A. Guess the Maximum

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

Alice and Bob came up with a rather strange game. They have an array of integers $a_1, a_2, \ldots, a_n$. Alice chooses a certain integer k and tells it to Bob, then the following happens:

- Bob chooses two integers i and j ($1 \le i < j \le n$), and then finds the maximum among the integers $a_i, a_{i+1}, \ldots, a_j$;
- If the obtained maximum is **strictly greater** than k, Alice wins, otherwise Bob wins.

Help Alice find the maximum k at which she is guaranteed to win.

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 a single integer n ($2 \le n \le 5 \cdot 10^4$) — the number of elements in the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the array.

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

Output

For each test case, output one integer — the maximum integer k at which Alice is guaranteed to win.

Standard Input	Standard Output
6 4 2 4 1 7 5 1 2 3 4 5 2 1 1 3 37 8 16 5 10 10 10 10 9 10 3 12 9 5 2 3 2 9 8 2	3 1 0 15 9 2

Note

In the first test case, all possible subsegments that Bob can choose look as follows: $[2, 4], [2, 4, 1], [2, 4, 1, 7], [4, 1], [4, 1, 7], [1, 7]$. The maximums on the subsegments are respectively equal to $4, 4, 7, 4, 7, 7$. It can be shown that 3 is the largest integer such that any of the maximums will be strictly greater than it.

In the third test case, the only segment that Bob can choose is $[1, 1]$. So the answer is 0.

,

B. XOR Sequences

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

You are given two distinct non-negative integers x and y. Consider two infinite sequences $a_1, a_2, a_3, \ldots$ and $b_1, b_2, b_3, \ldots$, where

- $a_n = n \oplus x$;
- $b_n = n \oplus y$.

Here, $x \oplus y$ denotes the bitwise XOR operation of integers x and y.

For example, with $x = 6$, the first 8 elements of sequence a will look as follows: $[7, 4, 5, 2, 3, 0, 1, 14, \ldots]$. Note that the indices of elements start with 1.

Your task is to find the length of the longest common subsegment† of sequences a and b. In other words, find the maximum integer m such that $a_i = b_j, a_{i+1} = b_{j+1}, \ldots, a_{i+m-1} = b_{j+m-1}$ for some $i, j \geq 1$.

†A subsegment of sequence p is a sequence $p_l, p_{l+1}, \ldots, p_r$, where $1 \leq l \leq r$.

Input

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

The only line of each test case contains two integers x and y ($0 \leq x, y \leq 10^9, x \neq y$) — the parameters of the sequences.

Output

For each test case, output a single integer — the length of the longest common subsegment.

Standard Input	Standard Output
4 0 1 12 4 57 37 316560849 14570961	1 8 4 33554432

Note

In the first test case, the first 7 elements of sequences a and b are as follows:

$a = [1, 2, 3, 4, 5, 6, 7, \ldots]$

$b = [0, 3, 2, 5, 4, 7, 6, \ldots]$

It can be shown that there isn't a positive integer k such that the sequence $[k, k + 1]$ occurs in b as a subsegment. So the answer is 1.

In the third test case, the first 20 elements of sequences a and b are as follows:

$a = [56, 59, 58, 61, 60, 63, 62, 49, 48, 51, 50, 53, 52, 55, 54, \mathbf{41, 40, 43, 42}, 45, \ldots]$

$b = [36, 39, 38, 33, 32, 35, 34, 45, 44, 47, 46, \mathbf{41, 40, 43, 42}, 53, 52, 55, 54, 49, \ldots]$

It can be shown that one of the longest common subsegments is the subsegment $[41, 40, 43, 42]$ with a length of 4.

,

C. Earning on Bets

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

You have been offered to play a game. In this game, there are n possible outcomes, and for each of them, you must bet a certain **integer** amount of coins. In the event that the i-th outcome turns out to be winning, you will receive back the amount of coins equal to your bet on that outcome, multiplied by k_i. Note that **exactly one** of the n outcomes will be winning.

Your task is to determine how to distribute the coins in such a way that you will come out ahead in the event of **any** winning outcome. More formally, the total amount of coins you bet on all outcomes must be **strictly less** than the number of coins received back for each possible winning outcome.

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 a single integer n ($1 \le n \le 50$) — the number of outcomes.

The second line of each test case contains n integers $k_1, k_2, \ldots, k_n$ ($2 \le k_i \le 20$) — the multiplier for the amount of coins if the i-th outcome turns out to be winning.

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 -1 if there is no way to distribute the coins as required. Otherwise, output n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 10^9$) — your bets on the outcomes.

It can be shown that if a solution exists, there is always a solution that satisfies these constraints.

If there are multiple suitable solutions, output any of them.

Standard Input	Standard Output
6 3 3 2 7 2 3 3 5 5 5 5 5 5 6 7 9 3 17 9 13 3 6 3 2 5 9 4 6 8 3	27 41 12 1 1 -1 1989 1547 4641 819 1547 1071 -1 8 18 12 9 24

Note

In the first test case, the coins can be distributed as follows: 27 coins on the first outcome, 41 coins on the second outcome, 12 coins on the third outcome. Then the total amount of coins bet on all outcomes is $27 + 41 + 12 = 80$ coins. If the first outcome turns out to be winning, you will receive back $3 \cdot 27 = 81$ coins, if the second outcome turns out to be winning, you will receive back $2 \cdot 41 = 82$ coins, if the third outcome turns out to be winning, you will receive back $7 \cdot 12 = 84$ coins. All these values are strictly greater than 80.

In the second test case, one way is to bet one coin on each of the outcomes.

,

D. Fixing a Binary String

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

You are given a binary string s of length n, consisting of zeros and ones. You can perform the following operation **exactly once**:

1. Choose an integer p ($1 \le p \le n$).
2. Reverse the substring $s_1 s_2 \ldots s_p$. After this step, the string $s_1 s_2 \ldots s_n$ will become $s_p s_{p-1} \ldots s_1 s_{p+1} s_{p+2} \ldots s_n$.
3. Then, perform a cyclic shift of the string s to the left p times. After this step, the initial string $s_1 s_2 \ldots s_n$ will become $s_{p+1} s_{p+2} \ldots s_n s_p s_{p-1} \ldots s_1$.

For example, if you apply the operation to the string `110001100110` with $p = 3$, after the second step, the string will become **`011`**`001100110`, and after the third step, it will become `001100110`**`011`**.

A string s is called k-*proper* if two conditions are met:

- $s_1 = s_2 = \ldots = s_k$;
- $s_{i+k} \neq s_i$ for any i ($1 \le i \le n - k$).

For example, with $k = 3$, the strings `000`, `111000111`, and `111000` are k-proper, while the strings `000000`, `001100`, and `1110000` are not.

You are given an integer k, which **is a divisor** of n. Find an integer p ($1 \le p \le n$) such that after performing the operation, the string s becomes k-proper, or determine that it is impossible. Note that if the string is initially k-proper, you still need to apply exactly one operation to it.

Input

Each test consists of multiple test cases. The first line contains one 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 two integers n and k ($1 \le k \le n$, $2 \le n \le 10^5$) — the length of the string s and the value of k. It is guaranteed that k **is a divisor** of n.

The second line of each test case contains a binary string s of length n, consisting of the characters `0` and `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, output a single integer — the value of p to make the string k-proper, or -1 if it is impossible.

If there are multiple solutions, output any of them.

Standard Input	Standard Output
7 8 4 11100001	3 -1 7

4 2 1110 12 3 111000100011 5 5 00000 6 1 101001 8 4 01110001 12 2 110001100110	5 4 -1 3

Note

In the first test case, if you apply the operation with $p = 3$, after the second step of the operation, the string becomes **111**00001, and after the third step, it becomes 00001**111**. This string is 4-proper.

In the second test case, it can be shown that there is no operation after which the string becomes 2-proper.

In the third test case, if you apply the operation with $p = 7$, after the second step of the operation, the string becomes **1000111**00011, and after the third step, it becomes 00011**1000111**. This string is 3-proper.

In the fourth test case, after the operation with any p, the string becomes 5-proper.

,

E. Manhattan Triangle

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

The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is defined as:

$$|x_1 - x_2| + |y_1 - y_2|.$$

We call a *Manhattan triangle* three points on the plane, the Manhattan distances between each pair of which are equal.

You are given a set of pairwise distinct points and an **even** integer d. Your task is to find any Manhattan triangle, composed of three distinct points from the given set, where the Manhattan distance between any pair of vertices is equal to d.

Input

Each test consists of multiple test cases. The first line contains one 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 two integers n and d ($3 \le n \le 2 \cdot 10^5$, $2 \le d \le 4 \cdot 10^5$, d is even) — the number of points and the required Manhattan distance between the vertices of the triangle.

The $(i + 1)$-th line of each test case contains two integers x_i and y_i ($-10^5 \le x_i, y_i \le 10^5$) — the coordinates of the i-th point. It is guaranteed that all points are pairwise distinct.

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 three distinct integers i, j, and k ($1 \le i, j, k \le n$) — the indices of the points forming the Manhattan triangle. If there is no solution, output "$0\ 0\ 0$" (without quotes).

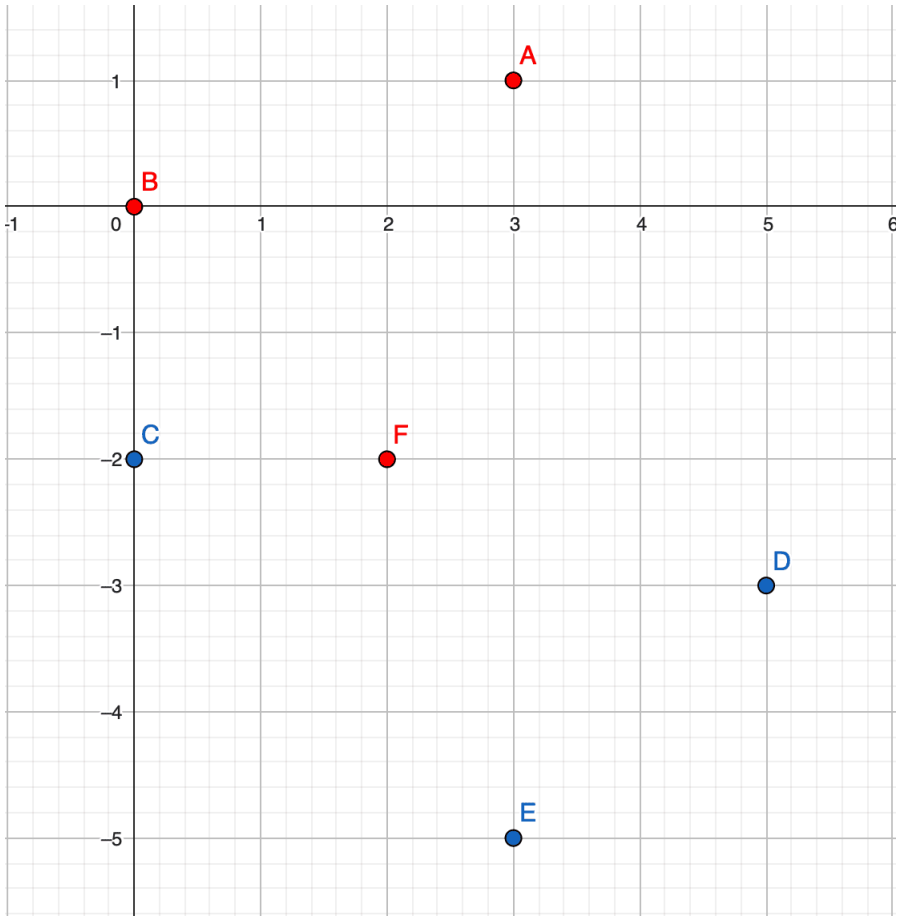
If there are multiple solutions, output any of them.

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

3 1 0 0 0 -2 5 -3 3 -5 2 -2 4 4 3 0 0 3 -3 0 0 -3 10 8 2 1 -5 -1 -4 -1 -5 -3 0 1 -2 5 -4 4 -4 2 0 0 -4 1 4 400000 100000 100000 -100000 100000 100000 -100000 -100000 -100000	

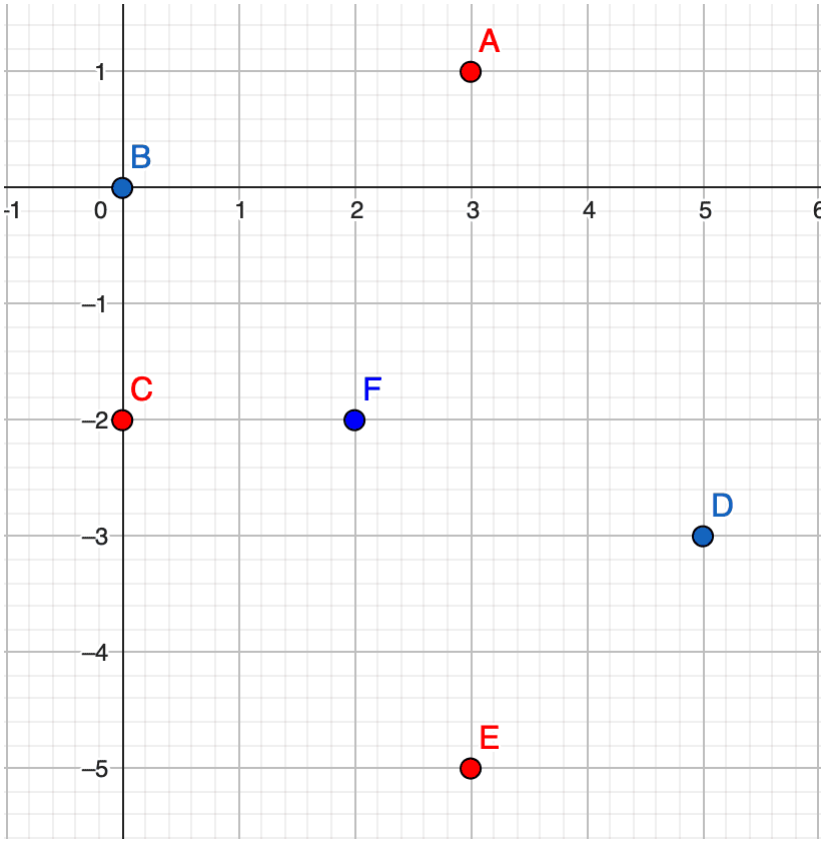
Note

In the first test case:



Points A, B, and F form a Manhattan triangle, the Manhattan distance between each pair of vertices is 4. Points D, E, and F can also be the answer.

In the third test case:



Points A, C, and E form a Manhattan triangle, the Manhattan distance between each pair of vertices is 6.

In the fourth test case, there are no two points with a Manhattan distance of 4, and therefore there is no suitable Manhattan triangle.

,

F. Kostyanych's Theorem

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

This is an interactive problem.

Kostyanych has chosen a complete undirected graph† with n vertices, and then removed exactly $(n-2)$ edges from it. You can ask queries of the following type:

- "? d" — Kostyanych tells you the number of vertex v with a degree **at least** d. Among all possible such vertices, he selects the vertex **with the minimum degree**, and if there are several such vertices, he selects the one with the minimum number. He also tells you the number of another vertex in the graph, with which v is not connected by an edge (if none is found, then 0 is reported). Among all possible such vertices, he selects the one with the minimum number. Then he removes the vertex v and all edges coming out of it. If the required vertex v is not found, then "$0\ 0$" is reported.

Find a Hamiltonian path‡ in the **original** graph in at most n queries. It can be proven that under these constraints, a Hamiltonian path always exists.

† A complete undirected graph is a graph in which there is exactly one undirected edge between any pair of distinct vertices. Thus, a complete undirected graph with n vertices contains $\frac{n(n-1)}{2}$ edges.

‡ A Hamiltonian path in a graph is a path that passes through each vertex exactly once.

Input

Each test consists of 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 only line of each test case contains a single integer n ($2 \le n \le 10^5$) — the number of vertices in the graph.

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

Interaction

Interaction for each test case begins with reading the integer n.

Then you can make no more than n queries.

To make a query, output a line in the format "? d" (without quotes) ($0 \le d \le n-1$). After each query, read two integers — the answer to your query.

When you are ready to report the answer, output a line in the format "! $v_1\ v_2\ \ldots\ v_n$" (without quotes) — the vertices in the order of their occurrence in the Hamiltonian path. Outputting the answer does not count as one of the n queries. After solving one test case, the program should immediately move on to the next one. After solving all test cases, the program should be terminated immediately.

If your program makes more than n queries for one test case or makes an incorrect query, then the **response to the query will be** -1, and after receiving such a response, your program should immediately terminate to receive the verdict `Wrong answer`. Otherwise, it may receive any other verdict.

After outputting a query, do not forget to output an end of line and flush the output buffer. Otherwise, you will receive the verdict `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.

The interactor is **non-adaptive**. The graph **does not change** during the interaction.

Hacks

To hack, use the following format:

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

The only line of each test case contains a single integer n ($2 \le n \le 10^5$) — the number of vertices in the graph.

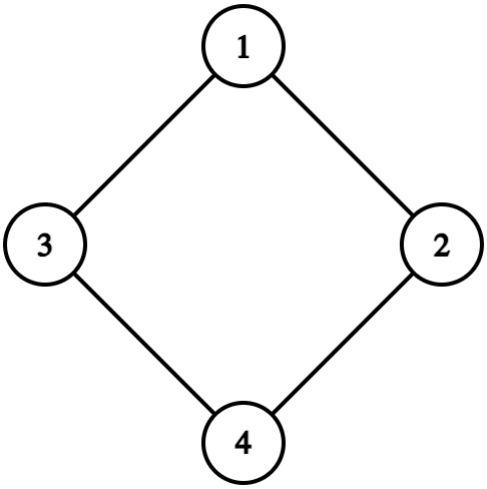
Each of the following $(n - 2)$ lines should contains two integers u and v ($1 \le u, v \le n, u \ne v$) — ends of the edge that was removed from the graph. Each edge must not occur more than once.

The sum of n over all test cases should not exceed 10^5.

Standard Input	Standard Output
3 4 0 0 1 4 2 3 4 1 0 4 2 2 1 0	 ? 3 ? 2 ? 1 ! 4 3 1 2 ? 3 ? 0 ! 4 1 2 3 ? 0 ! 2 1

Note

In the first test case, the original graph looks as follows:

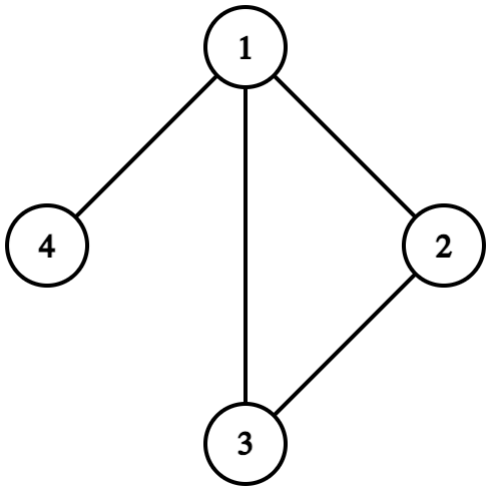


Consider the queries:

- There are no vertices with a degree of at least 3 in the graph, so "$0\ 0$" is reported.
- There are four vertices with a degree of at least 2, and all of them have a degree of exactly 2: 1, 2, 3, 4. Vertex 1 is reported, because it has the minimum number, and vertex 4 is reported, because it is the only one not connected to vertex 1. After this, vertex 1 is removed from the graph.
- There are three vertices with a degree of at least 1, among them vertices 2 and 3 have a minimum degree of 1 (vertex 4 has a degree of 2). Vertex 2 is reported, because it has the minimum number, and vertex 3 is reported, because it is the only one not connected to vertex 2. After this, vertex 2 is removed from the graph.

The path $4 - 3 - 1 - 2$ is a Hamiltonian path.

In the second test case, the original graph looks as follows:



Consider the queries:

- Vertex 1 has a degree of at least 3, but it is connected to all vertices, so "$1\ 0$" is reported. After this, vertex 1 is removed from the graph.
- The remaining vertices 2, 3, and 4 have a degree of at least 0, but among them vertex 4 has the minimum degree of 0 (vertices 2 and 3 have a degree of 1). Vertex 4 is not connected to both vertices 2 and 3, so vertex 2 is reported (as it has the minimum number). After this, vertex 4 is removed from the graph.

The path $4 - 1 - 2 - 3$ is a Hamiltonian path.

In the third test case, the graph consists of 2 vertices connected by an edge.