Codeforces Round #166 (Div. 2)

A. Beautiful Year

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

It seems like the year of 2013 came only yesterday. Do you know a curious fact? The year of 2013 is the first year after the old 1987 with only distinct digits.

Now you are suggested to solve the following problem: given a year number, find the minimum year number which is strictly larger than the given one and has only distinct digits.

Input

The single line contains integer y $(1000 \le y \le 9000)$ — the year number.

Output

Print a single integer — the minimum year number that is strictly larger than y and all it's digits are distinct. It is guaranteed that the answer exists.

Sample test(s)

input
1987
output
2013

input
2013
output
2014

B. Prime Matrix

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

You've got an $n \times m$ matrix. The matrix consists of integers. In one move, you can apply a single transformation to the matrix: choose an arbitrary element of the matrix and increase it by 1. Each element can be increased an arbitrary number of times.

You are really curious about prime numbers. Let us remind you that a *prime number* is a positive integer that has exactly two distinct positive integer divisors: itself and number one. For example, numbers 2, 3, 5 are prime and numbers 1, 4, 6 are not.

A matrix is *prime* if at least one of the two following conditions fulfills:

- the matrix has a row with prime numbers only;
- the matrix has a column with prime numbers only;

Your task is to count the minimum number of moves needed to get a prime matrix from the one you've got.

Input

The first line contains two integers n, m $(1 \le n, m \le 500)$ — the number of rows and columns in the matrix, correspondingly.

Each of the following n lines contains m integers — the initial matrix. All matrix elements are positive integers. All numbers in the initial matrix do not exceed 10^5.

The numbers in the lines are separated by single spaces.

Output

Print a single integer — the minimum number of moves needed to get a prime matrix from the one you've got. If you've got a prime matrix, print 0.

Sample test(s)

input
3 3 1 2 3 5 6 1 4 4 1

output
1

input
2 3 4 8 8 9 2 9

output
3

input
2 2 1 3 4 2

output
0

Note

In the first sample you need to increase number 1 in cell (1, 1). Thus, the first row will consist of prime numbers: 2, 2, 3.

In the second sample you need to increase number 8 in cell (1, 2) three times. Thus, the second column will consist of prime numbers: 11, 2.

In the third sample you don't have to do anything as the second column already consists of prime numbers: 3, 2.

C. Secret

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

The Greatest Secret Ever consists of n words, indexed by positive integers from 1 to n. The secret needs dividing between k Keepers (let's index them by positive integers from 1 to k), the i-th Keeper gets a **non-empty** set of words with numbers from the set $U_i = (u_{i,1}, u_{i,2}, ..., u_{i,|U_i|})$. Here and below we'll presuppose that the set elements are written in the increasing order.

We'll say that the *secret is safe* if the following conditions are hold:

- for any two indexes i, j $(1 \le i < j \le k)$ the intersection of sets U_i and U_j is an empty set;
- the union of sets $U_1, U_2, ..., U_k$ is set $(1, 2, ..., n)$;
- in each set U_i, its elements $u_{i,1}, u_{i,2}, ..., u_{i,|U_i|}$ **do not form** an arithmetic progression (in particular, $|U_i| \ge 3$ should hold).

Let us remind you that the elements of set $(u_1, u_2, ..., u_s)$ form an arithmetic progression if there is such number d, that for all i $(1 \le i < s)$ fulfills $u_i + d = u_{i+1}$. For example, the elements of sets (5), $(1, 10)$ and $(1, 5, 9)$ form arithmetic progressions and the elements of sets $(1, 2, 4)$ and $(3, 6, 8)$ don't.

Your task is to find any partition of the set of words into subsets $U_1, U_2, ..., U_k$ so that the secret is safe. Otherwise indicate that there's no such partition.

Input

The input consists of a single line which contains two integers n and k $(2 \le k \le n \le 10^6)$ — the number of words in the secret and the number of the Keepers. The numbers are separated by a single space.

Output

If there is no way to keep the secret safe, print a single integer "-1" (without the quotes). Otherwise, print n integers, the i-th of them representing the number of the Keeper who's got the i-th word of the secret.

If there are multiple solutions, print any of them.

Sample test(s)

input
11 3
output
3 1 2 1 1 2 3 2 2 3 1

input
5 2
output
-1

D. Good Substrings

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

You've got string s, consisting of small English letters. Some of the English letters are *good*, the rest are *bad*.

A substring $s[l...r]$ $(1 \le l \le r \le |s|)$ of string $s = s_1 s_2 ... s_{|s|}$ (where $|s|$ is the length of string s) is string $s_l s_{l+1} ... s_r$.

The substring $s[l...r]$ is *good*, if among the letters $s_l, s_{l+1}, ..., s_r$ there are **at most** k **bad** ones (look at the sample's explanation to understand it more clear).

Your task is to find the number of distinct good substrings of the given string s. Two substrings $s[x...y]$ and $s[p...q]$ are considered distinct if their content is different, i.e. $s[x...y] \neq s[p...q]$.

Input

The first line of the input is the non-empty string s, consisting of small English letters, the string's length is at most 1500 characters.

The second line of the input is the string of characters "0" and "1", the length is exactly 26 characters. If the i-th character of this string equals "1", then the i-th English letter is good, otherwise it's bad. That is, the first character of this string corresponds to letter "a", the second one corresponds to letter "b" and so on.

The third line of the input consists a single integer k $(0 \le k \le |s|)$ — the maximum acceptable number of bad characters in a good substring.

Output

Print a single integer — the number of distinct good substrings of string s.

Sample test(s)

input
ababab 01000000000000000000000000 1
output
5

input
acbacbacaa 00000000000000000000000000 2
output
8

Note

In the first example there are following good substrings: "a", "ab", "b", "ba", "bab".

In the second example there are following good substrings: "a", "aa", "ac", "b", "ba", "c", "ca", "cb".

E. Three Horses

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

There are three horses living in a horse land: one gray, one white and one gray-and-white. The horses are really amusing animals, which is why they adore special cards. Each of those cards must contain two integers, the first one on top, the second one in the bottom of the card. Let's denote a card with a on the top and b in the bottom as (a, b).

Each of the three horses can paint the special cards. If you show an (a, b) card to the gray horse, then the horse can paint a new $(a+1, b+1)$ card. If you show an (a, b) card, such that a and b are even integers, to the white horse, then the horse can paint a new $\left(\frac{a}{2}, \frac{b}{2}\right)$ card. If you show two cards (a, b) and (b, c) to the gray-and-white horse, then he can paint a new (a, c) card.

Polycarpus really wants to get n special cards $(1, a_1)$, $(1, a_2)$, ..., $(1, a_n)$. For that he is going to the horse land. He can take exactly one (x, y) card to the horse land, such that $1 \le x < y \le m$. How many ways are there to choose the card so that he can perform some actions in the horse land and get the required cards?

Polycarpus can get cards from the horses only as a result of the actions that are described above. Polycarpus is allowed to get additional cards besides the cards that he requires.

Input

The first line contains two integers n, m $(1 \le n \le 10^5, 2 \le m \le 10^9)$. The second line contains the sequence of integers $a_1, a_2, ..., a_n$ $(2 \le a_i \le 10^9)$. Note, that the numbers in the sequence can coincide.

The numbers in the lines are separated by single spaces.

Output

Print a single integer — the answer to the problem.

Please, do not use the `%lld` specifier to read or write 64-bit integers in *C++*. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

input
1 6 2

output
11

input
1 6 7

output
14

input
2 10 13 7

output
36