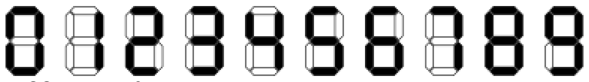
A. Digital Counter

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Malek lives in an apartment block with 100 floors numbered from 0 to 99. The apartment has an elevator with a digital counter showing the floor that the elevator is currently on. The elevator shows each digit of a number with 7 light sticks by turning them on or off. The picture below shows how the elevator shows each digit.



One day when Malek wanted to go from floor 88 to floor 0 using the elevator he noticed that the counter shows number 89 instead of 88. Then when the elevator started moving the number on the counter changed to 87. After a little thinking Malek came to the conclusion that there is only one explanation for this: One of the sticks of the counter was broken. Later that day Malek was thinking about the broken stick and suddenly he came up with the following problem.

Suppose the digital counter is showing number n. Malek calls an integer x ($0 \le x \le 99$) *good* if it's possible that the digital counter was supposed to show x but because of some(possibly none) broken sticks it's showing n instead. Malek wants to know number of good integers for a specific n. So you must write a program that calculates this number. Please note that the counter **always** shows two digits.

Input

The only line of input contains exactly two digits representing number n ($0 \le n \le 99$). Note that n may have a leading zero.

Output

In the only line of the output print the number of good integers.

Sample test(s)

input
```
89
```
output
```
2
```

input
```
00
```
output
```
4
```

input
```
73
```
output
```
15
```

Note

In the first sample the counter may be supposed to show 88 or 89.

In the second sample the good integers are 00, 08, 80 and 88.

In the third sample the good integers are $03, 08, 09, 33, 38, 39, 73, 78, 79, 83, 88, 89, 93, 98, 99$.

B. Modular Equations

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Last week, Hamed learned about a new type of equations in his math class called Modular Equations. Lets define i modulo j as the remainder of division of i by j and denote it by $i \bmod j$. A Modular Equation, as Hamed's teacher described, is an equation of the form $a \bmod x = b$ in which a and b are two non-negative integers and x is a variable. We call a positive integer x for which $a \bmod x = b$ a *solution* of our equation.

Hamed didn't pay much attention to the class since he was watching a movie. He only managed to understand the definitions of these equations.

Now he wants to write his math exercises but since he has no idea how to do that, he asked you for help. He has told you all he knows about Modular Equations and asked you to write a program which given two numbers a and b determines how many answers the Modular Equation $a \bmod x = b$ has.

Input

In the only line of the input two space-separated integers a and b ($0 \le a, b \le 10^9$) are given.

Output

If there is an infinite number of answers to our equation, print "infinity" (without the quotes). Otherwise print the number of solutions of the Modular Equation $a \bmod x = b$.

Sample test(s)

input
```
21 5
```
output
```
2
```

input
```
9435152 272
```
output
```
282
```

input
```
10 10
```
output
```
infinity
```

Note

In the first sample the answers of the Modular Equation are 8 and 16 since $21 \bmod 8 = 21 \bmod 16 = 5$

C. Treasure

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Malek has recently found a treasure map. While he was looking for a treasure he found a locked door. There was a string s written on the door consisting of characters '`(`', '`)`' and '`#`'. Below there was a manual on how to open the door. After spending a long time Malek managed to decode the manual and found out that the goal is to replace each '`#`' with one or more '`)`' characters so that the final string becomes *beautiful*.

Below there was also written that a string is called *beautiful* if for each i $(1 \le i \le |s|)$ there are no more '`)`' characters than '`(`' characters among the first i characters of s and also the total number of '`(`' characters is equal to the total number of '`)`' characters.

Help Malek open the door by telling him for each '`#`' character how many '`)`' characters he must replace it with.

Input

The first line of the input contains a string s $(1 \le |s| \le 10^5)$. Each character of this string is one of the characters '`(`', '`)`' or '`#`'. It is guaranteed that s contains at least one '`#`' character.

Output

If there is no way of replacing '`#`' characters which leads to a beautiful string print -1. Otherwise for each character '`#`' print a separate line containing a positive integer, the number of '`)`' characters this character must be replaced with.

