Good Bye 2017

A. New Year and Counting Cards

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

Your friend has n cards.

You know that each card has a lowercase English letter on one side and a digit on the other.

Currently, your friend has laid out the cards on a table so only one side of each card is visible.

You would like to know if the following statement is true for cards that your friend owns: "If a card has a vowel on one side, then it has an even digit on the other side." More specifically, a vowel is one of 'a', 'e', 'i', 'o' or 'u', and even digit is one of '0', '2', '4', '6' or '8'.

For example, if a card has 'a' on one side, and '6' on the other side, then this statement is true for it. Also, the statement is true, for example, for a card with 'b' and '4', and for a card with 'b' and '3' (since the letter is not a vowel). The statement is false, for example, for card with 'e' and '5'. You are interested if the statement is true for all cards. In particular, if no card has a vowel, the statement is true.

To determine this, you can flip over some cards to reveal the other side. You would like to know what is the minimum number of cards you need to flip in the worst case in order to verify that the statement is true.

Input

The first and only line of input will contain a string s $(1 \leq |s| \leq 50)$, denoting the sides of the cards that you can see on the table currently. Each character of s is either a lowercase English letter or a digit.

Output

Print a single integer, the minimum number of cards you must turn over to verify your claim.

Examples

input
ee
output
2

input
z
output
0

input
0ay1
output
2

Note

In the first sample, we must turn over both cards. Note that even though both cards have the same letter, they could possibly have different numbers on the other side.

In the second sample, we don't need to turn over any cards. The statement is vacuously true, since you know your friend has no cards with a vowel on them.

In the third sample, we need to flip the second and fourth cards.

B. New Year and Buggy Bot

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

Bob programmed a robot to navigate through a 2d maze.

The maze has some obstacles. Empty cells are denoted by the character '.', where obstacles are denoted by '#'.

There is a single robot in the maze. Its start position is denoted with the character 'S'. This position has no obstacle in it. There is also a single exit in the maze. Its position is denoted with the character 'E'. This position has no obstacle in it.

The robot can only move up, left, right, or down.

When Bob programmed the robot, he wrote down a string of digits consisting of the digits 0 to 3, inclusive. He intended for each digit to correspond to a distinct direction, and the robot would follow the directions in order to reach the exit. Unfortunately, he forgot to actually assign the directions to digits.

The robot will choose some random mapping of digits to distinct directions. The robot will map distinct digits to distinct directions. The robot will then follow the instructions according to the given string in order and chosen mapping. If an instruction would lead the robot to go off the edge of the maze or hit an obstacle, the robot will crash and break down. If the robot reaches the exit at any point, then the robot will stop following any further instructions.

Bob is having trouble debugging his robot, so he would like to determine the number of mappings of digits to directions that would lead the robot to the exit.

Input

The first line of input will contain two integers n and m ($2 \leq n, m \leq 50$), denoting the dimensions of the maze.

The next n lines will contain exactly m characters each, denoting the maze.

Each character of the maze will be '.', '#', 'S', or 'E'.

There will be exactly one 'S' and exactly one 'E' in the maze.

The last line will contain a single string s ($1 \leq |s| \leq 100$) — the instructions given to the robot. Each character of s is a digit from 0 to 3.

Output

Print a single integer, the number of mappings of digits to directions that will lead the robot to the exit.

Examples

input

```
5 6
.....#
S....#
.#....
.#....
...E..
333300012
```

output

```
1
```

input

```
6 6
......
......
..SE..
......
......
......
01232123212302123021
```

output

```
14
```

input

```
5 3
...
.S.
###
.E.
...
3
```

output

```
0
```

Note

For the first sample, the only valid mapping is $0 \to D, 1 \to L, 2 \to U, 3 \to R$, where D is down, L is left, U is up, R is right.

C. New Year and Curling

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

Carol is currently curling.

She has n disks each with radius r on the 2D plane.

Initially she has all these disks above the line $y = 10^{100}$.

She then will slide the disks towards the line $y = 0$ one by one in order from 1 to n.

When she slides the i-th disk, she will place its center at the point $(x_i, 10^{100})$. She will then push it so the disk's y coordinate continuously decreases, and x coordinate stays constant. The disk stops once it touches the line $y = 0$ or it touches any previous disk. Note that once a disk stops moving, it will not move again, even if hit by another disk.

Compute the y-coordinates of centers of all the disks after all disks have been pushed.

Input

The first line will contain two integers n and r ($1 \le n, r \le 1\,000$), the number of disks, and the radius of the disks, respectively.

The next line will contain n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 1\,000$) — the x-coordinates of the disks.

