Codeforces Round 1046 (Div. 2)

A. In the Dream

1 second, 256 megabytes

Two football teams, the RiOI team and the KDOI team, are about to have a football match. A football match consists of two halves — the first half and the second half. At the beginning of the match, both teams have a score of 0.

As a fan of both teams, Aquawave knows that the two teams have similar levels, so neither team will score **three consecutive** goals in the **same half**.

Aquawave had a dream the night before the match, in which:

- The score at the end of the first half was $a:b$, where a is the score of the RiOI team, and b is the score of the KDOI team;
- And, the score at the end of the second half was $c:d$, where c is the score of the RiOI team, and d is the score of the KDOI team.

You have to determine whether Aquawave's dream can come true according to the above information.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

The only line of each test case contains four integers a, b, c, and d ($0 \le a \le c \le 100, 0 \le b \le d \le 100$) — the score at the end of the first half and the score at the end of the second half.

Output

For each test case, print "`YES`" if Aquawave's dream can come true. Otherwise, print "`NO`".

You can output the answer in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "`YES`" will be recognized as positive responses.

input

```
11
1 4 1 4
4 1 4 1
1 4 2 5
0 100 0 100
1 4 2 9
3 1 13 5
8 11 17 36
19 41 30 50
20 38 30 60
0 0 0 0
100 100 100 100
```

output

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

We will use $\texttt{R}$ to represent the RiOI team's goal, and $\texttt{K}$ to represent the KDOI team's goal.

In the first test case, the only goal order is:

- First half: $\texttt{KKRKK}$. At the end of the first half, the score is $1:4$;
- Second half: No goals. At the end of the second half, the score is still $1:4$.

In the second test case, the only goal order is:

- First half: $\texttt{RRKRR}$. At the end of the first half, the score is $4:1$;
- Second half: No goals. At the end of the second half, the score is still $4:1$.

In the third test case, one possible goal order is:

- First half: $\texttt{KKRKK}$. At the end of the first half, the score is $1:4$;
- Second half: $\texttt{KR}$. At the end of the second half, the score is $2:5$.

In the fourth test case, at the end of the first half, the KDOI team has scored 100 goals, while the RiOI team did not score any goals. Thus, in Aquawave's dream, the KDOI team scored 100 consecutive goals in the first half, which is impossible.

B. Like the Bitset

1 second, 256 megabytes

You are given a binary string* s of length n, as well as an integer k.

Aquawave wants to construct a permutation† p of length n, so that for each $1 \le i \le n$, where $s_i = 1$, the following holds:

- For each interval $[l, r]$ ($1 \le l \le r \le n$) whose length is at least k (i.e. $r - l + 1 \ge k$), if it covers position i (i.e. $l \le i \le r$), then the maximum element among $p_l, p_{l+1}, \ldots, p_r$ is **not** p_i.

Note that there are **no** such constraints on indices with $s_i = 0$.

You have to find such a permutation, or determine that such permutations do not exist.

* A binary string is a string where each character is either 0 or 1.

† A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2,3,1,5,4]$ is a permutation, but $[1,2,2]$ is not a permutation (2 appears twice in the array), and $[1,3,4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le n \le 2 \cdot 10^5, 1 \le k \le n$) — the length of s and the integer in the statements.

The second line contains the binary string s of length n ($s_i =$ 0 or 1).

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

Output

For each test case:

- If there is at least one possible permutation:
  - Print "`YES`" in the first line of output;
  - Then, print n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$, all p_i-s are distinct) in the second line — the permutation you constructed.
- Otherwise, print "`NO`" in the single line of output.

You can output the tokens in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "`YES`" will be recognized as positive responses.

If there are multiple answers, you may output any of them.

input

```
6
2 1
00
4 3
0010
5 2
11011
7 5
1111110
8 4
00101011
10 2
1000000010
```

output

```
YES
1 2
YES
1 4 3 2
NO
NO
YES
6 5 2 3 4 8 1 7
YES
1 2 3 4 5 6 7 9 8 10
```

In the first test case, you can output an arbitrary permutation of length $n = 2$, since all s_i-s are equal to 0.

