**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

**PROGRAM**

**import java.util.\*;**

**public class Main**

**{**

**static String name(String s)**

**{**

**String sa="";**

**sa=s1.charAt(0)+" ";**

**for(int i=1;i<s.length();i++)**

**{**

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

**{**

**sa=sa+s.charAt(i+1)+" ";**

**}**

**}**

**return sa.toUpperCase();**

**}**

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

**String s="abhishek kumar singh";**

**System.out.println(name(s));**

**}**

**}**

# 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*

**Program: import java.util.\*;**

**public class Main**

**{**

**static int Gcd(int a,int b)**

**{**

**if (a == 0)**

**return b;**

**if (b == 0)**

**return a;**

**if (a == b)**

**return a;**

**if (a > b)**

**return Gcd(a - b, b);**

**return Gcd(a, b - a);**

**}**

**static int freLesHig(int[] a,int n)**

**{**

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

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

**{**

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

**}**

**int min=999999;**

**int max=hm.get(a[0]);**

**int MIn=0;**

**int Max=0;**

**for(Integer x:hm.keySet())**

**{**

**if(hm.get(x)<min)**

**{**

**min=hm.get(x);**

**MIn=x;**

**}**

**if(hm.get(x)>max)**

**{**

**max=hm.get(x);**

**Max=x;**

**}**

**}**

**//System.out.println(MIn+" "+Max);**

**return Gcd(MIn,Max);**

**}**

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

**int[] a=new int[]{2, 2, 4, 4, 5, 5, 6, 6, 6, 6};**

**//int[] a=new int[]{3,3, 22, 2,2, 44, 44, 44, 44};**

**System.out.println(freLesHig(a,a.length));**

**//System.out.println(Gcd(22,44));**

**}**

**}**

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)**

**Program:**

import java.util.\*;

public class Main

{

static boolean check\_duck(String N){

int n=N.length();

int c=0;

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

if(N.charAt(i)!='0' && N.charAt(i+1)=='0')

{

c++;

}

}

if(c>0)

{

return true;

}

return false;

}

public static void main(String[] args) {

//String s="707069";

// String s="02364";

String s="01203";

if(check\_duck(s))

{

System.out.println("is duck number");

}

else{

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

}

}

}

# b)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*

**Program:**

import java.util.\*;

public class Main

{

static int KthFreqEle(int[] a,int n,int k)

{

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

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

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

{

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

}

for(Integer x:hm.keySet())

{

l.add(hm.get(x));

}

Collections.sort(l);

int n1=l.size();

int K=l.get(n1-k);

for(Integer x:hm.keySet())

{

if(hm.get(x)==K)

{

return x;

}

}

return 0;

}

public static void main(String[] args) {

int[] a=new int[]{1, 2, 2, 2, 4, 4, 4, 5, 5, 5, 5, 5, 7, 7, 8, 8, 8, 8};

//KthFreqEle(a,a.length,3);

System.out.println(KthFreqEle(a,a.length,1));

System.out.println(KthFreqEle(a,a.length,2));

System.out.println(KthFreqEle(a,a.length,3));

}

}

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

**Program:**

import java.util.\*;

public class Main

{

static String orgString(String s)

{

StringBuilder s1=new StringBuilder();

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

{

if(s.charAt(i)>='A'&& s.charAt(i)<='Z'||s.charAt(i)>='a'&& s.charAt(i)<='z'||s.charAt(i)==' ')

{

s1.append(s.charAt(i));

}

}

return s1.toString();

}

public static void main(String[] args) {

System.out.println(orgString("Hello!!!, he said ---and went."));

System.out.println(orgString("%welcome' to @geeksforgeek<s"));

}

}

# B)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*

**Program:**

import java.util.\*;

public class Main

{

static void atLeastMfre(int[] a,int n,int m)

{

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

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

{

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

}

for(Integer x:hm.keySet())

{

if(hm.get(x)>=m)

{

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

}

}

}

public static void main(String[] args) {

//int[] a=new int[]{2, 3, 2, 2, 3, 5, 6, 3};

int[] a=new int[]{1, 32, 2, 1, 33, 5, 1, 5};

atLeastMfre(a,a.length,2);

//System.out.println("Hello World");

}

}

# 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

**Program:**

**public class Main**

**{**

**static boolean isPrime(int n)**

**{**

**if(n==1)**

**{**

**return false;**

**}**

**if(n==2)**

**{**

**return true;**

**}**

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

**{**

**if(n%i==0)**

**{**

**return false;**

**}**

**}**

**return true;**

**}**

**static boolean isEqulid(int n)**

**{**

**int pod=1;**

**int n1=n-1;**

**for(int i=1;pod<=n1;i++)**

**{**

**if(isPrime(i))**

**{**

**pod=pod\*i;**

**if(pod==n1)**

**{**

**break;**

**}**

**}**

**}**

**if(pod+1==n)**

**{**

**return true;**

**}**

**return false;**

**}**

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

**int n=211;**

**int n1=231;**

**if(isEqulid(n))**

**{**

**System.out.println(n+" is a euclid Number");**

**}**

**else{**

**System.out.println(n+"is not a euclid Number");**

**}**

**}**

**}**

# 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

**Program:**

import java.util.\*;

public class Main

{

static void printPattern(int n)

{

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

{

for (int j = 'A'; j < 'A' + (2 \* n) - 1; j++)

{

if (j >= ('A' + n - 1) + i)

System.out.print((char)(('A' + n - 1) -

(j % ('A' + n - 1))));

else if (j <= ('A' + n - 1) - i)

System.out.print((char)j);

else

System.out.print(" ");

}

System.out.println();

}

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

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

int n=sc.nextInt();

printPattern(n);

System.out.println();

printPattern(2);

System.out.println();

printPattern(7);

System.out.println();

printPattern(9);

}

}

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.*

**Program:**

public class Main

{

static boolean isPrefectNum(int n)

{

if(n==1)

{

return false;

}

int sum=1;

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

{

if(n%i==0)

{

if(i\*i==n)

{

sum=sum+i;

}

else

{

sum += i + (n / i);

}

}

}

if(sum==n)

{

return true;

}

else{

return false;

}

}

static long sumOfPre(int n,int m)

{

long sum=0;

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

{

if(isPrefectNum(i))

{

sum=sum+i;

