problem	spam	novci	mars

source file	`spam.pas`
	`spam.c`
	`spam.cpp`
		`novci.pas`
		`novci.c`
		`novci.cpp`
			`mars.pas`
			`mars.c`
			`mars.cpp`

input data	stdin		
output data	stdout		
memory limit (heap)	16 MB	32 MB	16 MB
memory limit (stack)	2 MB		
time limit (pentium4 1.6 ghz)	1 second	0,5 seconds	7 seconds
points	**50**	**70**	**80**
	200		

spam

Unfortunately , junk messages are very widespread today.

One method of protecting ourselves from spam is to scramble our address when publishing it on the web (or anywhere else) so that programs that automatically search for e-mail addresses won't notice ours.

For this problem, the following holds for valid e-mail addresses:

1.) an e-mail adrress is a string of **lowercase letters of the English alphabet** ('a'...'z'), as well as a number of **full stop** characters ('.') and exactly one character '**@**'.

2.) there has to be a **letter** to the **immediate left** and the **immediate right** of the '@' character, and the **first** and **last** characters in the address **may not be a full stop**.

For example, the addresses 'mama@ta..ta', 'm.am.a@t..a.t..a' and 'm@t' are all valid, while 'ma@', '.@ma.ma', '.mama@tata' i 'ma.ma@tata.tata.' aren't.

We will scramble our address as follows:

1.) we will first replace the '@' character with the string '**at**'

2.) we will then add the string **'nospam' exactly once** or **zero times** anywhere in the address (the beginning and the end included)

Write a program that will, given a scrambled e-mail address, determine **all different valid addresses** it could have originated from.

input data

The only line of the input will contain a string of **at most 100 characters**, the scrambled address.

output data

You should write all different valid e-mail address (**in any order**) that, when scrambled, can give the given scrambled address. Write every address in a separate line.

examples

input
```
natva
```
output
```
n@va
```

input
```
a.batc.dnospam
```
output
```
a.b@c.dnospam
a.b@c.d
```

input
```
nospammamaattatahr
```
output
```
mamaatt@ahr
nospammamaatt@ahr
mama@tatahr
nospammama@tatahr
```

novci

John and Ted are playing a very interesting game.

John throws a bunch of coins on the table, takes them back one by one, puts them in his pocket and writes **P** if the coin's upper side was **head** or **G** if it was **tail**.

This way, John will get a **random sequence of characters P and G**, and then make a number of statements of the form:

"the *i*-th letter is X or the *j*-th letter is Y"

where *i* and *j* are **different** numbers, X and Y are either P or G (independent of each other), and **at least one** of the claims in the statement is **true**.

Ted is sitting on his sofa on the other side of the room, listening to John and trying to determine his sequence based on the statements he makes.

Write a program that will help Ted determine **any sequence** (**if one exists**) of characters so that at least one claim is true in all of John's statements.

input data

The first line of input contains an integer L, 2 ≤ L ≤ 1000, the length of the sequence.

The second line of input contains an integer N, 1 ≤ N ≤ 100,000, the number of statements.

Each of the following lines contains a statement of the form:

```
i X j Y
```

1 ≤ *i*,*j* ≤ L, *i* and *j* are different, and X i Y are both either 'P' or 'G'.

output data

The first and only line of output should contain the discovered sequence.

If there is no such sequence, output **-1**.

Note: the solution need not be unique.

examples

input

```
2
3
1 P 2 G
1 G 2 P
1 P 2 P
```

output

```
PP
```

input

```
3
3
1 P 2 G
2 G 3 P
1 P 3 P
```

output

```
PGP
```

input

```
3
6
1 G 2 G
2 G 3 G
1 G 3 G
2 P 3 P
1 G 2 P
1 P 3 G
```

output

```
GPG
```

Person A discovered the DNA formula of Martians.

He wants to give the information to person B in exchange for a large amount of peanuts. They suspect that the Bureau of Prevention of the Spreading of the DNA Formula of Martians (BPSDFM) is easedropping on them, so they will meet in a deserted bakery to exchange goods.

Since the formula is very long, person A wants to **shorten** it some way so that he can give it to the other person as easily as possible. He does it the following way:

The formula is a **sequence of lowercase letters of the English alphabet**. We shorten the formula so that some **repeating strings** of characters are written as follows: `'abcabcabc'` as `'(abc)3'`, `'axyxyxyxyb'` as `'a(xy)4b'`, and we can also have nested shortening eg. `'mnmndefmnmndef'` as `'((mn)2def)2'`.

The length of a formula is defined as the **total number of characters**, **including** the **parentheses** and **digits**.

Write a program that will, given a formula, **shorten it so that the number of characters in the new formula is minimal**.

Note: the solution need not be unique.

input data

The first and only line of input contains the formula. The maximum length **is 1000 characters**.

output data

The first and only line of output should contain the shortest way of writing the formula. If there is **more than one way** of shortening the formula **with minimal length**, output **any of them**.

examples

input
```
ababab
```
output
```
(ab)3
```

input
```
adddddaaddddda
```
output
```
(a(d)5a)2
```

input
```
mnmnmnmnabnabnab
```
output
```
(mn)3m(nab)3
```