Codeforces Round #719 (Div. 3)

A. Do Not Be Distracted!

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

Polycarp has 26 tasks. Each task is designated by a capital letter of the Latin alphabet.

The teacher asked Polycarp to solve tasks in the following way: if Polycarp began to solve some task, then he must solve it to the end, without being distracted by another task. After switching to another task, Polycarp cannot return to the previous task.

Polycarp can only solve one task during the day. Every day he wrote down what task he solved. Now the teacher wants to know if Polycarp followed his advice.

For example, if Polycarp solved tasks in the following order: "DDBBCCCBBEZ", then the teacher will see that on the third day Polycarp began to solve the task 'B', then on the fifth day he got distracted and began to solve the task 'C', on the eighth day Polycarp returned to the task 'B'. Other examples of when the teacher is suspicious: "BAB", "AABBCCDDEEBZZ" and "AAAAZAAAAA".

If Polycarp solved the tasks as follows: "FFGZZZY", then the teacher cannot have any suspicions. Please note that Polycarp is not obligated to solve all tasks. Other examples of when the teacher doesn't have any suspicious: "BA", "AFFFCC" and "YYYYY".

Help Polycarp find out if his teacher might be suspicious.

Input

The first line contains an integer t ($1 \le t \le 1000$). Then t test cases follow.

The first line of each test case contains one integer n ($1 \le n \le 50$) — the number of days during which Polycarp solved tasks.

The second line contains a string of length n, consisting of uppercase Latin letters, which is the order in which Polycarp solved the tasks.

Output

For each test case output:

- "YES", if the teacher **cannot be suspicious**;
- "NO", otherwise.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES are all recognized as positive answer).

Example

input

```
5
3
ABA
11
DDBBCCCBBEZ
7
FFGZZZY
1
Z
2
AB
```

output

```
NO
NO
YES
YES
YES
```

B. Ordinary Numbers

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

Let's call a positive integer n ordinary if in the decimal notation all its digits are the same. For example, 1, 2 and 99 are ordinary numbers, but 719 and 2021 are not ordinary numbers.

For a given number n, find the number of ordinary numbers among the numbers from 1 to n.

Input

The first line contains one integer t ($1 \le t \le 10^4$). Then t test cases follow.

Each test case is characterized by one integer n ($1 \le n \le 10^9$).

Output

For each test case output the number of ordinary numbers among numbers from 1 to n.

Example

input

```
6
1
2
3
4
5
100
```

output

```
1
2
3
4
5
18
```

C. Not Adjacent Matrix

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

We will consider the numbers a and b as adjacent if they differ by exactly one, that is, $|a-b|=1$.

We will consider cells of a square matrix $n \times n$ as adjacent if they have a common side, that is, for cell (r,c) cells $(r,c-1)$, $(r,c+1)$, $(r-1,c)$ and $(r+1,c)$ are adjacent to it.

For a given number n, construct a square matrix $n \times n$ such that:

- Each integer from 1 to n^2 occurs in this matrix exactly once;
- If (r_1,c_1) and (r_2,c_2) are adjacent cells, then the numbers written in them **must not be adjacent**.

Input

The first line contains one integer t ($1 \le t \le 100$). Then t test cases follow.

Each test case is characterized by one integer n ($1 \le n \le 100$).

Output

For each test case, output:

- -1, if the required matrix does not exist;
- the required matrix, otherwise (any such matrix if many of them exist).

The matrix should be outputted as n lines, where each line contains n integers.

Example

input

```
3
1
2
3
```

output

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

D. Same Differences

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

You are given an array a of n integers. Count the number of pairs of indices (i, j) such that $i < j$ and $a_j - a_i = j - i$.

Input

The first line contains one integer t ($1 \le t \le 10^4$). Then t test cases follow.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — array a.

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

Output

For each test case output the number of pairs of indices (i, j) such that $i < j$ and $a_j - a_i = j - i$.

Example

input

```
4
6
3 5 1 4 6 6
3
1 2 3
4
1 3 3 4
6
1 6 3 4 5 6
```

output

```
1
3
3
10
```

E. Arranging The Sheep

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

You are playing the game "Arranging The Sheep". The goal of this game is to make the sheep line up. The level in the game is described by a string of length n, consisting of the characters '`.`' (empty space) and '*' (sheep). In one move, you can move any sheep one square to the left or one square to the right, if the corresponding square **exists and is empty**. The game ends as soon as the sheep are lined up, that is, there should be no empty cells between any sheep.