Output

Print a single line with n numbers. The i-th number denotes the y-coordinate of the center of the i-th disk. The output will be accepted if it has absolute or relative error at most 10^{-6}.

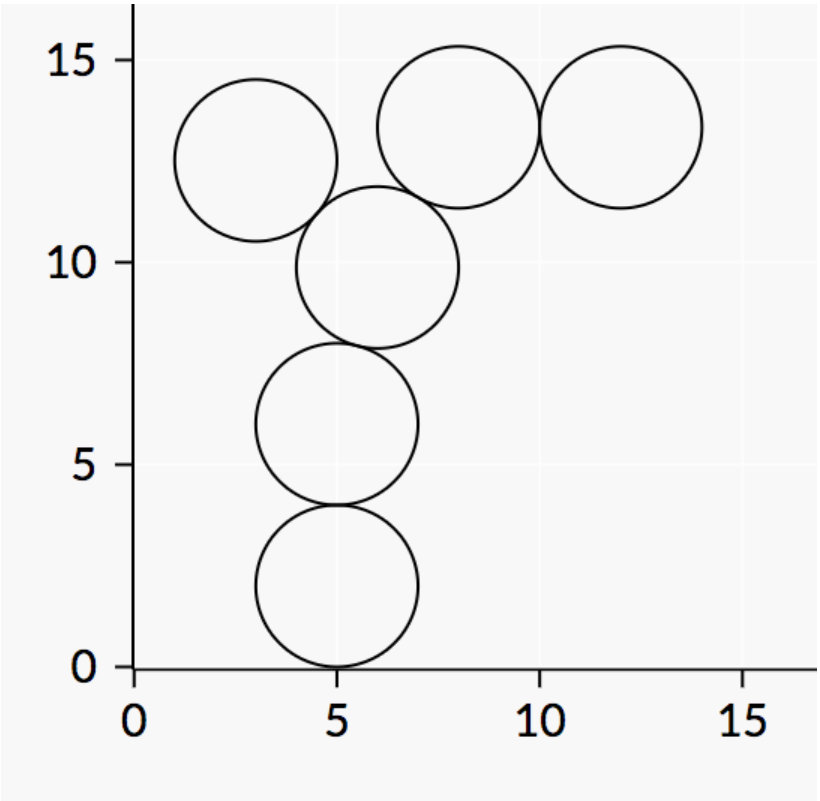
Namely, let's assume that your answer for a particular value of a coordinate is a and the answer of the jury is b. The checker program will consider your answer correct if $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$ for all coordinates.

Example

input
6 2 5 5 6 8 3 12
output
2 6.0 9.87298334621 13.3370849613 12.5187346573 13.3370849613

Note

The final positions of the disks will look as follows:



In particular, note the position of the last disk.

D. New Year and Arbitrary Arrangement

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

You are given three integers k, p_a and p_b.

You will construct a sequence with the following algorithm: Initially, start with the empty sequence. Each second, you do the following. With probability $p_a / (p_a + p_b)$, add 'a' to the end of the sequence. Otherwise (with probability $p_b / (p_a + p_b)$), add 'b' to the end of the sequence.

You stop once there are at least k subsequences that form 'ab'. Determine the expected number of times 'ab' is a subsequence in the resulting sequence. It can be shown that this can be represented by P / Q, where P and Q are coprime integers, and $Q \neq 0 \bmod (10^9 + 7)$. Print the value of $P \cdot Q^{-1} \bmod (10^9 + 7)$.

Input

The first line will contain three integers integer k, p_a, p_b ($1 \leq k \leq 1\,000$, $1 \leq p_a, p_b \leq 1\,000\,000$).

Output

Print a single integer, the answer to the problem.

Examples

input
1 1 1
output
2

input
3 1 4
output
370000006

Note

The first sample, we will keep appending to our sequence until we get the subsequence 'ab' at least once. For instance, we get the sequence 'ab' with probability 1/4, 'bbab' with probability 1/16, and 'aab' with probability 1/8. Note, it's impossible for us to end with a sequence like 'aabab', since we would have stopped our algorithm once we had the prefix 'aab'.

The expected amount of times that 'ab' will occur across all valid sequences is 2.

For the second sample, the answer is equal to $\frac{341}{100}$.

E. New Year and Entity Enumeration

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

You are given an integer m.

Let $M = 2^m - 1$.

You are also given a set of n integers denoted as the set T. The integers will be provided in base 2 as n binary strings of length m.

A set of integers S is called "good" if the following hold.

1. If $a \in S$, then $a \text{ XOR } M \in S$.
2. If $a, b \in S$, then $a \text{ AND } b \in S$
3. $T \subseteq S$
4. All elements of S are less than or equal to M.

