# Playing with Good Strings

A Good String is a string which contains only vowels (a,e,i,o,u) . Given a string S, print a single positive integer N where N is the length of the longest substring of S that is also a Good String.

**Note**: The time limit for this problem is 1 second, so you need to be clever in how you compute the substrings.

**Input Format**

A string 'S'

**Constraints**

Length of string < 10^5

**Output Format**

A single positive integer N, where N is the length of the longest sub-string of S that is also a Good String.

**Sample Input**

cbaeicde

**Sample Output**

3

**Explanation**

Longest good substring is "aei"

# Finding CB Numbers

Shubhanshu and Sneha are having a discussion on a new type of number that they call CB Number. They use following criteria to define a CB Number.

1. 0 and 1 are not a CB number.
2. 2,3,5,7,11,13,17,19,23,29 are CB numbers.
3. Any number not divisible by the numbers in point 2(Given above) are also CB numbers.

Shubhanshu said he loved CB numbers.Hearing it, Sneha throws a challenge to him.

Sneha will give Shubhanshu a string of digits. Shubhanshu's task is to find the number of CB numbers in the string.

* CB number once detected should not be sub-string or super-string of any other CB number.  
  Ex- In **4991**, both **499** and **991** are CB numbers but you can choose either **499** or **991**, not both.
* Further, the CB number formed can only be a sub-string of the string.  
  Ex - In **481**, you can not take **41** as CB number because 41 is not a sub-string of **481**.

As there can be multiple solutions, Sneha asks Shubhanshu to find the maximum number of CB numbers that can be formed from the given string.

Shubhanshu has to take class of Launchpad students. Help him by solving Sneha's challenge.

**Input Format**

First line contain size of the string.

Next line is A string of digits.

**Constraints**

1 <= Length of strings of digits <= 17

**Output Format**

Maximum number of CB numbers that can be formed.

**Sample Input**

5

81615

**Sample Output**

2

**Explanation**

61 and 5 are two CB numbers.

# String Compression

Take as input S, a string. Write a function that does basic string compression. Print the value returned. E.g. for input “aaabbccds” print out a3b2c2d1s1.

**Input Format**

A single String S

**Constraints**

1 < = length of String < = 1000

**Output Format**

The compressed String.

**Sample Input**

aaabbccds

**Sample Output**

a3b2c2d1s1

**Explanation**

In the given sample test case 'a' is repeated 3 times consecutively, 'b' is repeated twice, 'c' is repeated twice and 'd and 's' occurred only once.

# **Kartik Bhaiya And Strings**

Kartik Bhaiya has a string consisting of only 'a' and 'b' as the characters. Kartik Bhaiya describes perfectness of a string as the maximum length substring of equal characters. Kartik Bhaiya is given a number **k** which denotes the maximum number of characters he can change. Find the maximum perfectness he can generate by changing no more than **k** characters.

**Input Format**

The first line contains an integer denoting the value of K. The next line contains a string having only ‘a’ and ‘b’ as the characters.

**Constraints**

2 ≤ N ≤ 10^6

**Output Format**

A single integer denoting the maximum perfectness achievable.

**Sample Input**

2

abba

**Sample Output**

4

**Explanation**

We can swap the a's to b using the 2 swaps and obtain the string "bbbb". This would have all the b's and hence the answer 4.  
Alternatively, we can also swap the b's to make "aaaa". The final answer remains the same for both cases.

# **Non Repeating Character**

Given a string, find the first non-repeating character in it. For example, if the input string is “GeeksforGeeks”, then output should be ‘f’ and if input string is “GeeksQuiz”, then output should be ‘G’.

**Input Format**

The first line contains T denoting the number of testcases. Then follows description of testcases. Each case begins with a single integer N denoting the length of string. The next line contains the string S.

**Constraints**

**Output Format**

For each testcase, print the first non repeating character present in string. Print -1 if there is no non repeating character.

**Sample Input**

3

codingblocks

abbac

java

**Sample Output**

d

c

j

# **Form Biggest Number**

You are provided an array of numbers. You need to arrange them in a way that yields the largest value.

**Input Format**

First line contains integer t which is number of test case.  
For each test case, you are given a single integer n in the first line which is the size of array A[] and next line contains n space separated integers denoting the elements of the array A .

**Constraints**

1<=t<=100

1<=m<=100

1<=A[i]<=10^5

**Output Format**

Print the largest value.

**Sample Input**

1

4

54 546 548 60

**Sample Output**

6054854654

**Explanation**

Upon rearranging the elements of the array , 6054854654 is the largest possible number that can be generated.

