Create A New String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given two strings S and T. Print **2** lines that contain the following in the same order:

1. Print the **length** of S and T separated by space.
2. Print a **new string** that contains S and T separated by a space.

For more clarification see the example below.

Input

The first line contains a string S ($1 \leq |S| \leq 10^3$) where $|S|$ is the length of S.
The second line contains a string T ($1 \leq |T| \leq 10^3$) where $|T|$ is the length of T.

Output

Print the answer required above.

Examples

standard input	standard output
LEVEL ONE	5 3 LEVEL ONE
Programming contest	11 7 Programming contest

Let's use Getline

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Given a string S. Print the string S from the beginning to the first '\' character without printing the '\'.

Hint: use function **getline(cin, s)**.

Input

Only one line contains a string S ($1 \leq |S| \leq 10^6$) where |S| is the length of the string.

It's guaranteed that S will contain '\' symbol.

Output

Print the answer required above.

Examples

standard input	standard output
Egyptian collegiate programming textbackslash contest	Egyptian collegiate programming
google textbackslashor facebook	google

Compare

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

Given two strings X and Y . Print the **smallest lexicographical** one.

Note: Lexicographical is the way of ordering the words based on the alphabetical order of their component letters.

Input

Only one line contains two strings X, Y ($1 \le |X|, |Y| \le 20$) consists of lowercase English letters.

Output

Print the **smallest lexicographical** string.

Note: If both of X and Y are equal, print any of them.

Example

standard input	standard output
acm acpc	acm

Note

For more information visit Lexicographical order: https://en.wikipedia.org/wiki/Lexicographical_order

Strings

Input file:	standard input
Output file:	standard output
Time limit:	2 second
Memory limit:	64 megabytes

Given two strings A and B. Print three lines contain the following:

- The size of the string A and size of the string B separated by a space
- The string produced by **concatenating** A and B ($A + B$).
- The two strings separated by a single space respectively, after **swapping** their first character.

For more clarification see the example below.

Input

The first line contains a string A ($1 \leq |A| \leq 10$) where $|A|$ is the length of A.
The second line contains a string B ($1 \leq |B| \leq 10$) where $|B|$ is the length of B.

Output

Print the answer required above.

Examples

standard input	standard output
abcd ef	4 2 abcdef ebcd af

Note

Declaration:

```
string a = "abc";
```

Size:

```
int len = a.size();
```

Concatenate two strings:

```
string a = "abc";
string b = "def";
string c = a + b; // c = "abcdef".
```

Accessing i *element*:

```
string s = "abc";
char   c0 = s[0];   // c0 = 'a'
char   c1 = s[1];   // c1 = 'b'
char   c2 = s[2];   // c2 = 'c'
s[0] = 'z';         // s = "zbc"
```

Count

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Given a string S. Print the **summation** of its digits.

Input

Only one line contains a string S ($1 \leq |S| \leq 10^6$) where |S| is the length of the string.

It's guaranteed that S contains only digits from 0 to 9.

Output

Print the answer required above.

Example

standard input	standard output
351	9

Note

First Test :

$3 + 5 + 1 = 9$.

Way Too Long Words

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a string S. Print the **origin string** if it's **not too long** otherwise, print the **special abbreviation**.

Note: The string is called **too long**, if its length is strictly more than **10** characters. If the string is **too long** then you have to print the string in the following manner:

- Print the **first** character in the string.
- Print number of characters between the first and the last characters.
- Print the **last** character in the string.

For example: "localization" will be "l10n", and "internationalization" will be "i18n".

Input

The first line contains a number T ($1 \leq T \leq 100$) number of test cases.

Each of the T following lines contains a string S ($1 \leq |S| \leq 100$) where |S| is the length of the string.

It's guaranteed that S contains only lowercase Latin letters.

Output

For each test case, print the result string.

Example

standard input	standard output
4 word localization internationalization pneumonoultramicroscopicsilicovolcanoconiosis	word l10n i18n p43s

Conversion

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Given a string S. Print the **origin** string after replacing the following:

- Replace every **comma** character ',' with a space character.
- Replace every **capital character** in S with its respective **small character** and **Vice Versa**.

