

UCPC 2026 - I stage

Ukrainian Collegiate Programming Contest

October 3, 2026

Problems

A Antivirus
Time 1.5 s · Memory 256 MiB

B Collectible Pairs
Time 0.5 s · Memory 256 MiB

C Dance of the Remainders
Time 1 s · Memory 256 MiB

D Even and Odd
Time 1.5 s · Memory 256 MiB

E Four Numbers
Time 2 s · Memory 256 MiB

F Mixed-up Readings
Time 3 s · Memory 256 MiB

G Number Game
Time 1 s · Memory 256 MiB

H Pattern
Time 1 s · Memory 256 MiB

I Round Table
Time 2 s · Memory 512 MiB

J Scales
Time 0.5 s · Memory 256 MiB · Interactive

K Sum and Product
Time 1 s · Memory 256 MiB

L Three Computers
Time 3 s · Memory 256 MiB

M UCPC Garland
Time 4.5 s · Memory 256 MiB

N Weather Diary
Time 0.5 s · Memory 256 MiB

A Antivirus

Time limit 1.5 s · Memory limit 256 MiB · Standard input/output

Ostap's computer has n folders, numbered from 1 to n. Folder 1 is the root, and every other folder i ($2 \le i \le n$) lies directly inside folder a_i; it is guaranteed that $a_i < i$.

Ostap ran the antivirus k times. The j-th run checks folder c_j and all folders inside it that lie no deeper than d_j levels from c_j.

Formally, folder v is checked by the j-th run if v can be reached from c_j by moving from a folder to a folder that lies directly inside it, at most d_j times. In particular, folder c_j is always checked by the j-th run, even when $d_j = 0$.

The same folder could be checked multiple times. Find the number of folders checked by at least one run.

Input

The first line contains two integers n and k ($2 \le n \le 3 \cdot 10^5$, $1 \le k \le 3 \cdot 10^5$) — the number of folders and the number of antivirus runs.

The second line contains $n-1$ integers $a_2, a_3, \ldots, a_n$ ($1 \le a_i < i$), where a_i is the folder in which folder i was created.

Each of the next k lines contains two integers c_j and d_j ($1 \le c_j \le n$, $0 \le d_j \le 3 \cdot 10^5$) — the folder and the depth of the j-th run.

Output

Print a single integer — the number of folders checked by at least one run.

Examples

Input

```
8 3
1 1 2 2 3 5 5
2 1
3 0
8 2
```

Output

```
5
```

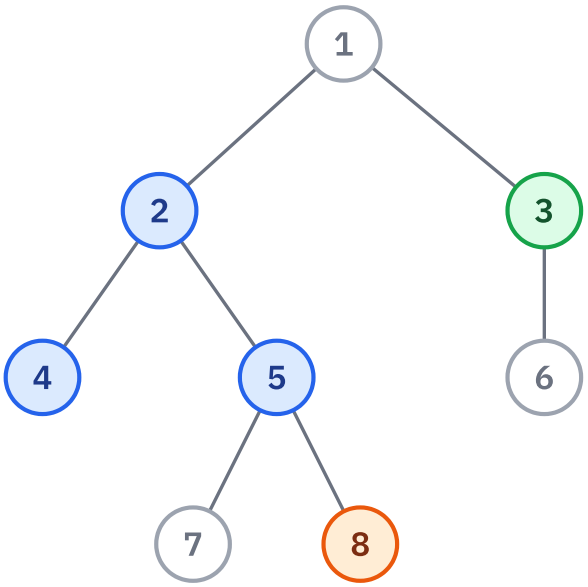
Input

```
6 4
1 2 2 4 1
2 2
2 0
1 0
4 0
```

Output

```
5
```

Note



In the first example, folder 1 contains folders 2 and 3, folder 2 contains folders 4 and 5, folder 3 contains folder 6, and folder 5 contains folders 7 and 8. The first run checks folders 2, 4, and 5 (blue color); the second checks only folder 3 (green color); the third checks only folder 8 (red color), because it has no other folders inside it. In total, 5 folders are checked.

In the second example, the first run checks folders 2, 3, 4, and 5, the second checks folder 2, the third checks folder 1, and the fourth checks folder 4. Folder 6 is not checked, so the answer is 5.

B Collectible Pairs

Time limit 0.5 s · Memory limit 256 MiB · Standard input/output

Marta collects banknotes and evaluates them by their serial numbers. A serial number is a positive integer, and Marta calls it *beautiful* if it is divisible by the sum of its digits in decimal notation. For example, 110 is beautiful, because the sum of its digits equals $1+1+0=2$ and $110=2\cdot 55$; the number 13 is not beautiful, because 13 is not divisible by $1+3=4$.

