Educational Codeforces Round 99 (Rated for Div. 2)

A. Strange Functions

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

Let's define a function $f(x)$ (x is a positive integer) as follows: write all digits of the decimal representation of x backwards, then get rid of the leading zeroes. For example, $f(321) = 123$, $f(120) = 21$, $f(1000000) = 1$, $f(111) = 111$.

Let's define another function $g(x) = \dfrac{x}{f(f(x))}$ (x is a positive integer as well).

Your task is the following: for the given positive integer n, calculate the number of different values of $g(x)$ among all numbers x such that $1 \le x \le n$.

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

Each test case consists of one line containing one integer n ($1 \le n < 10^{100}$). This integer is given without leading zeroes.

Output
For each test case, print one integer — the number of different values of the function $g(x)$, if x can be any integer from $[1, n]$.

Example

input
5 4 37 998244353 1000000007 12345678901337426966631415
output
1 2 9 10 26

Note
Explanations for the two first test cases of the example:

1. if $n = 4$, then for every integer x such that $1 \le x \le n$, $\dfrac{x}{f(f(x))} = 1$;
2. if $n = 37$, then for some integers x such that $1 \le x \le n$, $\dfrac{x}{f(f(x))} = 1$ (for example, if $x = 23$, $f(f(x)) = 23$, $\dfrac{x}{f(f(x))} = 1$); and for other values of x, $\dfrac{x}{f(f(x))} = 10$ (for example, if $x = 30$, $f(f(x)) = 3$, $\dfrac{x}{f(f(x))} = 10$). So, there are two different values of $g(x)$.

B. Jumps

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

You are standing on the OX-axis at point 0 and you want to move to an integer point $x > 0$.

You can make several jumps. Suppose you're currently at point y (y may be negative) and jump for the k-th time. You can:

- either jump to the point $y + k$
- or jump to the point $y - 1$.

What is the minimum number of jumps you need to reach the point x?

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

The first and only line of each test case contains the single integer x ($1 \le x \le 10^6$) — the destination point.

Output
For each test case, print the single integer — the minimum number of jumps to reach x. It can be proved that we can reach any integer point x.

Example

input
5 1 2 3 4 5
output
1 3 2 3 4

Note
In the first test case $x = 1$, so you need only one jump: the 1-st jump from 0 to $0 + 1 = 1$.

In the second test case $x = 2$. You need at least three jumps:

- the 1-st jump from 0 to $0 + 1 = 1$;
- the 2-nd jump from 1 to $1 + 2 = 3$;
- the 3-rd jump from 3 to $3 - 1 = 2$;

Two jumps are not enough because these are the only possible variants:

- the 1-st jump as -1 and the 2-nd one as -1 — you'll reach $0 - 1 - 1 = -2$;
- the 1-st jump as -1 and the 2-nd one as $+2$ — you'll reach $0 - 1 + 2 = 1$;
- the 1-st jump as $+1$ and the 2-nd one as -1 — you'll reach $0 + 1 - 1 = 0$;
- the 1-st jump as $+1$ and the 2-nd one as $+2$ — you'll reach $0 + 1 + 2 = 3$;

In the third test case, you need two jumps: the 1-st one as $+1$ and the 2-nd one as $+2$, so $0 + 1 + 2 = 3$.

In the fourth test case, you need three jumps: the 1-st one as -1, the 2-nd one as $+2$ and the 3-rd one as $+3$, so $0 - 1 + 2 + 3 = 4$.

C. Ping-pong

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

Alice and Bob play ping-pong with simplified rules.

During the game, the player serving the ball commences a play. The server strikes the ball then the receiver makes a return by hitting the ball back. Thereafter, the server and receiver must alternately make a return until one of them doesn't make a return.

The one who doesn't make a return loses this play. The winner of the play commences the next play. Alice starts the first play.

Alice has x stamina and Bob has y. To hit the ball (while serving or returning) each player spends 1 stamina, so if they don't have any stamina, they can't return the ball (and lose the play) or can't serve the ball (in this case, the other player serves the ball instead). If both players run out of stamina, the game is over.

Sometimes, it's strategically optimal not to return the ball, lose the current play, but save the stamina. On the contrary, when the server commences a play, they have to hit the ball, if they have some stamina left.

Both Alice and Bob play optimally and want to, firstly, maximize their number of wins and, secondly, minimize the number of wins of their opponent.

Calculate the resulting number of Alice's and Bob's wins.

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

The first and only line of each test case contains two integers x and y ($1 \le x, y \le 10^6$) — Alice's and Bob's initial stamina.

