n-1,r=n-1,v=1;
        while(t<=b||l<=r){
         if(t<=b){
             for(int i=l;i<=r;i++)
                a[t][i]=v++;
             t++;
         }
         if(l<=r){
             for(int i=t;i<=b;i++)
                 a[i][r]=v++;
             r--;
         }
         if(t<=b){
             for(int i=r;i>=l;i--)
             a[b][i]=v++;
             b--;
         }
         if(t<=b){
             for(int i=b;i>=t;i--)
             a[i][l]=v++;
         }
         l++;
        }
        return a;
    }
```

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]]

code:-

```
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;
    }
```