In the second test case, $p = [1, 4, 3, 2]$ is a valid answer because:

- The only position where $s_i = 1$ is $i = 3$. There are three distinct intervals $[l, r]$ covering index 3, whose length is at least $k = 3$: $[1, 3]$, $[1, 4]$, and $[2, 4]$;
- And, for each of the three intervals, the maximum element among $p_l, \ldots, p_r$ should be $p_2 = 4$, which is not equal to $p_3 = 3$.

C. Against the Difference

2 seconds, 256 megabytes

We define that a *block* is an array where all elements in it are equal to the length of the array. For example, $[3, 3, 3]$, $[1]$, and $[4, 4, 4, 4]$ are *blocks*, while $[1, 1, 1]$ and $[2, 3, 3]$ are not.

An array is called *neat* if it can be obtained by the concatenation of an arbitrary number of *blocks* (possibly zero). Note that an empty array is always *neat*.

You are given an array a consisting of n integers. Find the length of its longest *neat* subsequence*.

*A sequence c is a subsequence of a sequence a if c can be obtained from a by the deletion of several (possibly, zero or all) element from arbitrary positions.

Input

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 10^4)$. The description of the test cases follows.

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

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$ — the elements of 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 a single integer — the length of the longest *neat* subsequence of a.

input

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

output

```
1
2
4
5
0
5
```

In the first test case, the whole array $[1]$ is *neat* because it is a *block*.

In the second test case, the whole array $[2, 2]$ is *neat* because it is a *block*.

In the third test case, the whole array $[2, 2, 1, 1]$ is *neat* because it is the concatenation of three *blocks*: $[2, 2]$, $[1]$, and $[1]$.

In the fourth test case, one of the longest *neat* subsequences of a is $[1, 3, 3, 3, 1]$.

In the fifth test case, the longest *neat* subsequence of a is the empty subsequence.

D. For the Champion

2 seconds, 256 megabytes

This is an interactive problem.

The RiOI team is hosting a robot championship!

This time, your robot is teleported into an **infinite** 2D plane with the Cartesian coordinate system on it. There are n anchor points on the plane, and the coordinates of the i-th anchor point are (x_i, y_i) ($-10^9 \le x_i, y_i \le 10^9$). These are given to your robot by the jury as soon as it is teleported into the plane. However, your robot doesn't know its initial coordinates at first.

To test the IQ of your robot, the RiOI team has come up with an interesting game. Your robot needs to find out the initial coordinates (X, Y) ($-10^9 \le X, Y \le 10^9$) by making the following moves.

In one move, assuming that its current coordinates are (a, b), your robot can choose a non-negative integer k $(0 \le k \le 10^9)$ and do one of the following four types of operations:

- Move up by k units, i.e., your robot will move to $(a, b + k)$;
- Move down by k units, i.e., your robot will move to $(a, b - k)$;
- Move left by k units, i.e., your robot will move to $(a - k, b)$;
- Move right by k units, i.e., your robot will move to $(a + k, b)$.

After each move, the jury will give the minimum Manhattan Distance between the current coordinates of your robot and any anchor point. More formally, assuming that the coordinates of your robot are (c, d) after the move, the jury will output

$$\min_{1 \le i \le n} (|x_i - c| + |y_i - d|).$$

To win the prize, you must prove that your robot has a high IQ. So you have to write a program for your robot to find its **initial** coordinates (X, Y) in no more than 10 moves.

Input

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 100)$. The description of the test cases follows.

The first line of each test case contains a single integer n $(1 \le n \le 100)$ — the number of anchor points.

Then n lines follow, the i-th line contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) — the coordinates of the i-th anchor point.

It is guaranteed that the coordinates of the anchor points are pairwise distinct.

Interaction

For each test case, first read the integer n and the coordinates of anchor points. Then you may make up to 10 moves.

