**Q1.**

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

Given a string name, we have to find the initials of the name

**Examples:**

Input : prabhat kumar singh

Output : P K S

We take the first letter of all

words and print in capital letter.

Input : Jude Law

Output : J L

Input : abhishek kumar singh

Output : A K S

**Code:**

import java.util.\*;

public class Initial {

public static void main(String[] args) {

String name = null;

Scanner scan = new Scanner(System.in);

name = scan.nextLine();

System.out.print(Character.toUpperCase(name.charAt(0)) + " ");

for (int i = 1; i < name.length(); i++) {

if (name.charAt(i) == ' ') {

System.out.print(Character.toUpperCase(name.charAt(i + 1)) + " ");

}

}

}

}

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

Given an [array](https://www.geeksforgeeks.org/array-data-structure/) **arr[]** of size **n**, The task is to find the [GCD](http://www.geeksforgeeks.org/basic-and-extended-euclidean-algorithms/) of the [highest and lowest frequency element in the given array](https://www.geeksforgeeks.org/difference-between-highest-and-least-frequencies-in-an-array/).

**Examples:**

***Input:****arr[] = {2, 2, 4, 4, 5, 5, 6, 6, 6, 6}****Output:****2****Explanation:****The frequency of the elements in the above array is   
freq(2) = 2,   
freq(4) = 2,   
freq(5) = 2,   
freq(6) = 4,   
The minimum frequency is 2 (of elements 2, 4, and 5). So 2 will be picked as the least among 2, 4, and 5.   
The largest frequency is 4 (of element 6).   
Hence GCD of 2 and 6 = gcd(2, 6) is 2.*

***Input:****arr[] = {3, 2, 2, 44, 44, 44, 44}****Output:****1*

***Code:***

import java.util.\*;

public class MyClass{

static int gcd(int a,int b){

if(b==0)

return a;

else

return gcd(b,a%b);

}

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int arr[]=new int[n];

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

arr[i]=sc.nextInt();

}

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

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

if(h.containsKey(arr[i])){

h.put(arr[i],h.get(arr[i])+1);

}else{

h.put(arr[i],1);

}

}

int max=0,min=Integer.MAX\_VALUE;

int a=1,b=1;

for(Integer i:h.keySet()){

int t=h.get(i);

if(t>max){

max=t;

a=i;

}

if(t<min){

min=t;

b=i;

}

}

System.out.println(gcd(a,b));

}

}

Q2)

# Check Whether a number is Duck Number or not

A Duck number is a positive number which has zeroes present in it, For example 3210, 8050896, 70709 are all Duck numbers. Please note that a numbers with only leading 0s is not considered as Duck Number. For example, numbers like 035 or 0012 are not considered as Duck Numbers. A number like 01203 is considered as Duck because there is a non-leading 0 present in it.

**Examples :**

*Input : 707069   
Output : It is a duck number.   
Explanation: 707069 does not contains zeros at the beginning.*

*Input : 02364   
Output : It is not a duck number.   
Explanation: in 02364 there is a zero at the beginning of the number.*

# **Time Complexity:** **O(n)**where n is length of string. **Auxiliary Space: O(1)**

# ****Code:****

# import java.util.\*;

# public class DuckNumber {

# public static void main(String[] args) {

# String number=null;

# Scanner scan=new Scanner(System.in);

# number=scan.next();

# if(number.charAt(0)=='0'){

# System.out.println("It is not a duck number");

# }

# else{

# boolean check=false;

# for(int i=1;i<number.length();i++){

# if(number.charAt(i)=='0'){

# check=true;

# break;

# }

# }

# if(check){

# System.out.print("It is a duck number");

# }

# else{

# System.out.println("It is not a duck number");

# }

# }

# }

# }

# Find Kth most occurring element in an Array

Given an array of integers **arr[]** of size **N** and a number **K**, the task is to find the **Kth** most occurring element in this array.  
**Note:** If there are more than one numbers in the array with the same frequency, then both are considered to be at the same level of occurrence. Therefore print both the numbers.

**Examples:**

***Input:****arr[] = {1, 2, 2, 2, 4, 4, 4, 5, 5, 5, 5, 5, 7, 7, 8, 8, 8, 8}, K = 1****Output:****5****Explanation:****The occurrence of the elements are as follows:   
1 – 1   
2 – 3   
4 – 3   
5 – 5   
7 – 2   
8 – 4   
Clearly, 5 is the most occurring element in the array.*

***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*

**Code:**

import java.util.\*;

public class KthMostOccurringNumber {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int n;

n=scan.nextInt();

int[] arr=new int[n];

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

arr[i]=scan.nextInt();

}

int k=scan.nextInt();

Map<Integer, Integer> map= new HashMap<>();

for (int i:arr) {

map.put(i,map.getOrDefault(i,0) + 1);

}

Map<Integer, List<Integer>> map1=new TreeMap<>(Collections.reverseOrder());

for (Map.Entry<Integer, Integer> e:map.entrySet()) {

int key=e.getKey();

int value=e.getValue();

if (!map1.containsKey(value)) {

map1.put(value, new ArrayList<>());

}

map1.get(value).add(key);

}

int count=0;

for (Map.Entry<Integer, List<Integer>> e1:map1.entrySet()) {

List<Integer> l= e1.getValue();

Collections.sort(l);

for (int i:l) {

count++;

if (count==k) {

System.out.println(i);

return;

}

}

}

}

}

Q3.

# Removing punctuations from a given string

Given a string, remove the punctuation from the string if the given character is a punctuation character, as classified by the current C locale. The default C locale classifies these characters as punctuation:

**! " # $ % & ' ( ) \* + , - . / : ; ? @ [ \ ] ^ \_ ` { | } ~**

**Examples:**

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

**Output :** welcome to geeksforgeeks

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

**Output :** Hello he said and went

**Code:**

import java.util.\*;

public class RemovePunctuation {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String input = scan.nextLine();

String result = remove(input);

System.out.println(result);

}

private static String remove(String input) {

String punc="[!\"#$%&'()\*+,\\-./:;?@[\\\\]^\_`{|}~]";

String res=input.replaceAll(punc,"");

return res;

}

}

# Print all array elements occurring at least M times

Given an [array](https://www.geeksforgeeks.org/introduction-to-arrays/) **arr[]** consisting of **N** integers and a positive integer **M**, the task is to find the number of array elements that occur **at least M** times.

**Examples:**

***Input:****arr[] = {2, 3, 2, 2, 3, 5, 6, 3}, M = 2****Output:****2 3****Explanation:*** *In the given array arr[], the element that occurs at least M number of times are {2, 3}.*

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

**Code:**

import java.util.\*;

public class OccurringMTimes {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int n;

n=scan.nextInt();

int arr[]=new int[n];

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

arr[i]=scan.nextInt();

}

int m;

m=scan.nextInt();

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

HashSet<Integer> p=new HashSet<Integer>();

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

if(hm.containsKey(arr[i])){

int x=hm.get(arr[i]);

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

}

else{

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

}

}

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

if(hm.get(arr[i])>=m && !p.contains(arr[i])){

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

p.add(arr[i]);

}

}

}

}

# Q4) A) Check whether the given number is Euclid Number or not

Given a positive integer n, the task is to check if it is Euclid Number or not. Print ‘YES’ if the given number is Euclid Number otherwise print ‘NO’.

[**Euclid number :**](https://en.wikipedia.org/wiki/Euclid_number)In Mathematics, Euclid numbers are integers of the form –

E=P#+1

where P# is product of first n prime numbers.  
The first few Euclid numbers are-

*3, 7, 31, 211, 2311, 30031, 510511, 9699691, ……….*

**Input:** N = 31

**Output:** YES

31 can be expressed in the form of

pn# + 1 as p3# + 1

(First 3 prime numbers are 2, 3, 5 and their product is 30 )

**Input:** N = 43

**Output:** NO

1. Cannot be expressed in the form of pn# + 1

**Code:**

import java.util.\*;

public class Euclid {

static final int MAX = 10000;

static Vector<Integer> arr=new Vector<Integer>();

static void Sieve()

{

boolean[] p=new boolean[MAX];

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

p[i]=true;

for (int j=2;j\*j<MAX;j++) {

if (p[j]==true) {

for (int i=j\*2;i<MAX;i+=j)

p[i]=false;

}

}

for (int k=2;k<MAX;k++)

if (p[k])

arr.add(k);

}

static boolean isEuclid(long n)

{

long p=1;

int i=0;

while(p<n) {

p=p\*arr.get(i);

if (p+1==n)

return true;

i++;

}

return false;

}

public static void main(String[] args)

{

Sieve();

long n;

Scanner scan=new Scanner(System.in);

n=scan.nextLong();

if (isEuclid(n))

System.out.println("YES");

else

System.out.println("NO");

}

}

# B) Program to print reverse character bridge pattern

For a given value N, denoting the number of Charters starting from the A, print reverse character bridge pattern.  
**Examples :**

Input : n = 5

Output :

ABCDEDCBA

ABCD DCBA

ABC CBA

AB BA

A A

Input : n = 8

Output :

ABCDEFGHGFEDCBA

ABCDEFG GFEDCBA

ABCDEF FEDCBA

ABCDE EDCBA

ABCD DCBA

ABC CBA

AB BA

A A

**Code:**

import java.util.\*;

public class MyClass{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int c=0;

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

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

System.out.print((char)(c+65));

c++;

}

for(int j=0;j<2\*i;j++){

System.out.print(" ");

}

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

c--;

System.out.print((char)(c+65));

}

System.out.println();

}

}

}

Q5)

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

Given two numbers **L**, **R** which signifies the range **[L, R]**, the task is to find the sum of all [perfect numbers](https://www.geeksforgeeks.org/perfect-number/) lying in the range [L, R].