Input

Only one line contains a string S ($1 \leq |S| \leq 10^5$) where |S| is the length of the string and it consists of **lower** and **upper** English letters and **comma** character ','.

Output

Print the string after the **conversion**.

Example

standard input	standard output
happy,NewYear,enjoy	HAPPY nEWyEAR ENJOY

Good or Bad

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Given a string S. Determine whether this string is **Good** or **Bad**.

Note: The string is **Good** if and only if it has "**010**" or "**101**" as one of its sub-strings and it's not necessary to have both of them.

A substring of a string is a contiguous subsequence of that string. So, string "forces"is substring of string "codeforces but string "coder"is not.

Input

The first line contains a number T ($1 \leq T \leq 100$) number of test cases.

Each of the T following lines contains a string S ($1 \leq |S| \leq 10^5$) where |S| is the length of the string..

It's guaranteed that S contains only '1s' and '0s'.

Output

For each test case, print "**Good**" if the string is **Good** otherwise, print "**Bad**".

Example

standard input	standard output
2 11111110 10101010101010	Bad Good

Note

Example case 1:

The string doesn't contain **010** or **101** as sub-strings.

Example case 2:

The string contains both **010** and **101** as sub-strings.

Palindrome

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a string S. Determine whether S is **Palindrome** or **not**

Note: A string is said to be a **palindrome** if **the reverse** of the string is **same** as the string. For example, "abba" is **palindrome**, but "abbc" is not **palindrome**.

Input

Only one line contains a string S ($1 \leq |S| \leq 1000$) where |S| is the length of the string and it consists of **lowercase** letters only.

Output

Print "**YES**" if the string is **palindrome**, otherwise print "**NO**".

Examples

standard input	standard output
abba	YES
icpcassiut	NO
mam	YES

Count Letters

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Given a string S. Determine how many times does each letter **occurred** in S.

Input

Only one line contains the string S ($1 \leq |S| \leq 10^7$) where |S| is the length of the string and it consists of only **lowercase** English letters.

Output

For each character that appears in S, print a single line that contains the following format: "$X : Y$"where X is the letter and Y is the number of times that letter X occurred in S.

Note: you must print letters in **ascending** order.

Examples

standard input	standard output
aaabbc	a : 3 b : 2 c : 1
regff	e : 1 f : 2 g : 1 r : 1

I Love strings

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Given two strings S and T. Print a new string that contains the following:

1. The first letter of the string S followed by the first letter of the string T.
2. the second letter of the string S followed by the second letter of the string T.
3. and so on...

In other words, the new string should be ($S_0 + T_0 + S_1 + T_1 + ...$).

Note: If the length of S is greater than the length of T then you have to add the rest of S letters at the end of the new string and **vice versa**.

Input

The first line contains a number N ($1 \leq N \leq 50$) the number of test cases.

Each of the N following lines contains two string S, T ($1 \leq |S|, |T| \leq 50$) consists of **lower** and **upper** English letters.

Output

For each test case, print the required string.

Example

standard input	standard output
2 ipAsu ccsit ey gpt	icpcAssiut egypt

String Functions

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given two numbers N, Q and a string S of size N. Followed by Q lines of the following queries:

- **pop_back** : remove the last character in the string.
- **front** : print the first character in the string.
- **back** : print the last character in the string.
- **sort l r** : where $(1 \leq l, r \leq |S|)$ sort all characters of S from l to r.
- **reverse l r** : where $(1 \leq l, r \leq |S|)$ reverse all characters of S from l to r.
- **print pos** : where $(1 \leq pos \leq |S|)$ print the character in the index pos.
- **substr l r** : where $(1 \leq l, r \leq |S|)$ print sub-string of s from l to r.
- **push_back x** : add character x in the end of the string.

For each query, print the answer associated with it in a single line.

It's guaranteed that in the first 7 types of the query, the string is not empty.

it's recommended to use built-in functions of String.

