Educational Codeforces Round 71 (Rated for Div. 2)

A. There Are Two Types Of Burgers

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

There are two types of burgers in your restaurant — hamburgers and chicken burgers! To assemble a hamburger you need two buns and a beef patty. To assemble a chicken burger you need two buns and a chicken cutlet.

You have b buns, p beef patties and f chicken cutlets in your restaurant. You can sell one hamburger for h dollars and one chicken burger for c dollars. Calculate the maximum profit you can achieve.

You have to answer t independent queries.

Input

The first line contains one integer t ($1 \le t \le 100$) – the number of queries.

The first line of each query contains three integers b, p and f ($1 \le b,\ p,\ f \le 100$) — the number of buns, beef patties and chicken cutlets in your restaurant.

The second line of each query contains two integers h and c ($1 \le h,\ c \le 100$) — the hamburger and chicken burger prices in your restaurant.

Output

For each query print one integer — the maximum profit you can achieve.

Example

input
```
3
15 2 3
5 10
7 5 2
10 12
1 100 100
100 100
```
output
```
40
34
0
```

Note

In first query you have to sell two hamburgers and three chicken burgers. Your income is $2 \cdot 5 + 3 \cdot 10 = 40$.

In second query you have to ell one hamburgers and two chicken burgers. Your income is $1 \cdot 10 + 2 \cdot 12 = 34$.

In third query you can not create any type of burgers because because you have only one bun. So your income is zero.

B. Square Filling

time limit per test: 1 second
memory limit per test: 256 megabytes
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:

