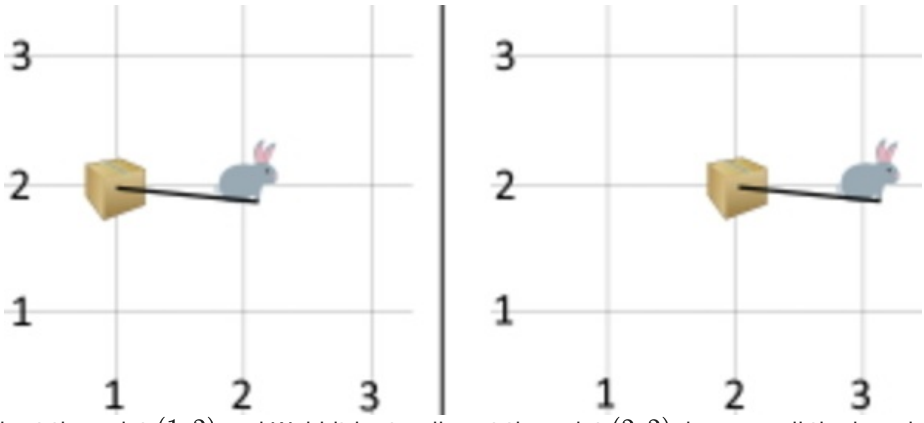
A. Box is Pull

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

Wabbit is trying to move a box containing food for the rest of the zoo in the coordinate plane from the point (x_1, y_1) to the point (x_2, y_2).

He has a rope, which he can use to pull the box. He can only pull the box if he stands **exactly** 1 unit away from the box in the direction of one of two coordinate axes. He will pull the box to where he is standing before moving out of the way in the same direction by 1 unit.



For example, if the box is at the point $(1, 2)$ and Wabbit is standing at the point $(2, 2)$, he can pull the box right by 1 unit, with the box ending up at the point $(2, 2)$ and Wabbit ending at the point $(3, 2)$.

Also, Wabbit can move 1 unit to the right, left, up, or down without pulling the box. In this case, it is not necessary for him to be in exactly 1 unit away from the box. If he wants to pull the box again, he must return to a point next to the box. Also, Wabbit can't move to the point where the box is located.

Wabbit can start at any point. It takes 1 second to travel 1 unit right, left, up, or down, regardless of whether he pulls the box while moving.

Determine the minimum amount of time he needs to move the box from (x_1, y_1) to (x_2, y_2). Note that the point where Wabbit ends up at does not matter.

Input

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

Each of the next t lines contains four space-separated integers x_1, y_1, x_2, y_2 $(1 \le x_1, y_1, x_2, y_2 \le 10^9)$, describing the next test case.

Output

For each test case, print a single integer: the minimum time in seconds Wabbit needs to bring the box from (x_1, y_1) to (x_2, y_2).

Example

input
2 1 2 2 2 1 1 2 2
output
1 4

Note

In the first test case, the starting and the ending points of the box are $(1, 2)$ and $(2, 2)$ respectively. This is the same as the picture in the statement. Wabbit needs only 1 second to move as shown in the picture in the statement.

In the second test case, Wabbit can start at the point $(2, 1)$. He pulls the box to $(2, 1)$ while moving to $(3, 1)$. He then moves to $(3, 2)$ and then to $(2, 2)$ without pulling the box. Then, he pulls the box to $(2, 2)$ while moving to $(2, 3)$. It takes 4 seconds.

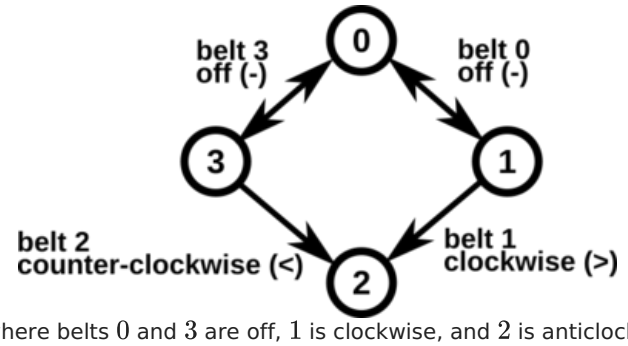
B. Belted Rooms

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

In the snake exhibition, there are n rooms (numbered 0 to $n - 1$) arranged in a circle, with a snake in each room. The rooms are connected by n conveyor belts, and the i-th conveyor belt connects the rooms i and $(i + 1) \bmod n$. In the other words, rooms 0 and 1, 1 and 2, $\ldots$, $n - 2$ and $n - 1$, $n - 1$ and 0 are connected with conveyor belts.

The i-th conveyor belt is in one of three states:

- If it is clockwise, snakes can only go from room i to $(i + 1) \bmod n$.
- If it is anticlockwise, snakes can only go from room $(i + 1) \bmod n$ to i.
- If it is off, snakes can travel in either direction.



Above is an example with 4 rooms, where belts 0 and 3 are off, 1 is clockwise, and 2 is anticlockwise.

Each snake wants to leave its room and come back to it later. A room is **returnable** if the snake there can leave the room, and later come back to it using the conveyor belts. How many such **returnable** rooms are there?

Input

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

The first line of each test case description contains a single integer n ($2 \le n \le 300\,000$): the number of rooms.

The next line of each test case description contains a string s of length n, consisting of only '<', '>' and '-'.

- If $s_i =$ '>', the i-th conveyor belt goes clockwise.
- If $s_i =$ '<', the i-th conveyor belt goes anticlockwise.
- If $s_i =$ '-', the i-th conveyor belt is off.

It is guaranteed that the sum of n among all test cases does not exceed $300\,000$.

Output

For each test case, output the number of returnable rooms.

Example

input
4 4 ->- 5 >>>>> 3 <-- 2 <>
output
3 5 3 0

Note

In the first test case, all rooms are returnable except room 2. The snake in the room 2 is trapped and cannot exit. This test case corresponds to the picture from the problem statement.

In the second test case, all rooms are returnable by traveling on the series of clockwise belts.

C. ABBB

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

output: standard output

Zookeeper is playing a game. In this game, Zookeeper must use bombs to bomb a string that consists of letters 'A' and 'B'. He can use bombs to bomb a substring which is either "AB" or "BB". When he bombs such a substring, the substring gets deleted from the string and the remaining parts of the string get concatenated.

For example, Zookeeper can use two such operations: AABABBA $\rightarrow$ AABBA $\rightarrow$ AAA.

Zookeeper wonders what the shortest string he can make is. Can you help him find the length of the shortest string?

Input

Each test contains multiple test cases. The first line contains a single integer t $(1 \leq t \leq 20000)$ — the number of test cases. The description of the test cases follows.

Each of the next t lines contains a single test case each, consisting of a non-empty string s: the string that Zookeeper needs to bomb. It is guaranteed that all symbols of s are either 'A' or 'B'.

It is guaranteed that the sum of $|s|$ (length of s) among all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer: the length of the shortest string that Zookeeper can make.

Example

input
3 AAA BABA AABBBABBBB
output
3 2 0

Note

For the first test case, you can't make any moves, so the answer is 3.

For the second test case, one optimal sequence of moves is BABA $\rightarrow$ BA. So, the answer is 2.

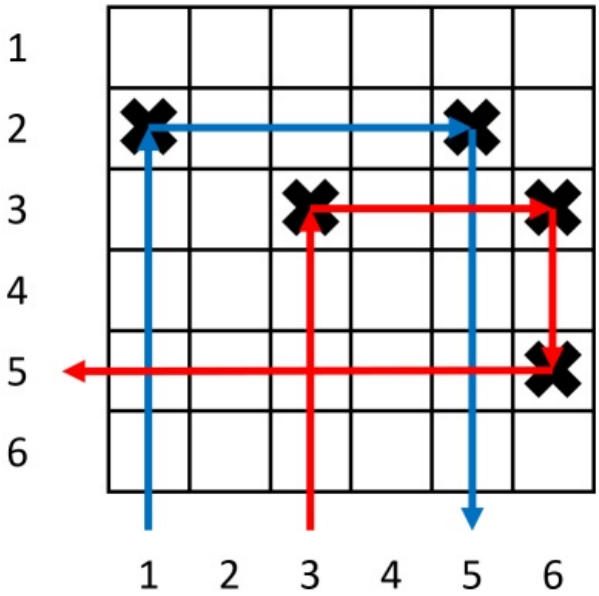
For the third test case, one optimal sequence of moves is AABBBABBBB $\rightarrow$ AABBBABB $\rightarrow$ AABBBB $\rightarrow$ ABBB $\rightarrow$ AB $\rightarrow$ (empty string). So, the answer is 0.

D. Bouncing Boomerangs

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

To improve the boomerang throwing skills of the animals, Zookeeper has set up an $n \times n$ grid with some targets, **where each row and each column has at most 2 targets each**. The rows are numbered from 1 to n from top to bottom, and the columns are numbered from 1 to n from left to right.

For each column, Zookeeper will throw a boomerang from the bottom of the column (below the grid) upwards. When the boomerang hits any target, it will bounce off, make a 90 degree turn to the right and fly off in a straight line in its new direction. The boomerang can hit multiple targets and does not stop until it leaves the grid.



In the above example, $n = 6$ and the black crosses are the targets. The boomerang in column 1 (blue arrows) bounces 2 times while the boomerang in column 3 (red arrows) bounces 3 times.

The boomerang in column i hits exactly a_i targets before flying out of the grid. **It is known that $a_i \leq 3$.**

However, Zookeeper has lost the original positions of the targets. Thus, he asks you to construct a valid configuration of targets that matches the number of hits for each column, or tell him that no such configuration exists. If multiple valid configurations exist, you may print any of them.

Input

The first line contains a single integer n $(1 \leq n \leq 10^5)$.

The next line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \leq a_i \leq 3)$.

Output

If no configuration of targets exist, print -1.

Otherwise, on the first line print a single integer t $(0 \leq t \leq 2n)$: the number of targets in your configuration.

Then print t lines with two spaced integers each per line. Each line should contain two integers r and c $(1 \leq r, c \leq n)$, where r is the target's row and c is the target's column. All targets should be different.

Every row and every column in your configuration should have at most two targets each.

Examples

input
6 2 0 3 0 1 1
output
5 2 1 2 5 3 3 3 6 5 6

input
1 0
output
0

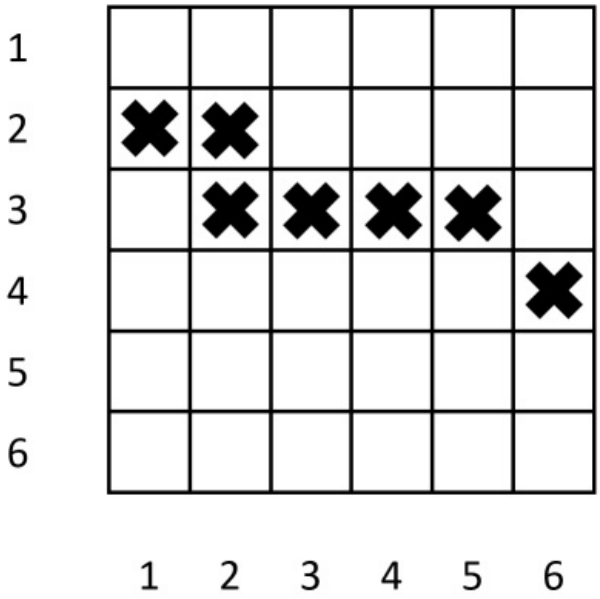
input
6 3 2 2 2 1 1
output
-1

Note

For the first test, the answer configuration is the same as in the picture from the statement.

For the second test, the boomerang is not supposed to hit anything, so we can place 0 targets.

For the third test, the following configuration of targets matches the number of hits, **but is not allowed as row 3 has 4 targets**.



It can be shown for this test case that **no valid configuration of targets** will result in the given number of target hits.

E. Carrots for Rabbits

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

There are some rabbits in Singapore Zoo. To feed them, Zookeeper bought n carrots with lengths $a_1, a_2, a_3, \ldots, a_n$. However, rabbits are very fertile and multiply very quickly. Zookeeper now has k rabbits and does not have enough carrots to feed all of them. To solve this problem, Zookeeper decided to cut the carrots into k pieces. For some reason, all resulting carrot lengths must be positive integers.

Big carrots are very difficult for rabbits to handle and eat, so the time needed to eat a carrot of size x is x^2.

Help Zookeeper split his carrots while minimizing the sum of time taken for rabbits to eat the carrots.

Input

The first line contains two integers n and k $(1 \leq n \leq k \leq 10^5)$: the initial number of carrots and the number of rabbits.

The next line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^6)$: lengths of carrots.

It is guaranteed that the sum of a_i is at least k.

Output

Output one integer: the minimum sum of time taken for rabbits to eat carrots.

Examples

input
3 6 5 3 1
output
15

input
1 4 19
output
91

Note

For the first test, the optimal sizes of carrots are $\{1, 1, 1, 2, 2, 2\}$. The time taken is $1^2 + 1^2 + 1^2 + 2^2 + 2^2 + 2^2 = 15$

For the second test, the optimal sizes of carrots are $\{4, 5, 5, 5\}$. The time taken is $4^2 + 5^2 + 5^2 + 5^2 = 91$.

F. Fruit Sequences

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

input: standard input
output: standard output

Zookeeper is buying a carton of fruit to feed his pet wabbit. The fruits are a sequence of apples and oranges, which is represented by a binary string $s_1 s_2 \ldots s_n$ of length n. 1 represents an apple and 0 represents an orange.

Since wabbit is allergic to eating oranges, Zookeeper would like to find the longest **contiguous** sequence of apples. Let $f(l, r)$ be the longest **contiguous** sequence of apples in the substring $s_l s_{l+1} \ldots s_r$.

Help Zookeeper find $\sum_{l=1}^{n} \sum_{r=l}^{n} f(l, r)$, or the sum of f across all substrings.

Input
The first line contains a single integer n $(1 \leq n \leq 5 \cdot 10^5)$.

The next line contains a binary string s of length n $(s_i \in \{0, 1\})$

Output
Print a single integer: $\sum_{l=1}^{n} \sum_{r=l}^{n} f(l, r)$.

Examples

input
4 0110
output
12

input
7 1101001
output
30

input
12 011100011100
output
156

Note
In the first test, there are ten substrings. The list of them (we let $[l, r]$ be the substring $s_l s_{l+1} \ldots s_r$):

- $[1, 1]$: 0
- $[1, 2]$: 01
- $[1, 3]$: 011
- $[1, 4]$: 0110
- $[2, 2]$: 1
- $[2, 3]$: 11
- $[2, 4]$: 110
- $[3, 3]$: 1
- $[3, 4]$: 10
- $[4, 4]$: 0

The lengths of the longest contiguous sequence of ones in each of these ten substrings are $0, 1, 2, 2, 1, 2, 2, 1, 1, 0$ respectively. Hence, the answer is $0 + 1 + 2 + 2 + 1 + 2 + 2 + 1 + 1 + 0 = 12$.

G1. Lucky Numbers (Easy Version)

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

This is the easy version of the problem. The only difference is that in this version $q = 1$. You can make hacks only if all versions of the problem are solved.

Zookeeper has been teaching his q sheep how to write and how to add. The i-th sheep has to write exactly k **non-negative integers** with the sum n_i.

Strangely, sheep have superstitions about digits and believe that the digits 3, 6, and 9 are lucky. To them, the fortune of a number depends on the decimal representation of the number; the fortune of a number is equal to the sum of fortunes of its digits, and the fortune of a digit depends on its value and position and can be described by the following table. For example, the number 319 has

fortune $F_2 + 3F_0$.

	Position					
Digit	*1*	*10*	*100*	*1000*	*10000*	*100000*
3	F_0	F_1	F_2	F_3	F_4	F_5
6	$2F_0$	$2F_1$	$2F_2$	$2F_3$	$2F_4$	$2F_5$
9	$3F_0$	$3F_1$	$3F_2$	$3F_3$	$3F_4$	$3F_5$
Other	0	0	0	0	0	0

Each sheep wants to maximize the **sum of fortune** among all its k written integers. Can you help them?

Input
The first line contains a single integer k ($1 \le k \le 999999$): the number of numbers each sheep has to write.

The next line contains six integers F_0, F_1, F_2, F_3, F_4, F_5 ($1 \le F_i \le 10^9$): the fortune assigned to each digit.

The next line contains a single integer q ($q = 1$): the number of sheep.

Each of the next q lines contains a single integer n_i ($1 \le n_i \le 999999$): the sum of numbers that i-th sheep has to write. In this version, there is only one line.

Output
Print q lines, where the i-th line contains the maximum sum of fortune of all numbers of the i-th sheep. In this version, you should print only one line.

Examples

input
3 1 2 3 4 5 6 1 57
output
11

input
3 1 2 3 4 5 6 1 63
output
8

Note
In the first test case, $57 = 9 + 9 + 39$. The three 9's contribute $1 \cdot 3$ and 3 at the tens position contributes $2 \cdot 1$. Hence the sum of fortune is 11.

In the second test case, $63 = 35 + 19 + 9$. The sum of fortune is 8.

G2. Lucky Numbers (Hard Version)

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

This is the hard version of the problem. The only difference is in the constraint on q. You can make hacks only if all versions of the problem are solved.

Zookeeper has been teaching his q sheep how to write and how to add. The i-th sheep has to write exactly k **non-negative integers** with the sum n_i.

Strangely, sheep have superstitions about digits and believe that the digits 3, 6, and 9 are lucky. To them, the fortune of a number depends on the decimal representation of the number; the fortune of a number is equal to the sum of fortunes of its digits, and the fortune of a digit depends on its value and position and can be described by the following table. For example, the number 319 has fortune $F_2 + 3F_0$.

	Position					
Digit	*1*	*10*	*100*	*1000*	*10000*	*100000*
3	F_0	F_1	F_2	F_3	F_4	F_5
6	$2F_0$	$2F_1$	$2F_2$	$2F_3$	$2F_4$	$2F_5$
9	$3F_0$	$3F_1$	$3F_2$	$3F_3$	$3F_4$	$3F_5$
Other	0	0	0	0	0	0

Each sheep wants to maximize the **sum of fortune** among all its k written integers. Can you help them?

Input
The first line contains a single integer k ($1 \le k \le 999999$): the number of numbers each sheep has to write.

The next line contains six integers F_0, F_1, F_2, F_3, F_4, F_5 ($1 \le F_i \le 10^9$): the fortune assigned to each digit.

The next line contains a single integer q ($1 \le q \le 100\,000$): the number of sheep.

Each of the next q lines contains a single integer n_i ($1 \le n_i \le 999999$): the sum of numbers that i-th sheep has to write.

Output
Print q lines, where the i-th line contains the maximum sum of fortune of all numbers of the i-th sheep.

Example

input
3 1 2 3 4 5 6 2 57 63
output
11 8

Note
In the first test case, $57 = 9 + 9 + 39$. The three 9's contribute $1 \cdot 3$ and 3 at the tens position contributes $2 \cdot 1$. Hence the sum of fortune is 11.

In the second test case, $63 = 35 + 19 + 9$. The sum of fortune is 8.

H. Rotary Laser Lock

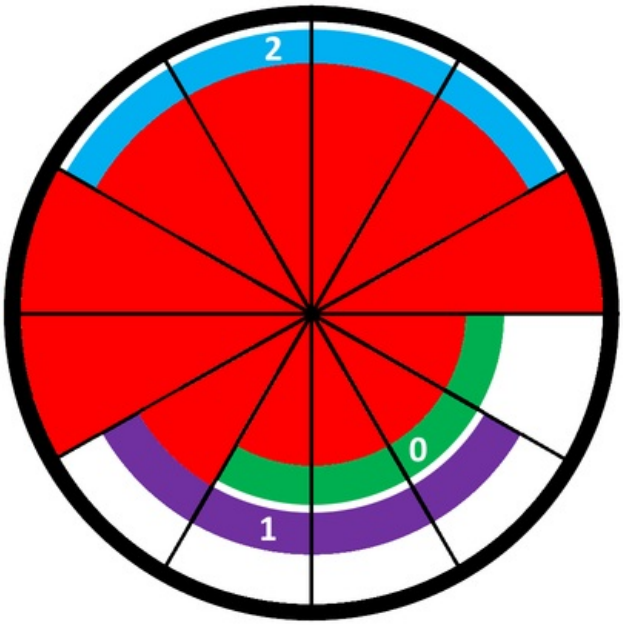
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem.

To prevent the mischievous rabbits from freely roaming around the zoo, Zookeeper has set up a special lock for the rabbit enclosure. This lock is called the Rotary Laser Lock.

The lock consists of n concentric rings numbered from 0 to $n-1$. The innermost ring is ring 0 and the outermost ring is ring $n-1$. All rings are split equally into nm sections each. Each of those rings contains a single metal arc that covers exactly m contiguous sections. At the center of the ring is a core and surrounding the entire lock are nm receivers aligned to the nm sections.

The core has nm lasers that shine outward from the center, one for each section. The lasers can be blocked by any of the arcs. A display on the outside of the lock shows how many lasers hit the outer receivers.



In the example above, there are $n = 3$ rings, each covering $m = 4$ sections. The arcs are colored in green (ring 0), purple (ring 1), and blue (ring 2) while the lasers beams are shown in red. There are $nm = 12$ sections and 3 of the lasers are not blocked by any arc, thus the display will show 3 in this case.

Wabbit is trying to open the lock to free the rabbits, but the lock is completely opaque, and he cannot see where any of the arcs are. Given the **relative positions** of the arcs, Wabbit can open the lock on his own.

To be precise, Wabbit needs $n - 1$ integers $p_1, p_2, \ldots, p_{n-1}$ satisfying $0 \leq p_i < nm$ such that for each i $(1 \leq i < n)$, Wabbit can rotate ring 0 clockwise exactly p_i times such that the sections that ring 0 covers perfectly aligns with the sections that ring i covers. In the example above, the relative positions are $p_1 = 1$ and $p_2 = 7$.

To operate the lock, he can pick any of the n rings and rotate them by 1 section either clockwise or anti-clockwise. You will see the number on the display after every rotation.

Because his paws are small, Wabbit has asked you to help him to find the **relative positions** of the arcs **after all of your rotations are completed**. You may perform up to 15000 rotations before Wabbit gets impatient.

Input

The first line consists of 2 integers n and m $(2 \leq n \leq 100, 2 \leq m \leq 20)$, indicating the number of rings and the number of sections each ring covers.

Interaction

To perform a rotation, print on a single line "`? x d`" where x $(0 \leq x < n)$ is the ring that you wish to rotate and d $(d \in \{-1, 1\})$ is the direction that you would like to rotate in. $d = 1$ indicates a clockwise rotation by 1 section while $d = -1$ indicates an anticlockwise rotation by 1 section.

For each query, you will receive a single integer a: the number of lasers that are not blocked by any of the arcs after the rotation has been performed.

Once you have figured out the relative positions of the arcs, print `!` followed by $n - 1$ integers $p_1, p_2, \ldots, p_{n-1}$.

Do note that the positions of the rings are predetermined for each test case and won't change during the interaction process.

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

To do this, use:

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

Hacks:

To hack, use the following format of test:

The first line should contain two integers n and m.

The next line of should contain $n - 1$ integers $p_1, p_2, \ldots, p_{n-1}$: relative positions of rings $1, 2, \ldots, n - 1$.

Example

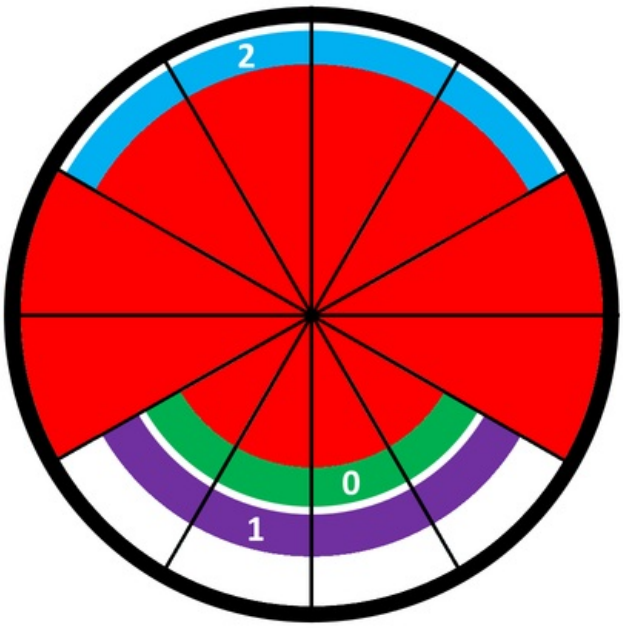
input
3 4

4 4 3
output
? 0 1 ? 2 -1 ? 1 1 ! 1 5

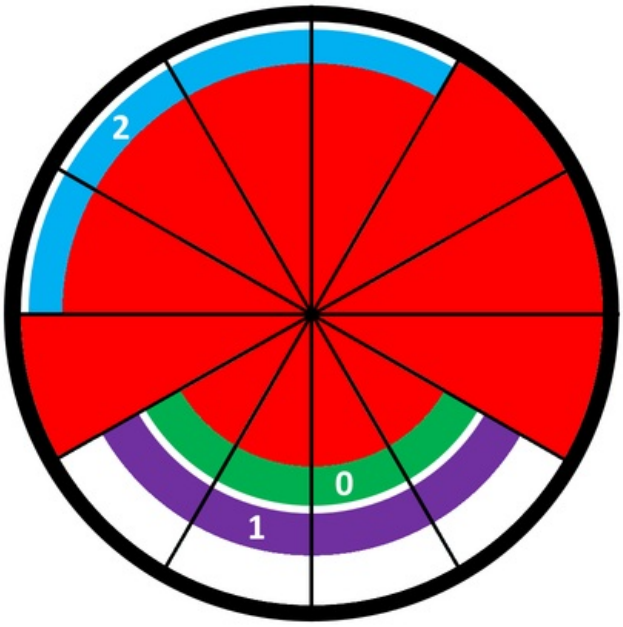
Note

For the first test, the configuration is the same as shown on the picture from the statement.

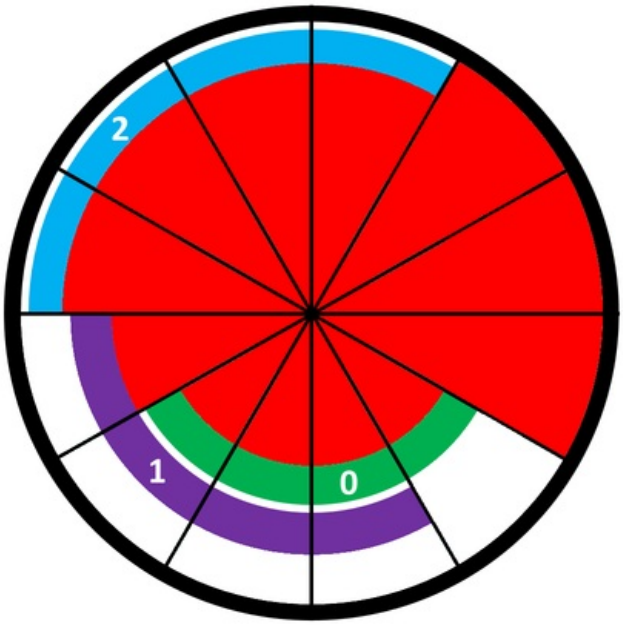
After the first rotation (which is rotating ring 0 clockwise by 1 section), we obtain the following configuration:



After the second rotation (which is rotating ring 2 counter-clockwise by 1 section), we obtain the following configuration:



After the third rotation (which is rotating ring 1 clockwise by 1 section), we obtain the following configuration:



If we rotate ring 0 clockwise once, we can see that the sections ring 0 covers will be the same as the sections that ring 1 covers, hence $p_1 = 1$.

If we rotate ring 0 clockwise five times, the sections ring 0 covers will be the same as the sections that ring 2 covers, hence $p_2 = 5$.

Note that if we will make a different set of rotations, we can end up with different values of p_1 and p_2 at the end.

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform