**MCCP 74 HARD**

**Q1.**

1. **Develop Program to find the initials of a name.**

import java.util.Scanner;

public class Initials {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String name = scan.nextLine();

        System.out.println(findInitials(name));

    }

    public static String findInitials(String name) {

        String[] words = name.split(" ");

        StringBuilder initials = new StringBuilder();

        for (String word : words) {

            initials.append(word.charAt(0)+" ");

        }

        return initials.toString().toUpperCase();

    }

}

**Input :** Jude Law

**Output :** J L

**Input :** abhishek kumar singh

**Output** : A K S

# Find GCD of most occurring and least occurring elements of given Array

    import java.util.HashMap;

    import java.util.Map;

    import java.util.Scanner;

    public class FreqGCD {

        public static void main(String[] args) {

            Scanner scan = new Scanner(System.in);

            int n = scan.nextInt();

            int[] arr = new int[n];

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

                arr[i] = scan.nextInt();

            countFreq(arr, n);

        }

        static void countFreq(int arr[], int n) {

            HashMap<Integer, Integer> hm = new HashMap<>();

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

                hm.put(arr[i], hm.getOrDefault(arr[i], 0) + 1);

            }

            int maxCount = -1, minCount = Integer.MAX\_VALUE, maxEle = -1, minEle = -1;

            for (Map.Entry<Integer, Integer> val : hm.entrySet()) {

                if (val.getValue() > maxCount) {

                    maxCount = val.getValue();

                    maxEle = val.getKey();

                }

                if (val.getValue() < minCount) {

                    minCount = val.getValue();

                    minEle = val.getKey();

                }

            }

            System.out.println(gcd(maxEle, minEle));

        }

        static int gcd(int a, int b) {

            if (b == 0) {

                return a;

            }

            return gcd(b, a % b);

        }

    }

**Input :** *arr[] = {2, 2, 4, 4, 5, 5, 6, 6, 6, 6}*

**Output :** 2

**Input :** *arr[] = {3, 2, 2, 44, 44, 44, 44}*

**Output** : 1

**Q2.**

# Check Whether a number is Duck Number or not

import java.util.Scanner;

public class DuckNumber {

    public static void main(String[] args) {

        boolean flag=false;

        Scanner scan = new Scanner(System.in);

        String s = scan.nextLine();

        int n = Integer.parseInt(s);

        s = Integer.toString(n);

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

            if (s.charAt(i) == '0') {

                flag = true;

                break;

            }

        }

        if (flag)

            System.out.println("It is a Duck Number.");

        else

            System.out.println("It is not a Duck Number.");

    }

}

**Input :** *707069*

**Output :** *It is a duck number.*

**Input :** *02364*

**Output** : *It is not a duck number.*

# Find Kth most occurring element in an Array

import java.util.LinkedHashMap;

import java.util.Scanner;

import java.util.TreeSet;

public class KthMostOccuringEle {

    static LinkedHashMap<Integer, Integer> hm = new LinkedHashMap<>();

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int k = scan.nextInt();

        int n = scan.nextInt();

        int[] arr = new int[n];

        for (int i = 0; i < n; i++) {

            arr[i] = scan.nextInt();

            hm.put(arr[i], hm.getOrDefault(arr[i], 0) + 1);

        }

        TreeSet<Integer> ts = new TreeSet<>();

        for (int i : hm.keySet()) {

            ts.add(hm.get(i));

        }

        int x = ts.size();

        k = x - k;

        int a[] = new int[x];

        int i = 0;

        for (Integer v : ts) {

            a[i] = v;

            i++;

        }

        for (int j : hm.keySet()) {

            if(hm.get(j) == a[k]) System.out.print(j+" ");

        }

    }

}

**Input :** *arr[] = {1, 2, 2, 2, 4, 4, 4, 5, 5, 5, 5, 5, 7, 7, 8, 8, 8, 8}, K = 1*

**Output :** *5*

**Input :** *arr[] = {1, 2, 2, 2, 4, 4, 4, 4, 5, 5, 5, 5, 5, 7, 7, 8, 8, 8, 8}, K = 3*

**Output** : *2*

**Q3.**

# Removing punctuations from a given string

import java.util.Scanner;

public class RemovePunc {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String s = scan.nextLine();

        StringBuilder sb = new StringBuilder();

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

            char c = s.charAt(i);

            if (c >= 'A' && c <= 'Z' || c >= 'a' && c <= 'z'|| c==' ')

                sb.append(c);

        }

        System.out.println(sb.toString());

    }

}

**Input :** %welcome' to @geeksforgeek<s

**Output :** welcome to geeksforgeeks

**Input :** Hello!!!, he said ---and went.

**Output** : Hello he said and went

# Print all array elements occurring at least M times

import java.util.LinkedHashMap;

import java.util.Scanner;

public class ElementsOccurMTimes {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int m = scan.nextInt();

        LinkedHashMap<Integer, Integer> hm = new LinkedHashMap<>();

        for (int i = 0; i < n; i++) {

            int e = scan.nextInt();

            hm.put(e, hm.getOrDefault(e, 0) + 1);

        }

        for (int i : hm.keySet()) if (hm.get(i) >= m) System.out.print(i + " ");

    }

}

***Input:****arr[] = {2, 3, 2, 2, 3, 5, 6, 3}, M = 2****Output:****2 3*

***Input:****arr[] = {1, 32, 2, 1, 33, 5, 1, 5}, M = 2****Output:****1 5*

# Q4.

# A) Check whether the given number is Euclid Number or not

import java.util.Scanner;

public class EuclidNumber {

    static boolean isPrime(int n) {

        for (int i = 2; i <= Math.sqrt(n); i++) {

            if (n % i == 0)

                return false;

        }

        return true;

    }

    static boolean euclidOrNot(int n) {

        int p = 1;

        for (int i = 2; i < n; i++) {

            if (p >= n)

                return false;

            if (isPrime(i)) {

                p \*= i;

                System.out.println(i+"---"+p);

                if (p + 1 == n)

                    return true;

            }

        }

        return false;

    }

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        boolean res;

        res = euclidOrNot(n);

        System.out.println(res);

    }

}

**Input :** 31

**Output :** true

**Input :** 43

**Output** : false

# B) Program to print reverse character bridge pattern

---TO BE DONE---

**Q5.**

# Sum of all Perfect numbers lying in the range [L, R]

import java.util.Scanner;

public class SumPerfectNumInRange {

    static boolean isPerfect(int n) {

        int p=0;

        for (int i = 1; i < n; i++)

            if (n % i == 0) {

                p += i;

            }

        if (p == n)

            return true;

        return false;

    }

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int l = scan.nextInt(), r = scan.nextInt();

        int s = 0;

        for (int i = l; i <= r; i++)

            if (isPerfect(i)) s += i;

        System.out.println(s);

    }

}

**Input :** *L = 6, R = 10*

**Output :** 6

**Input :** L = 6, R = 28

**Output** : 34

# B) Count words in a given string

import java.util.Scanner;

public class SplitOnSpaceRAndN {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String s = scan.nextLine();

        int count = 1;

        boolean flag = false;

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

            char c = s.charAt(i);

            if (c == ' ' || c == '\n' || c == '\t')

                flag = true;

            else if (flag) {

                count++;

                flag = false;

            }

        }

        System.out.println(count);

    }

}

**Input:** S = "abc def"

**Output:** 2

**Input:** S = "a\tyo\t"

**Output:** 2

**Q6.**

# Find one extra character in a string

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.Scanner;

public class RemoveExtraChar {

    public static void main(String[] args) {

        LinkedHashMap<Character, Integer> hm = new LinkedHashMap<>();

        Scanner scan = new Scanner(System.in);

        String s1=scan.nextLine(), s2 = scan.nextLine();

        for (int i = 0; i < s1.length(); i++) {

            hm.put(s1.charAt(i), hm.getOrDefault(s1.charAt(i), 0) + 1);

        }

        for(int i=0;i<s2.length();i++){

            hm.put(s2.charAt(i), hm.getOrDefault(s2.charAt(i), 0) + 1);

        }

        for (Character c : hm.keySet()) {

            if (hm.get(c) % 2 != 0) {

                System.out.println("Unique character is: " + c);

                System.exit(0);

            }

        }

        System.out.println("No unique character.");

    }

}

**Input:** String1 = "abcd";

String2 = "cbdae";

**Output:** e

**Input:** String1 = "kxml";

String2 = "klxml";

**Output:** l

B) Given an array A with N integers, find the count of unique integers in the array.

import java.util.LinkedHashMap;

import java.util.Scanner;

public class UniqueIntCount {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int c = 0;

        int n = scan.nextInt();

        int[] arr = new int[n];

        for (int i = 0; i < n; i++)

            arr[i] = scan.nextInt();

        LinkedHashMap<Integer, Integer> hm = new LinkedHashMap<>();

        for (int i = 0; i < n; i++)

            hm.put(arr[i], hm.getOrDefault(arr[i], 0) + 1);

        for (int i : hm.keySet())

            if (hm.get(i) == 1)

                c++;

        System.out.println(c);

    }

}

Input : arr = [3 2 4 1 2 3]

Output : 2

**Q7.**

# Program to print the initials of a name with the surname

import java.util.Scanner;

public class InitialsWithSurname {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String s = scan.nextLine();

        String ns = "";

        String[] sAr = s.split(" ");

        for (int i = 0; i < sAr.length - 1; i++)

            ns = ns + sAr[i].charAt(0) + ". ";

        ns += sAr[sAr.length - 1];

        System.out.println(ns);

    }

}

**Input:** Devashish Kumar Gupta

**Output:** D. K. Gupta

**Input:** Ishita Bhuiya

**Output:** I. Bhuiya

1. **Given an array of size N, which contains the voting ID's of students that have stood up for the elections for class monitor, the candidate with votes greater than half the strength of the class will become monitor, find the ID of candidate that can become monitor else return -1 if no one can become.**

import java.util.LinkedHashMap;

import java.util.Scanner;

public class CRElections {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int req = n/2;

        int[] arr = new int[n];

        for (int i = 0; i < n; i++)

            arr[i] = scan.nextInt();

        LinkedHashMap<Integer, Integer> hm = new LinkedHashMap<>();

        for (int i = 0; i < n; i++)

            hm.put(arr[i], hm.getOrDefault(arr[i], 0) + 1);

        for (int i : hm.keySet())

            if (hm.get(i) > req) {

                System.out.println(i);

                System.exit(0);

            }

        System.out.println(-1);

    }

}

Input : A = [1, 3, 2, 2, 2]

Output : 2

**Q8.**

# Check if frequency of character in one string is a factor or multiple of frequency of same character in other string

import java.util.LinkedHashMap;

import java.util.Scanner;

public class CharactersFactorsAnotherString {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String s1 = scan.nextLine();

        String s2 = scan.nextLine();

        LinkedHashMap<Character, Integer[]> hm = new LinkedHashMap<>();

        for (char i : s1.toCharArray()) {

            hm.put(i, new Integer[] { hm.getOrDefault(i, new Integer[] { 0, 0 })[0] + 1, 0 });

        }

        for (char i : s2.toCharArray()) {

            if (hm.containsKey(i)) {

                Integer[] value = hm.get(i);

                hm.put(i, new Integer[] { value[0], value[1] + 1 });

            } else {

                hm.put(i, new Integer[] { 0, 1 });

            }

        }

        for (Character c : hm.keySet()) {

            int v1 = hm.get(c)[0];

            int v2 = hm.get(c)[1];

            if (v1 == 0 || v2 == 0 || v1 % v2 == 0 || v2 % v1 == 0)

                continue;

            else {

                System.out.println("NO");

                System.exit(0);

            }

        }

        System.out.println("YES");

    }

}

***Input:****s1 = “aabccd”, s2 = “bbbaaaacc”****Output:****YES****Input:****s1 = “hhdwjwqq”, s2 = “qwjdddhhh”****Output:****NO*

1. Given marks of N students sitting on a bench and a value of K, print the index of the student whose marks matches with the value of K.

import java.util.Scanner;

public class MarksEqualsK {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int[] arr = new int[n];

        for (int i = 0; i < n; i++)

            arr[i] = scan.nextInt();

        int k = scan.nextInt();

        for (int i = 0; i < n; i++)

            if (arr[i] == k) {

                System.out.println(i);

                System.exit(0);

            }

        System.out.println(-1);

    }

}

Input : N = 10, K = 67

A[] = [60, 61, 62, 63, 63, 64, 65, 66, 67, 66]

Output : 8

**Q9.**

# Check if given String is Pangram or not

import java.util.Scanner;

import java.util.TreeSet;

public class PangramOrNot {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        System.out.println("Enter a sentence:");

        String input = scan.nextLine();

        TreeSet<Character> ts = new TreeSet<>();

        for (int i = 0; i < input.length(); i++) if(input.charAt(i)>='a' && input.charAt(i)<='z')

            ts.add(input.charAt(i));

        if (ts.size() == 26)

            System.out.println("YES");

        else

            System.out.println("NO");

    }

}

***Input:****“The quick brown fox jumps over the lazy dog”****Output:****is a Pangram*

**B) The Leaders**

Print all those elements that have no element greater than them in the right side of the array. Print elements from right to left.

import java.util.Scanner;

public class NoGreatRightElement {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int[] arr = new int[n];

        for (int i=0;i<n;i++)

            arr[i] = scan.nextInt();

        int max = arr[n - 1];

        for (int i = n - 1; i >= 0; i--) {

            if (arr[i] >= max) {

                System.out.print(arr[i] + " ");

                max = arr[i];

            }

        }

    }

}

Test Case 1:

Input : A[] = [1, 2, 3, 4, 5]

Output : 5

Test Case 2:

Input : A[] = [1, 4, 3, 2]

Output : 2 3 4

**Q10.**

# Missing characters to make a string Pangram

import java.util.ArrayList;

import java.util.Scanner;

public class MissingCharsInPangram {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        ArrayList<Character> al = new ArrayList<>();

        for (char ch = 'a'; ch <= 'z'; ch++)

            al.add(ch);

        String input = scan.nextLine().toLowerCase();

        ArrayList<Character> copy = new ArrayList<>(al);

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

            char currentChar = input.charAt(i);

            if (currentChar >= 'a' && currentChar <= 'z')

                copy.remove(Character.valueOf(currentChar));

        }

        StringBuilder ns = new StringBuilder();

        for (Character c : copy) {

            ns.append(c);

        }

        System.out.println(ns.toString());

    }

}

**Input:** welcome to geeksforgeeks

**Output:** abdhijnpquvxyz

**Input:** The quick brown fox jumps

**Output:** adglvyz

1. **Given an array A of N elements, your task is to print all those indexes that have values greater than its left and right neighbours. In case of extreme indexes like 0 and N-1, check with their single neighbour.**

import java.util.Scanner;

public class GreatestInNeighbourhood {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int[] a = new int[n];

        for (int i = 0; i < n; i++)

            a[i] = scan.nextInt();

        if(n==1) System.out.println(a[0]);

        else {

            if(a[0]>a[1]) System.out.print(0+" ");

            for (int i = 1; i < n - 1; i++) {

                if (a[i] > a[i - 1] && a[i] > a[i + 1])

                    System.out.print(i + " ");

            }

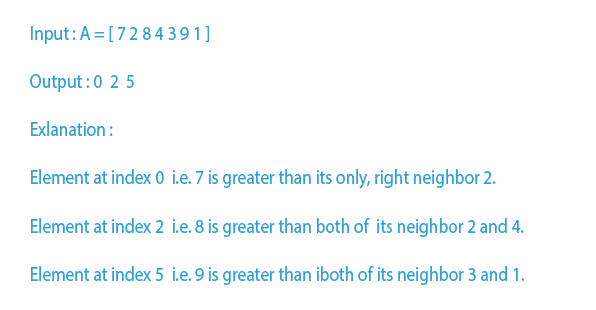
            if (a[n - 1] > a[n - 2])

                System.out.println(n - 1);

        }

    }

}



**Q11.**

# Check if max occurring character of one string appears same no. of times in other

import java.util.LinkedHashMap;

import java.util.Scanner;

public class MaxCharSameinBothStrings {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String s1 = scan.nextLine();

        String s2 = scan.nextLine();

        int maxCount = 0;

        char maxChar= s1.charAt(0);

        LinkedHashMap<Character, Integer> hm = new LinkedHashMap<>();

        for (int i = 0; i < s1.length(); i++)

            hm.put(s1.charAt(i), hm.getOrDefault(s1.charAt(i), 0) + 1);

        for (Character c : hm.keySet())

            if (hm.get(c) > maxCount) {

                maxChar = c;

                maxCount = hm.get(c);

            }

        boolean exists = false;

        hm = new LinkedHashMap<>();

        for (int i = 0; i < s2.length(); i++)

            hm.put(s2.charAt(i), hm.getOrDefault(s2.charAt(i), 0) + 1);

        for (Character c : hm.keySet())

            if (c == maxChar) {

                exists = true;

                if (hm.get(c) == maxCount)

                    System.out.println("YES.");

                else {

                    System.out.println("NO.");

                    break;

                }

            }

        if(!exists) System.out.println("NO.");

    }

}

**Input :** s1 = "sssgeek", s2 = "geeksss"

**Output :** Yes

**Input :** s1 = “ geekyarticle” s2 = “gfggfggfg”

**Output :** No

## **Cumulative Sum**

import java.util.Scanner;

public class CummulativeSum {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int[] arr = new int[n];

        for (int i = 0; i < n; i++)

            arr[i] = scan.nextInt();

        cumulativeSum(arr, n);

    }

    private static void cumulativeSum(int[] arr, int n) {

        int sum = 0;

        System.out.print("[ ");

        for (int i = 0; i < n; i++) {

            sum += arr[i];

            System.out.print(sum + " ");

        }

        System.out.println("]");

    }

}

Initial Array: [1, 2, 3, 4]

Cumulative Sum: [1, 3, 6, 10]

Initial Array: [1, 1, 1, 1, 1]

Cumulative Sum: [1, 2, 3, 4, 5]

Initial Array: [1, 3, 5, 7, 9]

Cumulative Sum: [1, 4, 9, 16, 25]

**Q12.**

1. **Given an array of words and a string, we need to count all words that are present in given string.**

import java.util.ArrayList;

import java.util.Scanner;

public class NumOfWordsMatchInArrayAndString {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        scan.nextLine();

        String[] inpStrList = new String[n];

        for (int i = 0; i < n; i++)

            inpStrList[i] = scan.nextLine();

        String s = scan.nextLine();

        System.out.println(countWordsMatched(s, inpStrList));

    }

    private static int countWordsMatched(String s, String[] inpList) {

        String[] strList = s.split(" ");

        int count = 0;

        ArrayList<String> al = new ArrayList<>();

        for (String i : inpList)

            for (String j : strList)

                if (i.equals(j) && notExists(al, j)) {

                    al.add(j);

                    count++;

                }

        return count;

    }

    private static boolean notExists(ArrayList<String> al, String s) {

        for (String i : al)

            if (i.equals(s))

                return false;

        return true;

    }

}

**Input:** words[] = { "welcome", "to", "geeks", "portal"}

str = "geeksforgeeks is a computer science portal for geeks."

**Output:** 2

**Input:** words[] = {"Save", "Water", "Save", "Yourself"}

str = "Save"

**Output:**1

## **B) Identical Twins**