Marta values *beautiful pairs* the most: two banknotes whose serial numbers are two consecutive beautiful integers x and $x+1$. For example, 110 and 111 form a beautiful pair, since 111 is divisible by $1+1+1=3$.

Marta is going to visit q collectors. The i-th collector has a bundle containing exactly one banknote for each serial number from m_i to $2m_i$ inclusive. For each collector, she wants to choose a beautiful pair in their bundle, that is, two consecutive **beautiful** integers x and $x+1$, for which

$$m_i \le x \quad \text{and} \quad x+1 \le 2m_i,$$

or determine that no such pair exists.

Input

The first line contains a single integer q ($1 \le q \le 10^5$) — the number of collectors.

Each of the following q lines contains a single integer m_i ($m_i \ge 1$), written in decimal notation without leading zeros. The total number of digits across all m_i does not exceed 10^6 ($\sum |m_i| \le 10^6$).

Output

Output q lines, the i-th of which is the answer for the i-th collector.

If there is a beautiful pair $(x, x+1)$ in the i-th collector's bundle, output the number x in decimal notation without leading zeros. If there are multiple possible answers, output any of them.

If there is no beautiful pair in the bundle, output a single number -1.

Examples

Input	Output
5 1 8 30 2718 987654321987654321	1 8 -1 2754 987654321987655230

Note

The first collector has the banknotes 1 and 2 in their bundle; both numbers are beautiful, so they form the only possible beautiful pair.

The second collector's serial numbers are $8, 9, \ldots, 16$. Among them, the beautiful ones are 8, 9, 10 and 12, so there are exactly two possible beautiful pairs: the pairs $(8, 9)$ and $(9, 10)$.

The third collector's serial numbers are $30, 31, \ldots, 60$; no two consecutive numbers among them are both beautiful.

For the fourth collector, one of the possible answers is $2754 = 18 \cdot 153$ and $2755 = 19 \cdot 145$, where 18 and 19 are their digit sums.

For the fifth collector, the digit sums of 987654321987655230 and 987654321987655231 equal 90 and 91, and it holds that $987654321987655230 = 90 \cdot 10973936910973947$ and $987654321987655231 = 91 \cdot 10853344197666541$.

C Dance of the Remainders

Time limit 1 s · Memory limit 256 MiB · Standard input/output

Three gnomes — Moss, Leaf, and Root — dance in a circle around an oak tree: Leaf follows Moss, Root follows Leaf, and Moss follows Root.

Before the dance, each gnome gathers some acorns for himself: Moss takes a acorns, where $1 \le a \le n$; Leaf takes b acorns, where $1 \le b \le m$; Root takes c acorns, where $1 \le c \le k$.

Each gnome arranges his acorns into as many piles as possible, each pile containing as many acorns as the next gnome has. The acorns that do not fit into piles are the gnome's *remainder*. Thus, the remainders of Moss, Leaf, and Root are equal to $a \bmod b$, $b \bmod c$, and $c \bmod a$, where $x \bmod y$ is the remainder of dividing x by y (for example, $7 \bmod 3 = 1$, $4 \bmod 11 = 4$).

The dance is called *harmonious* if the remainders of all three gnomes are equal:

$$a \bmod b = b \bmod c = c \bmod a.$$

For each of t test cases, find the number of triples (a, b, c) for which the dance is harmonious.

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases.

Each of the next t lines contains three integers n, m, and k ($1 \le n, m, k \le 10^9$).

Output

For each test case, output on a separate line a single integer — the number of triples (a, b, c) for which the dance is harmonious.

Examples

Input	Output
2 2 3 1 3 8 5	1 3

Note

In the first test case, Root can take only one acorn. The only suitable triple is $(1, 1, 1)$: all three remainders are equal to 0. For example, the triple $(2, 1, 1)$ is not suitable: the remainders of Moss and Leaf are 0, while the remainder of Root is $1 \bmod 2 = 1$.

D Even and Odd

Time limit 1.5 s · Memory limit 256 MiB · Standard input/output

Let A be a natural number, and let $a_{m-1}a_{m-2}...a_1a_0$ be its decimal representation without leading zeros, that is

$$A = a_0 + 10 \cdot a_1 + 10^2 \cdot a_2 + ... + 10^{m-1} \cdot a_{m-1}.$$

In other words, the digits are numbered **from right to left, starting at zero**: a_0 is the last digit (the units digit), a_1 is the tens digit, and so on. The number i is called the *position* of the digit a_i.

The number A is called *balanced* if

$$S_{\text{odd}}(A) - P_{\text{odd}}(A) = S_{\text{even}}(A) - P_{\text{even}}(A).$$

Here:

- $S_{\text{odd}}(A)$ is the sum of all digits a_i that are odd numbers;
- $S_{\text{even}}(A)$ is the sum of all digits a_i that are even numbers;
- $P_{\text{odd}}(A)$ is the sum of all digits a_i standing at odd positions i;
- $P_{\text{even}}(A)$ is the sum of all digits a_i standing at even positions i.

