Problem A. Binary Codes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given a finite alphabet S, a binary code over this alphabet S is a function that maps each element of S to some (possibly empty) string over the alphabet $\{0, 1\}$.

An example of such a code for $S = \{a, b, c, d\}$ is the function f defined by $f(a) = 1$, $f(b) = 1010$, $f(c) = 01$, $f(d) = 10101$.

Any binary code can be naturally extended to encode strings over the alphabet S simply by concatenating the codes of the string's letters, in order. For example, using the code mentioned above we can encode "`cac`" as $f(cac) = 01101$.

A code is called ambiguous if there are two different strings over S that have the same encoding. Obviously, in practice we want to avoid using an ambiguous code.

A code is called really ambiguous if there are three different strings over S that have the same encoding. For example, the code from the above example is really ambiguous: the strings "`ba`", "`acc`", and "`d`" are all encoded to 10101.

You will be given a code containing the strings over $\{0, 1\}$ used to encode letters of some alphabet S. Your method should check whether this code is really ambiguous. If it is really ambiguous, find a shortest string over $\{0, 1\}$ that is an encoding of (at least) three different strings over S, and output its length. If the given code is not really ambiguous, output -1.

Input

First line of input contains number $2 \le n \le 30$ — the number of strings in the code.

Following n lines contain strings in the code, length of each string is more than 0 and less than or equal to 50. Each string consists of characters '`0`' and '`1`'.

Output

Print a single integer — length of the shortest string that is an encoding of (at least) three different strings over S or -1, if the code is not really ambiguous.

Examples

standard input	standard output
4 1 1010 01 10101	5
2 0 1	-1
4 0 11 11 11	2
5 0000 001 01001 01010 01011	-1

Problem B. Beautiful board

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You have a $n \times n$ board and several checkers of different colors. You want to place all of the checkers on the board in such a way that no cell contains more than one checker.

Output the number of different possible placements modulo 1234567891. Two placements are equal if you can get one from the other by rotating the board. Note that checkers of the same color are indistinguishable. If you have more checkers than the number of cells on the board, there are no possible placements, so you should output 0.

Input

First line of input contains two integers n and m ($1 \le n \le 10^5$, $1 \le m \le 50$).

Second line contains m numbers between 1 and 10^5; i-th number is number of checkers of color i.

Output

Print the number of possible placements modulo 1234567891.

Examples

standard input	standard output
1 1 1	1
2 2 1 2	3
3 4 4 2 1 3	0

Note

You are *not allowed* to flip the board.

Problem C. Mountain Map

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

A mountain map is a $N \times M$ rectangular grid where each cell has a height. The heights are positive integers between 1 and $N \times M$, inclusive, and no two distinct cells have equal heights. Two cells are called neighboring if they share a side or a corner. A cell is called locally minimal if its height is less than the heights of all its neighboring cells.

You are given the grid. The j-th character of the i-th line of grid is '`X`' if the j-th cell in the i-th row of the mountain map is locally minimal, and '`.`' otherwise. Calculate the number of distinct mountain maps that match the grid. Return the result modulo 12345678.

Input

First line contains two integers N and M ($1 \le N \le 4$, $1 \le M \le 7$). Next N lines contain lines of grid, each one of length M and consists of '`X`' and '`.`' only.

Output

Output the number of distinct mountain maps modulo 12345678.

Examples

standard input	standard output
1 3 .X.	2
2 2 X. .X	0
2 3	0

Problem D. Wardrobe

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are attempting to assemble a wardrobe, but you have misplaced the instructions. The wardrobe has several holes, each of which is designed to accommodate a bolt of a specific size. A hole with size D should be matched with a bolt of size D. However, bolts with sizes $D-1$ and $D+1$ will also fit. Since you don't have the instructions, you decide to do the following: for each bolt, you will randomly choose an available hole in which the bolt will fit, and screw the bolt into that hole. If the bolt cannot fit into any of the available holes, you will skip it and move on to the next one.

You are given n integers containing the sizes of the bolts. For each element in bolts, there is a corresponding hole with the same size. Output the maximum number of unused holes that can remain at the end of this process.

Input

First line contains one integer $1 \le n \le 50$ — number of bolts.

Second line contains n space separated integers, sizes of bolts, each one from 1 to 100.

Output

Output the maximum number of unused holes that can remain at the end of this process.

Examples

standard input	standard output
3 1 2 3	1
5 1 2 3 2 4	1

Problem E. Balls and Bins

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Like all other software engineers, Petya likes to play with bins and balls. He has N bins, numbered 0 through $N-1$. Yesterday, Petya distributed all his balls into the bins, placing S_0 balls into bin 0, S_1 balls into bin 1, and so on. No two bins contained the same number of balls. It is possible that one of the bins contained zero balls.

This morning, Petya attended a lecture about sorting. After he got home, he decided to rearrange the balls in his bins into sorted order. More precisely, he wanted to reach a state with T_0 balls in bin 0, T_1 balls in bin 1, and so on, such that the following two conditions are met:

- T is a permutation of S;
- T is sorted in ascending order.

When rearranging the balls, Petya always moves them one ball at a time. In other words, in each move Petya takes a single ball from one bin and places it into another bin. Petya is very smart, so he always uses the smallest possible number of moves.

For example, when rearranging $S = \{2, 5, 0\}$ to $T = \{0, 2, 5\}$, Petya will make exactly 5 moves. One way of changing S to T in 5 moves: first Petya will move 3 balls from bin 1 to bin 2, and then he will move 2 balls from bin 0 to bin 2.

You are given the T and M — number of moves. We are interested in all possible initial configurations S such that Petya would produce the final configuration T, and in doing so he would move a ball exactly M times. Let C be the number of such configurations S. Since C may be very big, you must compute and output its value modulo $10^9 + 9$.

Input

First line of input contains two integers $2 \le N \le 50$ — size of T and $0 \le M \le 650$ — number of moves.

Second line contains N space-separated integers $0 \le T_i \le 50$.

T will be in a strictly ascending order.

Output

Output single integer — number of possible configurations.

Examples

standard input	standard output
3 1 1 2 3	2
3 2 1 2 3	3
2 650 0 1	0

Problem F. Your Bunny Wrote...

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Bunnies like programming and they sometimes make useless devices.

One group of bunnies made a simple computer, and one of them wrote code that calculates the following function f:

For a positive integer x: $f(x) = x/m$, if x is a multiple of m, and $f(x) = x + 1$ otherwise, where m is a fixed positive integer greater than 1.

For a positive integer x, we define a sequence $\{B_k\}$ by $B_0 = x$ and $B_{k+1} = f(B_k)$. Let $g(x)$ be the smallest index k of this sequence such that $B_k = 1$. Calculate the number of positive integers x satisfying $g(x) = n$, and output the answer modulo $10^9 + 9$.

Input

First line contains two integers m and n ($2 \le m \le 10^6$ and $0 \le n \le 10^6$).

Output

Print single integer — the number of positive x satisfying $g(x) = n$, modulo $10^9 + 9$.

Examples

standard input	standard output
5 3	4
487 0	1
19 10	512