CODEFORCES β
Sponsored by Telegram

acmASCIS Session 5 Wednesday groups

A. Fair Compare

time limit per test: 0.5 seconds
memory limit per test: 32 megabytes
input: standard input
output: standard output

You are given N words, write a program to compare these words and find the one with the longest length, and the one with the largest value.

A word's value is computed by summing up the values of its characters. i.e A=1, B=2..Z=26.

If two words have the same length or two words have the same value, consider the word that comes later in input.

Input

Input consists of an integer N $(1 < = N < = 1000)$, followed by N words, each word consists of uppercase letters only, lengths don't exceed 100 characters.

Output

Output the word with the largest value and the word with the largest length, each on a separate line.

Sample test(s)

input

```
1
BB
```

output

```
BB
BB
```

input

```
2
AAA
ZZ
```

output

```
ZZ
AAA
```

B. Froid's String

time limit per test: 2 seconds

memory limit per test: 64 megabytes

input: standard input

output: standard output

Young Froid was playing with a string. After lots of operations he wants to know how many characters have been changed in this string.

Input

You are given two strings S and T, both of the same length. Their lengths don't exceed 1000.

Output

Print the number of different characters.

Sample test(s)

input

```
acm
ACM
```

output

```
3
```

input

```
abc
abd
```

output

```
1
```

C. Sam

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Sam likes good strings, a string is good if all its characters have been repeated equally.

Input

Given a string S consists only of lowercase letters. |S|<=1000

Output

Print "YES" if S is good, "NO" otherwise.

Sample test(s)

input

```
aabbcc
```

output

```
YES
```

D. Game of Strings

time limit per test: 1 second
memory limit per test: 64 megabytes
input: standard input
output: standard output

Jon Snow, The Bastard of Winterfell's Ruler, Is serving on the Night's watch, the Night's watch is a curious brotherhood, they protect Westros of Dangerous Creatures like The Others, the Direwolves and the giants. Tyrion Lannister – The Imp- has recently befriended Jon. Jon being kindhearted and Tyrion being very smart, Jon asked Tyrion to design a special saddle for his Crippled half brother who can't use his legs,Bran Stark. So that Bran could ride horses again. Tyrion decided to play a game with Jon to see if he's Worthy of his help. He asked him to give him a string that when cleared of everything except for alphabets, it'd make a palindrome string. If he could do that, Tyrion will make the saddle, else, he can't help him

A Palindrome string is a string that still reads the same when reversed. For example "Madam, I'm Adam" if we remove everything that isn't an alphabetic letter and turn all the letters to lower cases, we'd get "madamimadam" which is a palindrome string because it reads the same no matter from which way you read it

Input

The input consists of a string S that contains any kind of characters from the ASCII table. the string will have a the following size (2<=S<=10^5)

Output

If the String is a Palindrome, Print "I'll make the saddle"

Else, Print "Alas, Jon, You failed my test"

Sample test(s)

input

```
Madam, I'm Adam.
```

output

```
I'll make the saddle
```

input

```
I'm a brother of the Night's Watch.
```

output

```
Alas, Jon, You failed my test
```

E. Sum of Digits

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

It's very elementary to add numbers. Given a number, keep calculating the sum of its digits until it becomes one digit.

Input

You are given a number N. (1 <= N <= 10^{1000})

Output

Print the remaining digit.

Sample test(s)

input

```
12345
```

output

```
6
```

F. WUB WUB!

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Kamal works as a DJ at the best Berland nightclub, and he often uses dubstep music in his performances. Recently, he has decided to take a couple of old songs and make dubstep remixes from them.

Let's assume that a song consists of some number of words. To make the dubstep remix of this song, Kamal inserts zero or more words "WUB" before the first word of the song, zero or more words "WUB" between any pair of neighboring words, zero or more words "WUB" after the last word. then Kamal glues together all the words, including "WUB" -if existed- in one string and plays the song at the club.

For example, a song with words "I AM X" can transform into a dubstep remix as "WUBWUBIWUBAMWUBWUBX" and cannot transform into "WUBWUBIAMWUBX".

Recently, Ali has heard Kamal's new dubstep track, but since he isn't into modern music, he decided to find out what was the initial song that Kamal remixed. Help Ali restore the original song.