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

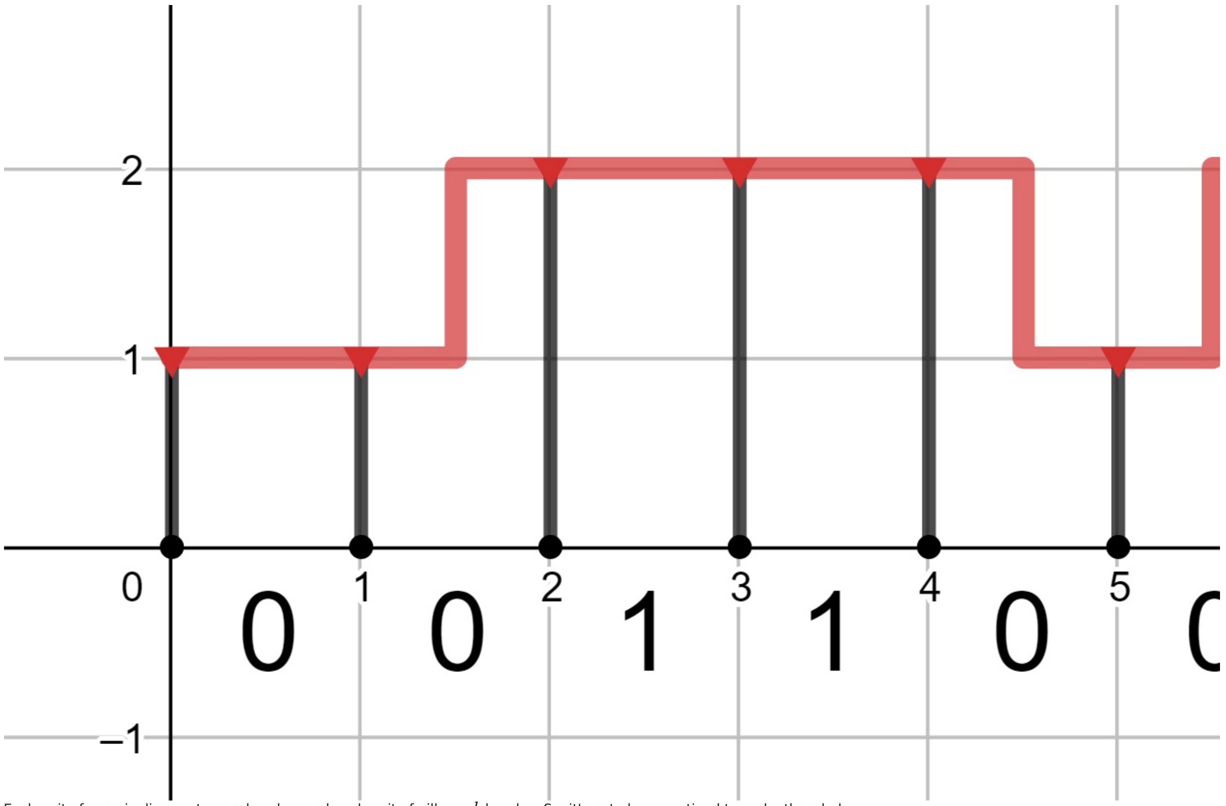
C. Gas Pipeline

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

You are responsible for installing a gas pipeline along a road. Let's consider the road (for simplicity) as a segment $[0, n]$ on OX axis. The road can have several crossroads, but for simplicity, we'll denote each crossroad as an interval $(x, x+1)$ with integer x. So we can represent the road as a binary string consisting of n characters, where character `0` means that current interval doesn't contain a crossroad, and `1` means that there is a crossroad.

Usually, we can install the pipeline along the road on height of 1 unit with supporting pillars in each integer point (so, if we are responsible for $[0, n]$ road, we must install $n+1$ pillars). But on crossroads we should lift the pipeline up to the height 2, so the pipeline won't obstruct the way for cars.

We can do so inserting several zig-zag-like lines. Each zig-zag can be represented as a segment $[x, x+1]$ with integer x consisting of three parts: 0.5 units of horizontal pipe + 1 unit of vertical pipe + 0.5 of horizontal. Note that if pipeline is currently on height 2, the pillars that support it should also have length equal to 2 units.



Each unit of gas pipeline costs us a bourles, and each unit of pillar — b bourles. So, it's not always optimal to make the whole pipeline on the height 2. Find the shape of the pipeline with minimum possible cost and calculate that cost.

Note that you **must** start and finish the pipeline on height 1 and, also, it's guaranteed that the first and last characters of the input string are equal to `0`.

Input

The fist line contains one integer T ($1 \le T \le 100$) — the number of queries. Next $2 \cdot T$ lines contain independent queries — one query per two lines.

The first line contains three integers n, a, b ($2 \le n \le 2 \cdot 10^5$, $1 \le a \le 10^8$, $1 \le b \le 10^8$) — the length of the road, the cost of one unit of the pipeline and the cost of one unit of the pillar, respectively.

The second line contains binary string s ($|s| = n$, $s_i \in \{0, 1\}$, $s_1 = s_n = 0$) — the description of the road.

It's guaranteed that the total length of all strings s doesn't exceed $2 \cdot 10^5$.

Output

Print T integers — one per query. For each query print the minimum possible cost of the constructed pipeline.

Example

input

```
4
8 2 5
00110010
8 1 1
00110010
9 100000000 100000000
010101010
2 5 1
00
```

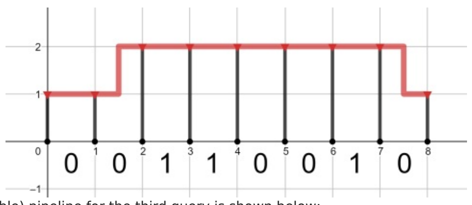
output

```
94
25
2900000000
13
```

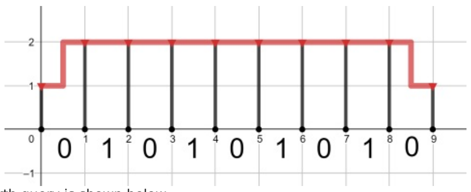
Note

The optimal pipeline for the first query is shown at the picture above.

