# Codeforces Round #409 (rated, Div. 2, based on VK Cup 2017 Round 2)

## A. Vicious Keyboard

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

Tonio has a keyboard with only two letters, "V" and "K".

One day, he has typed out a string $s$ with only these two letters. He really likes it when the string "VK" appears, so he wishes to change at most one letter in the string (or do no changes) to maximize the number of occurrences of that string. Compute the maximum number of times "VK" can appear as a substring (i. e. a letter "K" right after a letter "V") in the resulting string.

### Input

The first line will contain a string $s$ consisting only of uppercase English letters "V" and "K" with length not less than $1$ and not greater than $100$.

### Output

Output a single integer, the maximum number of times "VK" can appear as a substring of the given string after changing at most one character.

### Examples

input
```
VK
```
output
```
1
```

input
```
VV
```
output
```
1
```

input
```
V
```
output
```
0
```

input
```
VKKKKKKKKKVVVVVVVVVK
```
output
```
3
```

input
```
KVKV
```
output
```
1
```

### Note

For the first case, we do not change any letters. "VK" appears once, which is the maximum number of times it could appear.

For the second case, we can change the second character from a "V" to a "K". This will give us the string "VK". This has one occurrence of the string "VK" as a substring.

For the fourth case, we can change the fourth character from a "K" to a "V". This will give us the string "VKKVKKKKKKVVVVVVVVVK". This has three occurrences of the string "VK" as a substring. We can check no other moves can give us strictly more occurrences.

# B. Valued Keys

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

You found a mysterious function $f$. The function takes two strings $s_1$ and $s_2$. These strings must consist only of lowercase English letters, and must be the same length.

The output of the function $f$ is another string of the same length. The $i$-th character of the output is equal to the minimum of the $i$-th character of $s_1$ and the $i$-th character of $s_2$.

For example, $f($"`ab`", "`ba`"$)$ = "`aa`", and $f($"`nzwzl`", "`zizez`"$)$ = "`niwel`".

You found two strings $x$ and $y$ of the same length and consisting of only lowercase English letters. Find any string $z$ such that $f(x, z) = y$, or print `-1` if no such string $z$ exists.

## Input

The first line of input contains the string $x$.

The second line of input contains the string $y$.

Both $x$ and $y$ consist only of lowercase English letters, $x$ and $y$ have same length and this length is between $1$ and $100$.

## Output

If there is no string $z$ such that $f(x, z) = y$, print `-1`.

Otherwise, print a string $z$ such that $f(x, z) = y$. If there are multiple possible answers, print any of them. The string $z$ should be the same length as $x$ and $y$ and consist only of lowercase English letters.

### Examples

input
```
ab
aa
```
output
```
ba
```

input
```
nzwzl
niwel
```
output
```
xiyez
```

input
```
ab
ba
```
output
```
-1
```

## Note

The first case is from the statement.

Another solution for the second case is "`zizez`"

There is no solution for the third case. That is, there is no $z$ such that $f($"`ab`"$, z) =$ "`ba`".

# C. Voltage Keepsake

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

You have $n$ devices that you want to use simultaneously.

The $i$-th device uses $a_i$ units of power per second. This usage is continuous. That is, in $\lambda$ seconds, the device will use $\lambda \cdot a_i$ units of power. The $i$-th device currently has $b_i$ units of power stored. All devices can store an arbitrary amount of power.

You have a single charger that can plug to any single device. The charger will add $p$ units of power per second to a device. This charging is continuous. That is, if you plug in a device for $\lambda$ seconds, it will gain $\lambda \cdot p$ units of power. You can switch which device is charging at any arbitrary unit of time (including real numbers), and the time it takes to switch is negligible.

You are wondering, what is the maximum amount of time you can use the devices until one of them hits $0$ units of power.

If you can use the devices indefinitely, print `-1`. Otherwise, print the maximum amount of time before any one device hits $0$ power.

### Input

The first line contains two integers, $n$ and $p$ $(1 \le n \le 100\,000,\ 1 \le p \le 10^9)$ — the number of devices and the power of the charger.

This is followed by $n$ lines which contain two integers each. Line $i$ contains the integers $a_i$ and $b_i$ $(1 \le a_i, b_i \le 100\,000)$ — the power of the device and the amount of power stored in the device in the beginning.

### Output

If you can use the devices indefinitely, print `-1`. Otherwise, print the maximum amount of time before any one device hits $0$ power.

Your answer will be considered correct if its absolute or relative error does not exceed $10^{-4}$.

Namely, let's assume that your answer is $a$ and the answer of the jury is $b$. The checker program will consider your answer correct if .

**Examples**

input
```
2 1
2 2
2 1000
```
output
```
2.0000000000
```

input
```
1 100
1 1
```
output
```
-1
```

input
```
3 5
4 3
5 2
6 1
```
output
```
0.5000000000
```

### Note

In sample test 1, you can charge the first device for the entire time until it hits zero power. The second device has enough power to last this time without being charged.

In sample test 2, you can use the device indefinitely.

In sample test 3, we can charge the third device for $2 / 5$ of a second, then switch to charge the second device for a $1 / 10$ of a second.

# D. Volatile Kite

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

You are given a convex polygon $P$ with $n$ distinct vertices $p_1, p_2, ..., p_n$. Vertex $p_i$ has coordinates $(x_i, y_i)$ in the 2D plane. These vertices are listed in clockwise order.

You can choose a real number $D$ and move each vertex of the polygon a distance of at most $D$ from their original positions.

Find the maximum value of $D$ such that no matter how you move the vertices, the polygon does not intersect itself and stays convex.

## Input

The first line has one integer $n$ $(4 \leq n \leq 1\ 000)$ — the number of vertices.

The next $n$ lines contain the coordinates of the vertices. Line $i$ contains two integers $x_i$ and $y_i$ ( $-10^9 \leq x_i, y_i \leq 10^9$) — the coordinates of the $i$-th vertex. These points are guaranteed to be given in clockwise order, and will form a strictly convex polygon (in particular, no three consecutive points lie on the same straight line).

## Output

Print one real number $D$, which is the maximum real number such that no matter how you move the vertices, the polygon stays convex.

Your answer will be considered correct if its absolute or relative error does not exceed $10^{-6}$.

Namely, let's assume that your answer is $a$ and the answer of the jury is $b$. The checker program will consider your answer correct if .

## Examples

input

```
4
0 0
0 1
1 1
1 0
```

output

```
0.3535533906
```

input

```
6
5 0
10 0
12 -4
10 -8
5 -8
3 -4
```

output

```
1.0000000000
```

## Note

Here is a picture of the first sample

Here is an example of making the polygon non-convex.

This is not an optimal solution, since the maximum distance we moved one point is $\approx 0.4242640687$, whereas we can make it non-convex by only moving each point a distance of at most $\approx 0.3535533906$.

# E. Vulnerable Kerbals

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

You are given an integer $m$, and a list of $n$ distinct integers between $0$ and $m - 1$.

You would like to construct a sequence satisfying the properties:

- Each element is an integer between $0$ and $m - 1$, inclusive.
- All prefix products of the sequence modulo $m$ are distinct.
- No prefix product modulo $m$ appears as an element of the input list.
- The length of the sequence is maximized.

Construct any sequence satisfying the properties above.

## Input

The first line of input contains two integers $n$ and $m$ $(0 \le n < m \le 200\,000)$ — the number of forbidden prefix products and the modulus.

If $n$ is non-zero, the next line of input contains $n$ distinct integers between $0$ and $m - 1$, the forbidden prefix products. If $n$ is zero, this line doesn't exist.

## Output

On the first line, print the number $k$, denoting the length of your sequence.

On the second line, print $k$ space separated integers, denoting your sequence.

## Examples

input

```
0 5
```

output

```
5
1 2 4 3 0
```

input

```
3 10
2 9 1
```

output

```
6
3 9 2 9 8 0
```

## Note

For the first case, the prefix products of this sequence modulo $m$ are $[1, 2, 3, 4, 0]$.

For the second case, the prefix products of this sequence modulo $m$ are $[3, 7, 4, 6, 8, 0]$.