Input

The first line contains two integers N, Q $(1 \leq N, Q \leq 10^3)$ N denoting the size of the string and Q number of queries.

The second line contains the string S consists of only **lowercase** English letters.

Next Q lines contain the queries.

Output

For each query, print the answer associated with it in a single line.

Example

standard input	standard output
18 8 assiutinupperegypt substr 1 6 sort 5 8 pop_back back reverse 1 6 front push_back i print 4	assiut p n s

Subsequence String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given String S. Determine if there is a **Subsequence** in S that is **equal to** "hello" or **not**.

Note: A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

For example: The list of all **subsequence** for the word "apple"would be "a "ap "al "ae "app "apl "ape "ale "appl "appe "aple "apple "p "pp "pl "pe "ppl "ppe "ple "pple "l "le "e".

Input

Only one line contains a string S ($5 \leq |S| \leq 10^4$) where |S| is the length of the string and it consists of **lowercase** English letters.

Output

Print "**YES**" if there exists an **Subsequence** equal to "hello" otherwise, print "**NO**".

Examples

standard input	standard output
ahhellllloou	YES
hlelo	NO

Max Subsequence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a number N and a string S of size N. Determine the maximum possible size of the sub-sequence T derived from S such that no two adjacent characters in T are the same.

Note: A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

For example: The list of all **subsequence** for the word "apple"would be "a "ap "al "ae "app "apl "ape "ale "appl "appe "aple "apple "p "pp "pl "pe "ppl "ppe "ple "pple "l "le "e".

Input

The first line contains a number N ($1 \leq N \leq 10^5$) denoting the size of the string.

The second line contains a string S consists of lowercase English letters.

Output

Print a single line containing one number that represents the **maximum** size of the **subsequence** that satisfies the provided condition.

Examples

standard input	standard output
5 ababb	4
5 aaaac	2

Note

Test 1 : all **subsequence** strings such that no two adjacent characters in it is the same.

- a
- b
- ab
- aba
- abab
- bab
- ba

so the greatest one is **"abab"** and its size 4 so the answer is 4.

Sort String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	4 megabytes

Given a number N and a string S of size N. Print S after sorting it.

Note : don't use built-in function and it's recommended to not solve this problem with python language.

Input

The first line contains a number N ($1 \leq N \leq 10^7$) size of string S.

The second line contains a string S consists of lowercase English letters.

Output

Print S after sorting it.

Examples

standard input	standard output
4 deab	abde
5 egypt	egpty

Count Words

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a string S. Print number of words in it.

Word : consists of **lowercase** and **uppercase** English letters.

Input

Only one line contains S ($1 \leq |S| \leq 10^6$) where |S| is the length of the string and it consists of **lowercase** and **uppercase** English letters, spaces and ('!', '.', '?' and ',') symbols.

Output

Print the number of words in the given string.

Examples

standard input	standard output
`Meep Meep!`	`2`
`I tot I taw a putty tat.`	`7`
`I did! I did! I did taw a putty tat.`	`10`
`Shsssssssssh ... I am hunting wabbits. Heh Heh Heh Heh ...`	`8`

Reverse Words

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a string S. For each word in S **reverse** its letters then print it.

Note: words are separated by space.

Input

Only one line contains a strings S ($1 \leq |S| \leq 10^6$) where |S| is the length of the string and it consists of **lowercase** and **uppercase** English letters, spaces.

Output

Print the answer required above.

Examples

standard input	standard output
I love you	I evol uoy
You love me	uoY evol em
We are a happy family	eW era a yppah ylimaf

String Score

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a number N and a string S of size N consisting of **5** different uppercase characters only {**V**,**W**,**X**,**Y**,**Z**}.

- **V**: Adds to the score **5** points.
- **W**: Adds to the score **2** points.
- **X**: Removes the next consecutive character from the score.
- **Y**: Moves the next consecutive character to the end of the string.
- **Z**: If the next consecutive character is **V** it divides the total score by **5** but if it is **W** it divides the total score by **2**. Then it removes the next consecutive character from the string if and only if the next character is **V** or **W**.