For example, for $A = 1234$ we have $a_0 = 4$, $a_1 = 3$, $a_2 = 2$, $a_3 = 1$. Then:

- $S_{\text{odd}} = 1 + 3 = 4$;
- $S_{\text{even}} = 2 + 4 = 6$;
- $P_{\text{odd}} = 3 + 1 = 4$;
- $P_{\text{even}} = 4 + 2 = 6$.

Since $S_{\text{odd}} - P_{\text{odd}} = 4 - 4 = 0$ and $S_{\text{even}} - P_{\text{even}} = 6 - 6 = 0$ hold, the number 1234 is balanced.

However, for $A = 12340$ the positions of all the original digits are increased by one: $a_0 = 0$, $a_1 = 4$, $a_2 = 3$, $a_3 = 2$, $a_4 = 1$. Now:

- $S_{\text{odd}} = 4$;
- $S_{\text{even}} = 6$;
- $P_{\text{odd}} = 4 + 2 = 6$;
- $P_{\text{even}} = 0 + 3 + 1 = 4$.

Since $4 - 6 \neq 6 - 4$, the number 12340 is not balanced.

You are given Q queries. In each query, a natural number N is given. Find the **smallest** balanced number M whose decimal representation *begins with* the decimal representation of the number N. In other words, M is obtained by appending $k \geq 0$ more digits to the right of N (any digits, including zeros); the case $k = 0$, that is $M = N$, is allowed. It can be proven that such a number M always exists.

Input

The first line contains a single integer Q ($1 \leq Q \leq 10^6$) — the number of queries.

Each of the following Q lines contains a single natural number N, written without leading zeros. It is guaranteed that $1 \leq |N| \leq 10^6$ and the sum of $|N|$ over all queries does not exceed 10^6, where $|N|$ is the number of digits of the number N.

Output

For each query, output the required number M on a separate line in full, without leading zeros.

Examples

Input
```
5
1234
9
1
99
98
```

Output
```
1234
90
10
992081
98
```

Input
```
3
88
2025
999999999
```

Output
```
88107
20250
99999999904080808080
```

Note

In the first example:

- 1234 is itself balanced (see above), so $M = N$.
- 9 is not balanced: $S_{\text{odd}} - P_{\text{odd}} = 9 - 0 = 9$, while $S_{\text{even}} - P_{\text{even}} = 0 - 9 = -9$. The next candidates are $90, 91, \ldots, 99$; among them 90 is balanced.
- For 99 the answer is 992081: here $a_0 = 1$, $a_1 = 8$, $a_2 = 0$, $a_3 = 2$, $a_4 = 9$, $a_5 = 9$, so $S_{\text{odd}} - P_{\text{odd}} = (1 + 9 + 9) - (8 + 2 + 9) = 0$ and $S_{\text{even}} - P_{\text{even}} = (8 + 0 + 2) - (1 + 0 + 9) = 0$. No smaller number fits.

E Four Numbers

Time limit 2 s · Memory limit 256 MiB · Standard input/output

Olenka has a sequence of n integers $a_1, a_2, \ldots, a_n$ and an integer m.

She wants to choose four indices i, j, k, t ($1 \le i, j, k, t \le n$) such that $a_i + a_j + a_k + a_t \le m$. Note that the indices are **not necessarily** distinct (some of them may coincide).

Among all such quadruples, Olenka is interested in the largest possible value of $a_i + a_j + a_k + a_t$. Help her find it, or report that no such quadruple of indices exists.

Input

The first line contains two integers n and m ($1 \le n \le 3000$, $-4 \cdot 10^{18} \le m \le 4 \cdot 10^{18}$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^{18} \le a_i \le 10^{18}$).

Output

Print a single integer — the maximum value of $a_i + a_j + a_k + a_t$ among all quadruples of indices for which $a_i + a_j + a_k + a_t \le m$.

If no such quadruple exists, print -1.

Examples

Input	Output
4 20 3 9 1 7	20
3 -10 5 -2 8	-1
5 2 -5 4 10 -1 6	2

Note

In the first example, one can take $i = j = 2$ and $k = t = 3$: $a_2 + a_2 + a_3 + a_3 = 9 + 9 + 1 + 1 = 20 \le 20$.

In the second example, even the smallest possible sum $a_2 + a_2 + a_2 + a_2 = -8$ is greater than $m = -10$, so no answer exists.

In the third example, one of the optimal choices is $i = 1$, $j = k = 2$, $t = 4$: $(-5) + 4 + 4 + (-1) = 2$.

F Mixed-up Readings

Time limit 3 s · Memory limit 256 MiB · Standard input/output

The old alchemist Orest keeps a collection of n magic crystals; the i-th crystal has a positive integer *charge* a_i. Charges of different crystals may coincide.

Orest has two devices. If you place a non-empty group of crystals on the *scale*, it will show the total charge of the group, that is, the **sum** of the charges. If you place the same group on the *resonator*, it will show the **bitwise OR** of their charges.

One night Orest measured each of the $2^n - 1$ non-empty groups of crystals on both devices and recorded all the readings in a log. Thus, the log contains exactly $2 \cdot (2^n - 1)$ numbers. Unfortunately, the records were on separate sheets of paper that got mixed up: now it is unknown which group each reading belongs to and which device produced it. The crystals themselves have already been sold.

Given the $2 \cdot (2^n - 1)$ readings in an arbitrary order, recover the charges of the crystals.

Formally, for an array $a_1, a_2, \ldots, a_n$ and a non-empty set of indices $S \subseteq \{1, \ldots, n\}$ denote $\mathrm{sum}(S) = \sum_{i \in S} a_i$ and $\mathrm{or}(S) = a_{i_1} \mid a_{i_2} \mid \ldots \mid a_{i_k}$, where $S = \{i_1, \ldots, i_k\}$, and $\mid$ is the bitwise OR. The log is a multiset consisting of the numbers $\mathrm{sum}(S)$ and $\mathrm{or}(S)$ for all $2^n - 1$ non-empty sets S (if $\mathrm{sum}(S) = \mathrm{or}(S)$, then for this S that value appears twice).

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases.

Each test case consists of two lines. The first line contains an integer n ($1 \le n \le 20$) — the number of crystals. The second line contains $2 \cdot (2^n - 1)$ integers $b_1, b_2, \ldots, b_{2(2^n-1)}$ ($1 \le b_j \le 2 \cdot 10^{10}$) — the readings from the log in an arbitrary order.

It is guaranteed that the readings of each test case are exactly the log of some array of n integers $a_1, \ldots, a_n$, where $1 \le a_i \le 10^9$.

The sum of 2^n over all test cases does not exceed 2^{20}.

Output

For each test case, output on a separate line n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) in arbitrary order such that the log of this array matches the given readings as a multiset. If there are multiple answers, output any of them.

Examples

Input

```
4
1
7 7
2
3 1 2 3 2 1
2
2 4 2 2 2 2
3
7 5 3 14 6 7 11 5 8 7 3 9 6 7
```

Output

```
7
1 2
2 2
3 5 6
```

Note

In the first test case, the only crystal has charge 7: both devices show 7.

In the second test case, the charges are 1 and 2: the scale shows 1, 2, 3, and the resonator shows 1, 2, 3.

In the third test case, both crystals have charge 2: the scale shows 2, 2, 4, and the resonator shows 2, 2, 2.

In the fourth test case, the charges are 3, 5, 6. The scale shows 3, 5, 6, 8, 9, 11, 14, and the resonator shows 3, 5, 6, 7, 7, 7, 7.

G Number Game

Time limit 1 s · Memory limit 256 MiB · Standard input/output

Marta is playing a game with a number written on a board in binary without leading zeros. Initially, a positive number n is written on the board.

In one *move*, Marta performs two actions in the following order:

1. She erases any number of zeros (possibly none) from the representation and appends any number of zeros (possibly none) at any positions in the representation, so that it does not start with a zero. It is **not allowed** to erase or append ones.
2. She increases the written number by 1 and writes the result in its place, again in binary without leading zeros.

Marta wins if, after one of her moves, the positive number m is written on the board. If $n = m$, she wins immediately, without making a single move. Find the minimum number of moves in which Marta can win, or report that it is impossible to win.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

Each of the next t lines contains two positive numbers n and m ($1 \le n, m$), written in binary without leading zeros and separated by a space.

It is guaranteed that $1 \le |n|, |m| \le 1000$, where $|x|$ is the number of digits in the binary representation of x.

Output

For each test case, output on a separate line a single integer — the minimum number of moves, or -1 if it is impossible to win.

Examples

Input	Output
6 110 1110 1 10000 111 1 101 101 1000 11 11111 1001	2 4 -1 0 1 2

Note

In the first test case, $n = 6 =$ `110`, $m = 14 =$ `1110`. Marta can win in two moves:

- Move 1: Marta appends a zero at the end, `110` $\to$ `1100` $= 12$, and increases the number: $13 =$ `1101`.
- Move 2: Marta erases and appends nothing, and increases the number: $14 =$ `1110`.

In the third test case, it is impossible to obtain $m = 1$.

In the fourth test case, $n = m$, so Marta wins immediately.

H Pattern

Time limit 1 s · Memory limit 256 MiB · Standard input/output

Olenka is making a stamp for a pattern on a ribbon. A stamp is a strip of n cells, each of which is painted in one of k colors. Let us denote the colors of the cells from left to right as $w_1, w_2, \ldots, w_n$ $(1 \le w_i \le k)$.

So that the imprints on the ribbon cannot be confused, Olenka needs a *successful* stamp. A stamp is called *successful* if no prefix of it is colored the same as the suffix of the same length. That is, for every l from 1 to $n-1$ the first l cells differ from the last l:

$$w_1 w_2 \ldots w_l \neq w_{n-l+1} w_{n-l+2} \ldots w_n.$$

A stamp consisting of a single cell is always successful.

For example, for $k = 3$ the stamps `123`, `112` and `3` are successful, while the stamps `121` $(l = 1)$, `12312` $(l = 2)$ and `1111` $(l = 1, 2, 3)$ are not.

You are given t queries. For each query, count the number of successful stamps consisting of n cells painted in k colors. Since the answers can be very large, output them modulo $10^9 + 7$.

Input

The first line contains a single integer t $(1 \le t \le 10^5)$ — the number of queries.

Each of the following t lines contains two integers n and k $(1 \le n \le 10^6,\ 1 \le k \le 10^9)$ — the number of cells of the stamp and the number of colors.

It is guaranteed that the sum of n over all queries does not exceed 10^6.

Output

For each query, print on a separate line the number of successful stamps consisting of n cells painted in k colors, modulo $10^9 + 7$.

Examples

Input	Output
4 1 5 2 3 3 2 4 2	5 6 4 6
2 5 3 30 1000000000	144 508301895

Note

In the first example:

- For $n = 1$ all 5 stamps are successful.
- For $n = 2$, $k = 3$ a stamp is successful if and only if its two cells have different colors: $3 \cdot 2 = 6$ stamps.

- For $n = 3$, $k = 2$ the successful stamps are `112`, `122`, `211` and `221`.
- For $n = 4$, $k = 2$ the successful stamps are `1112`, `1122`, `1222`, `2111`, `2211` and `2221`. For example, the stamp `1212` is not successful because its first two cells `12` are colored the same as its last two, and the stamp `1221` is not successful because it starts and ends with color 1.

I Round Table

Time limit 2 s · Memory limit 512 MiB · Standard input/output

n students of a programming club are sitting at a round table. They are numbered from 1 to n in clockwise order, so that student n is sitting next to student 1.

The coach wants to form as many teams as possible from them. In order for team members to be able to discuss problems without shouting at each other across the whole table, he made a list of m possible teams, each consisting of students sitting consecutively. Teams can have any size.

The i-th possible team is given by the numbers l_i and r_i: it includes the students numbered from l_i to r_i inclusive, when moving clockwise. That is:

- if $l_i \le r_i$, the team consists of students $l_i, l_i + 1, \ldots, r_i$;
- if $l_i > r_i$, the team consists of students $l_i, l_i + 1, \ldots, n, 1, 2, \ldots, r_i$.

In particular, when $l_i = r_i$, the team consists of a single student, and when $l_i = r_i + 1$ or $l_i = 1$ and $r_i = n$, it consists of all n students. Some teams in the list may coincide.

Each student may belong to at most **one** formed team. Find the largest number of teams that can be formed.

Input

The first line contains a single integer t ($1 \le t \le 10^6$) — the number of test cases.

The first line of each test case contains two integers n and m ($1 \le n, m \le 10^6$) — the number of students and the number of possible teams.

Each of the next m lines contains two integers l_i and r_i ($1 \le l_i, r_i \le n$) — the i-th possible team.

It is guaranteed that the sum of n and the sum of m over all test cases do not exceed 10^6.

Output

For each test case, output on a separate line a single integer — the largest number of teams that can be formed simultaneously.

Examples

Input	Output
3 5 3 5 1 2 3 3 4 4 2 2 1 1 4 6 5 1 3 3 4 5 6 6 2 4 4	2 1 3

Input

```
1
10 6
9 2
3 5
6 8
1 4
5 7
4 4
```

Output

```
3
```

Note

In the first example:

- In the first test case, the possible teams are $\{5, 1\}$, $\{2, 3\}$ and $\{3, 4\}$. The teams $\{5, 1\}$ and $\{2, 3\}$ can be formed together, but all three cannot, because the last two have a common student 3.
- In the second test case, both teams consist of all four students, so only one can be formed.
- In the third test case, $\{1, 2, 3\}$, $\{4\}$ and $\{5, 6\}$ can be formed. No more can be formed.