For example, if $n = 6$ and the level is described by the string "`**.*..`", then the following game scenario is possible:

- the sheep at the 4 position moves to the right, the state of the level: "`**..*.`";
- the sheep at the 2 position moves to the right, the state of the level: "`*.*.*.`";
- the sheep at the 1 position moves to the right, the state of the level: "`.**.*.`";
- the sheep at the 3 position moves to the right, the state of the level: "`.*.**.`";
- the sheep at the 2 position moves to the right, the state of the level: "`..***.`";
- the sheep are lined up and the game ends.

For a given level, determine the minimum number of moves you need to make to complete the level.

Input

The first line contains one integer t ($1 \le t \le 10^4$). Then t test cases follow.

The first line of each test case contains one integer n ($1 \le n \le 10^6$).

The second line of each test case contains a string of length n, consisting of the characters '`.`' (empty space) and '*' (sheep) — the description of the level.

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

Output

For each test case output the minimum number of moves you need to make to complete the level.

Example

input

```
5
6
**.*..
5
*****
3
.*.
3
...
```

```
10
*.*...*.**
```

output

```
1
0
0
0
9
```

F1. Guess the K-th Zero (Easy version)

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

This is an interactive problem.

This is an easy version of the problem. The difference from the hard version is that in the easy version $t = 1$ and the number of queries is limited to 20.

Polycarp is playing a computer game. In this game, an array consisting of zeros and ones is hidden. Polycarp wins if he guesses the position of the k-th zero from the left t times.

Polycarp can make no more than 20 requests of the following type:

- ? l r — find out the sum of all elements in positions from l to r ($1 \le l \le r \le n$) inclusive.

In this (easy version) of the problem, this paragraph doesn't really make sense since $t = 1$ *always.* To make the game more interesting, each guessed zero turns into one and the game continues on the changed array. More formally, if the position of the k-th zero was x, then after Polycarp guesses this position, the x-th element of the array will be replaced from 0 to 1. Of course, this feature affects something only for $t > 1$.

Help Polycarp win the game.

Interaction

First, your program must read two integers n and t ($1 \le n \le 2 \cdot 10^5$, $t = 1$).

Then t lines follow, each of which contains one integer k ($1 \le k \le n$). It is guaranteed that at the moment of the request the array contains at least k zeros. In order to get the next value of k, you must output the answer for the current value of k.

After that, you can make no more than 20 requests.

Use the following format to output the answer (it is not a request, it doesn't count in 20):

- ! x — position of the k-th zero.

Positions in the array are numbered from left to right from 1 to n inclusive.

After printing t answers, your program should exit immediately.

In this task, the interactor is **not adaptive**. This means that within the same test, the hidden array and the queries **do not change**.

In case of an incorrect query, `-1` will be displayed. When this value is received, your program must immediately exit normally (for example, by calling `exit(0)`), otherwise, the testing system may issue an arbitrary verdict.

If the number of requests is exceeded, the verdict *wrong answer* will be displayed.

Your solution may get the verdict *Idleness limit exceeded* if you don't print anything or forget to flush the output buffer.

To flush the output buffer, you need to do the following immediately after the query output and the end-of-line character:

- `fflush(stdout)` or `cout.flush()` in C ++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

Hacks

Use the following format for hacks:

On the first line print the string s ($1 \le |s| \le 2 \cdot 10^5$), consisting of zeros and ones, and an integer t ($t = 1$) — hidden array and number of requests, respectively. In the next t lines output the number k ($1 \le k \le |s|$).

The hacked solution will not have direct access to the hidden array.

Example

input

```
6 1
2

2

1

1

0

0
```

output

```
? 4 6

? 1 1

? 1 2

? 2 2

? 5 5

! 5
```

Note

In the first test, the $[1, 0, 1, 1, 0, 1]$ array is hidden. In this test $k = 2$.

F2. Guess the K-th Zero (Hard version)

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

This is an interactive problem.

This is a hard version of the problem. The difference from the easy version is that in the hard version $1 \le t \le \min(n, 10^4)$ and the total number of queries is limited to $6 \cdot 10^4$.