Note: In case the string ends with **X** or **Y** or **Z** ignore their operations. The score is calculated from **left** to **right** and starts with **0**.

Input

The first line contains a number N ($1 \leq N \leq 10^6$) the length of the string.

The second line contains a string S.

It's guaranteed that S consists of only {V,W,X,Y,Z} letters.

Output

The total score of string.

Examples

standard input	standard output
5 VYWZW	4
6 WZYVXW	7

Note

Test 1:

String S = **"VYWZW"** and score initially = **0** .

First char is **'V'** so add 5 to score and become **5**.

Second char is **'Y'** then move 'W' to end of string and it become **"VYZWW"** and score = **5**.

Third char is **'Z'** then divide total score by 2 because next char is **'W'** and remove it so string become **"VYZW"** and score = **2**.

Fourth char is **'W'** so add 2 to score and become **4**.

So final answer is **4**.

Max Split

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a balanced string S consists of **['R', 'L']** characters. Split it into **maximum number of strings** such that the new strings are also balanced.

Note:

- Balanced strings are those who have **equal quantities of 'L' and 'R' characters**.
- You can not remove or reorder the characters while making the new strings.

Input

Only one line contains a string S ($2 \leq |S| \leq 1000$) where |S| is the length of the string.

It's guaranteed that S consists of only ['L', 'R'] letters, S is balanced and $|S|$ is even.

Output

Print maximum number of balanced strings then the balanced strings in any order.

Examples

standard input	standard output
LLRRLLLRRR	2 LLRR LLLRRR
LLLRRR	1 LLLRRR

URL

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a string S that represents a **URL** request. Print five lines contains the following format: "X: Y" where X is the parameter and Y is the parameter value.

Note: The parameters of the **URL**: **username**, **pwd**, **profile**, **role** and **key**.

Input

Only one line contains a string S ($100 \leq |S| \leq 10^4$) where |S| is the length of the string and it consists of **uppercase** and **lowercase** English letters, **digits**, and **special characters**.

The URL will be in the following format:

http://www.{word}.{word}/{word}/{word}?username={word}&pwd={word}&profile={word}&role={word}&key=

It's guaranteed that all parameters will have a value.

Output

Print **5** lines that contain the answer required above.

Example

standard input	standard output
http://www.cleartrip.com/signin/service?username=test&pwd=test&profile=developer&role=ELITE&key=m	username: test pwd: test profile: developer role: ELITE key: manager

New Words

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a string S. Print number of times that "**EGYPT**" word can be formed from S's characters.

Note: Case of the letters doesn't matter. For example: "**Egypt**", "**egypt**" and "**eGyPt**" are the same.

Input

Only one line contains a string $S(1 \leq |S| \leq 10^6)$ where |S| is the length of the string and it consists of **lowercase** and **uppercase** English letters.

Output

Print the answer required above.

Examples

standard input	standard output
EgYpTaz	1
pemigdbeigyypetet	2

Replace Word

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a string S. Print S after **replacing** every **sub-string** that is equal to "**EGYPT**" with space.

Input

Only one line contains a string S ($1 \leq |S| \leq 10^3$) where |S| is the length of the string and it consists of only **uppercase** English letters.

Output

Print the result as required above.

Examples

standard input	standard output
BRITISHEGYPTGHANA	BRITISH GHANA
ITALYKOREAEGYPTEGYPTALGERIAEGYPTZ	ITALYKOREA ALGERIA Z

Encrypt & Decrypt Message

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

Given a number Q and a string S. If Q is equal to **1** then print S after **encrypting** it otherwise, print S after **decrypting** it.

Key = "PgEfTYaWGHjDAmxQqFLRpCJBownyUKZXkbvzIdshurMilNSVOtec#@_!=.+-*/".

Original = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789".

Note:

- In Encryption: For each letter *x* in "**Original**" replace it by the equivalent letter *y* from "**Key**"
- In Decryption: For each letter *y* in "**Key**" replace it by the equivalent letter *x* from "**Original**".
- Key and Original have the same length.

Input

The first line contains a number Q ($1 \le Q \le 2$)

The second line contains a string S ($1 \le |S| \le 105$) where $|S|$ is the length of the string and it consists of **lowercase** and **uppercase** English letters.

Output

Print the answer required above.

Examples

standard input	standard output
1 Egypt	ZaoQR
2 #@_!=.+-*/	0123456789
2 ZaoQR	Egypt

Comparison

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

Given a string S. Print the **smallest** string that can be obtained by doing the following operations on the original string **only**:

- **Split** the string into two **non empty** consecutive strings (for example if you split the string into X and Y so $S = X + Y$).
- **Sort** every one of the separated strings.
- **Re-concatenate** the two stings into one string.

Note: If you couldn't split the string print the **original** string.

Input

Only one line contains a string S ($1 \leq |S| \leq 10^4$) where |S| is the length of the string and it consists of **lowercase** English letters.

Output

Print the smallest string that can be obtained.

Example

standard input	standard output
acmicpc	acccimp

Note

All possible strings that can be obtained :

- a + cmicpc : after sorting each part > a + cccimp = acccimp
- ac + micpc : after sorting each part > ac + ccimp = acccimp
- acm + icpc : after sorting each part > acm + ccip = acmccip
- acmi + cpc : after sorting each part > acim + ccp = acimccp
- acmic + pc : after sorting each part > accim + cp = accimcp
- acmicp + c : after sorting each part > accimp + c = accimpc

So the smallest one is "acccimp".

Min Cost String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a string S and **26** numbers that represents the cost of each letter. Print the minimum cost that can be achieved and the string S after replacing every '?' symbol in the string.

Note: The cost of the string will be the sum of the absolute difference of every two consecutive letters costs in the string. For example if cost of $'a' = 2$ and cost of $'x' = 5$ and cost of $'c' = 1$, if the string was equal "axc" then the cost of that string will = $|cost('a')\text{-}cost('x')| + |cost('x')\text{-}cost('c')| = |2-5| + |5-1| = 7$.

if there are more than one string with minimum cost print the smallest lexicographical one.

Input

The first line contains a string S ($2 \leq |S| \leq 10^6$) where |S| is the length of S.

The second line will contain 26 numbers ($0 \leq X_i \leq 10^5$) represents the cost of each letter from $'a'$ to $'z'$ respectively.

Output

Print the minimum cost that can be achieved by replacing every '?' in the first line followed by the string after the replacement.

Example

standard input	standard output
abc??def?gh 4 9 5 9 6 1 0 3 7 2 5 9 6 1 3 2 3 2 9 1 1 0 1 8 8 4	25 abcbbdeffgh

Note

Test 1 :

we can get string **"abcdddeffgh"** with cost 25 too but its **not smallest string**.

string **"abcbbdeffgh"** smallest one and smallest cost.

Clean Code

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

Given a source code in C++. Print the source code after cleaning it as follows :

1. If there are comments you have to remove them all whether it's a single comment (e.g. : //) or a block comment (e.g.: /* */).
2. If there are any blank lines or lines consists of spaces you must remove them all.

See the test cases carefully for more clarifications.

Input

Source code in C++ (maximum number of lines is 50).

Output

Print the clean code according to the problem statement.

Examples

standard input	standard output

// I am a single comment and you must remove me :((
/*
I am a block comment and you must remove me
*/
 	
#include<iostream>
using namespace std;
 	
int main() {
 	
int a, b;
cin » a » b; // Reading two variables from user (please remove me!! :( )
cout « a + b « endl;
 	
// End of the program (remove me please :))
return 0;
 	
}
	#include<iostream>
	using namespace std;
	int main() {
	int a, b;
	cin » a » b;
	cout « a + b « endl;
	return 0;
	}

Note

Please be careful in handling the priority of comments.

Comment symbols may be included in comments and it's not guaranteed that the resulting source code is a valid C++ syntax .