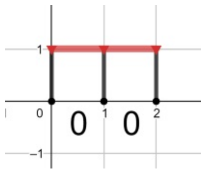
The optimal pipeline for the second query is pictured below:



The optimal (and the only possible) pipeline for the third query is shown below:



The optimal pipeline for the fourth query is shown below:



D. Number Of Permutations

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

You are given a sequence of n pairs of integers: $(a_1, b_1), (a_2, b_2), \dots, (a_n, b_n)$. This sequence is called bad if it is sorted in non-descending order by first elements or if it is sorted in non-descending order by second elements. Otherwise the sequence is good. There are examples of good and bad sequences:

- $s = [(1, 2), (3, 2), (3, 1)]$ is bad because the sequence of first elements is sorted: $[1, 3, 3]$;
- $s = [(1, 2), (3, 2), (1, 2)]$ is bad because the sequence of second elements is sorted: $[2, 2, 2]$;
- $s = [(1, 1), (2, 2), (3, 3)]$ is bad because both sequences (the sequence of first elements and the sequence of second elements) are sorted;
- $s = [(1, 3), (3, 3), (2, 2)]$ is good because neither the sequence of first elements ($[1, 3, 2]$) nor the sequence of second elements ($[3, 3, 2]$) is sorted.

Calculate the number of permutations of size n such that after applying this permutation to the sequence s it turns into a good sequence.

A permutation p of size n is a sequence $p_1, p_2, \dots, p_n$ consisting of n distinct integers from 1 to n ($1 \le p_i \le n$). If you apply permutation $p_1, p_2, \dots, p_n$ to the sequence $s_1, s_2, \dots, s_n$ you get the sequence $s_{p_1}, s_{p_2}, \dots, s_{p_n}$. For example, if $s = [(1, 2), (1, 3), (2, 3)]$ and $p = [2, 3, 1]$ then s turns into $[(1, 3), (2, 3), (1, 2)]$.

Input

The first line contains one integer n ($1 \le n \le 3 \cdot 10^5$).

The next n lines contains description of sequence s. The i-th line contains two integers a_i and b_i ($1 \le a_i, b_i \le n$) — the first and second elements of i-th pair in the sequence.

The sequence s may contain equal elements.

Output

Print the number of permutations of size n such that after applying this permutation to the sequence s it turns into a good sequence. Print the answer modulo 998244353 (a prime number).

Examples

input
```
3
1 1
2 2
3 1
```
output
```
3
```

input
```
4
2 3
2 2
2 1
2 4
```
output
```
0
```

input
```
3
1 1
1 1
2 3
```
output
```
4
```

Note

In first test case there are six permutations of size 3:

1. if $p = [1, 2, 3]$, then $s = [(1, 1), (2, 2), (3, 1)]$ — bad sequence (sorted by first elements);
2. if $p = [1, 3, 2]$, then $s = [(1, 1), (3, 1), (2, 2)]$ — bad sequence (sorted by second elements);
3. if $p = [2, 1, 3]$, then $s = [(2, 2), (1, 1), (3, 1)]$ — good sequence;
4. if $p = [2, 3, 1]$, then $s = [(2, 2), (3, 1), (1, 1)]$ — good sequence;
5. if $p = [3, 1, 2]$, then $s = [(3, 1), (1, 1), (2, 2)]$ — bad sequence (sorted by second elements);
6. if $p = [3, 2, 1]$, then $s = [(3, 1), (2, 2), (1, 1)]$ — good sequence.

E. XOR Guessing

time limit per test: 1 second
memory limit per test: 256 megabytes
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: $?\ a_1\ a_2\ \dots\ a_{100}$, 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.

F. Remainder Problem

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array a consisting of 500000 integers (numbered from 1 to 500000). Initially all elements of a are zero.

You have to process two types of queries to this array:

- $1\ x\ y$ — increase a_x by y;
- $2\ x\ y$ — compute $\sum\limits_{i \in R(x,y)} a_i$, where $R(x, y)$ is the set of all integers from 1 to 500000 which have remainder y modulo x.

Can you process all the queries?

Input

The first line contains one integer q ($1 \le q \le 500000$) — the number of queries.

Then q lines follow, each describing a query. The i-th line contains three integers t_i, x_i and y_i ($1 \le t_i \le 2$). If $t_i = 1$, then it is a query of the first type, $1 \le x_i \le 500000$, and $-1000 \le y_i \le 1000$. If $t_i = 2$, then it it a query of the second type, $1 \le x_i \le 500000$, and $0 \le y_i < x_i$.

It is guaranteed that there will be at least one query of type 2.

Output

For each query of type 2 print one integer — the answer to it.

Example

input

```
5
1 3 4
2 3 0
2 4 3
1 4 -4
2 1 0
```

output

```
4
4
0
```

G. Indie Album

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Mishka's favourite experimental indie band has recently dropped a new album! Songs of that album share one gimmick. Each name s_i is one of the following types:

- $1\ c$ — a single lowercase Latin letter;
- $2\ j\ c$ — name s_j ($1 \le j < i$) with a single lowercase Latin letter appended to its end.

Songs are numbered from 1 to n. It's guaranteed that the first song is always of type 1.

Vova is rather interested in the new album but he really doesn't have the time to listen to it entirely. Thus he asks Mishka some questions about it to determine if some song is worth listening to. Questions have the following format:

- $i\ t$ — count the number of occurrences of string t in s_i (the name of the i-th song of the album) as a continuous substring, t consists only of lowercase Latin letters.

Mishka doesn't question the purpose of that information, yet he struggles to provide it. Can you please help Mishka answer all Vova's questions?

Input

The first line contains a single integer n ($1 \le n \le 4 \cdot 10^5$) — the number of songs in the album.

Each of the next n lines contains the desciption of the i-th song of the album in the following format:

- $1\ c$ — s_i is a single lowercase Latin letter;
- $2\ j\ c$ — s_i is the name s_j ($1 \le j < i$) with a single lowercase Latin letter appended to its end.

The next line contains a single integer m ($1 \le m \le 4 \cdot 10^5$) — the number of Vova's questions.

Each of the next m lines contains the desciption of the j-th Vova's question in the following format:

- $i\ t$ ($1 \le i \le n$, $1 \le |t| \le 4 \cdot 10^5$) — count the number of occurrences of string t in s_i (the name of the i-th song of the album) as a continuous substring, t consists only of lowercase Latin letters.

It's guaranteed that the total length of question strings t doesn't exceed $4 \cdot 10^5$.

Output

For each question print a single integer — the number of occurrences of the question string t in the name of the i-th song of the album as a continuous substring.

Example

input

```
20
1 d
2 1 a
2 2 d
2 3 a
2 4 d
2 5 a
2 6 d
```

```
2 7 a
1 d
2 9 o
2 10 k
2 11 i
2 12 d
2 13 o
2 14 k
2 15 i
2 1 o
2 17 k
2 18 i
2 15 i
12
8 da
8 dada
8 ada
6 dada
3 dada
19 doki
19 ok
16 doki
15 doki
9 d
1 a
20 doki
```

output

```
4
3
3
2
0
1
1
2
1
1
0
2
```

Note

Song names of the first example:

1. d
2. da
3. dad
4. dada
5. dadad
6. dadada
7. dadadad
8. dadadada
9. d
10. do
11. dok
12. doki
13. dokid
14. dokido
15. dokidok
16. dokidoki
17. do
18. dok
19. doki
20. dokidoki

Thus the occurrences for each question string are:

1. string "da" starts in positions $[1, 3, 5, 7]$ in the name "dadadada";
2. string "dada" starts in positions $[1, 3, 5]$ in the name "dadadada";
3. string "ada" starts in positions $[2, 4, 6]$ in the name "dadadada";
4. string "dada" starts in positions $[1, 3]$ in the name "dadada";
5. no occurrences of string "dada" in the name "dad";
6. string "doki" starts in position $[1]$ in the name "doki";
7. string "ok" starts in position $[2]$ in the name "doki";
8. string "doki" starts in positions $[1, 5]$ in the name "dokidoki";
9. string "doki" starts in position $[1]$ in the name "dokidok";
10. string "d" starts in position $[1]$ in the name "d";
11. no occurrences of string "a" in the name "d";
12. string "doki" starts in positions $[1, 5]$ in the name "dokidoki".