Polycarp is playing a computer game. In this game, an array consisting of zeros and ones is hidden. Polycarp wins if he guesses the position of the k-th zero from the left t times.

Polycarp can make no more than $6 \cdot 10^4$ requests totally of the following type:

- ? l r — find out the sum of all elements in positions from l to r ($1 \le l \le r \le n$) inclusive.

To make the game more interesting, each guessed zero turns into one and the game continues on the changed array. More formally, if the position of the k-th zero was x, then after Polycarp guesses this position, the x-th element of the array will be replaced from 0 to 1.

Help Polycarp win the game.

Interaction

First, your program must read two integers n and t ($1 \le n \le 2 \cdot 10^5$, $1 \le t \le \min(n, 10^4)$).

Then t lines follow, each of which contains one integer k ($1 \le k \le n$). It is guaranteed that at the moment of the request the array contains at least k zeros. In order to get the next value of k, you must output the answer for the previous value of k.

After that, you can make no more than $6 \cdot 10^4$ requests in total.

Use the following format to output the answer (it is not a request, it doesn't count in $6 \cdot 10^4$):

- ! x — position of the k-th zero.

Positions in the array are numbered from left to right from 1 to n inclusive.

After printing t answers, your program should exit immediately.

In this task, the interactor is **not adaptive**. This means that within the same test, the hidden array and the queries **do not change**.

In case of an incorrect query, `-1` will be displayed. When this value is received, your program must immediately exit normally (for example, by calling `exit(0)`), otherwise, the testing system may issue an arbitrary verdict.

If the number of requests is exceeded, the verdict *wrong answer* will be displayed.

Your solution may get the verdict *Idleness limit exceeded* if you don't print anything or forget to flush the output buffer.

To flush the output buffer, you need to do the following immediately after the query output and the end-of-line character:

- `fflush(stdout)` or `cout.flush()` in C ++;

- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

Hacks

Use the following format for hacks:

On the first line print the string s ($1 \le |s| \le 2 \cdot 10^5$), consisting of zeros and ones, and an integer t ($1 \le t \le \min(|s|, 10^4)$) — hidden array and number of requests, respectively. In the next t lines output the number k ($1 \le k \le |s|$).

The hacked solution will not have direct access to the hidden array.

Example

input

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

output

```
? 4 6
? 1 1
? 1 2
? 5 5
! 5
? 2 2
! 2
```

Note

In the first test, the array $[1, 0, 1, 1, 0, 1]$ is hidden. After answering the query $k = 2$, the array changed to $[1, 0, 1, 1, 1, 1]$.

G. To Go Or Not To Go?

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

Dima overslept the alarm clock, which was supposed to raise him to school.

Dima wonders if he will have time to come to the first lesson. To do this, he needs to know the **minimum time** it will take him to get from home to school.

The city where Dima lives is a rectangular field of $n \times m$ size. Each cell (i, j) on this field is denoted by one number a_{ij}:

- The number -1 means that the passage through the cell is prohibited;
- The number 0 means that the cell is free and Dima can walk though it.
- The number x ($1 \le x \le 10^9$) means that the cell contains a portal with a cost of x. A cell with a portal is also considered free.

From any portal, Dima can go to any other portal, while the time of moving from the portal (i, j) to the portal (x, y) corresponds to the sum of their costs $a_{ij} + a_{xy}$.

In addition to moving between portals, Dima can also move between unoccupied cells adjacent to one side in time w. In particular, he can enter a cell with a portal and not use it.

Initially, Dima is in the upper-left cell $(1, 1)$, and the school is in the lower right cell (n, m).

Input

The first line contains three integers n, m and w ($2 \le n, m \le 2 \cdot 10^3$, $1 \le w \le 10^9$), where n and m are city size, w is time during which Dima moves between unoccupied cells.

The next n lines each contain m numbers ($-1 \le a_{ij} \le 10^9$) — descriptions of cells.

It is guaranteed that the cells $(1, 1)$ and (n, m) are free.

Output

Output the minimum time it will take for Dima to get to school. If he cannot get to school at all, then output "-1".

Example

input

```
5 5 1
0 -1 0 1 -1
0 20 0 0 -1
-1 -1 -1 -1 -1
3 0 0 0 0
-1 0 0 0 0
```

output

```
14
```

Note

Explanation for the first sample: