Kotlin Heroes: Practice 8

A. A+B (Trial Problem)

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given two integers a and b. Print $a + b$.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Then t test cases follow.

Each test case is given as a line of two integers a and b ($-1000 \le a, b \le 1000$).

Output

Print t integers — the required numbers $a + b$.

Example

input

```
4
1 5
314 15
-99 99
123 987
```

output

```
6
329
0
1110
```

B. Yellow Cards

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

The final match of the Berland Football Cup has been held recently. The referee has shown n yellow cards throughout the match. At the beginning of the match there were a_1 players in the first team and a_2 players in the second team.

The rules of sending players off the game are a bit different in Berland football. If a player from the first team receives k_1 yellow cards throughout the match, he can no longer participate in the match — he's sent off. And if a player from the second team receives k_2 yellow cards, he's sent off. After a player leaves the match, he can no longer receive any yellow cards. Each of n yellow cards was shown to exactly one player. Even if all players from one team (or even from both teams) leave the match, the game still continues.

The referee has lost his records on who has received each yellow card. Help him to determine the minimum and the maximum number of players that could have been thrown out of the game.

Input

The first line contains one integer a_1 $(1 \le a_1 \le 1\,000)$ — the number of players in the first team.

The second line contains one integer a_2 $(1 \le a_2 \le 1\,000)$ — the number of players in the second team.

The third line contains one integer k_1 $(1 \le k_1 \le 1\,000)$ — the maximum number of yellow cards a player from the first team can receive (after receiving that many yellow cards, he leaves the game).

The fourth line contains one integer k_2 $(1 \le k_2 \le 1\,000)$ — the maximum number of yellow cards a player from the second team can receive (after receiving that many yellow cards, he leaves the game).

The fifth line contains one integer n $(1 \le n \le a_1 \cdot k_1 + a_2 \cdot k_2)$ — the number of yellow cards that have been shown during the match.

Output

Print two integers — the minimum and the maximum number of players that could have been thrown out of the game.

Examples

input

```
2
3
5
1
8
```

output

```
0 4
```

input

```
3
1
6
7
25
```

output

```
4 4
```

input

```
6
4
9
10
89
```

output

```
5 9
```

Note

In the first example it could be possible that no player left the game, so the first number in the output is 0. The maximum possible number of players that could have been forced to leave the game is 4 — one player from the first team, and three players from the second.

In the second example the maximum possible number of yellow cards has been shown $(3 \cdot 6 + 1 \cdot 7 = 25)$, so in any case all players were sent off.

C. Shooting

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

Recently Vasya decided to improve his pistol shooting skills. Today his coach offered him the following exercise. He placed n cans in a row on a table. Cans are numbered from left to right from 1 to n. Vasya has to knock down each can exactly once to finish the exercise. He is allowed to choose **the order** in which he will knock the cans down.

Vasya knows that the *durability* of the i-th can is a_i. It means that if Vasya has already knocked x cans down and is now about to start shooting the i-th one, he will need $(a_i \cdot x + 1)$ shots to knock it down. You can assume that if Vasya starts shooting the i-th can, he will be shooting it until he knocks it down.

Your task is to choose such an order of shooting so that the number of shots required to knock each of the n given cans down exactly once is minimum possible.

Input

The first line of the input contains one integer n $(2 \le n \le 1\,000)$ — the number of cans.

The second line of the input contains the sequence $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 1\,000)$, where a_i is the durability of the i-th can.

Output

In the first line print the minimum number of shots required to knock each of the n given cans down exactly once.

In the second line print the sequence consisting of n **distinct** integers from 1 to n — the order of indices of cans that minimizes the number of shots required. If there are several answers, you can print any of them.

Examples

input

```
3
20 10 20
```

output

```
43
1 3 2
```

input

```
4
10 10 10 10
```

output

```
64
2 1 4 3
```

input

```
6
5 4 5 4 4 5
```

output

```
69
6 1 3 5 2 4
```

input

```
2
1 4
```

output

```
3
2 1
```

Note

In the first example Vasya can start shooting from the first can. He knocks it down with the first shot because he haven't knocked any other cans down before. After that he has to shoot the third can. To knock it down he shoots $20 \cdot 1 + 1 = 21$ times. After that only second can remains. To knock it down Vasya shoots $10 \cdot 2 + 1 = 21$ times. So the total number of shots is $1 + 21 + 21 = 43$.