To make a move, you should print a new line in one of the following formats:

- `? U` k — move up by k units;
- `? D` k — move down by k units;
- `? L` k — move left by k units;
- `? R` k — move right by k units.

You need to guarantee that $0 \le k \le 10^9$.

At the end of each move, the jury will print an integer s — the minimum Manhattan Distance between the current coordinates of your robot and any anchor point.

To report that you have found the initial coordinates of your robot, print a new line in the following format:

- `!` X Y — the initial coordinates of your robot are (X, Y).

Printing the answer does not count as one of the 10 moves.

After that, proceed to process the next test case or terminate the program if it was the last test case.

Assume that the initial coordinates of your robot are (X, Y). It is guaranteed that $-10^9 \le X, Y \le 10^9$.

After printing each query do not forget to output the end of line and flush[*] the output. Otherwise, you will get `Idleness limit exceeded` verdict.

If, at any interaction step, you read -1 instead of valid data, your solution must exit immediately. This means that your solution will receive `Wrong answer` because of an invalid query or any other mistake. Failing to exit can result in an arbitrary verdict because your solution will continue to read from a closed stream.

The interactor in this problem is **not** adaptive. In other words, the initial coordinates of the robot will not change during the interaction steps.

Hacks

To perform a hack, use the following format:

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

The first line of each test case contains a single integer n $(1 \le n \le 100)$ — the number of anchor points.

Then n lines follow, the i-th line contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) — the coordinates of the i-th anchor point.

In the next line print two integers X and Y $(-10^9 \le X, Y \le 10^9)$ — the initial coordinates of the robot.

[*] To flush, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `sys.stdout.flush()` in Python;
- see the documentation for other languages.

input

```
2
1
0 0

100

1

4
1 1
2 2
3 3
-1 -1

1

2

0
```

output

```



? D 99

? L 101

! 100 99





? L 0

? U 1

? R 2

! -1 0
```

Here is the progress of the example:

Solution	Jury	Explanation
	`2`	There are 2 test cases in this test.
	`1`	There is 1 anchor point in this test case.
	`0 0`	The coordinates of the only anchor point are $(0, 0)$.
		The jury chooses $(100, 99)$ as the robot's initial coordinates.
`? D 99`	`100`	The robot moves down by 99 units, and its current coordinates are $(100, 0)$. Then, the jury prints $\lvert 100 - 0\rvert + \lvert 0 - 0\rvert = 100$ as the response to this move.
`? L 101`	`1`	The robot moves left by 101 units, and its current coordinates are $(-1, 0)$. Then, the jury prints $\lvert(-1) - 0\rvert + \lvert 0 - 0\rvert = 1$ as the response to this move.
`! 100 99`		The robot has somehow determined its initial coordinates and reports the answer. Since the output is correct, the jury continues to the next test case.
	`4`	There are 4 anchor points in this test case.
	`1 1`	The coordinates of the first anchor point are $(1, 1)$.
	`2 2`	The coordinates of the second anchor point are $(2, 2)$.
	`3 3`	The coordinates of the third anchor point are $(3, 3)$.
	`-1 -1`	The coordinates of the fourth anchor point are $(-1, -1)$.
		The jury chooses $(-1, 0)$ as the robot's initial coordinates.
`? L 0`	`1`	The robot moves left by 0 units, i.e., stays at the same position. Then, the jury prints $\lvert(-1) - (-1)\rvert + \lvert 0 - (-1)\rvert = 1$ as the response to this move. It can be shown that this is the minimum Manhattan Distance between the robot and any anchor point.
`? U 1`	`2`	The robot moves up by 1 unit, and its current coordinates are $(-1, 1)$. Then, the jury prints $\lvert(-1) - (-1)\rvert + \lvert 1 - (-1)\rvert = 2$ as the response to this move.
`? R 2`	`0`	The robot moves right by 2 units, and its current coordinates are $(1, 1)$. Then, the jury prints $\lvert 1 - 1\rvert + \lvert 1 - 1\rvert = 0$ as the response to this move.
`! -1 0`		The robot has somehow determined its initial coordinates and reports the answer. Since the output is correct and there are no more test cases, the jury and the solution exit.