}

}

return sum;

}

public static void main(String[] args) {

System.out.println(sumOfPre(6,28

));

}

}

# 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

**Program:**

public class Main

{

public static void main(String args[])

{

// Custom input string

String str = "a\nyo\n";

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

System.out.println(s1.length);

}

}

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

**Program:**

import java.util.\*;

public class Main

{

static char extraChar(String s1,String s2)

{

String sMax;

String sMin;

if(s1.length()>s2.length())

{

sMax=s1;

sMin=s2;

}

else

{

sMax=s2;

sMin=s1;

}

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

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

{

l.add(sMax.charAt(i));

}

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

{

l.remove(Character.valueOf(sMin.charAt(i)));

}

return l.get(0);

}

public static void main(String[] args) {

String s1="kxml";

String s2="klxml";

System.out.println(extraChar(s1,s2));

}

}

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)

**Program:**

**import java.util.\*;**

**public class Main**

**{**

**static int uniqueNum(int[] a,int n)**

**{**

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

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

**{**

**h.add(a[i]);**

**}**

**return n-h.size();**

**}**

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

**int[] a=new int[]{1,2,3,4,3,2};**

**System.out.println(uniqueNum(a,a.length));**

**}**

**}**

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

**Program:**

import java.util.\*;

public class Main

{

static String NameIS(String s)

{

String ans="";

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

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

{

if(i==s1.length-1)

{

ans=ans+s1[i];

}

else{

ans=ans+s1[i].charAt(0)+" ";

}

}

return ans;

}

public static void main(String[] args) {

String s1="Ishita Bhuiya";

System.out.println(NameIS(s1));

}

}

B)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)

**Program:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import java.util.\*;

public class Main

{

static int MoniterIS(int[] a,int n)

{

int hal=n/2;

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

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

{

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

}

int max=-1;

for(Integer x:hm.keySet())

{

if(hm.get(x)>hal)

{

max=x;

}

}

return max;

}

public static void main(String[] args) {

int[] a=new int[]{1, 3, 2, 2, 2};

System.out.println(MoniterIS(a,a.length));

}

}

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*

**Program:**

import java.util.\*;

public class Main

{

static boolean frequChar(String s1,String s2)

{

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

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

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

{

h1.put(s1.charAt(i),h1.getOrDefault(s1.charAt(i),0)+1);

}

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

{

h2.put(s2.charAt(i),h2.getOrDefault(s2.charAt(i),0)+1);

}

for(Character x:h1.keySet())

{

int min=Math.min(h2.getOrDefault(x,0),h1.getOrDefault(x,0));

int max=Math.max(h2.getOrDefault(x,0),h1.getOrDefault(x,0));

if(min==0)

{

continue;

}

if(max%min!=0)

{

return false;

}

}

return true;

}

public static void main(String[] args) {

// String s1="hhdwjwqq";

// String s2="qwjdddhhh";

String s1 = "aabccdd", s2 = "bbbaaaaccddd";

if(frequChar(s1,s2))

{

System.out.println("YES");

}

else{

System.out.println("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.

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)

**Program:**

public class Main

{

static int indexOfStud(int[] a,int n,int key)

{

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

{

if(a[i]==key)

{

return i;

}

}

return -1;

}

public static void main(String[] args) {

int[] a=new int[]{60, 61, 62, 63, 63, 64, 65, 66, 67, 66};

int k=67;

System.out.println(indexOfStud(a,a.length,k));

}

}

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’]*

# Program:

public class Main

{

static boolean Letter(char ch)

{

if(!Character.isLetter(ch))

{

return false;

}

return true;

}

static boolean isPangram(String s,int n)

{

s=s.toLowerCase();

boolean[] alphabet=new boolean[26];

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

{

if(Letter(s.charAt(i))){

int ind=s.charAt(i)-'a';

alphabet[ind]=true;

}

}

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

{

if(!alphabet[i])

{

return false;

}

}

return true;

}

public static void main(String[] args) {

// String s="Abcdefghijklmonpqrstuvwxyz";

//String s="asd0";

String s="The quick brown fox jumps over the lazy dog";

if(isPangram(s,s.length()))

{

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

}

else

{

System.out.println("is not 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.

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.

**Program:**

import java.util.\*;

public class Main

{

static ArrayList<Integer> isLeaderEle(int[] a,int n)

{

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

int leader=a[n-1];

l.add(leader);

for(int i=n-2;i>=0;i--)

{

if(leader<a[i])

{

leader=a[i];

l.add(a[i]);

}

}

return l;

}

public static void main(String[] args) {

int[] a=new int[]{1, 4, 3, 2};

//int[] a=new int[]{1,2,3,4,5};

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

l=isLeaderEle(a,a.length);

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

{

System.out.print(l.get(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

**Program:**

import java.util.\*;

public class Main

{

static ArrayList<Character> missingChar(String s, int n)

{

boolean[] aplab=new boolean[26];

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

s=s.toLowerCase();

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

{

if(s.charAt(i)>='a' && s.charAt(i)<='z')

{

aplab[s.charAt(i)-'a']=true;

}

}

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

{

if(aplab[i]==false)

{

l.add((char) (i+'a'));

}

}

return l;

}

public static void main(String[] args) {

//String s="qwertyuiopasdfghjkl";

// String s="The quick brown fox jumps";

String s="welcome to geeksforgeeks";

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

l=missingChar(s,s.length());

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

{

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

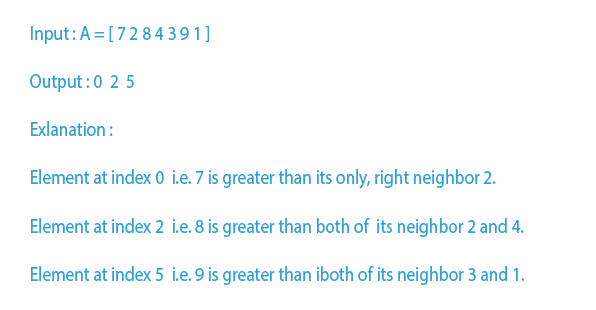
}

System.out.println();

}

}

B)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.



**Program:**

public class Main

{

static void niberBig(int[] a,int n)

{

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

{

System.out.print(0+" ");

}

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

{

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

{

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

}

}

}

public static void main(String[] args) {

int[] a=new int[]{7,2,8,4,3,9,1};

niberBig(a,a.length);

}

}

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

**Program:**

import java.util.\*;

public class Main

{

static boolean sameMaxNum(String s1,String s2)

{

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

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

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

{

h1.put(s1.charAt(i),h1.getOrDefault(s1.charAt(i),0)+1);

}

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

{

h2.put(s2.charAt(i),h2.getOrDefault(s2.charAt(i),0)+1);

}

int max=0;

Character maxC=null;

for(Character x:h1.keySet())

{

if(h1.get(x)>max)

{

max=h1.get(x);

maxC=x;

}

}

int max1=0;

Character maxC1=null;

for(Character x:h2.keySet())

{

if(h2.get(x)>max1)

{

max1=h2.get(x);

maxC1=x;

}

}

if(max==max1 && maxC==maxC1)

{

return true;

}

return false;

//return true;

}

public static void main(String[] args) {

String s="sssgeek", s2 = "geeksss";

// String s="geekyarticle",s2="gfggfggfg";

//String s="ewewwwy",s2="wwwwwqee";

if(sameMaxNum(s,s2))

{

System.out.println("Yes");

}

else{

System.out.println("No");

}

}

}

## B)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.

**Program:**

public class Main

{

static int[] cumSum(int[] a,int n)

{

int sum=0;

int[] a1=new int[n];

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

{

sum=sum+a[i];

a1[i]=sum;

}

return a1;

}

public static void main(String[] args) {

int[] a=new int []{1,2,3,4};

int[] a1=cumSum(a,a.length);

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

{

System.out.print(a1[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

**Program:**

import java.util.\*;

public class Main

{

static int noOfWords(String[] words,String str)

{

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

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

{

h.add(words[i]);

}

ArrayList<String> l=new ArrayList<String>(h);

int c=0;

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

{

if(str.contains(l.get(i)))

{

c++;

}

}

return c;

}

public static void main(String[] args) {

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

String[] words=new String[]{"welcome", "to", "geeks", "portal"};

/\*String str = "Save";

String[] words=new String[]{"Save", "Water", "Save", "Yourself"};\*/

System.out.println(noOfWords(words,str));

}

}

## 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.

**Program:**

import java.util.\*;

public class Main

{

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

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

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

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

}

int c = 0;

for (Integer x: hm.keySet()) {

c += (hm.get(x) \* (hm.get(x) - 1)) / 2;

}

return c;

}

public static void main(String[] args) {

//int[] a = {1, 2, 3, 2, 1};

int[] a = {1, 2, 2, 3, 2, 1};

System.out.println(noOfTwinsCount(a,a.length));

}

}

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.

**Program:**

import java.util.\*;

public class Main

{

static int twiceString(String[] words)

{

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

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

{

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

}

int c=0;

for(String x:m.keySet())

{

if(m.get(x)==2)

{

c++;

}

}

return c;

}

public static void main(String[] args) {

String[] s=new String[]{"Om", "Om", "Shankar", "Tripathi",

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

System.out.println(twiceString(s));

}

}

## 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.

**Program:**

public class Main

{

static int noOfDif(int n)

{

String s=Integer.toString(n);

return s.length();

}

static void evenDig(int[] a,int n)

{

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

{

if(a[i]<0)

{

a[i]=a[i]\*-1;

}

if(noOfDif(a[i])%2==0)

{

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

}

}

}

public static void main(String[] args) {

int[] a=new int[]{42, 564, 5775, 34, 123, 454, 1, 5, 45, -3556, 23442};

evenDig(a,a.length);

}

}

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*

**Program:**

public class Main

{

static boolean isUpperCh(char ch)

{

if(ch>='A'&&ch<='Z')

{

return true;

}

return false;

}

static int noOfcase(String s)

{

int c=0;

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

{

char ch=s.charAt(i);

if(isUpperCh(ch))

{

c++;

}

}

return c;

}

public static void main(String[] args) {

String s="ckjkUUYII";

//String s="asdafs";

System.out.println(noOfcase(s));

}

}

## 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

**Program:**

public class Main

{

static int maxNoOnes(int[] a,int n)

{

int max=0;

int c=0;

for(int i=0;i<n-1;i++)

{

if(a[i]==1&&a[i+1]==1)

{

c++;

max=Math.max(max,c);

}

else{

c=0;

}

}

return max+1;

}

public static void main(String[] args) {

int[] a=new int[]{1, 1, 3, 2, 3, 1, 1, 1,12,1,1,1,1,1,2,34,44};

System.out.println(maxNoOnes(a,a.length));

}

}

# 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”*

**Program:**

import java.util.\*;

public class Main

{

static String unCommonString(String s1,String s2)

{

StringBuffer s=new StringBuffer();

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

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

{

char ch=s1.charAt(i);

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

}

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

{

char ch=s2.charAt(i);

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

}

for(Character x:hm.keySet())

{

if(hm.get(x)==1)

{

s.append(x);

}

}

return s.toString();

}

public static void main(String[] args) {

// String S1 = "aacdb", S2 = "gafd";

String S1 = "abcs", S2 = "cxzca";

System.out.println(unCommonString(S1,S2));

}

}

# 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.

**Program:**

import java.util.\*;

public class Main

{

static int[] reArray (int[]a, int n)

{

Arrays.sort (a);

int ans[] = new int[n];

int p = 0, q = n - 1;

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

{

if ((i + 1) % 2 == 0)

{

ans[i] = a[q--];

}

else

{

ans[i]=a[p++];

}

}

return ans;

}

public static void main (String[]args)

{

//int[] a=new int[]{1,2,2,1};

int[] a=new int[]{1,3,2,4,8};

int[] ans=reArray(a,a.length);

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

{

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

}

}

}

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.

**Program:**

import java.util.\*;

public class Main

{

static int MaxConZero(String s,int k)

{

StringBuffer s1=new StringBuffer();

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

{

s1.append(s);

}

int max=0;

int cm=0;

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

{

if(s1.charAt(i)=='0')

{

cm++;

max=Math.max(cm,max);

}

else

{

cm=0;

}

}

return max;

}

public static void main(String[] args) {

// String s="110010";

// int k=2;

String s= "0010011000";

int k = 4 ;

System.out.println(MaxConZero(s,k));

}

}

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}*

**Program:**

public class Main

{

static int[] newArray(int[] a,int n)

{

int[] an=new int[n];

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

{

an[i]=0;

}

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

{

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

{

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

an[i]++;

}

}

return an;

}

public static void main(String[] args) {

int[] a=new int[]{12, 1, 2, 3, 0, 11, 4};

//int[] a=new int[]{5,4,3,2,1};

int[] an=newArray(a,a.length);

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

{

System.out.print(an[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]

]

**Program:**

import java.util.\*;

public class Main

{

static ArrayList<ArrayList<Integer>> antiDiag(int[][] mat)

{

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

int n=mat.length;

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

{

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

int c=i,r=0;

while(c>=0 && r<n)

{

tem.add(mat[r][c]);

c--;

r++;

}

l.add(tem);

}

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

{

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

int r=i,c=n-1;

while(r<n && c>=0)

{

tem.add(mat[r][c]);

c--;

r++;

}

l.add(tem);

}

return l;

}

public static void main(String[] args) {

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

int A[][]= { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

int a[][]={{1,2},{3,4}};

l=antiDiag(a);

for(ArrayList<Integer> k:l)

{

System.out.println(k);

}

System.out.println("Hello World");

}

}

# B) 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*

**Program:**

public class Main

{

static String freString(String s)

{

StringBuffer ans=new StringBuffer();

String temp="";

int j=0;

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

{

if(s.charAt(i)>=0)

{

int n= s.charAt(i)-'0';

if(s.charAt(i+1)=='(')

{

for(j=i+1;s.charAt(j)!=')';j++)

{

if(s.charAt(j)>='A'&& s.charAt(j)<='Z' || s.charAt(j)>='a'&& s.charAt(j)<='z')

{

temp=temp+s.charAt(j);

}

}

}

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

{

ans.append(temp);

}

temp="";

n=0;

if(j<s.length())

{

i=j;

}

}

}

return ans.toString();

}

public static void main(String[] args) {

String s="2(ab)4(cd)";

System.out.println(freString(s));

}

}

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.

**Program:**

public class MyClass {

static int eleFound(int[] a,int n)

{

int[] lmax=new int[n];

int[] rmin=new int[n];

lmax[0]=a[0];

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

{

lmax[i]=Math.max(lmax[i-1],a[i-1]);

}

rmin[n-1]=a[n-1];

for(int i=n-2;i>=0;i--)

{

rmin[i]=Math.min(rmin[i+1],a[i+1]);

}

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

if (a[i] > lmax[i] && a[i] < rmin[i]) {

return 1;

}

}

return 0;

}

public static void main(String args[]) {

int[] a={5, 1, 4, 3, 6, 8, 10, 7, 9};

// int[] a={5,1,4,4};

System.out.println(eleFound(a,a.length));

}

}

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

**Program:**

public class MyClass {

static int gcd(int a,int b)

{

if(a==0)

{

return b;

}

if(b==0)

{

return a;

}

return gcd(b,a%b);

}

static int gcdArray(int[] a,int n )

{

if(n==1)

{

return a[0];

}

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

{

a[i]=gcd(a[i-1],a[i]);

}

return a[n-1];

}

public static void main(String args[]) {

int[] a={2,4,6,8};

//int[] a={1,2,3};

System.out.println(gcdArray(a,a.length));

}

}

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]

**Program:**

public class MyClass {

static void spiralPrint( int a[][],int m, int n)

{

int i, r = 0, c = 0;

while (r < m && c < n) {

for (i = c; i < n; ++i) {

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

}

r++;

for (i = r; i < m; ++i) {

System.out.print(a[i][n - 1] + " ");

}

n--;

if (r < m) {

for (i = n - 1; i >= c; --i) {

System.out.print(a[m - 1][i] + " ");

}

m--;

}

if (c < n) {

for (i = m - 1; i >= r; --i) {

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

}

c++;

}

}

}

public static void main(String args[]) {

//int a[][] = { { 1, 2, 3,4 },{ 5, 6, 7, 8 },{ 9, 10, 11, 12 },{ 13, 14, 15, 16 } };

int a[][]={{1,2,3},{4,5,6},{7,8,9}};

spiralPrint(a, a.length, a[0].length);

}

}

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

**Program:**

public class MyClass {

static String newString(String s)

{

StringBuffer ans=new StringBuffer();

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

{

char ch=s.charAt(i);

for(int j=0;j<i+1;j++)

{

ans.append(ch);

}

}

return ans.toString();

}

public static void main(String args[]) {

String s="abcd";

String s1="geeks";

String s2="ABCD";

System.out.println(newString(s2));

}

}

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

# Program:

# import java.util.\*;

# public class MyClass {

# static void missingEle(int[] a,int[] b)

# {

# HashSet<Integer> s=new HashSet<Integer>();

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

# {

# s.add(b[i]);

# }

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

# {

# if(s.contains(a[i]))

# {

# continue;

# }

# else

# {

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

# }

# }

# }

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

# int[] a={1, 2, 3, 4, 5, 10};

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

# missingEle(a,b);

# 

# }

# }

# 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*

**Program:**

public class MyClass {

static boolean isTrue(String s,int k)

{

int n = s.length();

int i = 0;

while (i < n && i < k) {

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

return false;

}

i++;

}

return true;

}

public static void main(String args[]) {

String s="abcdefba";

int k=2;

// String s= "GeeksforGeeks";

// int k = 3;

if(isTrue(s,k))

{

System.out.println("Yes");

}

else

System.out.println("No");

}

}

# 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)

**Program:**

import java.util.\*;

public class Main

{

static boolean consArray(int[] a,int n)

{

Arrays.sort(a);

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

{

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

{

return false;

}

}

return true;

}

public static void main(String[] args) {

// int[] a={5,4,3,2,6};

int[] a={83, 78, 80, 81, 79, 82};

//int[] a={34, 23, 52, 12, 3};

//int[] a={7, 6, 5, 5, 3, 4};

System.out.println(consArray(a,a.length));

}

}

# 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.

**Program:**

public class Main

{

static boolean isRoOfPalindrome(String s) {

if (isPalindrome(s)) {

return true;

}

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

String rs = s.substring(i) + s.substring(0, i);

if (isPalindrome(rs)) {

return true;

}

}

return false;

}

static boolean isPalindrome(String s)

{

StringBuffer s1=new StringBuffer(s);

s1=s1.reverse();

if(s.equals(s1.toString()))

{

return true;

}

return false;

}

public static void main(String[] args) {

String s="aaaab";

String s1="abcd";

String s2="cbccc";

System.out.println(isRoOfPalindrome(s2));

}

}

# 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.*

**Program:**

import java.util.\*;

public class Main

{

static int disString(String[] str,int n)

{

int c=0;

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

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

{

char[] tem=str[i].toCharArray();

Arrays.sort(tem);

String tmp=new String(tem);

if(sS.contains(tmp))

{

continue;

}

else

{

sS.add(tmp);

c++;

}

}

return c;

}

public static void main(String[] args) {

//String[] arr = { "abcde", "abcce", "abcdf", "bcdea", "abcdf" };

String[] arr = { "ab", "abc", "abcd", "abcde", "a" };

System.out.println(disString(arr,arr.length));

}

}

# 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.*

**Program:**

import java.util.\*;

public class Main

{

static int eleCon(int[] a,int n,int k)

{

int c=0;

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

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

{

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

}

for(Integer x:hm.keySet())

{

if(hm.get(x)==k)

{

int num=x;

num=num\*2;

if(!hm.containsKey(num))

{

c++;

}

}

}

return c;

}

public static void main(String[] args) {

/\*int[] a = {10, 6, 12, 8, 10, 8};

int k = 2;

int[] a = {1, 3, 5, 3};

int k = 3;

int[] a={1,3,5,3,4,3};

int k =3;\*/

int[] a= {1, 2, 2, 3, 3, 4};

int k = 2;

System.out.println(eleCon(a,a.length,k));

}

}

# *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*

**Program:**

import java.util.\*;

public class Main

{

static boolean charFreq(String s,int k)

{

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

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

{

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

}

for(Character ch:hm.keySet())

{

if(hm.get(ch)!=k)

{

return false;

}

}

return true;

}

static int charArray(String[] s,int k)

{

int c=0;

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

{

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

{

c++;

}

}

return c;

}

public static void main(String[] args) {

// String[] arr = { "abab", "derdee", "erre" };

// int K = 2;

String[] arr = {"ag", "ka", "nanana"};

int K = 3;

System.out.println(charArray(arr,K));

}

}

# *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*

**Program:**

import java.util.\*;

public class Main

{

static int countOccurrence(int n, int[] arr, int k)

{

int c = 0;

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

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

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

{

if ( hm.getOrDefault(arr[i],false) == true)

continue;

if ( oc.getOrDefault(arr[i],0) >= k)

{

c++;

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

}

else

{

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

}

}

return c;

}

public static void main(String[] args) {

int[] arr = {1, 2, 1, 3};

//int[] arr={1, 2, 3, 2, 1, 3, 1, 2, 1};

int k = 1;

System.out.println(countOccurrence(arr.length,arr,k));

}

}

# *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*

**Program:**

import java.util.\*;

public class Main

{

static String freString(String s,int k)

{

StringBuffer ans=new StringBuffer();

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

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

{

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

}

for(Character ch:hm.keySet())

{

if(isPowerOfK(hm.get(ch),k))

{

for(int j=0;j<hm.get(ch);j++)

{

ans.append(ch);

}

}

}

return ans.toString();

}

static boolean isPowerOfK(int n, int k) {

while (n > 1) {

if (n % k != 0) {

return false;

}

n /= k;

}

return n == 1;

}

public static void main(String[] args) {

String str2 = "geeksgeekgeeks";

int k2 = 3;

System.out.println(freString(str2, k2));

String str1 = "aaacbb";

int k1 = 2;

System.out.println(freString(str1, k1));

}

}

# *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*

**Program:**

import java.util.\*;

public class Main

{

static ArrayList<Integer> minRmaxC(int[][] mat)

{

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

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

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

{

int minr=Integer.MAX\_VALUE;

for(int j=0;j<mat[i].length;j++)

{

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

}

s.add(minr);

}

for(int i=0;i<mat[0].length;i++)

{

int maxc=0;

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

{

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

}

if(s.contains(maxc))

{

l.add(maxc);

}

}

return l;

}

public static void main(String[] args) {

int[][] mat = { { 1, 10, 4 },{ 9, 3, 8 },{ 15, 16, 17 } };

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

l=minRmaxC(mat);

if(l.size()==0)

{

System.out.println("-1");

}

else

{

System.out.println(l);

}

}

}

# *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*

**Program:**

public class Main

{

static int[] newArray(int[] arr,int n)

{

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

{

if(i%2==0)

{

arr[i]=arr[i]\*arr[i]\*arr[i];

}

else

{

arr[i]=arr[i]\*arr[i];

}

}

return arr;

}

public static void main(String[] args) {

int[] arr= {2, 3, 4, 5};

//int[]arr = {3, 4, 5, 2}

int[] ans=newArray(arr,arr.length);

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

{

System.out.print(ans[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"

**Program:**

import java.util.\*;

public class Main

{

public static boolean isOvel(char ch){

if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U'){

return true;

}

return false;

}

public static String find(String s){

char s1[]=s.toCharArray();

int i=0,j=s.length()-1;

while(i<j){

if(isOvel(s1[i])&&isOvel(s1[j])){

char t=s1[i];

s1[i]=s1[j];

s1[j]=t;

i++;

j--;

}

else if(isOvel(s1[i])){

j--;

}

else{

i++;

}

}

String ans=new String(s1);

return ans;

}

public static void main(String[] args) {

String s="hello";

String s1="AEIOU";

String s2="DesignGUrus";

System.out.println(find(s));

System.out.println(find(s1));

System.out.println(find(s2));

}

}

*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

**Program:**

public class Main

{

static int shortDist(String[] words,String s1,String s2)

{

int dis=999;

int ind=-1;

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

{

if(words[i]==s1||words[i]==s2)

{

if(ind==-1)

{

ind=i;

}

else

{

if(words[ind]==words[i])

{

ind=i;

}

else

{

dis=Math.min(dis,((i-ind)));

ind=i;

}

}

}

}

return dis;

}

public static void main(String[] args) {

String[] words = {"the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"};

String word1 = "fox", word2 = "dog";

String[] words1 = {"a", "c", "d", "b", "a"};