In the second example, the possible teams are $\{9, 10, 1, 2\}$, $\{3, 4, 5\}$, $\{6, 7, 8\}$, $\{1, 2, 3, 4\}$, $\{5, 6, 7\}$ and $\{4\}$. The first three of them can be formed together, but four cannot.

J Scales

Time limit 0.5 s · Memory limit 256 MiB · Standard input/output · Interactive problem

The jury has old scales and two secret integers a and b, where $0 \le a, b < 2^n$, and the number n is known to you. Your task is to find both secret numbers.

The scales are a bit unusual. You can give the jury two integers x and y ($0 \le x, y < 2^n$). Then the jury places the number $a \oplus x$ on the left pan, and the number $b \oplus y$ on the right pan, and reports which pan is heavier, or that the scales are balanced. In other words, you learn the result of comparing $a \oplus x$ with $b \oplus y$. Here $\oplus$ denotes the bitwise exclusive OR (XOR).

You may use the scales no more than $2n$ times.

Input

The first line contains a single integer n ($2 \le n \le 60$).

Interaction

To perform a weighing, output a line `? x y` with two integers x and y ($0 \le x, y < 2^n$). In response, the jury will output a line with a single character:

- `<`, if $a \oplus x < b \oplus y$;
- `=`, if $a \oplus x = b \oplus y$;
- `>`, if $a \oplus x > b \oplus y$.

When you have determined the secret numbers, output a line `! a b` with the numbers a and b and terminate the program. This line is not counted as a weighing.

After outputting each line, make sure to output a newline character and flush the output buffer (for example, `fflush(stdout)` or `cout.flush()` in C++, `sys.stdout.flush()` in Python, `System.out.flush()` in Java). Otherwise, you may receive a "Time Limit Exceeded" verdict. If you perform more than $2n$ weighings, or a weighing with numbers outside the allowed bounds, or output a line in an incorrect format, you will receive a "Wrong Answer" verdict.

The jury may be **adaptive**: the numbers a and b are not fixed in advance and may depend on your weighings. However, at every moment there exists at least one pair (a, b) that is consistent with all the answers given so far. Therefore, output the answer only when the pair is uniquely determined: if there exists another pair that is also consistent with all the answers, the jury may consider that pair to be the secret one, and your answer will be judged incorrect.

Examples

Input
```
3

>

<

<

=

>
```

Output
```
? 0 0

? 4 0

? 0 4

? 4 2

? 4 3

! 5 3
```

Input
```
2

<

=

>
```

Output
```
? 0 0

? 0 2

? 0 3

! 1 3
```

Note

In the examples, the input column shows the lines output by the jury, and the output column shows the lines output by your program, in the order in which the exchange takes place. Blank lines are added only for column alignment; they are not part of the interaction.

In the first example, $n = 3$, and the secret numbers are $a = 5$ and $b = 3$.

- `? 0 0` compares $a = 5$ with $b = 3$, the answer is `>`;
- `? 4 0` compares $5 \oplus 4 = 1$ with 3, the answer is `<`;
- `? 0 4` compares 5 with $3 \oplus 4 = 7$, the answer is `<`;
- `? 4 2` compares $5 \oplus 4$ with $3 \oplus 2$, that is 1 with 1, the answer is `=`;
- `? 4 3` compares $5 \oplus 4$ with $3 \oplus 3$, that is 1 with 0, the answer is `>`.

To give the answer, the program outputs `! 5 3`. Five weighings were used, which is within the limit of $2n$.

In the second example, $n = 2$, $a = 1$, $b = 3$.

- `? 0 0`, the answer is `<`;
- `? 0 2` compares a with $b \oplus 2$, the answer is `=`;
- `? 0 3` compares a with $b \oplus 3$, the answer is `>`.

Three weighings were used, which is within the limit of $2n$.

K Sum and Product

Time limit 1 s · Memory limit 256 MiB · Standard input/output

Olenka just learned about real numbers and immediately invented a game. She names a real number s, and her older brother must write on the board one or several real numbers $a_1, a_2, \ldots, a_n$ so that their sum equals s, and their product equals 1:

$$a_1 + a_2 + \ldots + a_n = s, \qquad a_1 \cdot a_2 \cdot \ldots \cdot a_n = 1.$$

The brother is lazy, so he wants to write as few numbers as possible. Help him: for each number that Olenka names, find the smallest possible $n \geq 1$ and any suitable numbers $a_1, \ldots, a_n$.

Input

The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of numbers that Olenka named.

Each of the next t lines contains a single real number s ($|s| \leq 10^9$), written with exactly three digits after the decimal point.

Output

For each of the t numbers, output two lines. The first line should contain an integer n — the smallest number of real numbers whose sum is **exactly** equal to s, and whose product is **exactly** equal to 1. The second line should contain n real numbers $a_1, a_2, \ldots, a_n$, separated by spaces. It can be proven that such numbers always exist.

