A. Square String?

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

A string is called *square* if it is some string written twice in a row. For example, the strings "aa", "abcabc", "abab" and "baabaa" are square. But the strings "aaa", "abaaab" and "abcdabc" are not square.

For a given string s determine if it is square.

Input

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

This is followed by t lines, each containing a description of one test case. The given strings consist only of lowercase Latin letters and have lengths between 1 and 100 inclusive.

Output

For each test case, output on a separate line:

- YES if the string in the corresponding test case is square,
- NO otherwise.

You can output YES and NO in any case (for example, strings yEs, yes, Yes and YES will be recognized as a positive response).

Example

input

```
10
a
aa
aaa
aaaa
abab
abcabc
abacaba
xxyy
xyyx
xyxy
```

output

```
NO
YES
NO
YES
YES
YES
NO
NO
NO
YES
```

B. Squares and Cubes

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

Polycarp likes squares and cubes of positive integers. Here is the beginning of the sequence of numbers he likes: 1, 4, 8, 9, ...

For a given number n, count the number of integers from 1 to n that Polycarp likes. In other words, find the number of such x that x is a square of a positive integer number or a cube of a positive integer number (or both a square and a cube simultaneously).

Input

The first line contains an integer t ($1 \le t \le 20$) — the number of test cases.

Then t lines contain the test cases, one per line. Each of the lines contains one integer n ($1 \le n \le 10^9$).

Output

For each test case, print the answer you are looking for — the number of integers from 1 to n that Polycarp likes.

Example

input

```
6
10
1
25
1000000000
999999999
500000000
```

output

```
4
1
6
32591
32590
23125
```

C. Wrong Addition

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Tanya is learning how to add numbers, but so far she is not doing it correctly. She is adding two numbers a and b using the following algorithm:

1. If one of the numbers is shorter than the other, Tanya adds leading zeros so that the numbers are the same length.
2. The numbers are processed from right to left (that is, from the least significant digits to the most significant).
3. In the first step, she adds the last digit of a to the last digit of b and writes their sum in the answer.
4. At each next step, she performs the same operation on each pair of digits in the same place and writes the result to the **left** side of the answer.

For example, the numbers $a = 17236$ and $b = 3465$ Tanya adds up as follows:

$$\begin{array}{r} 17236 \\ +\ \ 03465 \\ \hline 1106911 \end{array}$$

- calculates the sum of $6 + 5 = 11$ and writes 11 in the answer.
- calculates the sum of $3 + 6 = 9$ and writes the result to the left side of the answer to get 911.
- calculates the sum of $2 + 4 = 6$ and writes the result to the left side of the answer to get 6911.
- calculates the sum of $7 + 3 = 10$, and writes the result to the left side of the answer to get 106911.
- calculates the sum of $1 + 0 = 1$ and writes the result to the left side of the answer and get 1106911.

As a result, she gets 1106911.

You are given two positive integers a and s. Find the number b such that by adding a and b as described above, Tanya will get s. Or determine that no suitable b exists.

Input

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

Each test case consists of a single line containing two positive integers a and s ($1 \le a < s \le 10^{18}$) separated by a space.

Output

For each test case print the answer on a separate line.

If the solution exists, print a single positive integer b. The answer must be written without leading zeros. If multiple answers exist, print any of them.

If no suitable number b exists, output -1.

Example

input

```
6
17236 1106911
1 5
108 112
12345 1023412
1 11
1 20
```

output

```
3465
```

```
4
-1
90007
10
-1
```

Note

The first test case is explained in the main part of the statement.

In the third test case, we cannot choose b that satisfies the problem statement.

D. New Year's Problem

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

Vlad has n friends, for each of whom he wants to buy *one* gift for the New Year.

There are m shops in the city, in each of which he can buy a gift for any of his friends. If the j-th friend ($1 \le j \le n$) receives a gift bought in the shop with the number i ($1 \le i \le m$), then the friend receives p_{ij} units of joy. The rectangular table p_{ij} is given in the input.

Vlad has time to visit at most $n-1$ shops (where n is the number of **friends**). He chooses which shops he will visit and for which friends he will buy gifts in each of them.