**Examples:** 

***Input:****L = 6, R = 10****Output:****6****Explanation:****From 6 to 10, the only perfect number is 6.****Input:****L = 6, R = 28****Output:****34****Explanation:****There are two perfect numbers in the range [6, 28]. They are, {6, 28}   
6 + 28 = 34.*

**Code:**

# import java.util.Scanner;

# public class PerfectNumberSum {

# public static void main(String[] args) {

# Scanner scan = new Scanner(System.in);

# int l = scan.nextInt();

# int r = scan.nextInt();

# int x=findPerfectNumbers(l,r);

# System.out.println(x);

# }

# static int findPerfectNumbers(int l,int r) {

# int sum=0;

# for (int i=l;i<=r;i++) {

# if (isPerfect(i)) {

# sum+=i;

# }

# }

# return sum;

# }

# static boolean isPerfect(int num) {

# if (num<=1) {

# return false;

# }

# int sum=1;

# for (int i=2;i\*i<=num;i++) {

# if (num%i==0) {

# sum+=i;

# if (i\*i!=num) {

# sum+=num/i;

# }

# }

# }

# return sum==num;

# }

# }

# B) Count words in a given string

Given a string, count the number of words in it. The words are separated by the following characters: space (‘ ‘) or new line (‘\n’) or tab (‘\t’) or a combination of these.

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

**Output:** 2

**Input:** S = "a\nyo\n"

**Output:** 2

**Code:**

import java.util.Scanner;

public class WordCount {

public static void main(String[] args) {

Scanner scan=new Scanner(System.in);

String s=null;

s=scan.nextLine();

int count=0;

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

if(s.charAt(i)==' '|| s.charAt(i)=='\n'||s.charAt(i)=='\t'){

count++;

}

}

System.out.println(count+1);

}

}

Q6)

# Find one extra character in a string

Given two strings which are of lengths n and n+1. The second string contains all the character of the first string, but there is one extra character. Your task to find the extra character in the second string.

**Examples :**

**Input :** string strA = "abcd";

string strB = "cbdae";

**Output :** e

string B contain all the element

there is a one extra character which is e

**Input :** string strA = "kxml";

string strB = "klxml";

**Output :** l

string B contain all the element

there is a one extra character which is l

**Code:**

import java.util.\*;

public class ExtraChar {

public static void main(String[] args) {

Scanner scan=new Scanner(System.in);

String s1=null;

String s2=null;

s1=scan.next();

s2=scan.next();

char x=find(s1,s2);

System.out.print(x);

}

public static char find(String str1,String str2) {

int res=0;

for(int i=0;i<str1.length();i++) {

res^= str1.charAt(i);

}

for(int i=0;i<str2.length();i++) {

res^= str2.charAt(i);

}

return (char) res;

}

}

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

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

Output : 2

Explanation: only 1 and 4 are unique in the array rest 2, 3 are occurring 2 times.

**Time Complexity: O(N)**

**Space Complexity** : O(N)

**Code:**

import java.util.\*;

public class FrequencyOf1 {

public static void main(String[] args) {

Scanner scan=new Scanner(System.in);

int n;

n=scan.nextInt();

int arr[]=new int[n];

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

arr[i]=scan.nextInt();

}

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

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

if(hm.containsKey(arr[i])){

int x=hm.get(arr[i]);

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

}

else{

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

}

}

int count=0;

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

if(hm.get(arr[i])==1)

count++;

}

System.out.println(count);

}

}

Q7)

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

Given a full name in the form of a string, the task is to print the initials of a name, in short, and surname in full.

**Examples:**

**Input:** Devashish Kumar Gupta

**Output:** D. K. Gupta

**Input:** Ishita Bhuiya

**Output:** I. Bhuiya

**Code:**

import java.util.\*;

public class Surname {

public static void main(String[] args) {

Scanner scan=new Scanner(System.in);

String s=null;

s=scan.nextLine();

System.out.print(Character.toUpperCase(s.charAt(0))+" ");

int x=0;

for(int k=s.length()-1;k>0;k--){

if(s.charAt(k)==' '){

x=k;

break;

}

}

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

if(s.charAt(i)==' '){

System.out.print(Character.toUpperCase(s.charAt(i+1))+" ");

}

}

for(int k1=x+1;k1<s.length();k1++){

System.out.print(s.charAt(k1));

}

}

}

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.

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

Output : 2

Explanation : 2 got 3 votes which is greater than half the strength of the class i.e. 5/2 = 2.

**Time Complexity: O(N)**

**Space Complexity** : O(N)

**Code:**

import java.util.\*;

public class Voting {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int n;

n = scan.nextInt();

int arr[] = new int[n];

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

arr[i] = scan.nextInt();

}

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

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

int x = 0;

if (hm.containsKey(arr[i])) {

x = hm.get(arr[i]);

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

} else {

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

}

}

int y = n / 2;

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

if (hm.get(arr[i]) > y) {

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

break;

}

}

}

}

Q8)

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

Given two strings, the task is to check whether the frequencies of a character(for each character) in one string are multiple or a factor in another string. If it is, then output “YES”, otherwise output “NO”.

**Examples:**

***Input:****s1 = “aabccd”, s2 = “bbbaaaacc”****Output:****YES   
Frequency of ‘a’ in s1 and s2 are 2 and 4 respectively, and 2 is a factor of 4   
Frequency of ‘b’ in s1 and s2 are 1 and 3 respectively, and 1 is a factor of 3   
Frequency of ‘c’ in s1 and s2 are same hence it also satisfies.   
Frequency of ‘d’ in s1 and s2 are 1 and 0 respectively, but 0 is a multiple of every number, hence satisfied.   
Hence, the answer YES.*

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

**Code:**

import java.util.\*;

public class MyClass{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s1=sc.nextLine();

HashMap<Character,Integer> h1=new HashMap<>();

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

char c=s1.charAt(i);

if(h1.containsKey(c)){

h1.put(c,h1.get(c)+1);

}else{

h1.put(c,1);

}

}

String s2=sc.nextLine();

HashMap<Character,Integer> h2=new HashMap<>();

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

char c=s2.charAt(i);

if(h2.containsKey(c)){

h2.put(c,h2.get(c)+1);

}else{

h2.put(c,1);

}

}

for(Character i:h2.keySet()){

if(h2.get(i)%h1.get(i)!=0){

System.out.print("NO");

return;

}

}

System.out.print("YES");

}

}

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.

Input : N = 10, K = 67

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

Output : 8

Explanation : 67 is present at 8th index (0-based indexing)

**Code:**

import java.util.\*;

public class Marks {

public static void main(String[] args) {

Scanner scan=new Scanner(System.in);

int n;

n=scan.nextInt();

int k;

k=scan.nextInt();

int arr[]=new int[n];

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

arr[i]=scan.nextInt();

}

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

if(arr[i]==k)

System.out.println(i);

}

}

}

Q9)

# Check if given String is Pangram or not

Given a string **Str**.The task is to check if it is Pangram or not.

*A****pangram****is a sentence containing every letter in the English Alphabet.*

**Examples:**

***Input:****“The quick brown fox jumps over the lazy dog”****Output:****is a Pangram****Explanation:****Contains all the characters from ‘a’ to ‘z’]*

***Code:***

import java.util.\*;

public class Pangram {

public static void main(String[] args) {

Scanner scan=new Scanner(System.in);

String str=null;

str=scan.nextLine();

str=str.toLowerCase();

Set<Character> hs=new HashSet<>();

for(char c:str.toCharArray()){

if(Character.isLetter(c)){

hs.add(c);

}

}

if(hs.size()==26){

System.out.println("is a Pangram");

}

else{

System.out.println("is not a Pangram");

}

}

}

1. 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.

Test Case 1:

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

Output : 5

Explanation : Only 5 has no element greater than it on its right.

Test Case 2:

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

Output : 2 3 4

Explanation : 2, 3, and 4 have no elements greater than them on their right while 1 has 4, 3 and 2 greater than it on its right.

**Code:**

import java.util.\*;

public class Leaders {

public static void main(String[] args) {

Scanner scan=new Scanner(System.in);

int n;

n=scan.nextInt();

int arr[]=new int[n];

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

arr[i]=scan.nextInt();

}

printElements(arr);

}

static void printElements(int[] arr) {

int n = arr.length;

Stack<Integer> stack = new Stack<>();

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

while (!stack.isEmpty() && stack.peek()<=arr[i]) {

stack.pop();

}

if (stack.isEmpty()) {

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

}

stack.push(arr[i]);

}

}

}

Q10)

# Missing characters to make a string Pangram

[Pangram](https://www.geeksforgeeks.org/pangram-checking/) is a sentence containing every letter in the English alphabet. Given a string, find all characters that are missing from the string, i.e., the characters that can make the string a Pangram. We need to print output in alphabetic order.

**Examples:**

Input : welcome to geeksforgeeks

Output : abdhijnpquvxyz

Input : The quick brown fox jumps

Output : adglvyz

**Code:**

import java.util.\*;

public class MissingPangram {

public static void main(String[] args) {

Scanner scan=new Scanner(System.in);

String str=null;

str=scan.nextLine();

str=str.toLowerCase();

HashSet<Character> hs=new HashSet<>();

HashSet<Character> missing=new HashSet<>();

for(char ch:str.toCharArray()){

if(Character.isLetter(ch)){

hs.add(ch);

}

}

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

if(!hs.contains(ch)){

missing.add(ch);

}

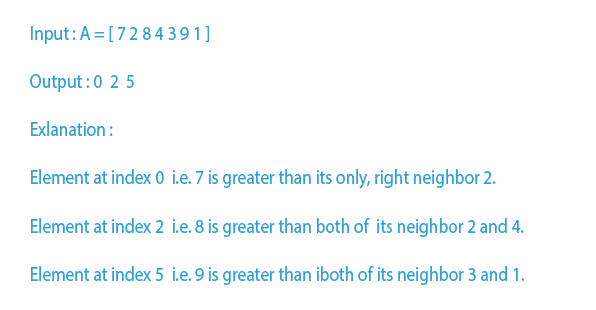
}

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

}

}

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 neighbors. In case of extreme indexes like 0 and N-1, check with their single neighbor.