Here, XOR and AND refer to the bitwise XOR and bitwise AND operators, respectively.

Count the number of good sets S, modulo $10^9 + 7$.

Input

The first line will contain two integers m and n ($1 \le m \le 1\,000$, $1 \le n \le min(2^m, 50)$).

The next n lines will contain the elements of T. Each line will contain exactly m zeros and ones. Elements of T will be distinct.

Output

Print a single integer, the number of good sets modulo $10^9 + 7$.

Examples

input
5 3 11010 00101 11000
output
4

input
30 2 01010101010100101010101010 11011011011011001101101101 1011
output
860616440

Note

An example of a valid set S is {00000, 00101, 00010, 00111, 11000, 11010, 11101, 11111}.

F. New Year and Rainbow Roads

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

Roy and Biv have a set of n points on the infinite number line.

Each point has one of 3 colors: red, green, or blue.

Roy and Biv would like to connect all the points with some edges. Edges can be drawn between any of the two of the given points. The cost of an edge is equal to the distance between the two points it connects.

They want to do this in such a way that they will both see that all the points are connected (either directly or indirectly).

However, there is a catch: Roy cannot see the color red and Biv cannot see the color blue.

Therefore, they have to choose the edges in such a way that if all the red points are removed, the remaining blue and green points are connected (and similarly, if all the blue points are removed, the remaining red and green points are connected).

Help them compute the minimum cost way to choose edges to satisfy the above constraints.

Input

The first line will contain an integer n ($1 \leq n \leq 300\,000$), the number of points.

The next n lines will contain two tokens p_i and c_i (p_i is an integer, $1 \leq p_i \leq 10^9$, c_i is a uppercase English letter 'R', 'G' or 'B'), denoting the position of the i-th point and the color of the i-th point. 'R' means red, 'G' denotes green, and 'B' means blue. The positions will be in strictly increasing order.

Output

Print a single integer, the minimum cost way to solve the problem.

Examples

input
4 1 G 5 R 10 B 15 G
output
23

input
4 1 G 2 R 3 B 10 G
output
12

Note

In the first sample, it is optimal to draw edges between the points (1,2), (1,4), (3,4). These have costs 4, 14, 5, respectively.

G. New Year and Original Order

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

Let $S(n)$ denote the number that represents the digits of n in sorted order. For example, $S(1) = 1$, $S(5) = 5$, $S(50394) = 3459$, $S(353535) = 333555$.

Given a number X, compute $\sum_{1 \le k \le X} S(k)$ modulo $10^9 + 7$.

Input

The first line of input will contain the integer X ($1 \le X \le 10^{700}$).

Output

Print a single integer, the answer to the question.

Examples

input
21
output
195

input
345342
output
390548434

Note

The first few values of S are $1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 12$. The sum of these values is 195.

H. New Year and Boolean Bridges

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Your friend has a hidden directed graph with n nodes.

Let $f(u, v)$ be true if there is a directed path from node u to node v, and false otherwise. For each pair of distinct nodes, u, v, you know at least one of the three statements is true:

1. $f(u,v)$ AND $f(v,u)$
2. $f(u,v)$ OR $f(v,u)$
3. $f(u,v)$ XOR $f(v,u)$

Here AND, OR and XOR mean AND, OR and exclusive OR operations, respectively.

You are given an n by n matrix saying which one of the three statements holds for each pair of vertices. The entry in the u-th row and v-th column has a single character.

1. If the first statement holds, this is represented by the character '`A`'.
2. If the second holds, this is represented by the character '`O`'.
3. If the third holds, this is represented by the character '`X`'.
4. The diagonal of this matrix will only contain the character '`-`'.

Note that it is possible that a pair of nodes may satisfy multiple statements, in which case, the character given will represent one of the true statements for that pair. This matrix is also guaranteed to be symmetric.

You would like to know if there is a directed graph that is consistent with this matrix. If it is impossible, print the integer `-1`. Otherwise, print the minimum number of edges that could be consistent with this information.

Input

The first line will contain an integer n ($1 \le n \le 47$), the number of nodes.

The next n lines will contain n characters each: the matrix of what you know about the graph connectivity in the format described in the statement.

Output

Print the minimum number of edges that is consistent with the given information, or `-1` if it is impossible.

Examples

input
4 -AAA A-AA AA-A AAA-
output
4

input
3 -XX X-X XX-
output
2

Note

Sample 1: The hidden graph is a strongly connected graph. We can put all four nodes in a cycle.

Sample 2: One valid graph is $3 \rightarrow 1 \rightarrow 2$. For each distinct pair, exactly one of $f(u, v)$, $f(v, u)$ holds.