Since the numbers are output with limited precision, an answer is accepted if n is the smallest possible, and the output numbers satisfy the following conditions:

- $|a_i| \leq 10^{10}$ for each i;
- $|(a_1 + a_2 + \ldots + a_n) - s| \leq 10^{-6} \cdot \max(1, |s|)$;
- $|a_1 \cdot a_2 \cdot \ldots \cdot a_n - 1| \leq 10^{-6}$.

The numbers can be output either in ordinary decimal notation (for example, `-0.618034`), or in exponential notation (for example, `1.5e-07`).

Examples

Input	Output
5 1.000 2.500 0.500 -2.000 0.000	1 1 2 0.5 2 3 -1 2 -0.5 2 -1 -1 3 -1 1.6180339887 -0.6180339887

Note

In the example, for $s = 1$ one number is enough. For $s = 2.5$ one can take $0.5 + 2 = 2.5$ and $0.5 \cdot 2 = 1$. For $s = 0.5$ neither one number nor a pair of numbers works, but the triple $-1, 2, -0.5$

works. For $s = 0$ the output numbers -1, 1.6180339887, -0.6180339887 are correct only approximately (the exact values are $\frac{1 \pm \sqrt{5}}{2}$), but they satisfy the conditions above.

The precision 10^{-6} can be achieved using ordinary double-precision floating-point arithmetic (the `double` type).

L Three Computers

Time limit 3 s · Memory limit 256 MiB · Standard input/output

A team of three students is participating in an unusual programming olympiad. Each team member has their own computer, so the team can work on three problems simultaneously.

The olympiad has n problems. Problem i can be solved only if one participant works on it without interruption from minute l_i to minute r_i inclusive. During one minute, a participant can work on only one problem, and a problem cannot be split between several participants. For example, one participant can solve problems that need minutes 1–3 and 4–5, but not problems that need minutes 1–3 and 3–5, because both need minute 3.

For solving problem i, the team receives w_i points. Find the greatest total number of points the team can get.

Input

The first line contains a single integer t ($1 \le t \le 2 \cdot 10^5$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of problems.

Each of the next n lines contains three integers l_i, r_i and w_i ($1 \le l_i \le r_i \le 10^9$, $1 \le w_i \le 10^9$) — the first and last minute of problem i and the number of points for it.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print on a separate line a single integer — the greatest total number of points.

Examples

Input

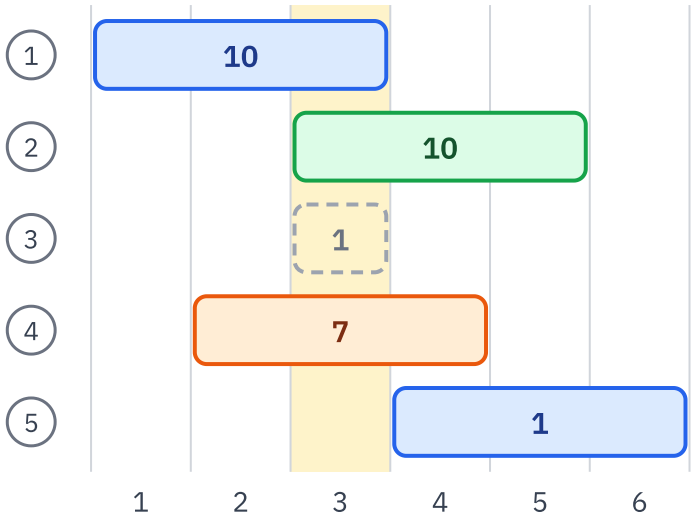
```
3
4
1 5 3
2 6 4
3 7 5
4 8 2
5
1 3 10
3 5 10
3 3 1
2 4 7
4 6 1
1
1000000000 1000000000 1000000000
```

Output

```
12
28
1000000000
```

Note

In the first test case, all four problems need minutes 4 and 5, so at most three of them can be solved. It is most profitable to give up the problem worth 2 points, and get $3 + 4 + 5 = 12$ points.



In the second test case, the first four problems need minute 3, so one of them has to be given up. It is most profitable to give up the third problem. The other four problems can be solved: the first participant solves problems 1 (minutes 1–3) and 5 (minutes 4–6), the second solves problem 2, and the third solves problem 4. In total, $10 + 1 + 10 + 7 = 28$ points.

In the third test case, the only problem can be solved.

M UCPC Garland

Time limit 4.5 s · Memory limit 256 MiB · Standard input/output

The organizers of UCPC are decorating the competition hall with a long garland of lamps shaped like letters. The garland forms a string s of length n consisting of uppercase Latin letters: the i-th lamp shows the letter s_i. Each lamp must be turned on in one of three colors: blue, yellow, or white.

Of course, the organizers' favorite word is `UCPC`. An *occurrence* of the word `UCPC` in the garland is a choice of four lamps at positions $i_1 < i_2 < i_3 < i_4$ that show the letters `U`, `C`, `P`, `C` respectively (the chosen lamps do not necessarily have to be adjacent). The *coloring of an occurrence* is the sequence of colors of the lamps i_1, i_2, i_3, i_4 in exactly that order.

A coloring of the entire garland is called *consistent* if all occurrences of the word `UCPC` have **the same coloring**. In other words, no matter which occurrence of the word `UCPC` a viewer reads in the garland, they will always see the same sequence of colors. In particular, if `UCPC` does not occur in the garland at all, then any coloring is consistent.

Count the number of consistent colorings of the garland. Two colorings are considered different if at least one lamp has a different color in them. Since the answer can be large, output it modulo 998 244 353.

The organizers are considering several garlands, and you need to answer this question for each of them.

Input

The first line contains an integer t ($1 \le t \le 10^6$) — the number of garlands.

Each of the next t lines contains a string s ($1 \le |s| \le 10^6$, where $|s|$ is the length of the string s) — the garland. It is guaranteed that the garland consists of uppercase Latin letters.

The total length of all t strings does not exceed 10^6.

Output

For each garland, output on a separate line a single integer — the number of its consistent colorings modulo 998 244 353.

Examples

Input	Output
3 UCPCPC UUCCPPCC CPU	27 81 27

Note

In the first example garland, the word `UCPC` occurs four times, at positions $(1, 2, 3, 4)$, $(1, 2, 3, 6)$, $(1, 2, 5, 6)$, and $(1, 4, 5, 6)$. Lamp 4 is the second letter of the last occurrence and the fourth letter of the first one, so both letters `C` of the word must have the same color. Hence all lamps with the letter `C` share one common color, and both lamps with the letter `P` share one common color. Together with the color of lamp 1, this gives $3^3 = 27$ consistent colorings.

In the second example garland, no lamp can simultaneously be the second and the fourth letter of an occurrence, so the two letters `C` of the word can have different colors: lamps $1, 2$ share one com-

mon color, lamps $3, 4$ — one common color, lamps $5, 6$ — one common color, and lamps $7, 8$ — one common color, giving $3^4 = 81$ colorings.

In the third example garland, the word `UCPC` does not occur at all, so all $3^3 = 27$ colorings are consistent.

N Weather Diary

Time limit 0.5 s · Memory limit 256 MiB · Standard input/output

For n consecutive days, amateur meteorologist Ostap recorded in his diary what each day was like: *sunny* or *rainy*.

Ostap's younger brother spilled tea on the diary, and the daily entries became illegible. Only a page with a summary at the end survived. For each pair of adjacent days (day i and day $i+1$, where $1 \le i < n$), Ostap counted what type it was:

- e pairs in which a sunny day is followed by a sunny day;
- g pairs in which a rainy day is followed by a rainy day;
- u pairs in which a sunny day is followed by a rainy day;
- v pairs in which a rainy day is followed by a sunny day.

Each of the $n-1$ pairs is counted exactly once, so $e+g+u+v=n-1$.

Ostap is not sure that he didn't make a mistake in his counting. Determine whether there exists at least one sequence of weather for n days that agrees with all four numbers of the summary.

The diary contains summaries for several different periods, so this question needs to be answered for t independent summaries.

Input

The first line contains an integer t ($1 \le t \le 10^5$) — the number of summaries.

Each of the next t lines describes one summary and contains five integers n, e, g, u, v ($1 \le n \le 2 \cdot 10^5$; $e, g, u, v \ge 0$; $e+g+u+v=n-1$).

The sum of n over all summaries does not exceed $4 \cdot 10^6$.

Output

For each summary, output a separate line: `YES` if there exists a weather sequence matching it, or `NO` otherwise.

The case of letters does not matter: `yes`, `Yes`, and `YES` are all accepted equally.

Examples

Input	Output
4 5 1 1 1 1 5 1 0 2 1 5 0 0 4 0 3 0 0 1 1	YES YES NO YES

Note

Let us denote a sunny day by the letter `S`, and a rainy day by the letter `R`.

In the first summary of the example, the sequence `SSRRS` fits: its pairs are `SS`, `SR`, `RR`, and `RS`.

In the second summary, the sequence `SSRSR` fits: its pairs are `SS`, `SR`, `RS`, and `SR`.

In the third summary, all four pairs would have to be “sunny, then rainy”. Then the second day would have to be rainy (as the second day of the first pair) and sunny (as the first day of the second

pair) at the same time, which is impossible.

In the fourth summary, the sequence SRS fits.