A. Dreamoon and Sums

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Dreamoon loves summing up something for no reason. One day he obtains two integers a and b occasionally. He wants to calculate the sum of all *nice* integers. Positive integer x is called *nice* if $\mathrm{mod}(x, b) \neq 0$ and $\frac{\mathrm{div}(x,b)}{\mathrm{mod}(x,b)} = k$, where k is some **integer** number in range $[1, a]$.

By $\mathrm{div}(x, y)$ we denote the *quotient* of integer division of x and y. By $\mathrm{mod}(x, y)$ we denote the *remainder* of integer division of x and y. You can read more about these operations here: `http://goo.gl/AcsXhT`.

The answer may be large, so please print its remainder modulo $1\,000\,000\,007$ $(10^9 + 7)$. Can you compute it faster than Dreamoon?

Input

The single line of the input contains two integers a, b ($1 \le a, b \le 10^7$).

Output

Print a single integer representing the answer modulo $1\,000\,000\,007$ $(10^9 + 7)$.

Sample test(s)

input
```
1 1
```
output
```
0
```

input
```
2 2
```
output
```
8
```

Note

For the first sample, there are no nice integers because $\mathrm{mod}(x, 1)$ is always zero.

For the second sample, the set of nice integers is $\{3, 5\}$.

B. Dreamoon and Sets

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Dreamoon likes to play with sets, integers and $\gcd$. $\gcd(a, b)$ is defined as the largest positive integer that divides both a and b.

Let S be a set of exactly four distinct integers greater than 0. Define S to be of rank k if and only if for all pairs of distinct elements s_i, s_j from S, $\gcd(s_i, s_j) = k$.

Given k and n, Dreamoon wants to make up n sets of rank k using integers from 1 to m such that no integer is used in two different sets (of course you can leave some integers without use). Calculate the minimum m that makes it possible and print one possible solution.

Input

The single line of the input contains two space separated integers n, k ($1 \le n \le 10\,000, 1 \le k \le 100$).

Output

On the first line print a single integer — the minimal possible m.

On each of the next n lines print four space separated integers representing the i-th set.

Neither the order of the sets nor the order of integers within a set is important. If there are multiple possible solutions with minimal m, print any one of them.

Sample test(s)

input

```
1 1
```

output

```
5
1 2 3 5
```

input

```
2 2
```

output

```
22
2 4 6 22
14 18 10 16
```

Note

For the first example it's easy to see that set $\{1, 2, 3, 4\}$ isn't a valid set of rank 1 since $\gcd(2, 4) = 2 \neq 1$.

C. Dreamoon and Strings

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

Dreamoon has a string s and a pattern string p. He first removes exactly x characters from s obtaining string s' as a result. Then he calculates $\operatorname{occ}(s', p)$ that is defined as the maximal number of non-overlapping substrings equal to p that can be found in s'. He wants to make this number as big as possible.

More formally, let's define $\operatorname{ans}(x)$ as maximum value of $\operatorname{occ}(s', p)$ over all s' that can be obtained by removing exactly x characters from s. Dreamoon wants to know $\operatorname{ans}(x)$ for all x from 0 to $|s|$ where $|s|$ denotes the length of string s.

Input

The first line of the input contains the string s $(1 \le |s| \le 2\ 000)$.

The second line of the input contains the string p $(1 \le |p| \le 500)$.

Both strings will only consist of lower case English letters.

Output

Print $|s| + 1$ space-separated integers in a single line representing the $\operatorname{ans}(x)$ for all x from 0 to $|s|$.

Sample test(s)

input

```
aaaaa
aa
```

output

```
2 2 1 1 0 0
```

input

```
axbaxxb
ab
```

output

```
0 1 1 2 1 1 0 0
```

Note

For the first sample, the corresponding optimal values of s' after removal 0 through $|s| = 5$ characters from s are {"**aaaa**a", "**aaaa**", "**aa**a", "**aa**", "a", ""}.