Input

The input consists of a single non-empty string, consisting only of uppercase English letters, the string's length doesn't exceed 200 characters. It is guaranteed that before Kamal remixed the song, no word contained substring "WUB" in it, Kamal didn't change the word order. It is also guaranteed that initially the song had at least one word.

Output

Print the words of the initial song that Kamal used to make a dubsteb remix. Separate the words with a space.

Sample test(s)

input

```
WUBWUBABCWUB
```

output

```
ABC
```

input

```
WUBWEWUBAREWUBWUBTHEWUBCHAMPIONSWUBMYWUBFRIENDWUB
```

output

```
WE ARE THE CHAMPIONS MY FRIEND
```

Note

In the first sample: "WUBWUBABCWUB" = "WUB" + "WUB" + "ABC" + "WUB". That means that the song originally consisted of a single word "ABC", and all words "WUB" were added by Kamal.

In the second sample Kamal added a single word "WUB" between all neighbouring words, in the beginning and in the end, except for words "ARE" and "THE" — between them Kamal added two "WUB".

G. Shorten!

time limit per test: 1 second

memory limit per test: 64 megabytes

input: standard input

output: standard output

Sometimes some words like "localization" or "internationalization" are so long that writing them many times in one text is quite tiresome.

Let's consider a word too long, if its length is strictly more than n characters. All too long words should be replaced with a special abbreviation.

This abbreviation is made like this: if the first letter and last letter are different, we write down the first and the last letter of a word and between them we write the number of letters (if existed) between the first and the last letters. Else, we write down the first letter followed by the number of letters after the first one. That number is in decimal system and doesn't contain any leading zeroes.

Thus, the abbreviation of "localization" will be $l10n$, and the abbreviation of "internationalization» will be as $i18n$.

Input

The first line contains an integer $n (1 \le n \le 100)$. The second line lines contains one word. All the words consist of lowercase Latin letters and possess the lengths of from 1 to 100 characters.

Output

Print one line containing the result of replacing the word from the input data.

Sample test(s)

input

```
10
localization
```

output

```
l10n
```

input

```
8
word
```

output

```
word
```

input

```
28
sneumonoultramicroscopicsilicovolcanoconiosis
```

output

```
s44
```

H. Zeko

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Our friend Zeko got so many zeros on his midterm exams, so he decided to eliminate all zeros from his life, but it wasn't easy.

In this problem he will have a simple example if he removed all zeros from his life, it's the addition operation. Let's assume you are given this equation a + b = c, where a and b are positive integers, and c is the sum of a and b. Now let's remove all zeros from this equation. Will the equation remain correct after removing all zeros?

For example if the equation is 102 + 103 = 205, if we removed all zeros it will be 12 + 13 = 25 which is still a correct equation.

But if the equation is 105 + 106 = 211, if we removed all zeros it will be 15 + 16 = 211 which is not a correct equation.

Input

The input will consist of two lines, the first line will contain the integer a, and the second line will contain the integer b which are in the equation as described above $(1 \leq a,\ b \leq 10^9)$. There won't be any leading zeros in both. The value of c should be calculated as c = a + b.

Output

The output will be just one line, you should print "YES" if the equation will remain correct after removing all zeros, and print "NO" otherwise.

Sample test(s)

input

```
102
103
```

output

```
YES
```

input

```
105
106
```

output

```
NO
```

I. Yoda's Task

time limit per test: 1 second
memory limit per test: 64 megabytes
input: standard input
output: standard output

"Young ACMer, solve this problem, you must. Give you a number, I will, if palindrome that number is not, reverse it, you must, and to itself add it, if palindrome it is still not, you should keep reversing and adding it to itself until, a palindrome, is it. With you may the force be. Hmmmmmm." said Yoda

Input

Each test case will consist of an integer P $(0 < P < 10^6 + 1)$ the number on which you'll do what Yoda asked you to do.

Output

For each of the N tests you will have to write a line with the following data : minimum number of iterations (additions) to get to the palindrome and the resulting palindrome itself separated by one space.