In the second example the order of shooting does not matter because all cans have the same durability.

D. Reachable Numbers

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

Let's denote a function $f(x)$ in such a way: we add 1 to x, then, while there is at least one trailing zero in the resulting number, we remove that zero. For example,

- $f(599) = 6$: $599 + 1 = 600 \rightarrow 60 \rightarrow 6$;
- $f(7) = 8$: $7 + 1 = 8$;
- $f(9) = 1$: $9 + 1 = 10 \rightarrow 1$;
- $f(10099) = 101$: $10099 + 1 = 10100 \rightarrow 1010 \rightarrow 101$.

We say that some number y is **reachable** from x if we can apply function f to x some (possibly zero) times so that we get y as a result. For example, 102 is reachable from 10098 because $f(f(f(10098))) = f(f(10099)) = f(101) = 102$; and any number is reachable from itself.

You are given a number n; your task is to count how many different numbers are reachable from n.

Input

The first line contains one integer n ($1 \le n \le 10^9$).

Output

Print one integer: the number of different numbers that are reachable from n.

Examples

input

```
1098
```

output

```
20
```

input

```
10
```

output

```
19
```

Note

The numbers that are reachable from 1098 are:

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1098, 1099.$$

E. Erasing Zeroes

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given a string s. Each character is either `0` or `1`.

You want all `1`'s in the string to form a contiguous subsegment. For example, if the string is `0`, `1`, `00111` or `01111100`, then all `1`'s form a contiguous subsegment, and if the string is `0101`, `100001` or `11111111111101`, then this condition is not met.

You may erase some (possibly none) `0`'s from the string. What is the minimum number of `0`'s that you have to erase?

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases.

Then t lines follow, each representing a test case. Each line contains one string s ($1 \le |s| \le 100$); each character of s is either `0` or `1`.

Output

Print t integers, where the i-th integer is the answer to the i-th testcase (the minimum number of `0`'s that you have to erase from s).

Example

input

```
3
010011
0
1111000
```

output

```
2
0
0
```

Note

In the first test case you have to delete the third and forth symbols from string `010011` (it turns into `0111`).

F. Square Filling

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given two matrices A and B. Each matrix contains exactly n rows and m columns. Each element of A is either 0 or 1; each element of B is initially 0.

You may perform some operations with matrix B. During each operation, you choose any submatrix of B having size 2×2, and replace every element in the chosen submatrix with 1. In other words, you choose two integers x and y such that $1 \le x < n$ and $1 \le y < m$, and then set $B_{x,y}$, $B_{x,y+1}$, $B_{x+1,y}$ and $B_{x+1,y+1}$ to 1.

Your goal is to make matrix B equal to matrix A. Two matrices A and B are equal if and only if every element of matrix A is equal to the corresponding element of matrix B.

Is it possible to make these matrices equal? If it is, you have to come up with a sequence of operations that makes B equal to A. Note that you don't have to minimize the number of operations.

Input

The first line contains two integers n and m ($2 \le n, m \le 50$).

Then n lines follow, each containing m integers. The j-th integer in the i-th line is $A_{i,j}$. Each integer is either 0 or 1.

Output

If it is impossible to make B equal to A, print one integer -1.

Otherwise, print any sequence of operations that transforms B into A in the following format: the first line should contain one integer k — the number of operations, and then k lines should follow, each line containing two integers x and y for the corresponding operation (set $B_{x,y}$, $B_{x,y+1}$, $B_{x+1,y}$ and $B_{x+1,y+1}$ to 1). The condition $0 \le k \le 2500$ should hold.

