A. Chess For Three

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Three friends gathered to play a few games of chess together.

In every game, two of them play against each other. The winner gets 2 points while the loser gets 0, and in case of a draw, both players get 1 point each. Note that the same pair of players could have played any non-negative number of times (possibly zero). It is also possible that no games were played at all.

You've been told that their scores after all the games were played were p_1, p_2 and p_3. Additionally, it is guaranteed that $p_1 \leq p_2 \leq p_3$ holds.

Find the maximum number of draws that could have happened and print it. If there isn't any way to obtain p_1, p_2 and p_3 as a result of a non-negative number of games between the three players, print -1 instead.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 500$). The description of the test cases follows.

The first line of each test case contains three integers p_1, p_2 and p_3 ($0 \leq p_1 \leq p_2 \leq p_3 \leq 30$) — the scores of the three players, sorted non-decreasingly.

Output

For each testcase, print one number — the maximum possible number of draws that could've happened, or -1 if the scores aren't consistent with any valid set of games and results.

Standard Input	Standard Output
7 0 0 0 0 1 1 1 1 1 1 1 2 3 3 3 3 4 5 1 1 10	0 1 -1 2 -1 6 2

Note

In the first example, no games were played at all, so no draws could occur either.

For the second example, exactly one game occurred between the second and the third player and it ended in draw, so the answer is 1.

It's easy to see that there's no set of games achieving the scores in third example, so the answer for it is -1.

,

B. Cat, Fox and the Lonely Array

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Today, Cat and Fox found an array a consisting of n non-negative integers.

Define the *loneliness* of a as the **smallest** positive integer k ($1 \le k \le n$) such that for any two positive integers i and j ($1 \le i, j \le n - k + 1$), the following holds:

$$a_i|a_{i+1}|\ldots|a_{i+k-1} = a_j|a_{j+1}|\ldots|a_{j+k-1},$$

where $x|y$ denotes the bitwise OR of x and y. In other words, for every k consecutive elements, their bitwise OR should be the same. Note that the loneliness of a is well-defined, because for $k = n$ the condition is satisfied.

Cat and Fox want to know how lonely the array a is. Help them calculate the loneliness of the found array.

Input

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

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{20}$) — the elements of the array.

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

Output

For each test case, print one integer — the loneliness of the given array.

Standard Input	Standard Output
7 1 0 3 2 2 2 3 1 0 2 5 3 0 1 4 2 5 2 0 4 0 2 7 0 0 0 0 1 2 4 8 0 1 3 2 2 1 0 3	1 1 3 4 4 7 3

Note

In the first example, the loneliness of an array with a single element is always 1, so the answer is 1.

In the second example, the OR of each subarray of length $k = 1$ is 2, so the loneliness of the whole array is 1.

In the seventh example, it's true that $(0|1|3) = (1|3|2) = (3|2|2) = (2|2|1) = (2|1|0) = (1|0|3) = 3$, so the condition is satisfied for $k = 3$. We can verify that the condition is not true for any smaller k, so the answer is indeed 3.

,

C. Cat, Fox and Double Maximum

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Fox loves permutations! She came up with the following problem and asked Cat to solve it:

You are given an **even** positive integer n and a permutation[†] p of length n.

The score of another permutation q of length n is the number of **local maximums** in the array a of length n, where $a_i = p_i + q_i$ for all i ($1 \le i \le n$). In other words, the score of q is the number of i such that $1 < i < n$ (note the **strict** inequalities), $a_{i-1} < a_i$, and $a_i > a_{i+1}$ (once again, note the strict inequalities).

Find the permutation q that achieves the maximum score for given n and p. If there exist multiple such permutations, you can pick any of them.

[†] 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

The first line of input contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input you will have to solve.

The first line of each test case contains one **even** integer n ($4 \le n \le 10^5$, n is even) — the length of the permutation p.

The second line of each test case contains the n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$). It is guaranteed that p is a permutation of length n.

It is guaranteed that the sum of n across all test cases doesn't exceed 10^5.

Output

For each test case, output one line containing any permutation of length n (the array q), such that q maximizes the score under the given constraints.

Standard Input	Standard Output
4 4 1 2 3 4 4 4 3 1 2 6 6 5 1 4 2 3 8 1 2 4 5 7 6 8 3	2 4 1 3 3 1 4 2 2 5 1 4 3 6 5 4 8 2 7 1 6 3

Note

In the first example, $a = [3, 6, 4, 7]$. The array has just one local maximum (on the second position), so the score of the chosen permutation q is 1. It can be proven that this score is optimal under the constraints.

In the last example, the resulting array $a = [6, 6, 12, 7, 14, 7, 14, 6]$ has 3 local maximums — on the third, fifth and seventh positions.

,

D. Cat, Fox and Maximum Array Split

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

This is an interactive problem.

Fox gave Cat two positive integers n and k. She has a hidden array $a_1, \ldots, a_n$ of length n, such that $1 \le a_i \le n$ for every i. Now they are going to play the following game:

For any two integers l, r such that $1 \le l \le r \le n$, define $f(l, r) = (r - l + 1) \cdot \max\limits_{x=l}^{r} a_x$. In other words, $f(l, r)$ is equal to the maximum of the subarray $a_l, \ldots, a_r$ multiplied by its size.

Cat can ask Fox at most $2n$ questions about the array. He will tell her two integers l and x ($1 \le l \le n, 1 \le x \le 10^9$), and she will tell him one integer p as the answer — the smallest positive integer r such that $f(l, r) = x$, or $n + 1$ if no such r exists.

Now, Cat needs to find the largest value m such that there exists a sequence $c_1, \ldots, c_{k-1}$ such that $1 \le c_1 < \ldots < c_{k-1} < n$ and $f(1, c_1) = f(c_1 + 1, c_2) = \ldots = f(c_{k-1} + 1, n) = m$. If no such m exists, he should indicate this and take -1 as the answer. Note that for $k = 1$, m is always equal to $f(1, n)$.

In other words, the goal is to find the largest m such that you can split the array into exactly k subarrays (k is the constant given to you in the beginning of the interaction) so that all the subarrays have the product of their length and their maximum equal to m, or determine that no such m exists. Every element should belong in exactly one of the subarrays.

Cat doesn't know what he should do, so he asked you to play the game for him.

Interaction

Each test contains multiple test cases. The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two positive integers n and k ($1 \le k \le n \le 10^4$) — the length of the hidden array and the number of subarrays in the desired split.

Now you are allowed to make queries in the following way — print one line of the form "**?** l x" (it must hold that $1 \le l \le n$, $1 \le x \le 10^9$) and you will receive the smallest integer r such that $l \le r \le n$ and $f(l, r) = x$, or $n + 1$ if no such r exists.

If you want to print the answer, output "**!** m" and you will recieve 1 if your answer is correct and -1 otherwise. In the first case, the interaction continues with the next test case. Note that printing the answer doesn't count towards the number of queries made. **Please note that you don't receive the values for the next test case immediately, you will first have to read whether your answer to the last test case was correct.**

If you receive the integer -1 at any moment, it means your program has made an invalid query, exceeded the query limit, or gave an incorrect answer. Your program must terminate immediately to receive a `Wrong Answer` verdict. Otherwise, you can get an arbitrary verdict because your solution will continue to read from a closed stream.

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

It is guaranteed that the total sum of n over the test cases won't exceed 10^4.

Hacks

The format of the hacks should be the following: the first line should contain one integer t ($1 \le t \le 10^3$) — the number of test cases. The description of the test cases should follow.

The first line of each test case should contain two integers n and k ($1 \le k \le n \le 10^4$) — the length of the array a and the number of subarrays you want to split it into.

The second line should contain n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

The sum of n over all test cases should not exceed 10^4.

Standard Input	Standard Output
3 1 1 1 2 2 1 3 1 6 3 7 2 3 6 1	! 1 ? 1 1 ? 2 1 ! -1 ? 1 9 ? 1 6 ? 3 6 ? 4 6 ! 6

Note

The hidden arrays in the three testcases are $[1]$, $[1, 2]$ and $[1, 3, 6, 1, 2, 1]$. In the second testcase, no split satisfies the constraints, so the answer is -1.

The answer for the first query of the third testcase is 7 since no valid r exists. For the second query of the third testcase, since $2 \cdot \max(1, 3) = 6$, we will get 2 as the answer, since $r = 1$ doesn't satisfy the constraint.

The sample interaction guessed all three answers (1, -1 and 6) correctly, so it received 1 after each answer.

,

E. Cat, Fox and Swaps

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Fox has found an array $p_1, p_2, \ldots, p_n$, that is a permutation of length $n^\dagger$ of the numbers $1, 2, \ldots, n$. She wants to sort the elements in increasing order. Cat wants to help her — he is able to swap any two numbers x and y in the array, but only if $l \leq x + y \leq r$ (note that the constraint is imposed on the values of the elements, not their positions). He can make such swaps any number of times.

They don't know the numbers l, r yet, they only know that it's true that $1 \leq l \leq r \leq 2 \cdot n$.

You are given the number n and the array $p_1, p_2, \ldots, p_n$. Determine how many pairs (l, r) satisfying the conditions are there such that you can sort the permutation if you can only swap two number (x, y) such that $l \leq x + y \leq r$ (arbitrary number of times, possibly 0).

† 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 ($1 \leq t \leq 10^4$). The description of the test cases follows.

Description of each test case consists of two lines. The first line contains one integer n ($1 \leq n \leq 10^5$).

The second line contains n integers: the array $p_1, p_2, \ldots, p_n$ ($1 \leq p_i \leq n$). It is guaranteed that this array is a permutation of length n.

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

Output

For each test case, print the number of pairs of integers (l, r) such that $1 \leq l \leq r \leq 2 \cdot n$, and you can sort the array under the constraints.

Standard Input	Standard Output
7 2 2 1 3 3 1 2 4 3 2 1 4 5 5 3 1 2 4 5 1 2 3 4 5 6 3 2 1 5 4 6 6	6 11 23 29 55 46 58

1 3 2 4 5 6	

Note

In the first example, we need to be able to swap 1 and 2, so we must be able to swap numbers with sum 3. There are exactly 6 pairs satisfying the condition: $(1, 3), (2, 3), (3, 3), (1, 4), (2, 4)$ and $(3, 4)$, so the answer is 6.

In the second example, the 11 pairs satisfying the condition are $(1, 4), (1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6), (4, 5)$ and $(4, 6)$. For example, if we pick the pair $(3, 4)$ we can first swap the numbers 1 and 2 and then the numbers 1 and 3, after this, the permutation is sorted.

,

F. Maximum GCD Sum Queries

Input file: standard input
Output file: standard output
Time limit: 5 seconds
Memory limit: 512 megabytes

For k positive integers $x_1, x_2, \ldots, x_k$, the value $\gcd(x_1, x_2, \ldots, x_k)$ is the greatest common divisor of the integers $x_1, x_2, \ldots, x_k$ — the largest integer z such that all the integers $x_1, x_2, \ldots, x_k$ are divisible by z.

You are given three arrays $a_1, a_2, \ldots, a_n$, $b_1, b_2, \ldots, b_n$ and $c_1, c_2, \ldots, c_n$ of length n, containing positive integers.

You also have a machine that allows you to swap a_i and b_i for any i ($1 \le i \le n$). Each swap costs you c_i coins.

Find the maximum possible value of

$$\gcd(a_1, a_2, \ldots, a_n) + \gcd(b_1, b_2, \ldots, b_n)$$

that you can get by paying in total at most d coins for swapping some elements. The amount of coins you have changes a lot, so find the answer to this question for each of the q possible values $d_1, d_2, \ldots, d_q$.

Input

There are two integers on the first line — the numbers n and q ($1 \le n \le 5 \cdot 10^5$, $1 \le q \le 5 \cdot 10^5$).

On the second line, there are n integers — the numbers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^8$).

On the third line, there are n integers — the numbers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^8$).

On the fourth line, there are n integers — the numbers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$).

On the fifth line, there are q integers — the numbers $d_1, d_2, \ldots, d_q$ ($0 \le d_i \le 10^{15}$).

Output

Print q integers — the maximum value you can get for each of the q possible values d.

Standard Input	Standard Output
3 4 1 2 3 4 5 6 1 1 1 0 1 2 3	2 3 3 3
5 5 3 4 6 8 4 8 3 4 9 3 10 20 30 40 50 5 55 13 1000 113	2 7 3 7 7
1 1 3 4	7

5 0	

Note

In the first query of the first example, we are not allowed to do any swaps at all, so the answer is $\gcd(1, 2, 3) + \gcd(4, 5, 6) = 2$. In the second query, one of the ways to achieve the optimal value is to swap a_2 and b_2, then the answer is $\gcd(1, 5, 3) + \gcd(4, 2, 6) = 3$.

In the second query of the second example, it's optimal to perform swaps on positions 1 and 3, then the answer is $\gcd(3, 3, 6, 9, 3) + \gcd(8, 4, 4, 8, 4) = 7$ and we have to pay 40 coins in total.