Output
For each test case, print two integers — the resulting number of Alice's and Bob's wins, if both of them play optimally.

Example

input
3 1 1 2 1 1 7

output
0 1 1 1 0 7

Note

In the first test case, Alice serves the ball and spends 1 stamina. Then Bob returns the ball and also spends 1 stamina. Alice can't return the ball since she has no stamina left and loses the play. Both of them ran out of stamina, so the game is over with 0 Alice's wins and 1 Bob's wins.

In the second test case, Alice serves the ball and spends 1 stamina. Bob decides not to return the ball — he loses the play but saves stamina. Alice, as the winner of the last play, serves the ball in the next play and spends 1 more stamina. This time, Bob returns the ball and spends 1 stamina. Alice doesn't have any stamina left, so she can't return the ball and loses the play. Both of them ran out of stamina, so the game is over with 1 Alice's and 1 Bob's win.

In the third test case, Alice serves the ball and spends 1 stamina. Bob returns the ball and spends 1 stamina. Alice ran out of stamina, so she can't return the ball and loses the play. Bob, as a winner, serves the ball in the next 6 plays. Each time Alice can't return the ball and loses each play. The game is over with 0 Alice's and 7 Bob's wins.

D. Sequence and Swaps

time limit per test: 1.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a sequence a consisting of n integers $a_1, a_2, \ldots, a_n$, and an integer x. Your task is to make the sequence a sorted (it is considered sorted if the condition $a_1 \le a_2 \le a_3 \le \cdots \le a_n$ holds).

To make the sequence sorted, you may perform the following operation any number of times you want (possibly zero): choose an integer i such that $1 \le i \le n$ and $a_i > x$, and swap the values of a_i and x.

For example, if $a = [0, 2, 3, 5, 4]$, $x = 1$, the following sequence of operations is possible:

1. choose $i = 2$ (it is possible since $a_2 > x$), then $a = [0, 1, 3, 5, 4]$, $x = 2$;
2. choose $i = 3$ (it is possible since $a_3 > x$), then $a = [0, 1, 2, 5, 4]$, $x = 3$;
3. choose $i = 4$ (it is possible since $a_4 > x$), then $a = [0, 1, 2, 3, 4]$, $x = 5$.

Calculate the minimum number of operations you have to perform so that a becomes sorted, or report that it is impossible.

Input

The first line contains one integer t ($1 \le t \le 500$) — the number of test cases.

Each test case consists of two lines. The first line contains two integers n and x ($1 \le n \le 500$, $0 \le x \le 500$) — the number of elements in the sequence and the initial value of x.

The second line contains n integers a_1, a_2, ..., a_n ($0 \le a_i \le 500$).

The sum of values of n over all test cases in the input does not exceed 500.

Output

For each test case, print one integer — the minimum number of operations you have to perform to make a sorted, or -1, if it is impossible.

Example

input
6 4 1 2 3 5 4 5 6 1 1 3 4 4 1 10 2 2 10 11 9 2 10 12 11 5 18 81 324 218 413 324

output
3 0 0 -1 1 3

E. Four Points

time limit per test: 2 seconds
memory limit per test: 256 megabytes

input: standard input
output: standard output

You are given four different integer points p_1, p_2, p_3 and p_4 on XY grid.

In one step you can choose one of the points p_i and move it in one of four directions by one. In other words, if you have chosen point $p_i = (x, y)$ you can move it to $(x, y + 1)$, $(x, y - 1)$, $(x + 1, y)$ or $(x - 1, y)$.

Your goal to move points in such a way that they will form a square with sides parallel to OX and OY axes (a square with side 0 is allowed).

What is the minimum number of steps you need to make such a square?

Input

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

Each test case consists of four lines. Each line contains two integers x and y ($0 \le x, y \le 10^9$) — coordinates of one of the points $p_i = (x, y)$.

All points are different in one test case.

Output

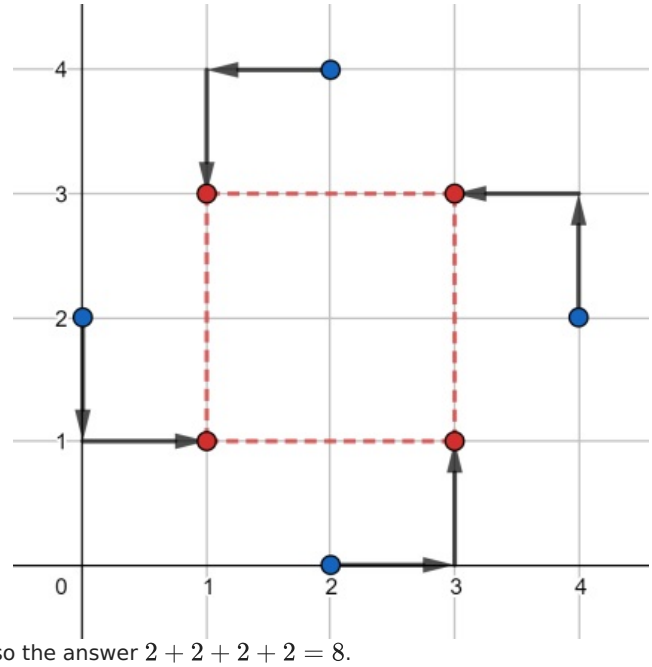
For each test case, print the single integer — the minimum number of steps to make a square.

Example

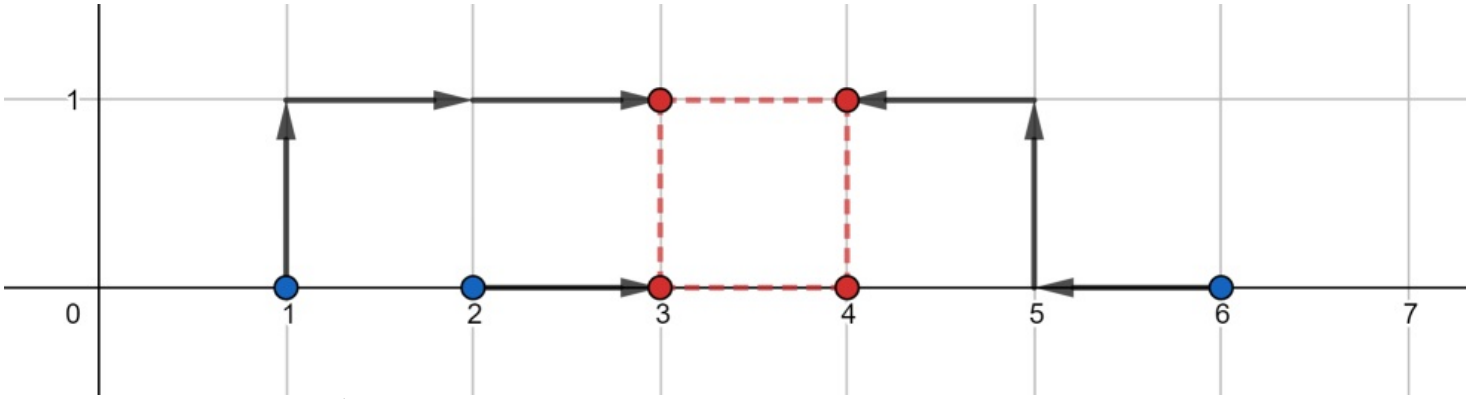
input
3 0 2 4 2 2 0 2 4 1 0 2 0 4 0 6 0 1 6 2 2 2 5 4 1
output
8 7 5

Note

In the first test case, one of the optimal solutions is shown below:

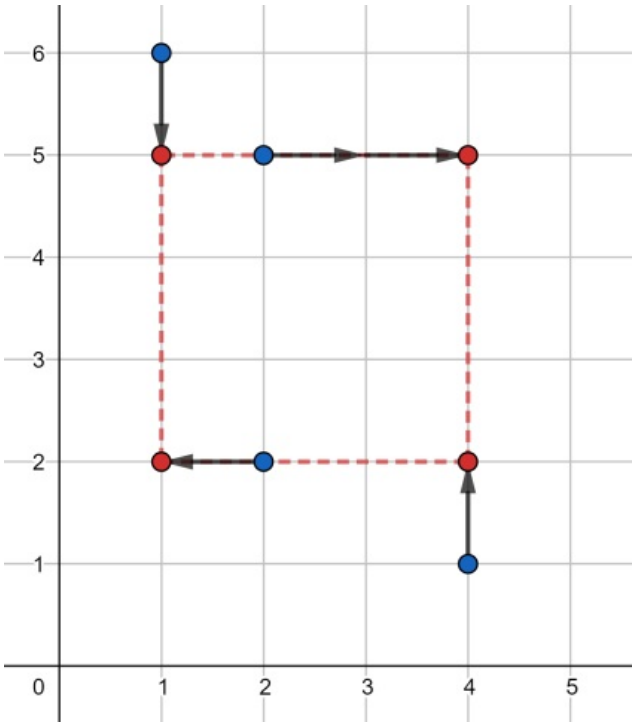


Each point was moved two times, so the answer $2 + 2 + 2 + 2 = 8$.
In the second test case, one of the optimal solutions is shown below:



The answer is $3 + 1 + 0 + 3 = 7$.