Sample test(s)

input

```
959139
```

output

```
27 178587688886785871
```

input

```
354
```

output

```
3 6666
```

J. Kuvo

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Kuvo has just had his session on strings and he was asked to do the following task. The task was to write a simple program. The program was supposed to do the following: in the given string, consisting of uppercase and lowercase Latin letters, it:

1) Deletes all the vowels.

2) Inserts a character "." before each consonant.

3) Replaces all uppercase consonants with corresponding lowercase ones.

Vowels are letters "A", "O", "Y", "E", "U", "I", and the rest are consonants. The program's input is exactly one string, it should return the output as a single string, resulting after the program processes the initial string.

Help Kuvo with this easy task.

Input

The first line represents input string of Kuvo's program. This string only consists of uppercase and lowercase Latin letters and its length is from 1 to 100, inclusive.

Output

Print the resulting string.

Sample test(s)

input

```
KuVo
```

output

```
.k.v
```

input

```
acmASCIS
```

output

```
.c.m.s.c.s
```

K. Maximum GCD

time limit per test: 1 second

memory limit per test: 64 megabytes

input: standard input

output: standard output

Given a definite number of integers, you have to find the maximum GCD(greatest common divisor) of every possible pair of these integers.

Input

Input consists of M (1<M<101) positive integers between 1 and 10^6, inclusive, that you have to find the maximum GCD among them.

Output

Print the maximum GCD of every possible pair.

If you have only 1 integer, print -1

Sample test(s)

input

```
7 5 12
```

output

```
1
```

input

```
1 2 3 4 5
```

output

```
2
```

L. Omar Loves Music

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

One day, I was having a chat with my friend Omar about the different genres of music, he stated that he likes listening to heavy metal.

I proposed to play a little game. We picked a book, and decided to search for a string that begins with the word "heavy" and ends with the word "metal". The game was funny at first but soon we got bored, so we decided to ask someone to write a program to find this string for us.

You're given the full text in the book (without spaces), Can you help us with this task?

Input

Input contains a single non-empty string consisting of the lowercase Latin alphabet letters. Length of this string will not be greater than $2 * 10^6$ characters.

Output

Print exactly one number — the number of strings in the book that starts with "heavy" and ends with "metal".

Sample test(s)

input

```
heavymetalisheavymetal
```

output

```
3
```

input

```
heavymetalismetal
```

output

```
2
```

input

```
trueheavymetalissotruewellitisalsosoheavythatyoucanalmostfeeltheweightofmetalonyou
```

output

```
3
```

Note

In the first sample the string "heavymetalisheavymetal" contains powerful substring "heavymetal" twice, also the whole string "heavymetalisheavymetal" is certainly powerful.

In the second sample the string "heavymetalismetal" contains two powerful substrings: "heavymetal" and "heavymetalismetal".

M. Tickets

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Each of you probably has your personal experience of riding public transportation and buying tickets. After a person buys a ticket (which traditionally has an even number of digits), he usually checks whether the ticket is lucky. Let us remind you that a ticket is lucky if the sum of digits in its first half matches the sum of digits in its second half.

But of course, not every ticket can be lucky. Far from it! Moreover, sometimes one look at a ticket can be enough to say right away that the ticket is not lucky. So, let's consider the following unluckiness criterion that can definitely determine an unlucky ticket. We'll say that a ticket is definitely unlucky if each digit from the first half corresponds to some digit from the second half so that each digit from the first half is strictly less than the corresponding digit from the second one or each digit from the first half is strictly more than the corresponding digit from the second one. Each digit should be used exactly once in the comparisons. In other words, there is such bijective correspondence between the digits of the first and the second half of the ticket, that either each digit of the first half turns out strictly less than the corresponding digit of the second half or each digit of the first half turns out strictly more than the corresponding digit from the second half.

For example, ticket 2421 meets the following unluckiness criterion and will not be considered lucky (the sought correspondence is $2 > 1$ and $4 > 2$), ticket 0135 also meets the criterion (the sought correspondence is $0 < 3$ and $1 < 5$), and ticket 3754 does not meet the criterion.

You have a ticket in your hands, it contains $2n$ digits. Your task is to check whether it meets the unluckiness criterion.

Input

The first line contains an integer n ($1 \le n \le 100$). The second line contains a string that consists of $2n$ digits and defines your ticket.