If there are several possible answers, you may output any of them.

Sample test(s)

input
```
(((#)((#)
```
output
```
1
2
```

input
```
()((#((#(#()
```
output
```
2
2
1
```

input
```
#
```
output
```
-1
```

input
```
(#)
```
output
```
-1
```

Note

$|s|$ denotes the length of the string s.

D. Obsessive String

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Hamed has recently found a string t and suddenly became quite fond of it. He spent several days trying to find all occurrences of t in other strings he had. Finally he became tired and started thinking about the following problem. Given a string s how many ways are there to extract $k \geq 1$ non-overlapping substrings from it such that each of them contains string t as a substring? More formally, you need to calculate the number of ways to choose two sequences $a_1, a_2, ..., a_k$ and $b_1, b_2, ..., b_k$ satisfying the following requirements:

- $k \geq 1$
- $\forall i \ (1 \leq i \leq k) \ 1 \leq a_i, b_i \leq |s|$
- $\forall i \ (1 \leq i \leq k) \ b_i \geq a_i$
- $\forall i \ (2 \leq i \leq k) \ a_i > b_{i-1}$
- $\forall i \ (1 \leq i \leq k)$ t is a substring of string $s_{a_i}s_{a_i+1} ... s_{b_i}$ (string s is considered as 1-indexed).

As the number of ways can be rather large print it modulo $10^9 + 7$.

Input

Input consists of two lines containing strings s and t ($1 \leq |s|, |t| \leq 10^5$). Each string consists of lowercase Latin letters.

Output

Print the answer in a single line.

Sample test(s)

input

```
ababa
aba
```

output

```
5
```

input

```
welcometoroundtwohundredandeightytwo
d
```

output

```
274201
```

input

```
ddd
d
```

output

```
12
```

E. Helping People

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Malek is a rich man. He also is very generous. That's why he decided to split his money between poor people. A charity institute knows n poor people numbered from 1 to n. The institute gave Malek q recommendations. A recommendation is a segment of people like $[l, r]$ which means the institute recommended that Malek gives one dollar to every person whose number is in this segment.

However this charity has very odd rules about the recommendations. Because of those rules the recommendations are given in such a way that for every two recommendation $[a, b]$ and $[c, d]$ one of the following conditions holds:

- The two segments are completely disjoint. More formally either $a \le b < c \le d$ or $c \le d < a \le b$
- One of the two segments are inside another. More formally either $a \le c \le d \le b$ or $c \le a \le b \le d$.

The *goodness* of a charity is the value of maximum money a person has after Malek finishes giving his money. The institute knows for each recommendation what is the probability that Malek will accept it. They want to know the expected value of *goodness* of this charity. So they asked you for help.

You have been given the list of recommendations and for each recommendation the probability of it being accepted by Malek. You have also been given how much money each person initially has. You must find the expected value of *goodness*.

Input

In the first line two space-separated integers n, q $(1 \le n \le 10^5, 1 \le q \le 5000)$ are given.

In the second line n space-separated integers $a_1, a_2, ..., a_n$ $(0 \le a_i \le 10^9)$ are given meaning that person number i initially has a_i dollars.

Each of the next q lines contains three space-separated numbers l_i, r_i, p_i $(1 \le l_i \le r_i \le n, 0 \le p \le 1)$ where l_i and r_i are two integers describing the segment of recommendation and p_i is a real number given with exactly three digits after decimal point which is equal to probability of Malek accepting this recommendation.

Note that a segment may appear several times in recommendations.

Output

Output the sought value. Your answer will be considered correct if its absolute or relative error is less than 10^{-6}.

Sample test(s)

input

```
5 2
1 7 2 4 3
1 3 0.500
2 2 0.500
```

output

```
8.000000000
```

input

```
5 2
281 280 279 278 282
1 4 1.000
1 4 0.000
```

output

```
282.000000000
```

input

```
3 5
1 2 3
1 3 0.500
2 2 0.250
1 2 0.800
1 1 0.120
2 2 0.900
```

output

```
4.465000000
```