Yandex.Algorithm 2011
Round 2

A. Reflection

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

For each positive integer n consider the integer $\psi(n)$ which is obtained from n by replacing every digit a in the decimal notation of n with the digit $(9 - a)$. We say that $\psi(n)$ is the *reflection* of n. For example, reflection of 192 equals 807. Note that leading zeros (if any) should be omitted. So reflection of 9 equals 0, reflection of 91 equals 8.

Let us call the *weight* of the number the product of the number and its reflection. Thus, the weight of the number 10 is equal to $10 \cdot 89 = 890$.

Your task is to find the maximum weight of the numbers in the given range $[l, r]$ (boundaries are included).

Input
Input contains two space-separated integers l and r ($1 \le l \le r \le 10^9$) — bounds of the range.

Output
Output should contain single integer number: maximum value of the product $n \cdot \psi(n)$, where $l \le n \le r$.

Please, do not use `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use `cout` (also you may use `%I64d`).

Sample test(s)

input
3 7
output
20

input
1 1
output
8

input
8 10
output
890

Note
In the third sample weight of 8 equals $8 \cdot 1 = 8$, weight of 9 equals $9 \cdot 0 = 0$, weight of 10 equals 890.

Thus, maximum value of the product is equal to 890.

B. Tetris revisited

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

Physicist Woll likes to play one relaxing game in between his search of the theory of everything.

Game interface consists of a rectangular $n \times m$ playing field and a dashboard. Initially some cells of the playing field are filled while others are empty. Dashboard contains images of all various connected (we mean connectivity by side) figures of 2, 3, 4 and 5 cells, with all their rotations and reflections. Player can copy any figure from the dashboard and place it anywhere at the still empty cells of the playing field. Of course any figure can be used as many times as needed.

Woll's aim is to fill the whole field in such a way that there are no empty cells left, and also... just have some fun.

Every initially empty cell should be filled with exactly one cell of some figure. Every figure should be entirely inside the board.

In the picture black cells stand for initially filled cells of the field, and one-colour regions represent the figures.

Input

First line contains integers n and m $(1 \le n, m \le 1000)$ — the height and the width of the field correspondingly. Next n lines contain m symbols each. They represent the field in a natural way: j-th character of the i-th line is "#" if the corresponding cell is filled, and "." if it is empty.

Output

If there is no chance to win the game output the only number "-1" (without the quotes). Otherwise output any filling of the field by the figures in the following format: each figure should be represented by some digit and figures that touch each other by side should be represented by distinct digits. Every initially filled cell should be represented by "#".

Sample test(s)

input
2 3 ... #.#
output
000 #0#

input
3 3 .#. ... ..#
output
5#1 511 55#

input
3 3 ... .## .#.
output
-1

input
1 2 ##
output
##

Note

In the third sample, there is no way to fill a cell with no empty neighbours.

In the forth sample, Woll does not have to fill anything, so we should output the field from the input.

C. Genetic engineering

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

"Multidimensional spaces are completely out of style these days, unlike genetics problems" — thought physicist Woll and changed his subject of study to bioinformatics. Analysing results of sequencing he faced the following problem concerning DNA sequences. We will further think of a DNA sequence as an arbitrary string of uppercase letters "A", "C", "G" and "T" (of course, this is a simplified interpretation).

Let w be a long DNA sequence and $s_1, s_2, ..., s_m$ — collection of short DNA sequences. Let us say that the collection *filters* w iff w can be covered with the sequences from the collection. Certainly, substrings corresponding to the different positions of the string may intersect or even cover each other. More formally: denote by $|w|$ the length of w, let symbols of w be numbered from 1 to $|w|$. Then for each position i in w there exist pair of indices l, r $(1 \le l \le i \le r \le |w|)$ such that the substring $w[l \ldots r]$ equals one of the elements $s_1, s_2, ..., s_m$ of the collection.

Woll wants to calculate the number of DNA sequences of a given length filtered by a given collection, but he doesn't know how to deal with it. Help him! Your task is to find the number of different DNA sequences of length n filtered by the collection $\{s_i\}$.

Answer may appear very large, so output it modulo 1000000009.

Input

First line contains two integer numbers n and m $(1 \le n \le 1000, 1 \le m \le 10)$ — the length of the string and the number of sequences in the collection correspondently.

Next m lines contain the collection sequences s_i, one per line. Each s_i is a nonempty string of length not greater than 10. All the strings consist of uppercase letters "A", "C", "G", "T". The collection may contain identical strings.

