TASK	VJEŠALA	KLJUČ	BIPALIN
input	standard input		
output	standard output		
time limit (per test case)	1 second		
memory limit (heap+stack)	64 MB		
points	**60**	**60**	**80**
	200		

Ivica is playing "Hangman" on his mobile phone. The goal of the game is to uncover letters in a secret phrase.

Initially, all letters in the secret phrase are obscured by dashes (one for each letter). Ivica chooses which letter to uncover next; if that letter appears in the secret phrase, then **all its occurrences** are revealed. The game ends after Ivica guesses all the letters.

Ivica plays this game all too often and has learned to recognize many of the phrases just by looking at the layout of the dashes. When he recognizes a phrase, Ivica wants to type it in as quickly as possible.

To select the next letter, Ivica uses three keys; LEFT, RIGHT and OK.

- One letter is always displayed on screen. Initially this is the letter A.
- When OK is pressed, the letter displayed is selected and all its occurrences revealed. The letter remains displayed.
- When LEFT is pressed, the previous letter is displayed (for example, C becomes B, Z becomes A).
- When RIGHT is pressed, the next letter is displayed (for example, B becomes C, A becomes Z).

Write a program that determines the **smallest number of key presses** Ivica needs to reveal the entire phrase, and **one possible ordering** of letters guessed that achieves the smallest number.
Note: the ordering of letters is not necessarily unique.

INPUT

The first and only line contains the secret phrase – at least 1 and at most 100 characters. All characters will be uppercase letters or spaces. The phrase starts and ends in a letter, and there is exactly one space between words.

OUTPUT

On the first line, output the smallest number of key presses needed to reveal the entire phrase.

On the second line, output an ordering which achieves that smallest number.

EXAMPLE TEST DATA

input	**input**	**input**
TURBO	PUTRA ZA SUTRA	IGRA BEZ GRANICA
output	**output**	**output**
19	18	28
BUTRO	AZUTSRP	AZBCEGINR

In the first example, Ivica can press keys in this order: RIGHT, OK, LEFT 7 times, OK, LEFT, OK, LEFT 2 times, OK, LEFT 3 times, OK.

Marica is sending Mirko a secret message. In order to stop the evil witch from listening, Marica **encrypts** her message using a simple algorithm which uses an additional **key**.

The message and the key are strings of letters in the English alphabet. The encryption method comprises several steps:

1. The key is cyclically repeated until it is of the same length as the message.
2. Each letter in the key and message is represented by a number between 0 and 25 (the letter 'a' is number 0; 'b' is 1 etc.).
3. Each letter in the message is **added** to the corresponding letter in the key (by adding their numeric values). If the result is 26 or more, we subtract 26 to make the result between 0 and 25.
4. The end result is represented as a string (0 for 'a', 1 for 'b' etc.).

For example, if we encrypt the message "sutraujutro" with the key "abz", we get the result "svsrbtjvsrp". Here is the example in both string and numeric form:

	s	u	t	r	a	u	j	u	t	r	o
+	a	b	z	a	b	z	a	b	z	a	b
	s	v	s	r	b	t	j	v	s	r	p

	18	20	19	17	0	20	9	20	19	17	14
+	0	1	25	0	1	25	0	1	25	0	1
	18	21	18	17	1	19	9	21	18	17	15

Unfortunately, the evil witch has learned the encryption method. Even though she doesn't know the key, she knows **one part of the original message** because she overheard it. That part of the message is always **at least twice as long as the key** and appears in the message **at least once**, but the witch doesn't know where.

Help the witch decrypt the message, catch Marica and Mirko and subsequently eat them. Write a program that, given the encrypted message and the known part of the original message, determines the original secret message.

Note: the input data will be such that a solution will exist and will be unique.

INPUT

The first line contains the encrypted message, a string of at most 1000 lowercase letters (no spaces).

The second line contains the known part of the message, a string of at most 100 lowercase letters.

OUTPUT

Output the secret message on a single line.

EXAMPLE TEST DATA

input
```
psinottfn
most
```

output
```
primosten
```

input
```
svsrbtjvsrp
ujutro
```

output
```
sutraujutro
```

input
```
yqyfjybckszapjezkqsqpok
vjesticu
```

output
```
bacitcemovjesticuuvatru
```

Pero and Slavko are two students who love math. Pero is very creative and often comes up with new ideas. Slavko later spends days thinking about Pero's ideas.

One day Pero and Slavko learned what a palindrome was: a string read the same in either direction (for example, "ANA", "1991" and "RADAR" are palindromes). Later that day Pero came up with a new concept – a bipalindrome (bipalin for short).

A bipalin is a **number**, composed of **two palindromes of the same length**. Both of these palindromes are strings of decimal digits, the first which **may not start with the digit 0** (therefore, the entire bipalin may not start with a 0). For example, 393020 is a bipalin (composed of the palindromes 393 and 020), while 222 and 010202 are not.

After hearing about bipalins, Slavko started thinking. After half an hour he found that there is only one bipalin of length 6 that is divisible by 12345. This is the bipalin 555525.

Shocked by this finding, he wants you to write a program that, given two integers N and M, calculates the number of **different bipalins of length N** that are **divisible by M**.

INPUT

Input consists of two integers N and M ($2 \leq N \leq 20$, $1 \leq M \leq 1000000$), where N is even.

OUTPUT

Output the number of different bipalins of length N that are divisible by M.

EXAMPLE TEST DATA

input 6 12345 **output** 1	**input** 2 10 **output** 9	**input** 6 123 **output** 71