Let the j-th friend receive a_j units of joy from Vlad's gift. Let's find the value $\alpha = \min\{a_1, a_2, \dots, a_n\}$. Vlad's goal is to buy gifts so that the value of α is as large as possible. In other words, Vlad wants to maximize the minimum of the joys of his friends.

For example, let $m = 2$, $n = 2$. Let the joy from the gifts that we can buy in the first shop: $p_{11} = 1$, $p_{12} = 2$, in the second shop: $p_{21} = 3$, $p_{22} = 4$.

Then it is enough for Vlad to go only to the second shop and buy a gift for the first friend, bringing joy 3, and for the second — bringing joy 4. In this case, the value α will be equal to $\min\{3, 4\} = 3$

Help Vlad choose gifts for his friends so that the value of α is as high as possible. Please note that each friend must receive one gift. Vlad can visit at most $n-1$ shops (where n is the number of **friends**). In the shop, he can buy any number of gifts.

Input

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

An empty line is written before each test case. Then there is a line containing integers m and n ($2 \le n$, $2 \le n \cdot m \le 10^5$) separated by a space — the number of shops and the number of friends, where $n \cdot m$ is the product of n and m.

Then m lines follow, each containing n numbers. The number in the i-th row of the j-th column p_{ij} ($1 \le p_{ij} \le 10^9$) is the joy of the product intended for friend number j in shop number i.

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

Output

Print t lines, each line must contain the answer to the corresponding test case — the maximum possible value of α, where α is the minimum of the joys from a gift for all of Vlad's friends.

Example

input

```
5

2 2
1 2
3 4

4 3
1 3 1
3 1 1
1 2 2
1 1 3

2 3
5 3 4
2 5 1

4 2
7 9
8 1
9 6
10 8

2 4
6 5 2 1
7 9 7 2
```

output

```
3
2
4
8
2
```

E. MEX and Increments

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

Dmitry has an array of n non-negative integers $a_1, a_2, \dots, a_n$.

In one operation, Dmitry can choose any index j ($1 \le j \le n$) and increase the value of the element a_j by 1. He can choose the same index j multiple times.

For each i from 0 to n, determine whether Dmitry can make the MEX of the array equal to exactly i. If it is possible, then determine the minimum number of operations to do it.

The MEX of the array is equal to the minimum non-negative integer that is not in the array. For example, the MEX of the array $[3, 1, 0]$ is equal to 2, and the array $[3, 3, 1, 4]$ is equal to 0.

Input

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

The descriptions of the test cases follow.

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

The second line of the description of each test case contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le n$) — elements of the array a.

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

Output

For each test case, output $n + 1$ integer — i-th number is equal to the minimum number of operations for which you can make the array MEX equal to i ($0 \le i \le n$), or -1 if this cannot be done.

Example

input

```
5
3
0 1 3
7
0 1 2 3 4 3 2
4
3 0 0 0
7
4 6 2 3 5 0 5
5
4 0 1 0 4
```

output

```
1 1 0 -1
1 1 2 2 1 0 2 6
3 0 1 4 3
1 0 -1 -1 -1 -1 -1 -1
2 1 0 2 -1 -1
```

Note

In the first set of example inputs, $n = 3$:

- to get $\mathrm{MEX} = 0$, it is enough to perform one increment: a_1++;
- to get $\mathrm{MEX} = 1$, it is enough to perform one increment: a_2++;
- $\mathrm{MEX} = 2$ for a given array, so there is no need to perform increments;
- it is impossible to get $\mathrm{MEX} = 3$ by performing increments.

F. Let's Play the Hat?

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

The Hat is a game of speedy explanation/guessing words (similar to Alias). It's fun. Try it! In this problem, we are talking about a variant of the game when the players are sitting at the table and everyone plays individually (i.e. not teams, but individual gamers *play).*

n people gathered in a room with m tables ($n \geq 2m$). They want to play the Hat k times. Thus, k games will be played at each table. Each player will play in k games.

To do this, they are distributed among the tables for each game. During each game, one player plays at exactly one table. A player can play at different tables.