Output

Output should contain a single integer — the number of strings filtered by the collection modulo 1000000009 $(10^9 + 9)$.

Sample test(s)

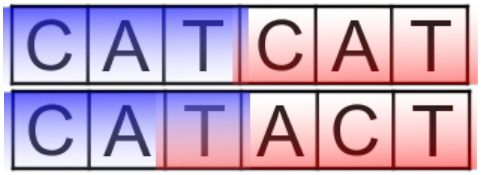
input
2 1 A
output
1

input
6 2 CAT TACT
output
2

Note

In the first sample, a string has to be filtered by "A". Clearly, there is only one such string: "AA".

In the second sample, there exist exactly two different strings satisfying the condition (see the pictures below).



D. Powerful array

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

An array of positive integers $a_1, a_2, ..., a_n$ is given. Let us consider its arbitrary subarray $a_l, a_{l+1}..., a_r$, where $1 \le l \le r \le n$. For every positive integer s denote by K_s the number of occurrences of s into the subarray. We call the *power* of the subarray the sum of products $K_s \cdot K_s \cdot s$ for every positive integer s. The sum contains only finite number of nonzero summands as the number of different values in the array is indeed finite.

You should calculate the power of t given subarrays.

Input

First line contains two integers n and t ($1 \le n, t \le 200000$) — the array length and the number of queries correspondingly.

Second line contains n positive integers a_i ($1 \le a_i \le 10^6$) — the elements of the array.

Next t lines contain two positive integers l, r ($1 \le l \le r \le n$) each — the indices of the left and the right ends of the corresponding subarray.

Output

Output t lines, the i-th line of the output should contain single positive integer — the power of the i-th query subarray.

Please, do not use `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use `cout` stream (also you may use `%I64d`).

Sample test(s)

input
3 2 1 2 1 1 2 1 3

output
3 6

input
8 3 1 1 2 2 1 3 1 1 2 7 1 6 2 7

output
20 20 20

Note

Consider the following array (see the second sample) and its [2, 7] subarray (elements of the subarray are colored):

Then $K_1 = 3, K_2 = 2, K_3 = 1$, so the power is equal to $3^2 \cdot 1 + 2^2 \cdot 2 + 1^2 \cdot 3 = 20$.

E. Long sequence

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

A sequence $a_0, a_1, ...$ is called a *recurrent binary sequence*, if each term a_i $(i = 0, 1, ...)$ is equal to 0 or 1 and there exist coefficients $c_1, c_2, \ldots, c_k \in \{0, 1\}$ such that

$$a_n = c_1 \cdot a_{n-1} + c_2 \cdot a_{n-2} + ... + c_k \cdot a_{n-k} \pmod 2,$$

for all $n \geq k$. Assume that not all of c_i are zeros.

Note that such a sequence can be uniquely recovered from any k-tuple $\{a_s, a_{s+1}, ..., a_{s+k-1}\}$ and so it is periodic. Moreover, if a k-tuple contains only zeros, then the sequence contains only zeros, so this case is not very interesting. Otherwise the minimal period of the sequence is not greater than $2^k - 1$, as k-tuple determines next element, and there are $2^k - 1$ non-zero k-tuples. Let us call a sequence *long* if its minimal period is exactly $2^k - 1$. Your task is to find a long sequence for a given k, if there is any.

Input

Input contains a single integer k $(2 \leq k \leq 50)$.

Output

If there is no long sequence for a given k, output "-1" (without quotes). Otherwise the first line of the output should contain k integer numbers: $c_1, c_2, ..., c_k$ (coefficients). The second line should contain first k elements of the sequence: $a_0, a_1, ..., a_{k-1}$. All of them (elements and coefficients) should be equal to 0 or 1, and at least one c_i has to be equal to 1.

If there are several solutions, output any.

Sample test(s)

input
2
output
1 1 1 0

input
3
output
0 1 1 1 1 1

Note

1. In the first sample: $c_1 = 1, c_2 = 1$, so $a_n = a_{n-1} + a_{n-2} \pmod 2$. Thus the sequence will be:

so its period equals $3 = 2^2 - 1$.

2. In the second sample: $c_1 = 0, c_2 = 1, c_3 = 1$, so $a_n = a_{n-2} + a_{n-3} \pmod 2$. Thus our sequence is:

and its period equals $7 = 2^3 - 1$.

Periods are colored.