Examples

input

```
3 3
1 1 1
1 1 1
0 1 1
```

output

```
3
1 1
```

```
1 2
2 2
```

input

```
3 3
1 0 1
1 0 1
0 0 0
```

output

```
-1
```

input

```
3 2
0 0
0 0
0 0
```

output

```
0
```

Note

The sequence of operations in the first example:

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{matrix}$$

G. XOR Guessing

time limit per test: 2.0 s

memory limit per test: 512 MB

input: standard input

output: standard output

This is an interactive problem. Remember to flush your output while communicating with the testing program. You may use `fflush(stdout)` in C++, `system.out.flush()` in Java, `stdout.flush()` in Python or `flush(output)` in Pascal to flush the output. If you use some other programming language, consult its documentation. You may also refer to the guide on interactive problems: https://codeforces.com/blog/entry/45307.

The jury picked an integer x not less than 0 and not greater than $2^{14}-1$. You have to guess this integer.

To do so, you may ask no more than 2 queries. Each query should consist of 100 integer numbers a_1, a_2, ..., a_{100} (each integer should be not less than 0 and not greater than $2^{14}-1$). In response to your query, the jury will pick one integer i ($1 \le i \le 100$) and tell you the value of $a_i \oplus x$ (the bitwise XOR of a_i and x). There is an additional constraint on the queries: all 200 integers you use in the queries should be distinct.

It is guaranteed that the value of x is fixed beforehand in each test, but the choice of i in every query may depend on the integers you send.

Output

To give the answer, your program should print one line $!\ x$ *with a line break in the end*. After that, it should flush the output and terminate gracefully.

Interaction

Before giving the answer, you may submit no more than 2 queries. To ask a query, print one line in the following format: $?\ a_1\ a_2$ ... a_{100}, where every a_j should be an integer from the range $[0, 2^{14}-1]$. *The line should be ended with a line break character*. After submitting a query, flush the output and read the answer to your query — the value of $a_i \oplus x$ for some $i \in [1, 100]$. **No integer can be used in queries more than once.**

If you submit an incorrect query (or ask more than 2 queries), the answer to it will be one integer -1. After receiving such an answer, your program should terminate immediately — otherwise you may receive verdict "Runtime error", "Time limit exceeded" or some other verdict instead of "Wrong answer".

Example

input

```
0
32
```

output

```
? 3 5 6
? 32 24 37
! 5
```

Note

The example of interaction **is not correct** — you should sumbit **exactly** 100 integers in each query. Everything else is correct.

Hacks are forbidden in this problem.

H. Chainword

time limit per test: 6.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

A chainword is a special type of crossword. As most of the crosswords do, it has cells that you put the letters in and some sort of hints to what these letters should be.

The letter cells in a chainword are put in a single row. We will consider chainwords of length m in this task.

A hint to a chainword is a sequence of segments such that the segments don't intersect with each other and cover all m letter cells. Each segment contains a description of the word in the corresponding cells.

The twist is that there are actually two hints: one sequence is the row above the letter cells and the other sequence is the row below the letter cells. When the sequences are different, they provide a way to resolve the ambiguity in the answers.

You are provided with a dictionary of n words, each word consists of lowercase Latin letters. All words are pairwise distinct.

An instance of a chainword is the following triple:

- a string of m lowercase Latin letters;
- the first hint: a sequence of segments such that the letters that correspond to each segment spell a word from the dictionary;
- the second hint: another sequence of segments such that the letters that correspond to each segment spell a word from the dictionary.

Note that the sequences of segments don't necessarily have to be distinct.

Two instances of chainwords are considered different if they have different strings, different first hints **or** different second hints.

Count the number of different instances of chainwords. Since the number might be pretty large, output it modulo $998\,244\,353$.

Input

The first line contains two integers n and m ($1 \le n \le 8$, $1 \le m \le 10^9$) — the number of words in the dictionary and the number of letter cells.

Each of the next n lines contains a word — a non-empty string of no more than 5 lowercase Latin letters. All words are pairwise distinct.

Output

Print a single integer — the number of different instances of chainwords of length m for the given dictionary modulo $998\,244\,353$.

