# A. Likes Display

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp is working on the implementation of displaying likes on the Codehorses social network. The number of likes should be displayed in a format that will be easy to read by users. It was decided that for large numbers of likes the format should be like 123K (one hundred twenty-three thousand) or like 56M (fifty-six million).

The following displaying strategy has been approved:

- the number will be displayed either as an integer number from 0 to 999, or as a positive integer number of thousands (from 1K to 999K), or as a positive integer number of millions (from 1M on),
- the specified exact number of likes $n$ when displaying should be rounded to the nearest view from the case above (if rounding is ambiguous, it must be rounded up): for example, $1785$ should be rounded to 2K instead of 1K, $4500000$ should be rounded to 5M.

Help Polycarp implement this part of the functionality: for a given non-negative integer number of likes $n$, print its view in the Codehorses interface.

### Input

The first line contains an integer $t$ ($1 \le t \le 1000$) — the number of test cases in the input. The following are descriptions of the $t$ input test cases, one per line.

The description of each test case consists of a single line that contains a non-negative integer $n$ ($0 \le n \le 2 \cdot 10^9$) — the number of likes.

### Output

Print $t$ answers to the given test cases in the order from the input. Each printed value must have one of the following types:

- either an integer from 0 to 999 which corresponds just to the number of likes,
- or a number of thousands from 1K to 999K,
- or a number of millions from 1M to 2000M.

The answer is equal to a view which is the closest (by difference) to the given number $n$. If this rounding is ambiguous, then round answer up (to a greater value).

**Example**

| input |
| --- |
| 9<br>999<br>123<br>0<br>1782<br>31415926<br>1500<br>999999<br>35499710<br>2000000000 |
| **output** |
| 999<br>123<br>0<br>2K<br>31M<br>2K<br>1M<br>35M<br>2000M |

### Note

Let's describe some test cases:

- $1782$ can be displayed either as 1K or as 2K but 2K is the nearest view;
- $1500$ have same difference with 1K and 2K so it should be rounded up;
- $999999$ should be displayed as 1M since it's closer to it than to 999K.

# B. Cartoons

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Tanya likes cartoons. She knows that $n$ new cartoons will be released in his favorite cinema: the $i$-th of them will be airing from the day $a_i$ to the day $b_i$ ($1 \le a_i \le b_i \le 10^9$).

The cinema has a special offer: there is a huge discount every day when only one cartoon is airing.

Tanya doesn't care which cartoon she will watch but she'd like to save some money. That's why she asks you to find any day $x$ when only one cartoon will be airing. Formally: find $x$ such that there is exactly one $i$ ($1 \le i \le n$) with $a_i \le x \le b_i$. If there are several

possible answers, print any of them. If there is no such day, print -1.

### Input
The first line contains single integer $t$ ($1 \le t \le 1000$) — the number of test cases. The following are descriptions of the $t$ test cases.

The first line of each test case contains a single integer $n$ ($1 \le n \le 2000$) — the number of cartoons.

In the next $n$ lines, the cartoons themselves are described, one per line, by a pair of integers $a_i$, $b_i$ ($1 \le a_i \le b_i \le 10^9$) — the first and last airing days for the $i$-th cartoon.

It is guaranteed that the sum of the values $n$ for all test cases in the input does not exceed $2000$.

### Output
Print $t$ answers to given test cases in the order in which they appear in the input: the $i$-th answer is such $x$, that only one cartoon will be airing on day $x$ or -1 if there are no such days.

**Example**

| input |
| --- |
| 5<br>1<br>1 1<br>3<br>2 1000000000<br>2 500000000<br>500000002 1000000000<br>3<br>1 2<br>3 4<br>1 4<br>2<br>4 11<br>4 9<br>3<br>1 5<br>10 10<br>1 5 |
| **output** |
| 1<br>500000001<br>-1<br>10<br>10 |

### Note
In the third test case: at day $1$ and $2$, first and third cartoons will be airing, and days $3$ and $4$, second and third cartoons will be airing. So, there is no day when only one cartoon will be airing.

In the fourth test case, $11$ is also a possible answer.