String word11 = "a", word21 = "b";

System.out.println(shortDist(words1,word11,word21));

}

}

***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.

**Porgarm:**

public class Main

{

static int numIdenticalPairs(int[] nums) {

int c=0;

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

{

for(int j=i+1;j<nums.length;j++)

{

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

{

c++;

}

}

}

return c;

}

public static void main(String[] args) {

int[] nums = {1,2,3,1,1,3};

int[] a={1,1,1,1};

int[] a1={1,2,3};

System.out.println(numIdenticalPairs(a1));

}

}

# *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*

**Program:**

import java.util.\*;

public class Main

{

static boolean isIsom(String s)

{

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

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

{

se.add(s.charAt(i));

}

if(se.size()==s.length())

{

return true;

}

return false;

}

static boolean checkInArr(String[] words)

{

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

{

if(!isIsom(words[i]))

{

return false;

}

}

return true;

}

public static void main(String[] args) {

//String[] arr = {"abcd", "derg", "erty"};

String[] arr = {"agka", "lkmn"};

if(checkInArr(arr)){

System.out.println("yes");

}

else

System.out.println("No");

}

}

# *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

**Program:**

import java.util.\*;

public class Main

{

static int sumOfFre(int[][] mat)

{

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

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

{

for(int j=0;j<mat[i].length;j++)

{

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

}

}

int sum=0;

for(Integer x:hm.keySet())

{

if(hm.get(x)%2!=0)

{

sum=sum+(hm.get(x)\*x);

}

}

return sum;

}

public static void main(String[] args) {

int[][] mat = {{1, 1, 2},{2, 3, 3},{4, 5, 3}};

int[][] mat1= {{10, 20},{40, 40}};

System.out.println(sumOfFre(mat1));

}

}

# *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*

**Program:**

import java.util.\*;

public class Main

{

static String pattern(String s)

{

StringBuffer s1=new StringBuffer();

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

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

{

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

}

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

{

char ch=s.charAt(i);

int n=hm.get(ch);

char ch1=(char)(ch+n);

s1.append(ch1);

}

return s1.toString();

}

public static void main(String[] args) {

String S = "geeksforgeeks";

String S1 = "aabcadb";

System.out.println(pattern(S));

System.out.println(pattern(S1));

}

}

# *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.

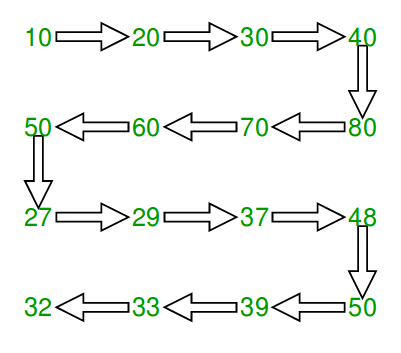
**Examples :**

***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*



**Program:**

public class Main

{

static void pattern(int[][] mat)

{

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

{

if(i%2==0)

{

for(int j=0;j<mat[i].length;j++)

{

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

}

}

else

{

for(int j=mat[i].length-1;j>=0;j--)

{

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

}

}

}

}

public static void main(String[] args) {

int[][] mat = { {10, 20, 30, 40},{15, 25, 35, 45},{27, 29, 37, 48},{32, 33, 39, 50}};

pattern(mat);

}

}

# *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*

**Program:**

import java.util.\*;

public class Main

{

static boolean evenFre(String s)

{

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

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

{

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

}

for(Character ch:hm.keySet())

{

if(hm.get(ch)%2!=0)

{

return false;

}

}

return true;

}

public static void main(String[] args) {

String s= "abaccaba";

String s1="hthth";

if(evenFre(s1))

System.out.println("Yes");

else

System.out.println("No");

}

}

# *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

**Program:**

import java.util.\*;

public class Main

{

static void diaEle(int[][]mat)

{

ArrayList<Integer> d1=new ArrayList<>();

ArrayList<Integer> d2=new ArrayList<>();

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

{

int n=mat[i][i]\*mat[i][i];

d1.add(n);

}

for(int i=0 ,j=mat[i].length-1;i<mat.length&&j>=0;i++,j--)

{

int n=mat[i][j]\*mat[i][j];

d2.add(n);

}

System.out.println("Diagonal one : "+d1 );

System.out.println("Diagonal two : "+d2 );

}

public static void main(String[] args) {

int[][] mat={{1,2,3},{4,5,6},{7,8,9}};

int[][] mat1={{2,5,7},{3,7,2},{5,6,9}};

diaEle(mat);

System.out.println();

diaEle(mat1);

}

}

# *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*

**Program:**

import java.util.\*;

public class Main

{

static boolean check(HashSet<Character> s,String str)

{

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

{

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

{

return false;

}

}

return true;

}

static int coStrcon(String[] words,String str)

{

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

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

{

s.add(str.charAt(i));

}

int c=0;

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

{

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

{

c++;

}

}

return c;

}

public static void main(String[] args) {

String[] wo={"abcd", "hijk", "xyz", "ayt"};

String str="apple";

String[] wo1={"apple", "banana", "pear"};

String str1="nil";

System.out.println(coStrcon(wo1,str1));

}

}

*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.

**Program:**

import java.util.\*;

public class MyClass {

static void printCommonElements(int mat[][])

{

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

for (int j = 0; j < mat[0].length; j++)

{

mp.put(mat[0][j],1);

}

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

{

for (int j = 0; j < mat[i].length; j++)

{

if (mp.get(mat[i][j]) != null && mp.get(mat[i][j]) == i)

{

mp.put(mat[i][j], i + 1);

if (i == mat.length - 1)

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

}

}

}

}

public static void main(String[] args) {

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

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

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

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

};

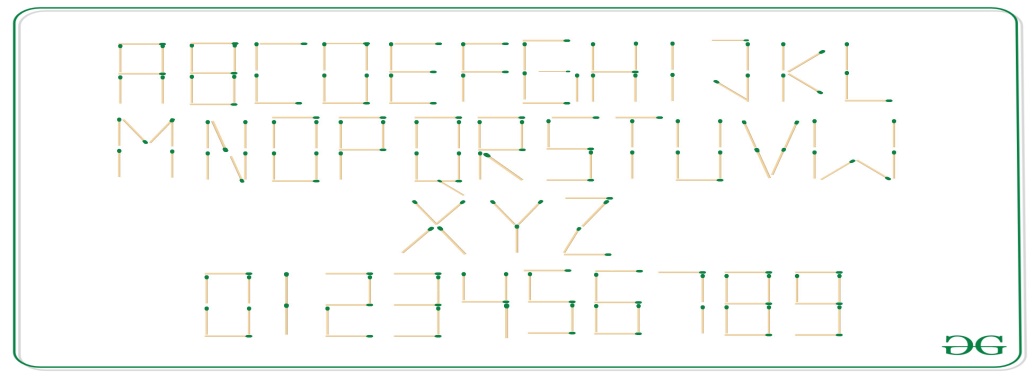
printCommonElements(mat);

}

}

# *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.*

**Program:**

public class Main

{

static int noOfStick(String s)

{

int aplha[] = { 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 num[] = { 6, 2, 5, 5, 4, 5, 6,3, 7, 6 };

int c=0;

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

{

char ch=s.charAt(i);

if(ch>='A'&&ch<='Z')

{

c=c+aplha[ch-'A'];

}

else

{

c=c+num[ch-'0'];

}

}

return c;

}

public static void main(String[] args) {

String s="ABC2";

String s1="GEEKSFORGEEKS";

System.out.println(noOfStick(s));

System.out.println(noOfStick(s1));

}

}

# *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

**Program:**

import java.util.\*;

public class Main

{

public static int maxDifference(int[] arr) {

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

int maxD = 0;

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

int num = a[i];

if (!hm.containsKey(num)) {

hm.put(num, i);

}

else {

int dis = i - hm.get(num);

maxD = Math.max(maxD, dis);

}

}

return maxD;

}

public static void main(String[] args) {

int[] arr1 = {2, 1, 3, 4, 2, 1, 5, 1, 7};

int[] arr2 = {2, 2, 1, 1, 8, 8, 3, 5, 3};

int[] arr3 = {1,4,2,6,78,1};

System.out.println("Output for arr1: " + maxDifference(arr1));

System.out.println("Output for arr2: " + maxDifference(arr2));

System.out.println("Output for arr2: " + maxDifference(arr3));

}

}

# *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.*

**Program:**

import java.util.\*;

public class Main

{

static int maxMatch(HashSet<Character> se,String s)

{

int c=0;

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

{

char ch=s.charAt(i);

if(se.contains(ch))

{

c++;

}

}

return c;

}

static String mostComm(String[] words,String s)

{

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

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

{

char ch=s.charAt(i);

se.add(ch);

}

int maxM=0;

int ind=-1;

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

{

int c=0;

c=maxMatch(se,words[i]);

if(c>maxM)

{

maxM=c;

ind=i;

}

}

return words[ind];

}

public static void main(String[] args) {

String[]words ={"preeti", "khusbu", "katherina"};

String s="vikas";

String[] words2={"goal","fog","abc"};

String s2="gfg";

System.out.println(mostComm(words,s));

System.out.println(mostComm(words2,s2));

}

}

# *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.*

**Program:**

public class Main

{

static void cNofFC(int a[][], int n, int k)

{

int[] ans = new int[k];

int[] rV = new int[n];

int[] cV = new int[n];

int rc = 0;

int colc = 0;

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

int r = a[i][0];

int c = a[i][1];

if (rV[r] == 0) {

rc++;

rV[r] = 1;

}

if (cV[c] == 0) {

colc++;

cV[c]= 1;

}

ans[i] = ((n \* n) - (rc \* n) - (colc \* n) + (rc \* colc));

}

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

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

}

}

public static void main (String[] args) {

int n=3, k=3;

int[][] arr = { { 1, 1 }, { 1, 2 }, { 2, 1 } };

cNofFC(arr, n, k);

System.out.println();

int n1=2,k1=2;

int[][] arr1={{0,1},{0,0}};

cNofFC(arr1, n1, k1);

}

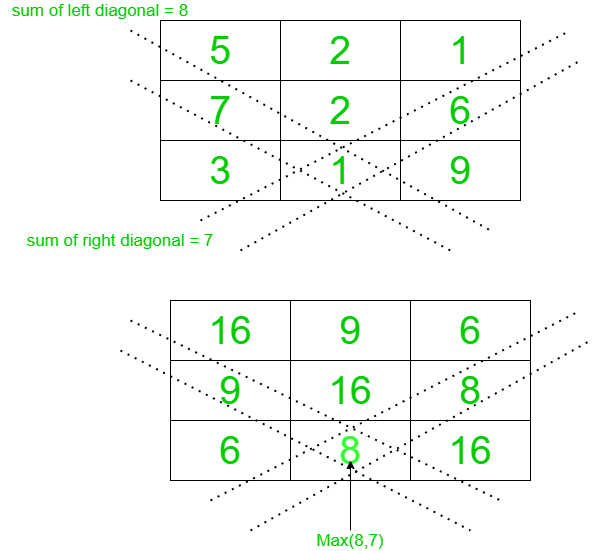
}

# *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*

**Program:**

import java.util.\*;

public class Main

{

static void updateMatrix(int mat[][])

{

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

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

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

{

for(int j = 0; j < mat[i].length; j++)

{

if (!r.containsKey(i + j))

r.put(i + j, mat[i][j]);

else

r.put(i + j,

r.get(i + j) + mat[i][j]);

if (!l.containsKey(i - j))

l.put(i - j, mat[i][j]);

else

l.put(i - j,

l.get(i - j) + mat[i][j]);

}

}

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

{

for(int j = 0; j < mat[i].length; j++)

{

mat[i][j] = Math.max(r.get(i + j),

l.get(i - j));

}

}

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

{

for(int j = 0; j < mat[i].length; j++)

{

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

}

System.out.println();

}

}

public static void main (String[] args) {

int[][] mat = {{ 5, 2, 1 },{ 7, 2, 6 },{ 3, 1, 9 }};

int[][] mat1={{1,2},{3,4}};

updateMatrix(mat);

updateMatrix(mat1);

}

}

# *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***Program:**

import java.util.\*;

public class Main

{

static void printChar(char []arr, int len)

{

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

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

{

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

}

int n = hm.size();

while (n > 0)

{

int cM = 0;

char am = 0;

for (Map.Entry<Character,Integer> it : hm.entrySet())

{

if (it.getValue() > cM ||(it.getValue() == cM && it.getKey() > am))

{

arg\_max = it.getKey();

currentMax = it.getValue();

}

}

System.out.print(am + " - " + cM + "\n");

hm.remove(am);

n--

}

}

public static void main(String[] args)

{

String str = "geeksforgeeks";

int len = str.length();

printChar(str.toCharArray(), len);

}

}

# *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*

*Program:*

*import java.util.\*;*

*public class Main*

*{*

*static int uniqueRow(char[][] c)*

*{*

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

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

*{*

*StringBuffer s=new StringBuffer();*

*for(int j=0;j<c[i].length;j++)*

*{*

*s.append(c[i][j]);*

*}*

*hm.put(s.toString(),hm.getOrDefault(s.toString(),0)+1);*

*}*

*int cr=0;*

*for(String s1:hm.keySet())*

*{*

*if(hm.get(s1)==1)*

*{*

*cr++;*

*}*

*}*

*return cr;*

*}*

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

*char[][] arr = {*

*{ 'a', 'b', 'c', 'd' },*

*{ 'a', 'e', 'f', 'r' },*

*{ 'a', 'b', 'c', 'd' },*

*{ 'z', 'c', 'e', 'f' },*

*};*

*System.out.println(uniqueRow(arr));*

*}*

*}*

# *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*

**Program:**

import java.util.\*;

public class Main

{

static boolean validFreq(String s,int x)

{

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

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

{

char ch=s.charAt(i);

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

}

for(int num:hm.values())

{

if(num>x)

{

return false;

}

}

return true;

}

public static int countStrings(String[] arr, int X, int Y) {

int co = 0;

for (String str : arr) {

if (validFreq(str,X) && str.length()>=Y) {

co++;

}

}

return co;

}

public static void main(String[] args) {

String[] arr1 = {"ab", "derdee", "erre"};

int X1 = 2, Y1 = 4;

System.out.println(countStrings(arr1, X1, Y1));

String[] arr2 = {"ag", "ka", "nanana"};

int X2 = 3, Y2 = 2;

System.out.println(countStrings(arr2, X2, Y2));

}

}

# *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

***Program:***

***import java.util.\*;***

***public class Main***

***{***

***static int sumOfFreK(int[] a,int n,int k)***

***{***

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

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

***{***

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

***}***

***int sum=0;***

***for(Integer x:hm.keySet())***

***{***

***if(hm.get(x)==k)***

***{***

***sum=sum+x;***

***}***

***}***

***return sum;***

***}***

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

***int[] a={2,3,9,9};***

***int[] arr = {9, 8, 8, 8, 10, 4};***

***System.out.println(sumOfFreK(a,a.length,1));***

***System.out.println(sumOfFreK(arr,arr.length,3));***

***}***

***}***

# *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”*

**Program:**

import java.util.\*;

public class Main

{

static String findString(String s1,String s2)

{

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

String[] ch2=s2.split(" ");

for(int i=0;i<ch2.length;i++)

{

hm1.put(ch2[i],hm1.getOrDefault(ch2[i],0)+1);

}

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

String[] ch1=s1.split(" ");

int fre=0;

String ans="";

for(int i=0;i<ch1.length;i++)

{

if(hm1.get(ch1[i])==null)

{

hm.put(ch1[i],hm.getOrDefault(ch1[i],0)+1);

if(hm.get(ch1[i])>fre||hm.get(ch1[i])==fre && ch1[i].compareTo(ans)<0)

{

ans=ch1[i];

fre=hm.get(ch1[i]);

}

}

}

return ans;

}

public static void main(String[] args) {

String S1 = "geeks for geeks is best place to learn";

String S2 = "bad place";

System.out.println(findString(S1, S2));

String s3="the quick brown fox jumps over the lazy dog";

String s4="the brown fox jumps";

System.out.println(findString(s3,s4));

}

}

# *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.*

***Program:***

import java.util.\*;

public class Main

{

static int maxRepFre(int[] a,int n)

{

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

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

{

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

}

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

for(Integer freq:hm.values())

{

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

}

int max=0;

int maxf=0;

for(Integer x:hm1.keySet())

{

if(hm1.get(x)>max)

{

max=hm1.get(x);

maxf=x;

}

}

return maxf;

}

public static void main(String[] args) {

int[] arr = {1, 1, 1, 2, 3, 2, 2, 3, 5, 5, 5, 5, 4, 4, 4, 4, 4};

int[] arr1 = { 3, 5, 15, 51, 15, 14, 14, 14, 14, 4};

System.out.println(maxRepFre(arr,arr.length));

System.out.println(maxRepFre(arr1,arr1.length));

}

}

# *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.*

**Program:**

import java.util.\*;

public class Main

{

static String ansString(String s)

{

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

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

StringBuilder s1=new StringBuilder(s);

String rev=s1.reverse().toString();

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

{

if(se.contains(rev.charAt(i)))

{

continue;

}

else

{

se.add(rev.charAt(i));

stack.push(rev.charAt(i));

}

}

StringBuilder ans=new StringBuilder();

while(stack.size()!=0)

{

ans.append(stack.pop());

}

return ans.toString();

}

public static void main(String[] args) {

System.out.println(ansString("geeksforgeeks"));

System.out.println(ansString("hi this is sample test"));

}

}

# *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

**Program:**

public class Main

{

static int find(int a[], int n)

{

return a[n / 2];

}

public static void main(String args[])

{

int[] a={2, 3 ,3, 4};

int[] b={3 ,4 ,5 ,5, 5 };

int[] c={1 ,1, 1 ,2, 3};

System.out.println(find(b, b.length));

System.out.println(find(a, a.length));

System.out.println(find(c, c.length));

}

}

# *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

**Program:**

import java.util.\*;

public class Main

{

static String oddString(String s)

{

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

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

{

char ch=s.charAt(i);

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

}

StringBuilder ans=new StringBuilder();

for(Character ch:hm.keySet())

{

if(hm.get(ch)%2!=0)

{

for(int i=0;i<hm.get(ch);i++)

{

ans.append(ch);

}

}

}

return ans.toString();

}

public static void main(String[] args) {

System.out.println(oddString("aabbbddeeecc"));

System.out.println(oddString("zzzxxweeerr"));

}

}