Players want to have the most "fair" schedule of games. For this reason, they are looking for a schedule (table distribution for each game) such that:

- At any table in each game there are either $\lfloor \frac{n}{m} \rfloor$ people or $\lceil \frac{n}{m} \rceil$ people (that is, either n/m rounded down, or n/m rounded up). Different numbers of people can play different games at the same table.
- Let's calculate for each player the value b_i — the number of times the i-th player played at a table with $\lceil \frac{n}{m} \rceil$ persons (n/m rounded up). Any two values of $b_i$must differ by no more than 1. In other words, for any two players i and j, it must be true $|b_i - b_j| \leq 1$.

For example, if $n = 5$, $m = 2$ and $k = 2$, then at the request of the first item either two players or three players should play at each table. Consider the following schedules:

- First game: $1, 2, 3$ are played at the first table, and $4, 5$ at the second one. The second game: at the first table they play $5, 1$, and at the second $-2, 3, 4$. This schedule is **not "fair"** since $b_2 = 2$ (the second player played twice at a big table) and $b_5 = 0$ (the fifth player did not play at a big table).
- First game: $1, 2, 3$ are played at the first table, and $4, 5$ at the second one. The second game: at the first table they play $4, 5, 2$, and at the second one $-1, 3$. This schedule is **"fair"**: $b = [1, 2, 1, 1, 1]$ (any two values of b_i differ by no more than 1).

Find any "fair" game schedule for n people if they play on the m tables of k games.

Input

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

Each test case consists of one line that contains three integers n, m and k ($2 \leq n \leq 2 \cdot 10^5$, $1 \leq m \leq \lfloor \frac{n}{2} \rfloor$, $1 \leq k \leq 10^5$) — the number of people, tables and games, respectively.

It is guaranteed that the sum of nk (n multiplied by k) over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case print a required schedule — a sequence of k blocks of m lines. Each block corresponds to one game, a line in a block corresponds to one table. In each line print the number of players at the table and the indices of the players (numbers from 1 to n) who should play at this table.

If there are several required schedules, then output any of them. We can show that a valid solution always exists.

You can output additional blank lines to separate responses to different sets of inputs.

Example

input

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

output

```
3 1 2 3
2 4 5
3 4 5 2
2 1 3

2 6 2
3 3 5 1
3 4 7 8

2 2 1
2 2 1
2 2 1
```

G. Unusual Minesweeper

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

Polycarp is very fond of playing the game Minesweeper. Recently he found a similar game and there are such rules.

There are mines on the field, for each the coordinates of its location are known (x_i, y_i). Each mine has a lifetime in seconds, after which it will explode. After the explosion, the mine also detonates all mines vertically and horizontally at a distance of k (two perpendicular lines). As a result, we get an explosion on the field in the form of a "plus" symbol ('**+**'). Thus, one explosion can cause new explosions, and so on.

Also, Polycarp can detonate anyone mine every second, starting from zero seconds. After that, a chain reaction of explosions also takes place. Mines explode **instantly** and also **instantly** detonate other mines according to the rules described above.

Polycarp wants to set a new record and asks you to help him calculate in what minimum number of seconds all mines can be detonated.

Input

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

An empty line is written in front of each test suite.

Next comes a line that contains integers n and k ($1 \le n \le 2 \cdot 10^5$, $0 \le k \le 10^9$) — the number of mines and the distance that hit by mines during the explosion, respectively.

Then n lines follow, the i-th of which describes the x and y coordinates of the i-th mine and the time until its explosion ($-10^9 \le x, y \le 10^9$, $0 \le timer \le 10^9$). It is guaranteed that all mines have different coordinates.

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

Output

Print t lines, each of the lines must contain the answer to the corresponding set of input data — the minimum number of seconds it takes to explode all the mines.