# **Minimum Window Size Substring**

Given a string S and a string T, find the minimum window in S which will contain all the characters in T in complexity O(n).

**Input Format**

First Line Contains 2 strings of length not more than 10^5

**Constraints**

1<=|S|<=10^5

**Output Format**

A single Line a containing String

**Sample Input**

ADOBECODEBANC ABC

**Sample Output**

BANC

# **Strings-Remove Duplicates**

Take as input S, a string. Write a function that removes all consecutive duplicates. Print the value returned.

**Input Format**

String

**Constraints**

A string of length between 1 to 1000

**Output Format**

String

**Sample Input**

aabccba

**Sample Output**

abcba

**Explanation**

For the given example, "aabccba", Consecutive Occurrence of a is 2, b is 1, and c is 2.

After removing all of the consecutive occurences, the Final ans will be : - "abcba".

# **Subarrays Having Product less than k**

Given an array of positive numbers, the task is to find the number of possible contiguous subarrays having a product less than a given number k.

**Input Format**

First line contains Integer where N is the Size of Array  
Second line contains Integer k  
Next N Line Contains an Integer which denotes element of array

**Constraints**

1<=n<=10^5  
1<=k<=10^15  
1<=a[i]<=10^5

**Output Format**

Print number of possible contiguous subarrays having product less than a given number k.

**Sample Input**

4

10

1

2

3

4

**Sample Output**

7

**Explanation**

The contiguous subarrays are {1}, {2}, {3}, {4} {1, 2}, {1, 2, 3} and {2, 3} whose count is 7.

# Strings-Toggle Case

Take as input S, a string. Write a function that toggles the case of all characters in the string. Print the value returned.

**Input Format**

String

**Constraints**

Length of string should be between 1 to 1000.

**Output Format**

String

**Sample Input**

abC

**Sample Output**

ABc

**Explanation**

Toggle Case means to change UpperCase character to LowerCase character and vice-versa.

# **Strings-Odd Even Character**

Take as input S, a string. Write a function that replaces every even character with the character having just higher ASCII code and every odd character with the character having just lower ASCII code. Print the value returned.

**Input Format**

String

**Constraints**

Length of string should be between 1 to 1000.

**Output Format**

String

**Sample Input**

abcg

**Sample Output**

badf

# **Strings-String Compression**

Take as input S, a string. Write a function that does basic string compression. Print the value returned. E.g. for input “aaabbccds” print out a3b2c2ds.

**Input Format**

A single String S.

**Constraints**

A string of length between 1 to 1000

**Output Format**

The compressed String.

**Sample Input**

aaabbccds

**Sample Output**

a3b2c2ds

**Explanation**

In the given sample test case 'a' is repeated 3 times consecutively, 'b' is repeated twice, 'c' is repeated twice. But, 'd' and 's' occurred only once that's why we do not write their occurrence.

# **Strings-Difference in Ascii Codes**

Take as input S, a string. Write a program that inserts between each pair of characters the difference between their ascii codes and print the ans.

**Input Format**

String

**Constraints**

Length of String should be between 2 to 1000.

**Output Format**

String

**Sample Input**

acb

**Sample Output**

a2c-1b

**Explanation**

For the sample case, the Ascii code of a=97 and c=99 ,the difference between c and a is 2.Similarly ,the Ascii code of b=98 and c=99 and their difference is -1. So the ans is a2c-1b.

# **Strings-isPalindrome**

Take as input S, a string. Write a function that returns true if the string is a palindrome and false otherwise. Print the value returned.

**Input Format**

String

**Constraints**

String length between 1 to 1000 characters

**Output Format**

Boolean

**Sample Input**

abba

**Sample Output**

true

**Explanation**

A string is said to be palindrome if reverse of the string is same as string. For example, “abba” is palindrome as it's reverse is "abba", but “abbc” is not palindrome as it's reverse is "cbba".

# **Strings-Max Frequency Character**

Take as input S, a string. Write a function that returns the character with maximum frequency. Print the value returned.

**Input Format**

String

**Constraints**

A string of length between 1 to 1000.

**Output Format**

Character

**Sample Input**

aaabacb

**Sample Output**

a

**Explanation**

For the given input string, a appear 4 times. Hence, it is the most frequent character.

# Strings-Count Palindromic Substrings

Take as input S, a string. Write a program that gives the count of substrings of this string which are palindromes and Print the ans.

**Input Format**

Single line input containing a string

**Constraints**

Length of string is between 1 to 1000.

**Output Format**

Integer output showing the number of palindromic substrings.

**Sample Input**

abc

**Sample Output**

3

**Explanation**

For the given sample case , the palindromic substrings of the string abc are "a","b" and "c".So, the ans is 3.