E. By the Assignment

4 seconds, 512 megabytes

For an undirected connected graph of n vertices, where the i-th **vertex** has a weight of v_i, we define the *value* of a simple path[*] $l_1, l_2, \dots, l_m$ as $v_{l_1} \oplus v_{l_2} \oplus \dots \oplus v_{l_m}$[†]. We call the graph *balanced* if and only if:

- For every $1 \le p < q \le n$, all simple paths from p to q have the same *value*.

Aquawave has given you an undirected connected graph of n vertices and m edges, and the i-th vertex in the graph has a weight of a_i. However, some of the weights are missing, represented by -1.

Aquawave wants to assign an integer weight between 0 and $V - 1$ to each vertex with $a_i = -1$, so that the graph will be *balanced*.

You have to help Aquawave find the number of ways to assign weights to achieve the goal, modulo $998\,244\,353$.

*A simple path from c to d is a sequence of vertices $l_1, l_2, \ldots, l_m$, where $l_1 = c$, $l_m = d$, such that there is an edge between l_i and l_{i+1} for every $1 \le i \le m-1$, and there are no repeated vertices, i.e. $l_i \ne l_j$ for $1 \le i < j \le n$.

$^\dagger \oplus$ denotes the bitwise XOR operation.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains three integers n, m, and V ($2 \le n \le 2 \cdot 10^5$, $n-1 \le m \le \min\left(\frac{n(n-1)}{2}, 4 \cdot 10^5\right)$, $1 \le V \le 10^9$) — the number of vertices, the number of edges, and the upper bound of weights.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-1 \le a_i \le V-1$) — the weights of the vertices. $a_i = -1$ represents that the weight of the i-th vertex is missing.

Then m lines follow, the i-th line containing two integers u and v ($1 \le u, v \le n$) — the two vertices that the i-th edge connects.

It is guaranteed that the given graph is simple and connected.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$, and the sum of m over all test cases does not exceed $4 \cdot 10^5$.

Output

For each test case, output a single integer — the number of ways to assign weights to make the graph *balanced*, modulo $998\,244\,353$.

input

```
5
4 4 4
-1 -1 -1 -1
1 2
2 3
1 3
4 3
5 6 7
2 2 -1 2 2
1 2
1 3
1 4
2 5
3 5
4 5
7 8 9
-1 -1 -1 -1 0 -1 0
1 2
2 3
3 4
1 4
1 5
5 6
7 6
7 5
5 8 1000000000
1 2 3 4 -1
1 2
3 2
3 5
5 1
2 4
4 3
2 5
1 4
5 4 1000000000
-1 2 -1 3 -1
1 2
1 3
2 4
2 5
```

output

```
4
1
9
0
747068572
```

In the first test case, there are four possible assignments:

- $a = [0, 0, 0, 0]$;
- $a = [0, 0, 0, 1]$;
- $a = [0, 0, 0, 2]$;
- $a = [0, 0, 0, 3]$.

It can be shown that all of these assignments can make the graph *balanced*.

In the second test case, we will pick $(p, q) = (1, 5)$. The simple path $1 \to 2 \to 5$ has a *value* of $2 \oplus 2 \oplus 2 = 2$, and the simple path $1 \to 3 \to 5$ has a *value* of $2 \oplus a_3 \oplus 2 = a_3$, so the only possible value for a_3 is 2. It can be shown that $a_3 = 2$ can make the graph *balanced*.

In the fifth test case, the given graph is a tree, so there is only one simple path between any two vertices. Thus, we can assign an arbitrary value between 0 and $V-1$ to each a_i, and the answer is $1\,000\,000\,000^3 \bmod 998\,244\,353 = 747\,068\,572$.

F1. From the Unknown (Easy Version)

3 seconds, 512 megabytes

This is the easy version of the problem. The difference between the versions is that in this version, there are no constraints on the sum of the lengths of the articles over all queries. You can hack only if you solved all versions of this problem.

This is an interactive problem.

The RiOI Team has recently developed a text editor named *RiOI Editor*. The editor works with exactly one integer parameter W — the width of each line. It is known that $1 \le W \le 10^5$.

As you cannot understand the RiOI Language, from your point of view, words differ from each other only by their length. Hence, an *article* of length n is defined as a sequence a consisting of n positive integers, where a_i is the length of the i-th word in the *article*. The *RiOI Editor* displays the *article* $[a_1, a_2, \ldots, a_n]$ on screen as follows:

- If $\max(a_1, a_2, \ldots, a_n) > W$, the editor is unable to display the *article*;
- Otherwise, the editor is able to display the *article* by the following process:
  - Initially, $l = 1$, and $s = 0$. During the whole process, l always denotes the current number of lines in the editor, and s always denotes the sum of lengths of words in the last line;
  - Then, for each $1 \le i \le n$:
    - If $s + a_i \le W$, the word is inserted at the end of the current line. Thus, l remains unchanged, and s gets increased by a_i.
    - Otherwise, the word is inserted into a new line. Thus, l becomes $l+1$, and s becomes a_i.
  - The number of lines needed to display the article is the final value of l.

You are very interested in the editor, so you decide to find out the value of W by inputting some *articles* into the editor and observing the number of lines needed to display each *article*.

Formally, you can query the jury **at most 2 times**. In each query, you input an *article* $[a_1, a_2, \ldots, a_n]$ ($1 \le n \le 10^5$) to the editor, and the jury will respond to you with:

- The number of lines needed to display the *article*, if the editor is able to display it;
- 0, if the editor is unable to display the *article*.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10$). The description of the test cases follows.

Interaction

For each test case, you can make up to 2 queries to find out the value of W. It is guaranteed that $1 \le W \le 10^5$.

To make a query, you should print a new line in the following format:

- `?` n a_1 a_2 $\ldots$ a_n ($1 \le n, a_i \le 10^5$) — the *article* you input to the editor.

At the end of each query, the jury will print an integer, as described in the statements.

To report that you have found the value of W, print a new line in the following format:

- `!` W — the parameter of the editor.

Printing the answer does not count as one of the 2 queries.

After that, proceed to process the next test case or terminate the program if it was the last test case.

After printing each query do not forget to output the end of line and flush* the output. Otherwise, you will get `Idleness limit exceeded` verdict.

If, at any interaction step, you read -1 instead of valid data, your solution must exit immediately. This means that your solution will receive `Wrong answer` because of an invalid query or any other mistake. Failing to exit can result in an arbitrary verdict because your solution will continue to read from a closed stream.

The interactor in this problem is **adaptive**. It means that the value of W may change during the interaction, but it is guaranteed that there is always at least one integer W satisfying all the queries.

Hacks

To perform a hack, only the following format is available:

The first line of the input contains an integer t $(1 \le t \le 10)$ — the number of test cases. Then, output a string "`manual`" in the same line.

The only line of each test case contains a single integer W ($1 \le W \le 10^5$) — the parameter of the editor.

The hacking format for the example is as follows.

```
2 manual
20
1
```

* To flush, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `sys.stdout.flush()` in Python;
- see the documentation for other languages.

input

```
2

2

1


0
```

output

```

? 5 1 9 4 6 1

? 2 10 10

! 20
? 1 2

! 1
```

In the first test case:

- For the first query, the total length of the words is $1+9+4+6+1=21$, and the *article* is displayed in two lines, so $W < 21$;
- For the second query, the total length of the words is $10+10=20$, and the *article* is displayed in one line, so $W \ge 20$.

Thus, we have determined $W = 20$.

In the second test case, the editor cannot display the *article* in the only query. Thus, $W < 2$, so it can only be 1.

F2. From the Unknown (Hard Version)

3 seconds, 512 megabytes

This is the hard version of the problem. The difference between the versions is that in this version, the sum of the lengths of the articles over all queries must not exceed $2.5 \cdot 10^4$. You can hack only if you solved all versions of this problem.

This is an interactive problem.

The RiOI Team has recently developed a text editor named *RiOI Editor*. The editor works with exactly one integer parameter W — the width of each line. It is known that $1 \le W \le 10^5$.

As you cannot understand the RiOI Language, from your point of view, words differ from each other only by their length. Hence, an *article* of length n is defined as a sequence a consisting of n positive integers, where a_i is the length of the i-th word in the *article*. The *RiOI Editor* displays the *article* $[a_1, a_2, \ldots, a_n]$ on screen as follows:

- If $\max(a_1, a_2, \ldots, a_n) > W$, the editor is unable to display the *article*;
- Otherwise, the editor is able to display the *article* by the following process:
  - Initially, $l = 1$, and $s = 0$. During the whole process, l always denotes the current number of lines in the editor, and s always denotes the sum of lengths of words in the last line;
  - Then, for each $1 \le i \le n$:
    - If $s + a_i \le W$, the word is inserted at the end of the current line. Thus, l remains unchanged, and s gets increased by a_i.
    - Otherwise, the word is inserted into a new line. Thus, l becomes $l+1$, and s becomes a_i.
  - The number of lines needed to display the article is the final value of l.

You are very interested in the editor, so you decide to find out the value of W by inputting some *articles* into the editor and observing the number of lines needed to display each *article*.

Formally, you can query the jury **at most 2 times**. In each query, you input an *article* $[a_1, a_2, \ldots, a_n]$ $(1 \le n \le 10^5)$ to the editor, and the jury will respond to you with:

- The number of lines needed to display the *article*, if the editor is able to display it;
- 0, if the editor is unable to display the *article*.

Additional constraint in this version: the sum of the lengths of the *articles* (i.e., n) over all queries must not exceed $2.5 \cdot 10^4$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 10)$. The description of the test cases follows.

Interaction

For each test case, you can make up to 2 queries to find out the value of W. It is guaranteed that $1 \le W \le 10^5$.

To make a query, you should print a new line in the following format:

- `?` n a_1 a_2 $\ldots$ a_n $(1 \le n, a_i \le 10^5)$ — the *article* you input to the editor.

The sum of n over all queries must not exceed $2.5 \cdot 10^4$.

At the end of each query, the jury will print an integer, as described in the statements.

To report that you have found the value of W, print a new line in the following format:

- `!` W — the parameter of the editor.

Printing the answer does not count as one of the 2 queries.

After that, proceed to process the next test case or terminate the program if it was the last test case.

After printing each query do not forget to output the end of line and flush* the output. Otherwise, you will get `Idleness limit exceeded` verdict.

If, at any interaction step, you read -1 instead of valid data, your solution must exit immediately. This means that your solution will receive `Wrong answer` because of an invalid query or any other mistake. Failing to exit can result in an arbitrary verdict because your solution will continue to read from a closed stream.

The interactor in this problem is **adaptive**. It means that the value of W may change during the interaction, but it is guaranteed that there is always at least one integer W satisfying all the queries.

Hacks

To perform a hack, only the following format is available:

The first line of the input contains an integer t $(1 \le t \le 10)$ — the number of test cases. Then, output a string "`manual`" in the same line.

The only line of each test case contains a single integer W ($1 \le W \le 10^5$) — the parameter of the editor.

The hacking format for the example is as follows.

```
2 manual
20
1
```

* To flush, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `sys.stdout.flush()` in Python;
- see the documentation for other languages.

input

```
2

2

1


0
```

output

```

? 5 1 9 4 6 1

? 2 10 10

! 20
? 1 2

! 1
```

In the first test case:

- For the first query, the total length of the words is $1+9+4+6+1=21$, and the *article* is displayed in two lines, so $W < 21$;
- For the second query, the total length of the words is $10+10=20$, and the *article* is displayed in one line, so $W \ge 20$.

Thus, we have determined $W=20$.

In the second test case, the editor cannot display the *article* in the only query. Thus, $W < 2$, so it can only be 1.