Example

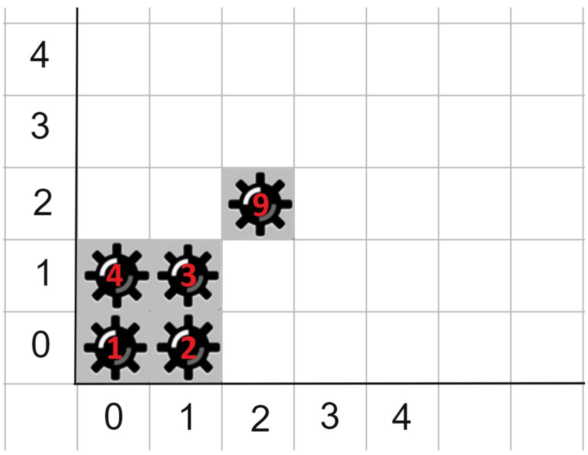
input

```
3

5 0
0 0 1
0 1 4
1 0 2
1 1 3
2 2 9

5 2
0 0 1
0 1 4
1 0 2
1 1 3
2 2 9

6 1
1 -1 3
0 -1 9
0 1 7
-1 0 1
-1 1 9
-1 -1 7
```

output

```
2
1
0
```

Note



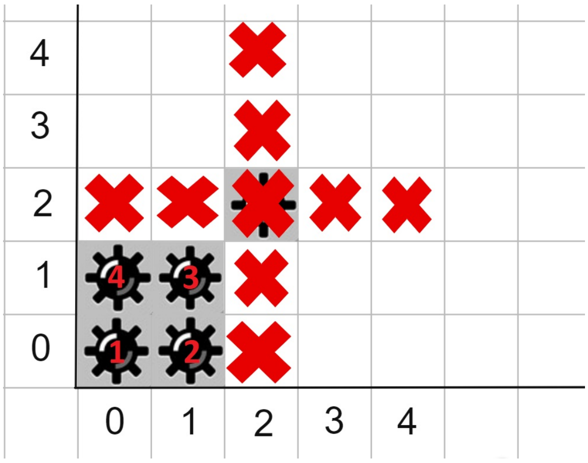
Picture from examples

First example:

- 0 second: we explode a mine at the cell $(2, 2)$, it does not detonate any other mine since $k = 0$.
- 1 second: we explode the mine at the cell $(0, 1)$, and the mine at the cell $(0, 0)$ explodes itself.
- 2 second: we explode the mine at the cell $(1, 1)$, and the mine at the cell $(1, 0)$ explodes itself.

Second example:

- 0 second: we explode a mine at the cell $(2, 2)$ we get:



- 1 second: the mine at coordinate $(0, 0)$ explodes and since $k = 2$ the explosion detonates mines at the cells $(0, 1)$ and $(1, 0)$, and their explosions detonate the mine at the cell $(1, 1)$ and there are no mines left on the field.

H. Permutation and Queries

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a permutation p of n elements. A permutation of n elements is an array of length n containing each integer from 1 to n exactly once. For example, $[1, 2, 3]$ and $[4, 3, 5, 1, 2]$ are permutations, but $[1, 2, 4]$ and $[4, 3, 2, 1, 2]$ are not permutations. You should perform q queries.

There are two types of queries:

- $1\ x\ y$ — swap p_x and p_y.
- $2\ i\ k$ — print the number that i will become if we assign $i = p_i$ k times.

Input

The first line contains two integers n and q ($1 \le n, q \le 10^5$).

The second line contains n integers $p_1, p_2, \dots, p_n$.

Each of the next q lines contains three integers. The first integer is t ($1 \le t \le 2$) — type of query. If $t = 1$, then the next two integers are x and y ($1 \le x, y \le n$; $x \ne y$) — first-type query. If $t = 2$, then the next two integers are i and k ($1 \le i, k \le n$) — second-type query.

It is guaranteed that there is at least one second-type query.

Output

For every second-type query, print one integer in a new line — answer to this query.

Examples

input

```
5 4
5 3 4 2 1
2 3 1
2 1 2
1 1 3
2 1 2
```

output

```
4
1
2
```

input

```
5 9
2 3 5 1 4
2 3 5
2 5 5
2 5 1
2 5 3
2 5 4
1 5 4
2 5 3
2 2 5
2 5 1
```

output

```
3
5
4
2
3
3
3
1
```

Note

In the first example $p = \{5, 3, 4, 2, 1\}$.

The first query is to print p_3. The answer is 4.

The second query is to print p_{p_1}. The answer is 1.

The third query is to swap p_1 and p_3. Now $p = \{4, 3, 5, 2, 1\}$.

The fourth query is to print p_{p_1}. The answer is 2.