DSA ASSIGNMENT-6(STRING)

Solution 1:

```java
class Solution {

    public int[] diStringMatch(String s) {
        int n = s.length();
        int[] perm = new int[n + 1];

        int start = 0;
        int end = n;
        for (int i = 0; i < n; i++) {
            if (s.charAt(i) == 'I') {
                perm[i] = start++;
            } else {
                perm[i] = end--;
            }
        }
        perm[n] = start;

        return perm;
    }
}
```

Solution 2:

```java
class Solution {
    public boolean searchMatrix(int[][] arr, int target) {
         int n=arr.length, m=arr[0].length;
        int st=0,end=(n*m)-1;
        while (st <= end) {
            int mid = st+(end-st)/2;
            int midElem=arr[mid/m][mid%m];
            if (midElem == target) {
                return true;
            }else if (target < midElem) {
                end=mid-1;
            }else{
                st=mid+1;
            }
        }
        return false;
    }
}
```

Solution 3:

```
class Solution {
    public boolean validMountainArray(int[] arr) {
      //if size is < 2 then it not mountain
    if(arr.length<3) return false;


    int topidx=0;
    int top=0;

    //find max value and that index
    for(int i=0;i<arr.length;i++)
        {
            if(arr[i]>top)
            {
            top = arr[i];
            topidx=i;
            }
        }

    //check that one side mountain or not .
    if(top==arr[arr.length-1] || top==arr[0]) return false;

            //check perfact mountain or not
            int i=0;
            while(i<topidx)
            {
                if(arr[i] >= arr[i+1]) return false;
                i++;

            }

            while(topidx<arr.length-1)
            {
                if(arr[topidx] <= arr[topidx+1]) return false;
                topidx++;

            }
            return true;
    }
}
```

Solution 4:

```
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;
    }
}
```

Solution 5:

```
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;
    }
}
```

Solution 6:

```
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;
    }
}
```

Solution 7:

```
public class Solution {
    public int[][] generateMatrix(int n) {
        // Start typing your Java solution below
        // DO NOT write main() function
        if(n<=0) return new int[0][];
        int[][] result=new int[n][n];
        int xBeg=0,xEnd=n-1;
        int yBeg=0,yEnd=n-1;
        int cur=1;
        while(true){
            for(int i=yBeg;i<=yEnd;i++) result[xBeg][i]=cur++;
            if(++xBeg>xEnd) break;
            for(int i=xBeg;i<=xEnd;i++) result[i][yEnd]=cur++;
            if(--yEnd<yBeg) break;
            for(int i=yEnd;i>=yBeg;i--) result[xEnd][i]=cur++;
            if(--xEnd<xBeg) break;
            for(int i=xEnd;i>=xBeg;i--) result[i][yBeg]=cur++;
            if(++yBeg>yEnd) break;
        }
        return result;
    }
}
```

Solution 8:

```
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;
    }
}
```