Output

In the first line print "YES" if the ticket meets the unluckiness criterion. Otherwise, print "NO" (without the quotes).

Sample test(s)

input

```
2
2421
```

output

```
YES
```

input

```
2
0135
```

output

```
YES
```

N. Layla and her interesting puzzle

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Layla is a girl,she is 8 years old,she loves all kinds of puzzles and spends hours trying to solve a puzzle but she has a specific most interesting puzzle. A Layla's puzzle that was popular 30 years ago consisted of a 5x5 frame which contained 24 small squares of equal size. A unique letter of the alphabet was printed on each small square. Since there were only 24 squares within the frame, the frame also contained an empty position which was the same size as a small square. A square could be moved into that empty position if it were immediately to the right,left, above, or below the empty position. The object of the puzzle was to slide squares into the empty position so that the frame displayed the letters in alphabetical order.

The illustration below represents a puzzle in its original configuration and in its configuration after the following sequence of 6 moves:

1. The square above the empty position moves.

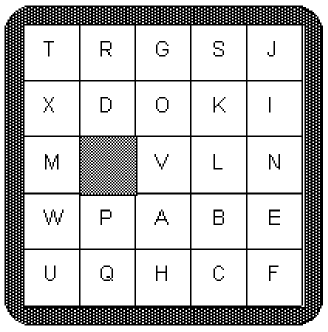
2. The square to the right of the empty position moves.

3. The square to the right of the empty position moves.

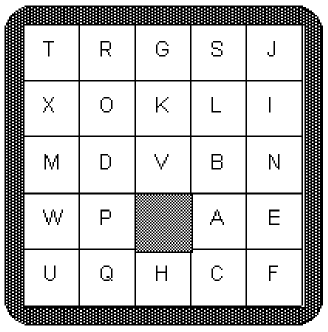
4. The square below the empty position moves.

5. The square below the empty position moves.

6. The square to the left of the empty position moves.



Original puzzle configuration.



Puzzle configuration after the sequence of described moves.

Now , Can you help Layla by Writing a program to display resulting frames given their initial configurations and sequences of moves.

Input

Input for your program consists of several puzzles. Each is described by its initial configuration and the sequence of moves on the puzzle. The first 5 lines of each puzzle description are the starting configuration. Subsequent lines give the sequence of moves.

The first line of the frame display corresponds to the top line of squares in the puzzle. The other lines follow in order. The empty position in a frame is indicated by a blank. Each display line contains exactly 5 characters, beginning with the character on the leftmost square (or a blank if the leftmost square is actually the empty frame position). The display lines will correspond to a legitimate puzzle.

The sequence of moves is represented by a sequence of As, Bs, Rs, and Ls to denote which square moves into the empty position. A denotes that the square above the empty position moves; B denotes that the square below the empty position moves; L denotes that the square to the left of the empty position moves; R denotes that the square to the right of the empty position moves. It is possible that there is an illegal move, even when it is represented by one of the 4 move characters. If an illegal move occurs, the puzzle is considered to have no final configuration. This sequence of moves may be spread over several lines, but it always ends in the digit 0. The end of data is denoted by the character Z.

Output

Output for each puzzle begins with an appropriately labeled number (Puzzle #1, Puzzle #2, etc.). If the puzzle has no final configuration, print "This puzzle has no final configuration." without the quotes, Otherwise that final configuration should be displayed.

Format each line for a final configuration so that there is a single blank character between two adjacent letters. Treat the empty square the same as a letter. For example, if the blank is an interior position, then it will appear as a sequence of 3 blanks - one to separate it from the square to the left, one for the empty position itself, and one to separate it from the square to the right.

Separate output from different puzzle records by one blank line.

Sample test(s)

input

```
TRGSJ
```

```
XDOKI
M VLN
WPABE
UQHCF
ARRBBL0
ABCDE
FGHIJ
KLMNO
PQRS
TUVWX
AAA
LLLL0
Z
```

output

```
Puzzle #1:
T R G S J
X O K L I
M D V B N
W P   A E
U Q H C F

Puzzle #2:
  A B C D
F G H I E
K L M N J
P Q R S O
T U V W X
```

Note

The first record of the sample input corresponds to the puzzle illustrated above.