For the second sample, possible corresponding optimal values of s' are {"axbaxxb", "**ab**axxb", "axb**ab**", "**abab**", "**ab**a", "**ab**", "a", ""}.

D. Dreamoon and Binary

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

Dreamoon saw a large integer x written on the ground and wants to print its binary form out. Dreamoon has accomplished the part of turning x into its binary format. Now he is going to print it in the following manner.

He has an integer $n = 0$ and can only perform the following two operations in any order for unlimited times each:

1. Print n in binary form without leading zeros, each print will append to the right of previous prints.
2. Increase n by 1.

Let's define an *ideal sequence* as a sequence of operations that can successfully print binary representation of x without leading zeros and ends with a print operation (i.e. operation 1). Dreamoon wants to know how many different ideal sequences are there and the length (in operations) of the shortest ideal sequence.

The answers might be large so please print them modulo 1000000007 ($10^9 + 7$).

Let's define the string representation of an ideal sequence as a string of `'1'` and `'2'` where the i-th character in the string matches the i-th operation performed. Two ideal sequences are called different if their string representations are different.

Input

The single line of the input contains a binary integer representing x ($1 \leq x < 2^{5000}$) without leading zeros.

Output

The first line of the output should contain an integer representing the number of different ideal sequences modulo 1000000007 ($10^9 + 7$).

The second line of the output contains an integer representing the minimal length of an ideal sequence modulo 1000000007 ($10^9 + 7$).

Sample test(s)

input

```
101
```

output

```
1
6
```

input

```
11010
```

output

```
3
5
```

Note

For the first sample, the shortest and the only ideal sequence is «`222221`» of length 6.

For the second sample, there are three ideal sequences «`21211`», «`212222222221`», «`222222222222222222222222221`». Among them the shortest one has length 5.

E. Dreamoon and Notepad

time limit per test: 3.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Dreamoon has just created a document of hard problems using notepad.exe. The document consists of n lines of text, a_i denotes the length of the i-th line. He now wants to know what is the fastest way to move the cursor around because the document is really long.

Let (r, c) be a current cursor position, where r is row number and c is position of cursor in the row. We have $1 \le r \le n$ and $0 \le c \le a_r$.

We can use following six operations in notepad.exe to move our cursor assuming the current cursor position is at (r, c):

1. up key: the new cursor position $(nr, nc) = (max(r - 1, 1), min(a_{nr}, c))$
2. down key: the new cursor position $(nr, nc) = (min(r + 1, n), min(a_{nr}, c))$
3. left key: the new cursor position $(nr, nc) = (r, max(0, c - 1))$
4. right key: the new cursor position $(nr, nc) = (r, min(a_{nr}, c + 1))$
5. HOME key: the new cursor position $(nr, nc) = (r, 0)$
6. END key: the new cursor position $(nr, nc) = (r, a_r)$

You're given the document description (n and sequence a_i) and q queries from Dreamoon. Each query asks what minimal number of key presses is needed to move the cursor from (r_1, c_1) to (r_2, c_2).

Input

The first line contains an integer $n (1 \le n \le 400, 000)$ — the number of lines of text.

The second line contains n integers $a_1, a_2, ..., a_n (1 \le a_i \le 10^8)$.

The third line contains an integer $q (1 \le q \le 400, 000)$.

Each of the next q lines contains four integers r_1, c_1, r_2, c_2 representing a query $(1 \le r_1, r_2 \le n, 0 \le c_1 \le a_{r1}, 0 \le c_2 \le a_{r2})$.

Output

For each query print the result of the query.

Sample test(s)

input

```
9
1 3 5 3 1 3 5 3 1
4
3 5 3 1
3 3 7 3
1 0 3 3
6 0 7 3
```

output

```
2
5
3
2
```

input

```
2
10 5
1
1 0 1 5
```

output

```
3
```

Note

In the first sample, the first query can be solved with keys: HOME, right.

The second query can be solved with keys: down, down, down, END, down.

The third query can be solved with keys: down, END, down.

The fourth query can be solved with keys: END, down.