**Code:**

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

if(a[0]>a[1])

System.out.println(0);

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

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

System.out.println(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

Given two strings, we need to take the character which has the maximum occurrence in the first string, and then we have to check if that particular character is present in the second string the same number of times as it is present in the first string.  
**Examples:** 

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

Output : Yes

Max occurring character in s1 is

's'. It occurs same number of times

in s2.

Input : geekyarticle

gfggfggfg

Output : No

**Code:**

import java.util.\*;

public class MaxOccurringCharacter {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String s1 = scan.nextLine();

String s2 = scan.nextLine();

HashMap<Character, Integer> hm1 = new HashMap<>();

HashMap<Character, Integer> hm2 = new HashMap<>();

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

char ch = s1.charAt(i);

hm1.put(ch, hm1.getOrDefault(ch, 0) + 1);

}

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

char ch = s2.charAt(i);

hm2.put(ch, hm2.getOrDefault(ch, 0) + 1);

}

char maxChar = ' ';

int maxCount = Integer.MIN\_VALUE;

for (Map.Entry<Character, Integer> entry : hm1.entrySet()) {

char ch = entry.getKey();

int count = entry.getValue();

if (count > maxCount) {

maxCount = count;

maxChar = ch;

}

}

if (hm2.containsKey(maxChar) && hm2.get(maxChar) == maxCount) {

System.out.println("Yes");

} else {

System.out.println("No");

}

}

}

## Cumulative Sum

The cumulative sum of an array at index i is defined as the sum of all elements of the array from index 0 to index i.

##### Examples

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]

Given an array, return its cumulative sum.

**Code:**

import java.util.\*;

public class CumulativeSum {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int n;

n=scan.nextInt();

int arr[]=new int[n];

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

arr[i]=scan.nextInt();

}

int sum1[]=new int[n];

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

int sum=0;

for(int k=i;k>=0;k--){

sum+=arr[k];

}

sum1[i]=sum;

}

for(int i=0;i<sum1.length;i++){

System.out.print(sum1[i]+" ");

}

}

}

Q12) A) Given an array of words and a string, we need to count all words that are present in given string.

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

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

**Output :** 2

Two words "portal" and "geeks" is present in str.

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

str = "Save"

**Output :**1

**Code:**

import java.util.\*;

public class CountingWords {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String s1=scan.nextLine();

String words[]=scan.nextLine().split("\\s+");

String split[]=s1.split("\\s+");

HashSet<String> hs=new HashSet<>(Arrays.asList(words));

int count=0;

for(String c:split){

if(hs.contains(c)){

count++;

}

}

System.out.println(count);

}

}

## B) Identical Twins

For an array of integers nums, an identical twin is defined as pair (i, j) where nums[i] is equal to nums[j] and i < j.

Test Case 1: Array: [1, 2, 3, 2, 1]

Number of Identical Twins: 2

Explanation:

Identical Twins: [[1, 1], [2, 2]]

Indexes: (0, 4), (1, 3)

Test Case 2: Array: [1, 2, 2, 3, 2, 1]

Number of Identical Twins: 4

Explanation:

Identical Twins: [[1, 1], [2, 2], [2, 2], [2, 2]]

Indexes: (0, 5), (1, 2), (1, 4), (2, 4)

Test Case 3: Array: [1, 1, 1, 1]

Number of Identical Twins: 6

Explanation:

Identical Twins: [[1, 1], [1, 1], [1, 1], [1, 1], [1, 1], [1, 1]]

Indexes: (0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)

Given an array nums, find the number of identical twins.

**Code:**

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int c=0;

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

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

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

if(a[i]==a[j])

c++;

}

}

System.out.println(c);

}

}

Q13)

# Count words that appear exactly two times in an array of words

Given an array of n words. Some words are repeated twice, we need to count such words.

**Examples:**

**Input :** s[] = {"hate", "love", "peace", "love",

"peace", "hate", "love", "peace",

"love", "peace"};

**Output :** 1

There is only one word "hate" that appears twice

**Input :** s[] = {"Om", "Om", "Shankar", "Tripathi",

"Tom", "Jerry", "Jerry"};

**Output :** 2

There are two words "Om" and "Jerry" that appear

twice.

**Code:**

import java.util.\*;

public class CountingWordsTwice {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String[] words=scan.nextLine().split("\\s+");

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

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

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

}

int count=0;

for(int i=0;i<hm.size();i++){

if(hm.get(words[i])==2)

count++;

}

System.out.println(count);

}

}

## Even Number of Digits

Given an array of integers, find the elements which have an even number of digits.

##### Example

Array: [42, 564, 5775, 34, 123, 454, 1, 5, 45, 3556, 23442]

Answer: 42, 5775, 34, 45, 3556

The order of the returned elements should be the same as the order of the initial array.

**Code:**

import java.util.\*;

public class EvenNumberOfDigits {

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();

}

List<Integer> res=new ArrayList<>();

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

int count=0;

int x=arr[i];

while(x!=0){

int rem=x%10;

count++;

x=x/10;

}

if(count%2==0){

res.add(arr[i]);

}

}

for(int i:res){

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

}

}

}

Q14)

# Count of camel case characters present in a given string

Given a [string](https://www.geeksforgeeks.org/string-data-structure/) **S**, the task is to count the number of[camel case characters](https://www.geeksforgeeks.org/camel-case-given-sentence/) present in the given string.

*The camel case character is defined as the number of uppercase characters in the given string.*

**Examples:**

***Input:****S = “ckjkUUYII”****Output:****5****Explanation:****Camel case characters present are U, U, Y, I and I.*

***Input:****S = “abcd”****Output:****0*

**Code:**

import java.util.\*;

public class CountCamelCase {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String s=scan.nextLine();

int count=0;

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

if(Character.isUpperCase(s.charAt(i))){

count++;

}

}

System.out.print(count);

}

}

## Max Consecutive Ones

Given an array A, find the maximum number of consecutive 1s in the array.

#### Examples

A: [1, 1, 3, 2, 3, 1, 1, 1]  
Max consecutive 1s: 3

**Code:**

import java.util.\*;

import java.lang.\*;

public class ConsecutiveOnesCount {

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 count=0;

int count1=0;

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

if (arr[i] == 1) {

count++;

count1 = Math.max(count1, count);

}

else {

count = 0;

}

}

System.out.print(count1);

}

}

# Q15)

# A)Find resultant string after concatenating uncommon characters of given strings

Given two strings **S1** and**S2.**The task is to concatenate uncommon characters of the **S2**to **S1**and return the resultant string **S1 .**

**Examples:**

***Input:******S1****= “aacdb”,****S2****= “gafd”****Output:****“cbgf”*

***Input:******S1****= “abcs”,****S2****= “cxzca”;****Output:****“bsxz”*

***Code:***

import java.util.\*;

public class ConcatenateUncommonCharacters {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String s1=scan.nextLine();

String s2=scan.nextLine();

String res=concatenatechar(s1,s2);

System.out.println(res);

}

public static String concatenatechar(String S1, String S2) {

Map<Character, Integer> c= new HashMap<>();

for (char ch : S1.toCharArray()) {

c.put(ch, c.getOrDefault(ch, 0) + 1);

}

for (char ch : S2.toCharArray()) {

c.put(ch, c.getOrDefault(ch, 0) + 1);

}

StringBuilder result = new StringBuilder();

for (Map.Entry<Character, Integer> entry : c.entrySet()) {

if (entry.getValue() == 1) {

result.append(entry.getKey());

}

}

return result.toString();

}

}

# B) Rearrange array such that even positioned are greater than odd

Given an array A of n elements, sort the array according to the following relations :

* A[i]>=A[i-1], if i is even.
* A[i]<=A[i-1], if i is odd.

Print the resultant array.

**Examples :**

Input : A[] = {1, 2, 2, 1}

Output : 1 2 1 2

Explanation :

For 1st element, 1 1, i = 2 is even.

3rd element, 1 1, i = 4 is even.

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

Output : 1 3 2

Explanation :

Here, the array is also sorted as per the conditions.

1 1 and 2 < 3.

**Code:**

import java.util.\*;

public class Sort{

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();

}

sortarray(arr);

System.out.println(Arrays.toString(arr));

}

private static void sortarray(int[] A) {

int n = A.length;

for (int i = 1; i < n; i += 2) {

for (int j = i + 2; j < n; j += 2) {

if (A[i] > A[j]) {

swap(A, i, j);

}

}

}

for (int i = 0; i < n; i += 2) {

for (int j = i + 2; j < n; j += 2) {

if (A[i] > A[j]) {

swap(A, i, j);

}

}

}

}

private static void swap(int[] arr, int i, int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

Q16)

# Maximum Consecutive Zeroes in Concatenated Binary String

You are given a binary string **str** of length **n**. Suppose you create another string of size n \* k by concatenating **k** copies of str together. What is the maximum size of a substring of the concatenated string consisting only of 0’s? Given that k > 1.

**Examples:**

***Input :****str = “110010”, k = 2****Output :****2   
String becomes 110010110010 after two concatenations. This string has two zeroes.*

***Input :****str = “00100110”, k = 4****Output :****3*

**Complexity Analysis:**

***Time Complexity:*O(N)**, where N represents the length of the given string.

**Code:**

import java.util.\*;

public class MyClass{

static int fun(String s){

int max=0;

int n=s.length();

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

int c=0;

if(s.charAt(i)=='0'){

c++;

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

if(s.charAt(k)=='0')

c++;

else

break;

}

}

if(max<c){

max=c;

}

}

return max;

}

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

int n=sc.nextInt();

String t="";

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

t+=s;

}

System.out.print(fun(t));

}

}

1. **Count smaller elements on Right side**

Given an unsorted array arr[] of distinct integers, construct another array countSmaller[] such that countSmaller[i] contains the count of smaller elements on the right side of each element arr[i] in the array.

**Examples:**

***Input:****arr[] =  {12, 1, 2, 3, 0, 11, 4}****Output:****countSmaller[]  =  {6, 1, 1, 1, 0, 1, 0}*

***Input:****arr[] =  {5, 4, 3, 2, 1}****Output:****countSmaller[]  =  {4, 3, 2, 1, 0}*

**Code:**

import java.util.\*;

public class MyClass{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

int t[]=new int[n];

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

a[i]=sc.nextInt();

}

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

int c=0;

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

if(a[i]>a[j])

c++;

}

t[i]=c;

}

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

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

}

}

17). A) Give a N\*N square matrix, return an array of its anti-diagonals. Look at the example for more details.

**Example:**

Input:

1 2 3

4 5 6

7 8 9

Return the following:

[

[1],

[2, 4],

[3, 5, 7],

[6, 8],

[9]

]

Input:

1 2

3 4

Return the following:

[

[1],

[2, 3],

[4]

]

**Code:**

import java.util.\*;

public class MyClass{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[][]=new int[n][n];

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

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

a[i][j]=sc.nextInt();

}

}

int c=0;

System.out.println("[");

for(int i=0;i<2\*n-1;i++){

System.out.print("[");

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

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

if(k+j==c){

System.out.print(a[j][k]+" ");

}

}

System.out.println("]");

c++;

}

System.out.print("]");

}

}

# Expand the string according to the given conditions

Given string **str** of the type **“3(ab)4(cd)”**, the task is to expand it to “abababcdcdcdcd” where integers are from the range **[1, 9]**.

**Examples:**

***Input:****str = “3(ab)4(cd)”****Output:****abababcdcdcdcd*

***Input:****str = “2(kl)3(ap)”****Output:****klklapapap*

**Code:**

import java.util.\*;

public class MyClass{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

String ans="";

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

char c=s.charAt(i);

String k="";

if(c>='0' && c<='9'){

for(int j=i+2;j<s.length();j++){

char m=s.charAt(j);

if(m==')')

break;

k+=m;

i++;

}

for(int j=0;j<(int)(c-48);j++){

ans+=k;

}

}

}

System.out.print(ans);

}

}

18) A) Given an integer array **A** of size **N**. You need to check that whether there exist a element which is **strictly greater than all the elements on left** of it and **strictly smaller than all the elements** on right of it.If it exists return **1** else return **0**.

**NOTE:** Do not consider the corner elements i.e **A[0] and A[N-1]** as the answer.

**Problem Constraints:** 3 <= N <= 105

1 <= A[i] <= 109

**Input Format:** First and only argument is an integer array **A** containing **N** integers.

**Output Format :** Return **1** if there exist a element that is **strictly greater than all the elements on left** of it and **strictly smaller than all the elements** on right of it else return 0.

**Example Input**

Input 1: A = [5, 1, 4, 3, 6, 8, 10, 7, 9]

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

**Example Output:**

Output 1: 1

Output 2: 0

**Example Explanation**

Explanation 1: A[4] = 6 is the element we are looking for.

All elements on left of A[4] are smaller than it and all elements on right are greater.

Explanation 2: No such element exits.

**Code:**

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

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

if(a[i]>a[i-1]){

int j=i+1;

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

if(a[j]>=a[j+1])

break;

}

if(j==n)

{

System.out.println(1);

return;

}

}

}

System.out.println(0);

}

}

B) GCD of more than two (or array) numbers

Given an array of numbers, find GCD of the array elements. In a previous post we [find GCD of two number](https://www.geeksforgeeks.org/c-program-find-gcd-hcf-two-numbers/).

**Examples:**

Input : arr[] = {1, 2, 3}

Output : 1

Input : arr[] = {2, 4, 6, 8}

Output : 2

**Code:**

import java.util.\*;

public class MyClass {

static int fun(int a,int b){

if(a<b){

int t=a;

a=b;

b=t;

}

if(b==0)

return a;

else

return fun(b,a%b);

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

int gcd=fun(a[0],a[1]);

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

gcd=fun(a[i],gcd);

}

System.out.println(gcd);

}

}

19) A) Given a matrix of **M \* N** elements (M rows, N columns), return all elements of the matrix in spiral order.

**Problem Constraints**

1 <= **M, N** <= 1000

**Input Format**

The first argument is a matrix A.

**Output Format**

Return an array of integers representing all elements of the matrix in spiral order.

**Example Input**

A =

[

[ 1, 2, 3 ],

[ 4, 5, 6 ],

[ 7, 8, 9 ]

]

**Example Output**

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

**Code:**

B) Encrypt a string by repeating i-th character i times

Given string **str**, the task is to encrypt the string with the given encryption algorithm. The **1st** character of the string will be repeated **once** in the encrypted string, the **2nd** character will be repeated twice, …, **nth** character will be repeated **n** times.

**Examples:**

Input: str = "geeks"

Output: geeeeekkkksssss

Input: str = "abcd"

Output: abbcccdddd

**Code:**

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

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

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

System.out.print(s.charAt(i));

}

}

}

}

20)

# Find elements which are present in first array and not in second

Given two arrays, the task is that we find numbers which are present in first array, but not present in the second array.

**Examples :**

Input : a[] = {1, 2, 3, 4, 5, 10};

b[] = {2, 3, 1, 0, 5};

Output : 4 10

4 and 10 are present in first array, but

not in second array.

Input : a[] = {4, 3, 5, 9, 11};

b[] = {4, 9, 3, 11, 10};

Output : 5

**Code:**

# import java.util.\*;

# public class MyClass {

# public static void main(String args[]) {

# Scanner sc=new Scanner(System.in);

# int m=sc.nextInt();

# int n=sc.nextInt();

# ArrayList<Integer> a=new ArrayList<>();

# ArrayList<Integer> b=new ArrayList<>();

# for(int i=0;i<m;i++){

# a.add(sc.nextInt());

# }

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

# b.add(sc.nextInt());

# }

# for(int i=0;i<m;i++){

# if(!b.contains(a.get(i)))

# System.out.print(a.get(i)+" ");

# }

# }

# }

# B) Check if String formed by first and last X characters of a String is a Palindrome

Given a [string](https://www.geeksforgeeks.org/category/data-structures/c-strings/) **str** and an integer **X**. The task is to find whether the first**X** characters of both [string](https://www.geeksforgeeks.org/category/data-structures/c-strings/) **str**and [reversed](https://www.geeksforgeeks.org/reverse-a-string-in-java/)[string](https://www.geeksforgeeks.org/category/data-structures/c-strings/) **str** are same or not. If it is equal then print **true**, otherwise print **false**.

**Examples:**

***Input****: str = abcdefba, X = 2****Output****: true****Explanation****:   
First 2 characters of both string****str****and reversed string****str****are same.*

***Input****: str = GeeksforGeeks, X = 3****Output****: false*

**Code:**

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

int x=sc.nextInt();

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

if(s.charAt(i)!=s.charAt(s.length()-1-i)){

System.out.println("false");

return;

}

}

System.out.println("true");

}

}

# 21) A) Check if array elements are consecutive. 30M

# Given an unsorted array of numbers, write a function that returns true if the array consists of consecutive numbers.

# **Test Cases:**

# **a)** If the array is {5, 2, 3, 1, 4}, then the function should return true because the array has consecutive numbers from 1 to 5.

# **b)** If the array is {83, 78, 80, 81, 79, 82}, then the function should return true because the array has consecutive numbers from 78 to 83.

# **c)**If the array is {34, 23, 52, 12, 3}, then the function should return false because the elements are not consecutive.

# **d)**If the array is {7, 6, 5, 5, 3, 4}, then the function should return false because 5 and 5 are not consecutive.

Expected **Time Complexity:** O(n)   
**Auxiliary Space:** O(n)

For Time Complexity O(nlogn): 15 Marks

For Time Complexity O(n): 30 Marks

**Code:**

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

Arrays.sort(a);

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

if(a[i]-a[i-1]!=1){

System.out.print("false");

return;

}

}

System.out.print("true");

}

}

# B) Check if a given string is a rotation of a palindrome 30 M

Given a string, check if it is a rotation of a palindrome. For example your function should return true for “aab” as it is a rotation of “aba”.

**Examples:**

Input: str = "aaaad"

Output: 1

// "aaaad" is a rotation of a palindrome "aadaa"

Input: str = "abcd"

Output: 0

// "abcd" is not a rotation of any palindrome.

**Code:**

import java.util.\*;

public class MyClass {

static String rotate(String s){

return s.substring(1,s.length())+s.charAt(0);

}

static Boolean isPalindrome(String s){

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

if(s.charAt(i)!=s.charAt(s.length()-i-1))

return false;

}

return true;

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

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

s=rotate(s);

if(isPalindrome(s)){

System.out.println("true");

return;

}

}

System.out.println("false");

}

}

# 22) A) Count Distinct Strings present in an array

Given an [array of strings](https://www.geeksforgeeks.org/array-strings-c-3-different-ways-create/) **arr[]**, the task is to find the count of distinct strings present in the array.

**Examples:**

***Input:****arr[] = { “abcde”, “abcce”, “abcdf”, “abcde”, “abcdf” }****Output:****3****Explanation:****Distinct strings in the array are { “abcde”, “abcce”, “abcdf” }.   
Therefore, the required output is 3.*

***Input:****arr[] = { “ab”, “abc”, “abcd”, “abcde”, “a” }****Output:****5****Explanation:****Distinct strings in the array are { “abcde”, “abcd”, “abc”, “ab”, “a” }.   
Therefore, the required output is 5.*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

String[] words=new String[n];

sc.nextLine();

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

words[i]=sc.nextLine();

}

HashSet<String> h=new HashSet<>();

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

h.add(words[i]);

}

System.out.println(h.size());

}

}

# Count of elements in Array which are present K times & their double isn’t present

Given an [array](https://www.geeksforgeeks.org/array-data-structure/) **arr[]** of N integers, the task is to find the count of elements in the array that are present **K times** and their double are not present in the array.

**Examples:**

***Input:****arr[] = {10, 6, 12, 8, 10, 8}, K = 2****Output:****2****Explanation:****10 is a valid number since it appears exactly two times and 20 does not appear in array.  
8 is a valid number since it appears two times and 16 does not appear in array.*

***Input:****arr[] = {1, 3, 5, 3}, K = 3****Output:****0****Explanation:****No element in the given array satisfy the condition.*

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

***Output****:0*

***Input:****arr[] = {1, 2, 2, 3, 3, 4}, K = 2****Output:****1****Explanation:****Only 3 is valid element.  
Though 2 is present twice but its double is also present.*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

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

int n=sc.nextInt();

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

int k=sc.nextInt();

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

if(h.containsKey(a[i]))

h.put(a[i],h.get(a[i])+1);

else

h.put(a[i],1);

}

int c=0;

for(Integer i:h.keySet()){

if(h.get(i)==k){

if(!h.containsKey(i\*2)){

c++;

}

}

}

System.out.println(c);

}

}

# *23) A)* Count of strings with frequency of each character at most K

Given an [array](https://www.geeksforgeeks.org/array-data-structure/) **arr[]** containing **N** [strings](http://www.geeksforgeeks.org/frequent-word-array-strings/) and an integer **K**, the task is to find the count of [strings](https://www.geeksforgeeks.org/category/data-structures/c-strings/) with the frequency of each character at most **K**

**Examples:**

***Input:****arr[] = { “abab”, “derdee”, “erre” }, K = 2****Output:****2****Explanation:****Strings with character frequency at most 2 are “abab”, “erre”. Hence count is 2*

***Input:****arr[] = {“ag”, “ka”, “nanana”}, K = 3****Output:****1*

***Code:***

import java.util.\*;

public class MyClass {

static Boolean count(String s,int k){

HashMap<Character,Integer> h=new HashMap<>();

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

char c=s.charAt(i);

if(h.containsKey(c)){

h.put(c,h.get(c)+1);

}else{

h.put(c,1);

}

}

for(Character i:h.keySet()){

if(h.get(i)==k)

return true;

}

return false;

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

sc.nextLine();

String s[]=new String[n];

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

s[i]=sc.nextLine();

}

int c=0;

int k=sc.nextInt();

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

if(count(s[i],k)){

c++;

}

}

System.out.println(c);

}

}

# *B)* Count all elements in the array which appears at least K times after their first occurrence

Given an array **arr[]** of **N** integer elements and an integer **K**. The task is to count all distinct **arr[i]** such that **arr[i]** appears at least **K** times in the index range **i + 1** to **n – 1**.

**Examples:**

***Input:****arr[] = {1, 2, 1, 3}, K = 1****Output:****1   
arr[0] = 1 is the only element that appears at least once in the index range [1, 3] i.e. arr[2]*

***Input:****arr[] = {1, 2, 3, 2, 1, 3, 1, 2, 1}, K = 2****Output:****2*

***Code:***

import java.util.\*;

public class MyClass{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

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

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

a[i]=sc.nextInt();

if(h.containsKey(a[i])){

h.put(a[i],h.get(a[i])+1);

}else{

h.put(a[i],1);

}

}

int c=0;

int k=sc.nextInt();

for(Integer i:h.keySet()){

if(h.get(i)==k+1)

c++;

}

System.out.print(c);

}

}

# *24) A)* Print all characters of string whose frequency is a power of K

Given [string](https://www.geeksforgeeks.org/string-class-in-java/) **str** of size **N**, the task is to print the characters of string whose frequency is a power of **K** in a lexicographically sorted order.

**Examples:**

***Input:****str = “aaacbb” K = 2****Output:****bbc****Explanation:****Frequency of a is 3 which is not the power of 2. Frequency of c is 1 and frequency of b is 2 which are the power of 2.*

***Input:****str = “geeksgeekgeeks” K = 3****Output:****eeeeeegggkkk*

***Code:***

import java.util.\*;

public class MyClass {

static Boolean isPower(int n,int k){

int i=1;

while(i<=n){

if(i==n)

return true;

i=i\*k;

}

return false;

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

int k=sc.nextInt();

HashMap<Character,Integer> h=new HashMap<>();

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

char c=s.charAt(i);

if(h.containsKey(c))

h.put(c,h.get(c)+1);

else

h.put(c,1);

}

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

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

char c=s.charAt(i);

if(isPower(h.get(c),k))

ans.add(c);

}

Collections.sort(ans);

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

}

}

# *B)* Find all matrix elements which are minimum in their row and maximum in their column

Given a matrix **mat[][]** of size **M \* N**, the task is to find all matrix elements which are minimum in their respective row and maximum in their respective column. If no such element is present, print **-1**.

**Examples:**

***Input:****mat[][] = {{1, 10, 4}, {9, 3, 8}, {15, 16, 17}}****Output:****15****Explanation:*** *15 is the only element which is maximum in its column {1, 9,****15****} and minimum in its row {****15****, 16, 17}.*

***Input:****m[][] = {{10, 41}, {3, 5}, {16, 2}}****Output:****-1*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int m=sc.nextInt();

int n=sc.nextInt();

int a[][]=new int[m][n];

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

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

a[i][j]=sc.nextInt();

}

}

int row[]=new int[m];

int col[]=new int[n];

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

int min=Integer.MAX\_VALUE;

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

min=Math.min(min,a[i][j]);

}

row[i]=min;

}

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

int max=Integer.MIN\_VALUE;

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

max=Math.max(max,a[j][i]);

}

col[i]=max;

}

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

if(row[i]==col[n-1-i]){

System.out.println(row[i]);

return;

}

}

System.out.println(-1);

}

}

# *25) A)* Replace the odd positioned elements with their cubes and even positioned elements with their squares

Given an array **arr[]** of **n** elements, the task is to replace all the odd positioned elements with their cubes and even positioned elements with their squares i.e. the resultant array must be **{arr[0]3, arr[1]2, arr[2]3, arr[3]2, …}**.  
**Examples:**

***Input:****arr[]= {2, 3, 4, 5}****Output:****8 9 64 25   
Updated array will be {23, 32, 43, 52} -> {8, 9, 64, 25}****Input:****arr[] = {3, 4, 5, 2}****Output:****27 16 125 4*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

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

if(i%2==0)

a[i]=a[i]\*a[i]\*a[i];

else

a[i]=a[i]\*a[i];

}

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

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

}

}

*B)* Given a string s, reverse only all the vowels in the string and return it.

The vowels are 'a', 'e', 'i', 'o', and 'u', and they can appear in both lower and upper cases, more than once.

Example 1:

Input: s= "hello"

Output: "holle"

Example 2:

Input: s= "AEIOU"

Output: "UOIEA"

Example 3:

Input: s= "DesignGUrus"

Output: "DusUgnGires"

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

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

k.add('a');

k.add('e');

k.add('i');

k.add('o');

k.add('u');

String t="";

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

char c=s.charAt(i);

if(k.contains(c)){

t+=c;

}

}

StringBuffer b=new StringBuffer(t);

b.reverse();

String l=b.toString();

String q="";

int j=0;

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

char c=s.charAt(i);

if(k.contains(c)){

q+=l.charAt(j);

j++;

}else{

q+=c;

}

}

System.out.println(q);

}

}

*26) A)* Given an array of strings words and two different strings that already exist in the array word1 and word2, return the shortest distance between these two words in the list.

Example 1:

Input: words = ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"], word1 = "fox", word2 = "dog"

Output: 5

Explanation: The distance between "fox" and "dog" is 5 words.

Example 2:

Input: words = ["a", "c", "d", "b", "a"], word1 = "a", word2 = "b"

Output: 1

Explanation: The shortest distance between "a" and "b" is 1 word

**Code:**

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

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

sc.nextLine();

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

a.add(sc.nextLine());

}

String word1=sc.nextLine();

String word2=sc.nextLine();

System.out.println(Math.abs(a.indexOf(word2)-a.lastIndexOf(word1)));

}

}

*B)* Given an array of integers nums, return the number of good pairs.

A pair (i, j) is called good if nums[i] == nums[j] and i < j.

Example 1:

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

Output: 4

Explanation: There are 4 good pairs, here are the indices: (0,3), (0,4), (3,4), (2,5).

Example 2:

Input: nums = [1,1,1,1]

Output: 6

Explanation: Each pair in the array is a 'good pair'.

Example 3:

Input: words = nums = [1,2,3]

Output: 0

Explanation: No number is repeating.

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

int c=0;

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

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

if(a[i]==a[j])

c++;

}

}

System.out.println(c);

}

}

# *27) A)* Check if all given strings are isograms or not

Given an array **arr** containing **N** strings, the task is to check if all strings are [isogram](https://www.geeksforgeeks.org/check-string-isogram-not/) or not. If they are, print **Yes**, otherwise **No**.

*An****Isogram****is a word in which no letter occurs more than once.*

**Examples:**

***Input:****arr[] = {“abcd”, “derg”, “erty”}****Output:****Yes*

***Input:****arr[] = {“agka”, “lkmn”}****Output:****No*

***Code:***

import java.util.\*;

public class MyClass {

static Boolean isIsogram(String s){

HashMap<Character,Integer> h=new HashMap<>();

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

char c=s.charAt(i);

if(h.containsKey(c))

return false;

h.put(c,1);

}

return true;

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

String s[]=new String[n];

sc.nextLine();

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

s[i]=sc.nextLine();

}

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

if(!isIsogram(s[i])){

System.out.println("No");

return;

}

}

System.out.println("Yes");

}

}

# *B)* Sum of all odd frequency elements in a Matrix

Given a NxM matrix of integers containing duplicate elements. The task is to find the sum of all odd occurring elements in the given matrix. That is the sum of all such elements whose frequency is odd in the matrix.

**Examples**:

**Input** : mat[] = {{1, 1, 2},

{2, 3, 3},

{4, 5, 3}}

**Output** : 18

The odd occurring elements are 3, 4, 5 and their number

of occurrences are 3, 1, 1 respectively. Therefore,

sum = 3+3+3+4+5 = 18.

**Input** : mat[] = {{10, 20},

{40, 40}}

**Output** : 30

**Code:**

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int m=sc.nextInt();

int a[][]=new int[n][m];

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

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

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

a[i][j]=sc.nextInt();

}

}

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

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

int k=a[i][j];

if(h.containsKey(k))

h.put(k,h.get(k)+1);

else

h.put(k,1);

}

}

int s=0;

for(Integer i:h.keySet()){

if(h.get(i)%2!=0)

s+=i\*h.get(i);

}

System.out.println(s);

}

}

# *28) A)* Modify a string by circularly shifting each character to the right by respective frequencies

Given a [string](https://www.geeksforgeeks.org/string-data-structure/) **S** consisting of lowercase English alphabets, the task is to right shift each character of the given string **S** circularly by its frequency.

*Circular shifting of characters refers to shifting character ‘z’ to ‘a’, as its next character.*

**Examples:**

***Input:****S = “geeksforgeeks”****Output:****iiimugpsiiimu****Explanation:*** *Following changes are made on the string S:*

1. *Frequency of ‘g’ is 2. Therefore, shifting the character ‘g’ by 2 becomes ‘i’.*
2. *Frequency of ‘e’ is 4. Therefore, shifting the character ‘e’ by 4 becomes ‘i’.*
3. *Frequency of ‘k’ is 2. Therefore, shifting the character ‘k’ by 2 becomes ‘m’.*
4. *Frequency of ‘s’ is 2. Therefore, shifting the character ‘s’ by 2 becomes ‘u’.*
5. *Frequency of ‘f’ is 1. Therefore, shifting the character ‘f’ by 1 becomes ‘g’.*
6. *Frequency of ‘o’ is 1. Therefore, shifting the character ‘o’ by 1 becomes ‘p’.*
7. *Frequency of ‘r’ is 1. Therefore, shifting the character ‘r’ by 1 becomes ‘s’.*

*After the above shifting of characters, the string modifies to “iiimugpsiiimu”.*

***Input:****S = “aabcadb”****Output:****ddddded*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

HashMap<Character,Integer> h=new HashMap<>();

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

char c=s.charAt(i);

if(h.containsKey(c))

h.put(c,h.get(c)+1);

else

h.put(c,1);

}

String k="";

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

char c=s.charAt(i);

k+=(char)((int)c+h.get(c));

}

System.out.println(k);

}

}

# *B)* Print matrix in snake pattern

Given an n x n matrix. In the given matrix, you have to print the elements of the matrix in the snake pattern.

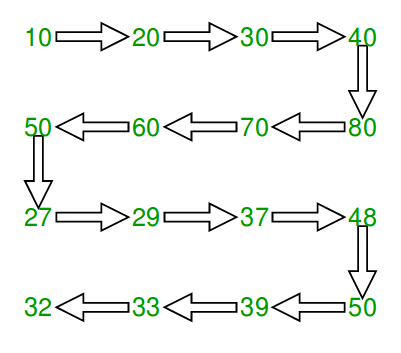
**Examples :**

***Input:****mat[][] = { {10, 20, 30, 40},*

*{15, 25, 35, 45},*

*{27, 29, 37, 48},*

*{32, 33, 39, 50}};****Output:****10 20 30 40 45 35 25 15 27 29 37 48 50 39 33 32*



***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[][]=new int[n][n];

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

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

a[i][j]=sc.nextInt();

}

}

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

if(i%2==0){

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

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

}

}else{

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

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

}

}

System.out.println();

}

}

}

# *29) A)* Program to check if all characters have even frequency

Given a string S consisting only of lowercase letters check if the string has all characters appearing even times.

**Examples:**

***Input :****abaccaba****Output :****Yes****Explanation:****‘a’ occurs four times, ‘b’ occurs twice, ‘c’ occurs twice and the other letters occur zero times.*

***Input:****hthth****Output :****No*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

HashMap<Character> h=new HashMap<>();

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

char c=s.charAt(i);

if(h.containsKey(c)){

h.put(c,h.get(c)+1);

}

else{

h.put(c,1);

}

}

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

if(h.get(c)%2!=0){

System.out.println("No");

return;

}

}

System.out.println("Yes");

}

}

# *B)* Squares of Matrix Diagonal Elements

You have given an integer matrix with odd dimensions. Find the square of the diagonal elements on both sides.

**Examples:**

Input : 1 2 3

4 5 6

7 8 9

Output : Diagonal one: 1 25 81

Diagonal two: 9 25 49

Input : 2 5 7

3 7 2

5 6 9

Output : Diagonal one : 4 49 81

Diagonal two : 49 49 25

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[][]=new int[n][n];

String s1="",s2="";

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

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

a[i][j]=sc.nextInt();

}

}

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

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

if(i==j){

s1+=a[i][j]\*a[i][j]+" ";

}

if(i+j==n-1){

s2+=a[i][j]\*a[i][j]+" ";

}

}

}

System.out.println("Diagonal one: "+s1+"\n"+"Diagonal two: "+s2);

}

}

# *30) A)* Count of strings that does not contain any character of a given string

Given an array **arr** containing **N** strings and a string **str**, the task is to find the number of strings that do not contain any character of string **str**.

**Examples:**

***Input:****arr[] = {“abcd”, “hijk”, “xyz”, “ayt”}, str=”apple”****Output:****2****Explanation:****“hijk” and “xyz” are the strings that do not contain any character of str*

***Input:****arr[] = {“apple”, “banana”, “pear”}, str=”nil”****Output:****1*

**Code:**

import java.util.\*;

public class MyClass {

static Boolean check(String s,String k){

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

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

a.add(k.charAt(i));

}

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

if(a.contains(s.charAt(i)))

return false;

}

return true;

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

sc.nextLine();

String s[]=new String[n];

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

s[i]=sc.nextLine();

}

int c=0;

String k=sc.nextLine();

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

if(check(s[i],k))

c++;

}

System.out.println(c);

}

}

*B)* Given an m x n matrix, find all common elements present in all rows in O(mn) time and one traversal of matrix.

**Example:**

**Input:**

mat[4][5] = {{1, 2, 1, 4, 8},

{3, 7, 8, 5, 1},

{8, 7, 7, 3, 1},

{8, 1, 2, 7, 9},

};

**Output:**

1 8 or 8 1

8 and 1 are present in all rows.

**Code:**

import java.util.\*;

public class MyClass {

static int[] fun(int[] a,int[] b){

HashSet<Integer> h=new HashSet<>();

int c=0;

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

for(int j=0;j<b.length;j++){

if(a[i]==b[j])

h.add(a[i]);

}

}

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

for(int j=0;j<a.length;j++){

if(b[i]==a[j])

h.add(b[i]);

}

}

int[] ans=new int[h.size()];

for(Integer i:h){

ans[c++]=i;

}

return ans;

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int m=sc.nextInt();

int n=sc.nextInt();

int a[][]=new int[m][n];

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

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

a[i][j]=sc.nextInt();

}

int common[]=fun(a[0],a[1]);

for(int i=1;i<m;i++){

common=fun(a[i],common);

}

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

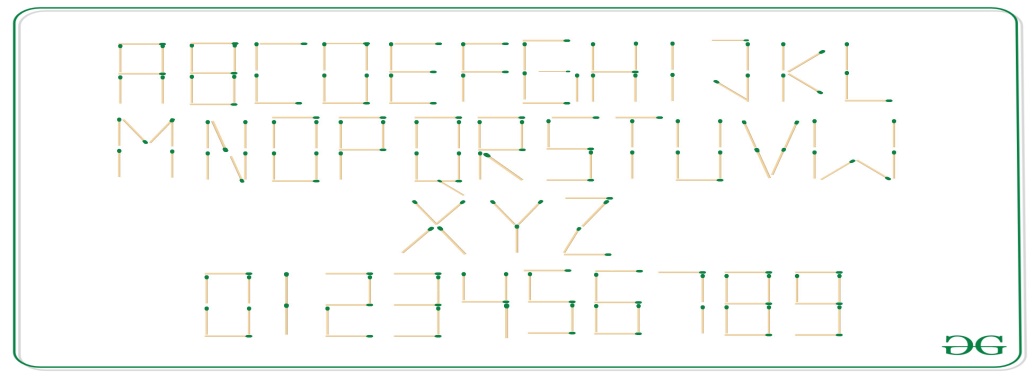
System.out.println(common[i]);

}

}

# *31) A)* Count of sticks required to represent the given string

Given a string **str** of uppercase alphabets and numbers, the task is to find the number of matchsticks required to represent it.



***Input:****str = “ABC2”****Output:****22****Explanation:****6 sticks are required to represent A,   
7 sticks are required to represent B,   
4 sticks are required to represent C,   
5 sticks are required to represent 2.   
Therefore the total number of matchsticks required is 6 + 7 + 4 + 5 = 22.****Input:****str = “GEEKSFORGEEKS”****Output:****66****Explanation:****6 sticks are required to represent G,   
5 sticks are required to represent E,   
4 sticks are required to represent K,   
5 sticks are required to represent S,   
4 sticks are required to represent F,   
6 sticks are required to represent O,   
6 sticks are required to represent R.   
Therefore the total number of matchsticks required is 6 + 5 + 5 + 4 + 5 + 4 + 6 + 6 + 6 + 5 + 5 + 4 + 5 = 17.*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int a[] = { 6, 7, 4, 6, 5, 4, 6,

5, 2, 4, 4, 3, 6, 6,

6, 5, 7, 6, 5, 3, 5,

4, 6, 4, 3, 4 };

int n[] = { 6, 2, 5, 5, 4, 5, 6,

3, 7, 6 };

String s=sc.nextLine();

int sum=0;

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

char c=s.charAt(i);

if(c>='a' && c<='z'){

sum+=a[c-'a'];

}else{

sum+=n[c-'0'];

}

}

System.out.println(sum);

}

}

# *B)* Maximum difference between first and last indexes of an element in array

Given an array of n integers. The task is to find the difference of first and last index of each distinct element so as to maximize the difference.

**Examples:**

Input : {2, 1, 3, 4, 2, 1, 5, 1, 7}

Output : 6

Element **1** has its **first index** = 1 and **last index** = 7 **Difference** = 7 - 1 = 6

Other elements have a smaller first and last index difference

Input : {2, 2, 1, 1, 8, 8, 3, 5, 3}

Output : 2

Maximum difference is for indexes of element 3.

**Code:**

import java.util.\*;

public class MyClass {

static int fun(int a[],int n){

int r=a.length-1,l=0;

while(l<=r){

if(a[l]==a[r]){

break;

}

l++;

r--;

}

return r-l;

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

int max=0,diff=0;

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

diff=fun(a,a[i]);

max=Math.max(max,diff);

}

System.out.println(max);

}

}

# *32) A)* Most similar string

Given a string **str** and an [array](https://www.geeksforgeeks.org/introduction-to-arrays/) of strings **arr[]** of size **N**, the task is to print a string from arr[], which has maximum count of matching characters with str.

**Examples:**

***Input:****str = “vikas”, N = 3, arr[] = [“preeti”, “khusbu”, “katherina”]****Output:****“katherina”****Explanation:****Number of similar characters between Str and each string in D[ ] are,    
“preeti” = 1   
“khusbu” = 2   
“katherina” = 3   
Hence, “katherina” has maximum matching characters.*

***Input:****str = “gfg”, N = 3, arr[] = [“goal”, “fog”, “abc”]****Output:****“fog”****Explanation:*** *Number of similar characters between Str and each string in D[ ] are,    
“goal” = 1   
“fog” = 2   
“abc” = 0   
Hence, “fog” has maximum matching characters.*

***Code:***

import java.util.\*;

public class MyClass {

static int matching(String a,String b){

int k1[]=new int[26];

int k2[]=new int[26];

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

k1[a.charAt(i)-'a']++;

}

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

k2[b.charAt(i)-'a']++;

}

int c=0;

for(int i=0;i<26;i++){

if((k1[i]>=k2[i] || k2[i]>=k1[i]) && k1[i]!=0 && k2[i]!=0)

c++;

}

return c;

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

sc.nextLine();

String words[]=new String[n];

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

words[i]=sc.nextLine();

}

String s=sc.nextLine();

int max=0;

String ans="";

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

int m=matching(words[i],s);

if(max<m){

max=m;

ans=words[i];

}

}

System.out.println(ans);

}

}

# *B)* Count number of free cell present in the Matrix

Given a Matrix size **N\*N** and an integer **K**. Initially, the matrix contains only **0**. You are given K tasks and for each task, you are given two coordinates (**r, c**). Where coordinates (r, c) denotes the **rth** row and the **cth** column of the given matrix. You have to perform each task sequentially in the given order. For each task, You have to put 1 in all rthrow cells and all the cth column cells. After each task, You have to calculate the number of 0 present in the matrix and return it.

**Examples:**

***Input****: N = 3, K = 3  
1 1  
1 2  
2 1****Output****: 4 2 1****Explanation:****After 1st Operation, all the cells of the 2nd row and  2nd column will be filled by 1. So, the remaining cell with the value 0 will be 4. After 2nd operation, all the second row and all the third columns will be filled by 1. So, the remaining cell with value will be 2. After 3rd operation number of cells having the value 0 will be 1.*

***Input****: N = 2, K = 2  
0 1  
0 0****Output****: 1 0****Explanation:****After 1st operation, all the cells of the 1st row and 2nd column will be filled by 1. So, the remaining cell with the value 0 will be 1. After 2nd operation, all the cells of the 1st row and 1st column will be filled by 1. So, the remaining cell with the value 0 will be 0.*

***Code:***

import java.util.\*;

public class MyClass {

static int count(int[][] a,int n){

int c=0;

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

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

if(a[i][i]==0){

c++;

}

}

}

return c;

}

static int[][] change(int[][] a,int n,int k1,int k2){

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

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

if(i==k1 || j==k2)

a[i][j]=1;

}

}

return a;

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int k=sc.nextInt();

int a[][]=new int[n][n];

int t[][]=new int[k][2];

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

for(int j=0;j<2;j++){

t[i][j]=sc.nextInt();

}

}

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

for(int j=0;j<2;j++){

a=change(a,n,i,j);

}

System.out.print(count(a,n)+" ");

}

}

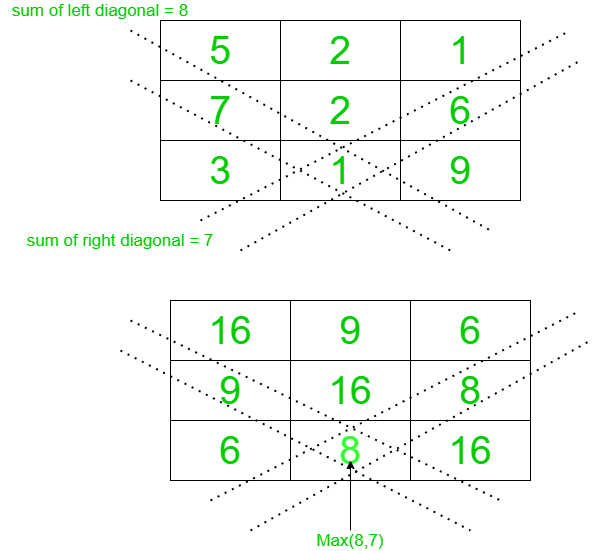
}

# *33)A)* Modify a matrix by replacing each element with the maximum of its left or right diagonal sum

Given a matrix **mat[][]** with dimensions **M \* N**, the task is to replace each matrix elements with the maximum sum of its left or right diagonal.

**Examples:**

***Input:****mat[][] = {{5, 2, 1}, {7, 2, 6}, {3, 1, 9}}****Output:****16  9   6   
9 16   8  
6   8 16****Explanation:*** *Replace each element with max(sum of right diagonal, sum of left diagonal).  
Follow the diagram below to understand more clearly.*



***nput:****mat[][] = {{1, 2}, {3, 4}}****Output:****5 5  
5 5*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[][]=new int[n][n];

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

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

a[i][j]=sc.nextInt();

}

}

int t[][]=new int[n][n];

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

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

int cl=a[i][j],cr=a[i][j];

int i1=i-1,j1=j-1;

while(i1>=0 && j1>=0){

cl+=a[i1][j1];

i1--;

j1--;

}

i1=i+1;j1=j+1;

while(i1<=n-1 && j1<=n-1){

cl+=a[i1][j1];

i1++;

j1++;

}

i1=i-1;j1=j+1;

while(i1>=0 && j1<=n-1){

cr+=a[i1][j1];

i1--;

j1++;

}

i1=i+1;j1=j-1;

while(i1<=n-1 && j1>=0){

cr+=a[i1][j1];

i1++;

j1--;

}

t[i][j]=Math.max(cl,cr);

}

}

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

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

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

}

System.out.println();

}

}

}

# *B)* Print characters in decreasing order of frequency

Given string **str**, the task is to print the characters in decreasing order of their frequency. If the frequency of two characters is the same then sort them in descending order alphabetically.  
**Examples:**   
 ***Input:****str = “geeksforgeeks”****Output:****e – 4   
s – 2   
k – 2   
g – 2   
r – 1   
o – 1   
f – 1****Input:****str = “bbcc”****Output:****c – 2   
b – 2*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

HashMap<Character,Integer> h=new HashMap<>();

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

char c=s.charAt(i);

if(h.containsKey(c))

h.put(c,h.get(c)+1);

else

h.put(c,1);

}

ArrayList<Integer> a=new ArrayList<>();

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

a.add(h.get(c));

}

Collections.sort(a,Collections.reverseOrder());

int i=0;

while(i<a.size()){

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

if(h.get(c)==a.get(i)){

System.out.println(c+" - "+a.get(i));

i++;

}

}

}

}

}

# *34) A)* Count of unique rows in a given Matrix

Given a **2D** matrix arr of size **N\*M**containing lowercase English letters, the task is to find the number of unique rows in the given matrix.

**Examples:**

***Input:****arr[][]= { {‘a’, ‘b’, ‘c’, ‘d’},    
                          {‘a’, ‘e’, ‘f’, ‘r’},    
                          {‘a’, ‘b’, ‘c’, ‘d’},    
                          {‘z’, ‘c’, ‘e’, ‘f’} }****Output:****2****Explanation:****The 2nd and the 4th row are unique.*

***Input:****arr[][]={{‘a’, ‘c’},    
                       {‘b’, ‘d’},    
                       {‘e’, ‘f’}}****Output:****3*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int m=sc.nextInt();

sc.nextLine();

String words[]=new String[n];

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

words[i]=sc.nextLine();

}

HashMap<String,Integer> h=new HashMap<>();

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

if(h.containsKey(words[i]))

h.put(words[i],h.get(words[i])+1);

else

h.put(words[i],1);

}

int c=0;

for(String k:h.keySet())

if(h.get(k)==1)

c++;

System.out.println(c);

}

}

# *B)* Count of strings with frequency of each character at most X and length at least Y

Given an array **arr[]** of strings and integers **X** and **Y**, the task is to find the count of strings with **frequency** of each character **at most X** and **length** of the string **at least Y**.

**Examples:**

***Input:****arr[] = { “ab”, “derdee”, “erre” }, X = 2, Y = 4****Output:****1****Explanation:****Strings with character frequency at most 2 and   
length at least 4 is “erre”. Hence count is 1*

***Input:****arr[] = {“ag”, “ka”, “nanana”}, X = 3, Y = 2****Output:****3*

***Code:***

import java.util.Arrays;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int n = sc.nextInt();

sc.nextLine();

String[] arr = new String[n];

System.out.print("Enter the elements of the array: ");

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

arr[i] = sc.nextLine();

}

System.out.print("Enter the value of X: ");

int X = sc.nextInt();

System.out.print("Enter the value of Y: ");

int Y = sc.nextInt();

int count = 0;

for (String s : arr) {

Map<Character, Integer> map = new HashMap<>();

for (char c : s.toCharArray()) {

map.put(c, map.getOrDefault(c, 0) + 1);

}

boolean flag = true;

for (int i : map.values()) {

if (i > X) {

flag = false;

break;

}

}

if (flag && s.length() >= Y) {

count++;

}

}

System.out.println("Input: arr[] = " + Arrays.toString(arr) + ", X = " + X + ", Y = "

+ Y);

System.out.println("Output: " + count);

System.out.println("Explanation: Strings with character frequency at most " + X + "

and length at least " + Y + " is " + Arrays.toString(arr) + ". Hence count is " + count +

".");

}

}

# *35) A)* Sum of all elements repeating ‘k’ times in an array

Given an array, we have to find the sum of all the elements repeating k times in an array. We need to consider every repeating element just once in the sum.

**Examples:**

**Input :** arr[] = {2, 3, 9, 9}

k = 1

**Output :** 5

2 + 3 = 5

**Input :** arr[] = {9, 8, 8, 8, 10, 4}

k = 3

**Output :** 8

**Code:**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int n = sc.nextInt();

int[] arr = new int[n];

System.out.print("Enter the elements of the array:

");

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

arr[i] = sc.nextInt();

}

System.out.print("Enter the value of k: ");

int k = sc.nextInt();

Map<Integer, Integer> map = new HashMap<>();

for (int i : arr) {

map.put(i, map.getOrDefault(i, 0) + 1);

}

int sum = 0;

for (Map.Entry<Integer, Integer> entry :

map.entrySet()) {

if (entry.getValue() == k) {

sum += entry.getKey();

}

}

System.out.println("The sum of all elements repeating

" + k + " times in the array is " + sum + ".");

}

}

# *B)* Most frequent word in first String which is not present in second String

Given two string ‘S1’ and ‘S2’, the task is to return the most frequent (which is used the maximum number of times) word from ‘S1’ that is not present in ‘S2’. If more than one word is possible then print lexicographically smallest among them.

**Examples:**

***Input:****S1 = “geeks for geeks is best place to learn”, S2 = “bad place”****Output:****geeks   
“geeks” is the most frequent word in S1 and is also not present in S2.   
The frequency of “geeks” is 2*

***Input:****S1 = “the quick brown fox jumps over the lazy dog”, S2 = “the brown fox jumps”****Output:****dog   
All the words have frequency 1.   
The lexicographically smallest word is “dog”*

***Code:***

import java.util.\*;

public class MyClass {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String a=sc.nextLine();

String b=sc.nextLine();

String[] s1=a.split(" ");

String[] s2=b.split(" ");

HashMap<String,Integer> h=new HashMap<>();

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

String c=s1[i];

if(h.containsKey(c))

h.put(c,h.get(c)+1);

else

h.put(c,1);

}

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

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

k.add(s2[i]);

}

ArrayList<Integer> d=new ArrayList<>();

for(String c:h.keySet()){

d.add(h.get(c));

}

Collections.sort(d,Collections.reverseOrder());

int i=0;

while(i<d.size()){

for(String c:h.keySet()){

if(h.get(c)==d.get(i) && !k.contains(c)){

System.out.println(c);

return;

}

}

i++;

}

}

}

# *36) A)* Find element with highest frequency in given nested Array

Given an [array](https://www.geeksforgeeks.org/introduction-to-arrays/) **arr[]** of **N** integers. The task is to create a frequency array **freq[]** of the given array **arr[]** and find the maximum element of the frequency array. If two elements have the same frequency in the array **freq[]**, then return the element which has a smaller value.

**Examples:**

***Input:****arr[] = {1, 1, 1, 2, 3, 2, 2, 3, 5, 5, 5, 5, 4, 4, 4, 4, 4};****Output:****3****Explanation:****frequency of elements is given by:   
val -> freq[]   
1 -> 3   
2 -> 3   
3 -> 2   
4 -> 5   
5 -> 4   
Element 3 has the maximum frequency in the frequency array.*

***Input:****arr[] = { 3, 5, 15, 51, 15, 14, 14, 14, 14, 4};****Output:****1****Explanation:****frequency of elements is given by:   
val -> freq[]   
3 -> 1   
4 -> 1   
5 -> 1   
14 -> 4   
15 -> 2   
51 -> 1   
Element 1 has the maximum frequency in the frequency array.*

***Code:***

import java.util.Arrays;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int n = sc.nextInt();

int[] arr = new int[n];

System.out.print("Enter the elements of the array: ");

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

arr[i] = sc.nextInt();

}

int maxCount = 0;

int elementHavingMaxFreq = 0;

Map<Integer, Integer> map1 = new HashMap<>();

Map<Integer, Integer> map2 = new HashMap<>();

for (int i : arr) {

map1.put(i, map1.getOrDefault(i, 0) + 1);

}

for (int i : map1.values()) {

map2.put(i, map2.getOrDefault(i, 0) + 1);

}

for (Map.Entry<Integer, Integer> entry : map2.entrySet()) {

if (entry.getValue() > maxCount) {

maxCount = entry.getValue();

elementHavingMaxFreq = entry.getKey();

} else if (entry.getValue() == maxCount) {

elementHavingMaxFreq = Math.min(elementHavingMaxFreq, entry.getKey());

}

}

System.out.println(elementHavingMaxFreq);

}

}

# *B)* Remove duplicates from string keeping the order according to last occurrences

Given a string, remove duplicate characters from the string, retaining the last occurrence of the duplicate characters. Assume the characters are case-sensitive.

**Examples:**

***Input :****geeksforgeeks****Output :****forgeks   
Explanation : Please note that we keep only last occurrences of repeating characters in same order as they appear in input. If we see result from right side, we can notice that we keep last ‘s’, then last ‘k’ , and so on.*

***Input :****hi this is sample test****Output :****hiampl est   
Explanation : Here, the output contains last occurrence of every character, even ” “(spaces), and removing the duplicates. Like in this example, there are 4 spaces count, so we have only the last occurrence of space in it removing the others. And there is only last occurrence of each character without repetition.*

***Code:***

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = sc.nextLine();

String output = removeDuplicates(input);

System.out.println("Output: " + output);

}public static String removeDuplicates(String str) {

int n = str.length();

StringBuilder sb = new StringBuilder();

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

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

char c = str.charAt(i);

if (!map.containsKey(c)) {

map.put(c, i);

}

}

for (Map.Entry<Character, Integer> entry : map.entrySet()) {

sb.append(entry.getKey());

}

return sb.reverse().toString();

}

}

# *37) A)* Find element in a sorted array whose frequency is greater than or equal to n/2.

Given a sorted array of length n, find the number in array that appears more than or equal to n/2 times. It is given that such element always exists.

**Examples:**

Input : 2 3 3 4

Output : 3

Input : 3 4 5 5 5

Output : 5

Input : 1 1 1 2 3

Output : 1

**Code:**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the array: ");

String[] input = sc.nextLine().split(" ");

int[] arr = new int[input.length];

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

arr[i] = Integer.parseInt(input[i]);

}

int majority = arr[arr.length / 2];

System.out.println("Output: " + majority);

}

}

# *B)* Remove even frequency characters from the string

Given a string ‘str’, the task is to remove all the characters from the string that have even frequencies.

**Examples:**

**Input:** str = "aabbbddeeecc"

**Output:** bbbeee

The characters a, d, c have even frequencies

So, they are removed from the string.

**Input:** str = "zzzxxweeerr"

**Output:** zzzweee

***Code:***

import java.util.HashMap;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the string: ");

String str = sc.nextLine();

HashMap<Character, Integer> freq = new HashMap<>();

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

char c = str.charAt(i);

freq.put(c, freq.getOrDefault(c, 0) + 1);

}

StringBuilder result = new StringBuilder();

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

char c = str.charAt(i);

if (freq.get(c) % 2 != 0) {

result.append(c);

}

}

System.out.println("Output: " + result.toString());

}

}