In the third test case, one of the optimal solutions is shown below:



The answer is $1 + 1 + 2 + 1 = 5$.

F. String and Operations

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

You are given a string s consisting of n characters. These characters are among the first k lowercase letters of the Latin alphabet. You have to perform n operations with the string.

During the i-th operation, you take the character that **initially occupied** the i-th position, and perform **one** of the following actions with it:

- swap it with the previous character in the string (if it exists). This operation is represented as L;
- swap it with the next character in the string (if it exists). This operation is represented as R;
- cyclically change it to the previous character in the alphabet (b becomes a, c becomes b, and so on; a becomes the k-th letter of the Latin alphabet). This operation is represented as D;
- cyclically change it to the next character in the alphabet (a becomes b, b becomes c, and so on; the k-th letter of the Latin alphabet becomes a). This operation is represented as U;
- do nothing. This operation is represented as 0.

For example, suppose the initial string is test, $k = 20$, and the sequence of operations is URLD. Then the string is transformed as follows:

1. the first operation is U, so we change the underlined letter in $\underline{\texttt{t}}\texttt{est}$ to the next one in the first 20 Latin letters, which is a. The string is now aest;
2. the second operation is R, so we swap the underlined letter with the next one in the string $\texttt{a}\underline{\texttt{e}}\texttt{st}$. The string is now aset;
3. the third operation is L, so we swap the underlined letter with the previous one in the string $\texttt{a}\underline{\texttt{s}}\texttt{et}$ (note that this is now the 2-nd character of the string, but it was initially the 3-rd one, so the 3-rd operation is performed to it). The resulting string is saet;
4. the fourth operation is D, so we change the underlined letter in $\texttt{sae}\underline{\texttt{t}}$ to the previous one in the first 20 Latin letters, which is s. The string is now saes.

The result of performing the sequence of operations is saes.

Given the string s and the value of k, find the lexicographically smallest string that can be obtained after applying a sequence of operations to s.

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

Each test case consists of two lines. The first line contains two integers n and k ($1 \le n \le 500$; $2 \le k \le 26$).

The second line contains a string s consisting of n characters. Each character is one of the k first letters of the Latin alphabet (in lower case).

Output
For each test case, print one line containing the lexicographically smallest string that can be obtained from s using one sequence of operations.

Example

input
6 4 2 bbab 7 5 cceddda 6 5 ecdaed 7 4 dcdbdaa 8 3 ccabbaca 5 7 eabba
output
aaaa baccacd aabdac aabacad aaaaaaaa abadb

G. Forbidden Value

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

Polycarp is editing a complicated computer program. First, variable x is declared and assigned to 0. Then there are instructions of two types:

1. `set` y v — assign x a value y or spend v burles to remove that instruction (thus, not reassign x);
2. `if` y `...` `end` block — execute instructions inside the `if` block if the value of x is y and ignore the block otherwise.

`if` blocks can contain `set` instructions and other `if` blocks inside them.

However, when the value of x gets assigned to s, the computer breaks and immediately catches fire. Polycarp wants to prevent that from happening and spend as few burles as possible.

What is the minimum amount of burles he can spend on removing `set` instructions to never assign x to s?

Input
The first line contains two integers n and s ($1 \le n \le 2 \cdot 10^5$, $1 \le s \le 2 \cdot 10^5$) — the number of lines in the program and the forbidden value of x.

The following n lines describe the program. Each line is one of three types:

1. `set` y v ($0 \le y \le 2 \cdot 10^5, 1 \le v \le 10^9$);
2. `if` y ($0 \le y \le 2 \cdot 10^5$);
3. `end`.

Each `if` instruction is matched by an `end` instruction. Each `end` instruction has an `if` instruction to match.

Output
Print a single integer — the minimum amount of burles Polycarp can spend on removing `set` instructions to never assign x to s.

Examples

input
5 1 set 1 10 set 2 15 if 2 set 1 7 end

output

```
17
```

input

```
7 2
set 3 4
if 3
set 10 4
set 2 7
set 10 1
end
set 4 2
```

output

```
4
```

input

```
9 200
if 0
set 5 5
if 5
set 100 13
end
if 100
set 200 1
end
end
```

output

```
1
```

input

```
1 10
set 1 15
```

output

```
0
```