## C. Dream Team

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp is the project manager in the IT-company. Right now, he needs to choose developers for his team to start a new project. The company has $n$ developers "on the bench" (i.e not involved in other projects). Polycarp assessed the skills of each of them: $a_i$ ($-10^4 \le a_i \le 10^4$) — an integer characteristic of the $i$-th developer. This value can be either positive, zero or even negative (some developers cause distractions).

After Polycarp chooses a subset of developers for his team, the strength of the team will be determined by the sum of $a_i$ values for all selected developers.

Polycarp fears that if he chooses a team in such a way that maximizes the sum of the characteristics of $a_i$, other managers may find this unacceptable. For this reason, he plans to create such a team that the sum of the $a_i$ values for it is **strictly less** than the maximum possible value.

Help Polycarp choose any team that:

- the sum of the characteristics $a_i$ for all members of the selected team is strictly less than the maximum value that can be achieved by choosing the team in some other way
- and at the same time, the sum of the characteristics of $a_i$ for all members of the selected team is the greatest possible.

If, following the requirements above, you can select a team in several ways, then just find any of them.

It's guaranteed that **the sum of the characteristics in the desired subset is strictly positive** (i.e. Polycarp can always choose a non-empty team).

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

Each test case begins with a line containing one integer $n$ ($2 \le n \le 10^5$) — the number of developers "on the bench".

The second line of a test case contains a sequence of integers $a_1, a_2, \ldots, a_n$ ($-10^4 \le a_i \le 10^4$) — the characteristics of the $n$ developers. It is guaranteed that the characteristics are such that **the sum of the characteristics in the answer is strictly positive**.

It is guaranteed that the sum of $n$ over all test cases in the input doesn't exceed $10^5$.

### Output
Print the answers for the given $t$ test cases in the order that they appear in the input. In the first line of each answer, print a positive integer $s$ — the sum of the characteristics in the desired subset. The second line should contain only the characters 0 and 1 and

match the answer:

- the character in the $i$-th position should be equal to 1 if the $i$-th developer belongs to the team;
- the character in the $i$-th position should be equal to 0 if the $i$-th developer does not belong to the team.

If there are several answers, print any of them.

**Example**

| input |
| --- |
| 5<br>5<br>1 -1 1 -1 1<br>2<br>11 1<br>3<br>5 -3 4<br>3<br>5 3 -4<br>5<br>-1 0 3 -3 0 |
| **output** |
| 2<br>11101<br>11<br>10<br>6<br>111<br>5<br>100<br>2<br>10100 |

### Note

In the first test case, the maximum subset $a_1, a_3, a_5$ has a sum equal to $3$, so Polycarp should choose a team with the maximum total sum which is less than $3$.

In the second test case, the maximum subset $a_1, a_2$ has a sum equal to $12$, so Polycarp should choose a team with the maximum total sum which is less than $12$.

In the third test case, the maximum subset $a_1, a_3$ has a sum equal to $9$.

In the fourth test case, the maximum subset $a_1, a_2$ has a sum equal to $8$.

In the fifth test case, there are several subsets with a maximum sum equal to $3$, so Polycarp should choose a team with a lower total sum.

## D. Bonus Distribution

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

For the first time, Polycarp's startup ended the year with a profit! Now he is about to distribute $k$ burles as a bonus among $n$ employees.

It is known that the current salary of the $i$-th employee is $a_i$ and all the values of $a_i$ in the company are **different**.

Polycarp wants to distribute the $k$ burles between $n$ employees so this month the $i$-th employee will be paid not $a_i$, but $a_i + d_i$ ( $d_i \geq 0$, $d_i$ is an integer), where $d_i$ is the bonus for the $i$-th employee. Of course, $d_1 + d_2 + \dots + d_n = k$.

Polycarp will follow two rules for choosing the values $d_i$:

- the relative order of the salaries should not be changed: the employee with originally the highest salary ($a_i$ is the maximum) should have the highest total payment after receiving their bonus ($a_i + d_i$ is also the maximum), the employee whose salary was originally the second-largest should receive the second-largest total payment after receiving their bonus and so on.
- to emphasize that annual profit is a group effort, Polycarp wants to **minimize** the maximum total payment to an employee (i.e minimize the maximum value of $a_i + d_i$).

Help Polycarp decide the non-negative integer bonuses $d_i$ such that:

- their sum is $k$,
- for each employee, the number of those who receive **strictly more** than them remains unchanged (that is, if you sort employees by $a_i$ and by $a_i + d_i$, you get the same order of employees),
- all $a_i + d_i$ are different,
- the maximum of the values $a_i + d_i$ is the minimum possible.

Help Polycarp and print any of the possible answers $d_1, d_2, \dots, d_n$.

### Input

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

The first line of each test case contains two integers $n$ and $k$ ($1 \leq n \leq 10^5$, $1 \leq k \leq 10^9$) — the number of employees and the total bonus.

The second line of each test case contains $n$ **different** integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^9$), where $a_i$ is the current salary of the $i$-th employee.

It is guaranteed that the sum of all $n$ values in the input does not exceed $10^5$.

### Output

Print the answers to $t$ test cases in the order they appear in the input. Print each answer as a sequence of non-negative integers

$d_1, d_2, \ldots, d_n$. If there are several answers, print any of them.

**Example**

| input |
| --- |
| 5<br>4 1<br>3 1 4 2<br>2 3<br>10 2<br>4 1000000000<br>987654321 1000000000 999999999 500000000<br>8 9<br>5 6 1 8 3 4 2 7<br>6 1<br>6 3 1 8 5 9 |
| **output** |
| 0 0 1 0<br>0 3<br>134259259 121913582 121913582 621913577<br>2 2 0 2 0 1 0 2<br>1 0 0 0 0 0 |

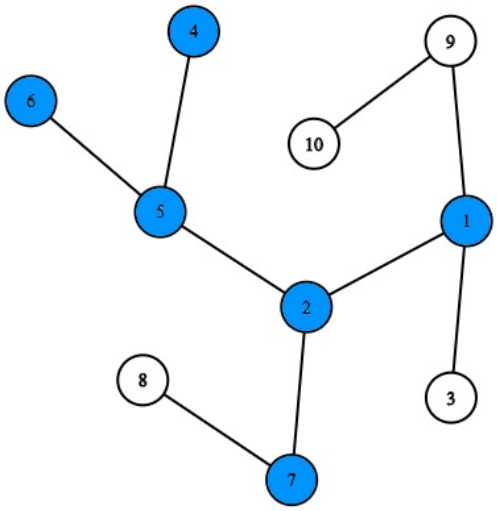
## E. Modernization of Treeland

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Treeland consists of $n$ cities and $n - 1$ two-way roads connecting pairs of cities. From every city, you can reach every other city moving only by the roads. You are right, the system of cities and roads in this country forms an undirected tree.

The government has announced a program for the modernization of urban infrastructure of some cities. You have been assigned to select an arbitrary subset of cities $S$ to upgrade (potentially all the cities) that satisfies the following requirements:

- the subset of cities must be "connected", that is, from any city of the subset $S$ you can get to any other city of the subset $S$ by roads, moving only through cities from $S$,
- the number of "dead-ends" in $S$ must be equal to the given number $k$. A city is a "dead-end" if it is the only city in $S$ or connected to exactly one another city from $S$.




This shows one of the possible ways to select $S$ (blue vertices) for a given configuration and $k = 4$. Dead-ends are vertices with numbers $1$, $4$, $6$ and $7$.

Help Treeland upgrade its cities. Find any of the possible subsets $S$ or determine that such a subset does not exist.

### Input

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

Each test case begins with a line that contains two integers $n$ and $k$ ($2 \le n \le 3 \cdot 10^5$, $1 \le k \le n$) — the number of cities in Treeland and the number of "dead-end" cities required in the subset $S$.

This is followed by $n - 1$ lines with road descriptions. Each road is given by two integers $x$ and $y$ ($1 \le x, y \le n$; $x \ne y$) — the numbers of the cities that are connected by this road. It is guaranteed that from every city you can reach any other, moving only by the roads.

The sum of the values of $n$ for all test cases in the input does not exceed $3 \cdot 10^5$.

### Output

For each test case print Yes or No (in any case, upper or lower), depending on whether the answer exists or not. If the answer exists, then print an integer $m$ ($1 \le m \le n$) — the number of cities in the found subset. Then print $m$ different numbers from $1$ to $n$ — the numbers of the cities in the found subset. City numbers can be printed in any order. If there are several answers, print any of them.

**Example**

| input |
| --- |
| 4<br>10 4<br>4 5<br>5 2<br>2 1 |

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

**output**

```
Yes
9
1 2 4 5 6 7 8 9 10
No
Yes
4
1 3 4 5
Yes
1
4
```

## F. Movie Fan

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Recently, Polycarp has been a fan of cinema novelties and is trying not to miss them!

In the near future, $n$ new movies will be released: the $i$-th of them will be airing from the day $a_i$ and to the day $b_i$. This means that if Polycarp wants to watch the $i$-th movie in the cinema, he must do so in the period from $a_i$ to $b_i$ inclusive.

If perhaps Polycarp will not have the opportunity to watch a movie in a cinema, he can then do it after day $b_i$ by watching it using an online service. Of course, this is an undesirable outcome for Polycarp because the whole world will have time to discuss this movie on social networks!

Polycarp can watch no more than $m$ movies per day. Help Polycarp find a movie-watching schedule such that every movie will be watched in the cinema. If such a schedule does not exist, then Polycarp wants to watch movies so that:

- for each movie that he doesn't have time to watch in the cinema, we will find the number of days between the end of its airing and the day when Polycarpus watches the movie,
- **the maximum** of the values from the previous point should be **as small as possible**.

### Input

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

The first line of each test case contains two integers $n$ and $m$ ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 10^9$) — the number of movies and the maximum number of movies that Polycarp can view per day.

In the next $n$ lines, the movies themselves are described, one per line, by a pair of integers $a_i$, $b_i$ ($1 \le a_i \le b_i \le 10^9$) — the first and last airing days for the $i$-th movie.

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

### Output

Print $t$ answers to given test cases in the order in which they appear in the input: the $i$-th answer should consist of two lines. Print the integer $d$ in the first line of each test case answer:

- $d = 0$, if there is a schedule such that all movies are watched during airing,
- $d > 0$, if such a schedule does not exist — in this case, $d$ is equal to the minimum value of maximum among all the watching "delays" after the end of airing.

In the second line of the answer to each test case, print $n$ positive integers $t_1, t_2, \ldots, t_n$, where $t_i$ is the number of the day when Polycarp needs to watch the $i$-th movie in the optimal schedule.

If there are several answers, print any of them.

**Example**

**input**

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

```
6 1
13 13
31 31
25 25
12 12
14 14
10 10
```

**output**

```
1
1 3 2 3 1 4 2
1
1 1 1 2 2
0
13 31 25 12 14 10
```

## G. M-numbers

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

For a given positive integer $m$, a positive number is called a $m$-number if the product of its digits is $m$. For example, the beginning of a series of $24$-numbers are as follows: $38$, $46$, $64$, $83$, $138$, $146$, $164$, $183$, $226$ ...

You are given a positive integer $m$ and $k$. Print $k$-th among $m$-numbers if all $m$-numbers are sorted in ascending order.

### Input

A single line of input contains two integers $m$ and $k$ ($2 \le m \le 10^9$, $1 \le k \le 10^9$).

### Output

Print the desired number — $k$-th among all $m$-numbers if $m$-numbers are sorted in ascending order. If the answer does not exist, print $-1$.

**Examples**

| input |
| --- |
| 24 9 |
| **output** |
| 226 |

| input |
| --- |
| 24 1 |
| **output** |
| 38 |

| input |
| --- |
| 5040 1000000000 |
| **output** |
| 111121111315213227111 |

| input |
| --- |
| 2020 2020 |
| **output** |
| -1 |

## H. Paint the String

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

You are given a string $s$ of lowercase Latin letters. It is required to paint each letter of the string in one of two colors (red or blue) so that if you write all the red letters from left to right and write all the blue letters from left to right, then the lexicographically maximum of the two written strings is lexicographically minimal. For each index, in the string $s$ you can choose either of two colors.

Formally, we write out:

- the string $r$ (can be empty) — all red letters in the order from left to right (red subsequence),
- the string $b$ (can be empty) — all blue letters in the order from left to right (blue subsequence).

Your task is to paint the string such that $\max(r, b)$ is minimal. Small reminder: the empty string is the lexicographically smallest string.

### Input

The first line contains an integer $t$ ($1 \le t \le 100$) — the number of test cases in the test. Next, the test cases are given, one per line.

Each test case is a non-empty string $s$ of length between $2$ to $100$ characters, inclusive, which consists of lowercase Latin letters.

### Output

Print $t$ lines, the $i$-th of them should contain the answer to the $i$-th test case of the input. Print a string of length $n$, where $n$ is the length of the given string $s$: the $j$-th character of the string should be either 'R'or 'B' depending on the color of the $j$-th character in the answer (painted in red or blue). If there are several possible answers, print any of them.

**Example**

| input |
| --- |
| 5<br>kotlin<br>codeforces<br>abacaba<br>ffccgc<br>yz |

| output |
| --- |
| RRRRBB<br>RRRRRRRBBB<br>RRRRBBB<br>RBBBBR<br>RR |

# I. Falling Blocks

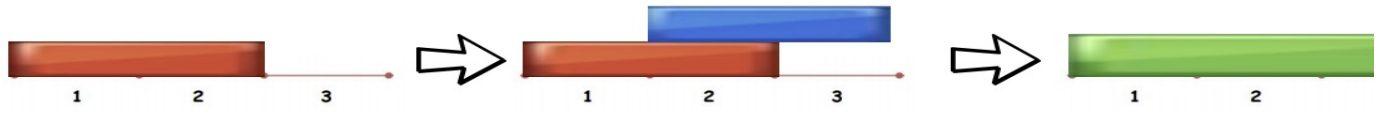
time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Recently, Polycarp has invented a new mobile game with falling blocks.

In the game, $n$ blocks are falling down, one at a time, towards a flat surface with length $d$ units. Each block can be represented as a rectangle with coordinates from $l_i$ to $r_i$ and unit height, dropped downwards from very high up. A block falls until it comes in contact with the flat surface or any other block. Let's define that block $a$ covers block $b$ if $l_a \leq l_b \leq r_b \leq r_a$.

Consider what happens when a new block $i$ falls. If the new (upper) block $i$ comes in contact with **any** block $j$ such that block $i$ **does not** cover block $j$, block $i$ will stick to block $j$, and **no blocks will disappear**. Otherwise, all blocks that block $i$ covers and is in contact with will be vaporized, and block $i$ will continue falling with the ability to vaporize lower blocks.

For example, consider what happens when three blocks $(1, 2)$, $(2, 3)$ and $(1, 3)$ fall, in that order. The first block will stick to the flat surface. Then, the second block will stick to the first block. Finally, the third block will vaporize the second block, keep falling, vaporize the first block, and stick to the flat surface.




Here is a graphic for the first example.

After each block falls, help Polycarp determine how many blocks will remain!

## Input

The first line contains two integers $n$ and $d$ ($1 \leq n, d \leq 10^5$) — the number of falling blocks and the length of the flat surface.

The $i$-th of the following $n$ lines contains integers $l_i$ and $r_i$ ($1 \leq l_i \leq r_i \leq d$) — the coordinates of the $i$-th block.

## Output

Output $n$ integers. The $i$-th integer should be the number of blocks that will be left after the $i$-th block falls.

**Examples**

| input |
| --- |
| 3 3<br>1 2<br>2 3<br>1 3 |

| output |
| --- |
| 1<br>2<br>1 |

| input |
| --- |
| 8 6<br>1 2<br>3 3<br>2 3<br>1 3<br>2 4<br>3 6<br>1 5<br>1 5 |

| output |
| --- |
| 1<br>2<br>3<br>1<br>2<br>3<br>4<br>4 |

## Note

The first example is explained above.

In the second example, this is what happens after each block falls:

- Block $1$ will stick to the flat surface.

- Block $2$ will stick to the flat surface.
- Block $3$ will stick to blocks $1$ and $2$. Note that block $3$ will not vaporize block $2$ because it does not cover block $1$ and is in contact with it.
- Block $4$ will vaporize all the blocks and stick to the flat surface.
- Block $5$ will stick to block $4$
- Block $6$ will stick to block $5$.
- Block $7$ will stick to block $6$. Note that no blocks are vaporized because although block $7$ covers block $4$ and block $5$, they never come in contact.
- Block $8$ vaporizes block $7$ and sticks to block $6$.

---