Examples

input

```
3 5
ababa
ab
a
```

output

```
11
```

input

```
2 4
ab
cd
```

output

```
4
```

input

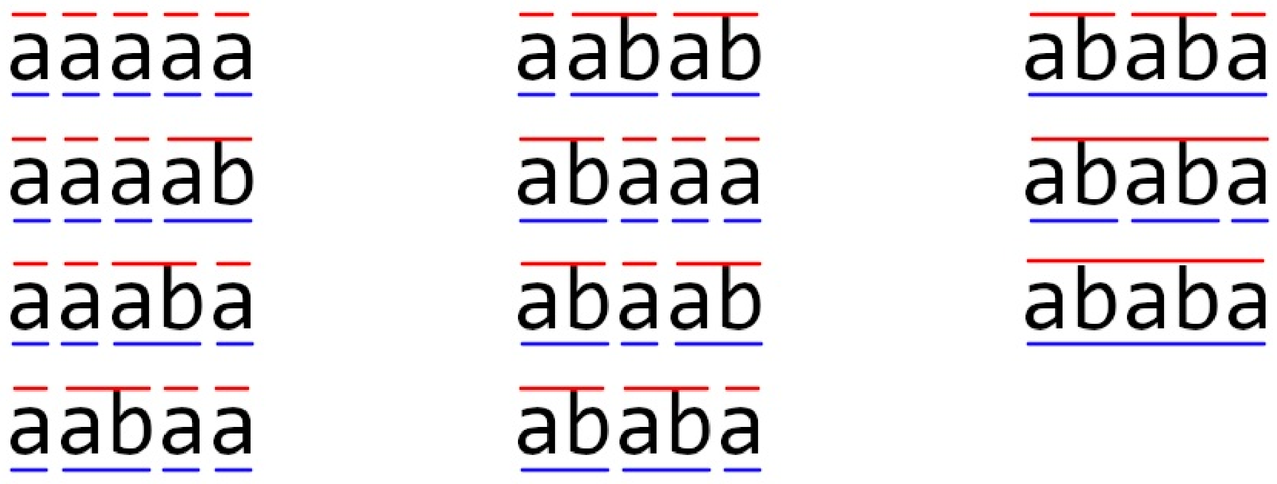
```
5 100
a
aa
aaa
aaaa
aaaaa
```

output

```
142528942
```

Note

Here are all the instances of the valid chainwords for the first example:



The red lines above the letters denote the segments of the first hint, the blue lines below the letters denote the segments of the second hint.

In the second example the possible strings are: "abab", "abcd", "cdab" and "cdcd". All the hints are segments that cover the first two letters and the last two letters.

I. Minimum Difference

time limit per test: 10.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given an integer array a of size n.

You have to perform m queries. Each query has one of two types:

- "$1\ l\ r\ k$" — calculate the minimum value dif such that there are exist k **distinct** integers $x_1, x_2, \dots, x_k$ such that $cnt_i > 0$ (for every $i \in [1, k]$) and $|cnt_i - cnt_j| \le dif$ (for every $i \in [1, k], j \in [1, k]$), where cnt_i is the number of occurrences of x_i in the subarray $a[l..r]$. If it is impossible to choose k integers, report it;
- "$2\ p\ x$" — assign $a_p := x$.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$) — the size of the array a and the number of queries.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^5$).

Next m lines contain queries (one per line). Each query has one of two types:

- "$1\ l\ r\ k$" ($1 \le l \le r \le n; 1 \le k \le 10^5$)
- "$2\ p\ x$" ($1 \le p \le n; 1 \le x \le 10^5$).

It's guaranteed that there is at least one query of the first type.

Output

For each query of the first type, print the minimum value of dif that satisfies all required conditions, or -1 if it is impossible to choose k distinct integers.

Example

input

```
12 11
2 1 1 2 1 1 3 2 1 1 3 3
1 2 10 3
1 2 11 3
2 7 2
1 3 9 2
1 1 12 1
1 1 12 4
2 12 4
1 1 12 4
2 1 5
1 3 12 2
1 1 4 3
```

output

```
5
4
1
0
-1
5
```

```
0
1
```