A. Too Min Too Max

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

Given an array a of n elements, find the maximum value of the expression:

$$|a_i - a_j| + |a_j - a_k| + |a_k - a_l| + |a_l - a_i|$$

where i, j, k, and l are four **distinct** indices of the array a, with $1 \le i, j, k, l \le n$.

Here $|x|$ denotes the absolute value of x.

Input

The first line contains one integer t ($1 \le t \le 500$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($4 \le n \le 100$) — the length of the given array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^6 \le a_i \le 10^6$).

Output

For each test case, print a single integer — the maximum value.

Standard Input	Standard Output
5 4 1 1 1 1 5 1 1 2 2 3 8 5 1 3 2 -3 -1 10 3 4 3 3 1 1 4 1 2 2 -1	0 6 38 8 8

Note

In the first test case, for any selection of i, j, k, l, the answer will be 0. For example,
$|a_1 - a_2| + |a_2 - a_3| + |a_3 - a_4| + |a_4 - a_1| = |1 - 1| + |1 - 1| + |1 - 1| + |1 - 1| = 0 + 0 + 0 + 0 = 0$.

In the second test case, for $i = 1$, $j = 3$, $k = 2$, and $l = 5$, the answer will be 6.
$|a_1 - a_3| + |a_3 - a_2| + |a_2 - a_5| + |a_5 - a_1| = |1 - 2| + |2 - 1| + |1 - 3| + |3 - 1| = 1 + 1 + 2 + 2 = 6$.

,

B. Yet Another Coin Problem

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

You have 5 different types of coins, each with a value equal to one of the first 5 triangular numbers: $1, 3, 6, 10$, and 15. These coin types are available in abundance. Your goal is to find the minimum number of these coins required such that their total value sums up to exactly n.

We can show that the answer always exists.

Input

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 an integer n ($1 \le n \le 10^9$) — the target value.

Output

For each test case, output a single number — the minimum number of coins required.

Standard Input	Standard Output
14 1 2 3 5 7 11 12 14 16 17 18 20 98 402931328	1 2 1 3 2 2 2 3 2 3 2 2 8 26862090

Note

In the first test case, for $n = 1$, the answer is 1 since only one 1 value coin is sufficient. $1 = 1 \cdot 1$.

In the fourth test case, for $n = 5$, the answer is 3, which can be achieved using two 1 value coins and one 3 value coin. $5 = 2 \cdot 1 + 1 \cdot 3$.

In the seventh test case, for $n = 12$, the answer is 2, which can be achieved using two 6 value coins.

In the ninth test case, for $n = 16$, the answer is 2, which can be achieved using one 1 value coin and one 15 value coin or using one 10 value coin and one 6 value coin. $16 = 1 \cdot 1 + 1 \cdot 15 = 1 \cdot 6 + 1 \cdot 10$.

,

C. Find a Mine

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

This is an interactive problem.

You are given a grid with n rows and m columns. The coordinates (x, y) represent the cell on the grid, where x ($1 \le x \le n$) is the row number counting from the top and y ($1 \le y \le m$) is the column number counting from the left. It is guaranteed that there are exactly 2 mines in the grid at **distinct** cells, denoted as (x_1, y_1) and (x_2, y_2). You are allowed to make no more than 4 queries to the interactor, and after these queries, you need to provide the location of **one of the mines**.

In each query, you can choose any grid cell (x, y), and in return, you will receive the minimum Manhattan distance from both the mines to the chosen cell, i.e., you will receive the value $\min(|x - x_1| + |y - y_1|, |x - x_2| + |y - y_2|)$.

Your task is to determine the location of one of the mines after making the queries.

Input

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

The only line of each test case contains two integers n and m ($2 \le n \le 10^8$, $2 \le m \le 10^8$) — the number of rows and columns.

Interaction

For each test case, the interaction starts with reading n and m.

Then you are allowed to make at most 4 queries in the following way:

"? x y" ($1 \le x \le n$ and $1 \le y \le m$)

After each one, you should read an integer d which is equal to $\min(|x - x_1| + |y - y_1|, |x - x_2| + |y - y_2|)$.

When you have found the location of any one of the mines, print a single line "! x y" (without quotes), representing the row and the column of one of the mines. Outputting the answer does not count as a query.

After printing the answer, your program must then continue to solve the remaining test cases, or exit if all test cases have been solved.

The interactor for this problem is not adaptive: cells of mines are fixed before any queries are made.

After printing a query, do not forget to output the 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 the documentation for other languages.

Hacks:

To make a hack, use the following format:

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

The description of each test case should consist of three lines.

The first line contains two integers n and m ($2 \le n \le 10^8$, $2 \le m \le 10^8$) — the number of rows and columns.

The second line contains the coordinates of the first mine x_1 and y_1 ($1 \le x_1 \le n$, $1 \le y_1 \le m$).

The third line contains the coordinates of the second mine x_2 and y_2 ($1 \le x_2 \le n$, $1 \le y_2 \le m$).

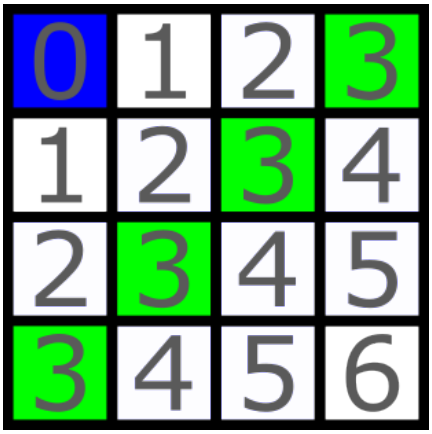
The mines should be located at different positions.

Standard Input	Standard Output
2 4 4 3 2 2 0 5 5 1 2 3	 ? 1 1 ? 1 4 ? 4 1 ? 2 3 ! 2 3 ? 5 5 ? 2 2 ? 3 3 ! 1 1

Note

In the first test case, we start by querying the upper-left corner $(1, 1)$ and get the result 3, which means that there is a mine on the counter diagonal, and there is no mine above it.

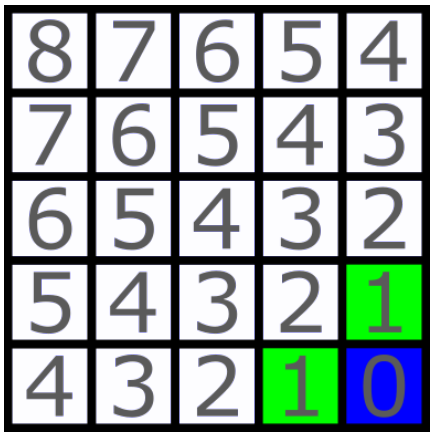
In the image below, each cell contains a number indicating the distance to the blue cell. The green cells are candidates to contain the nearest mine.



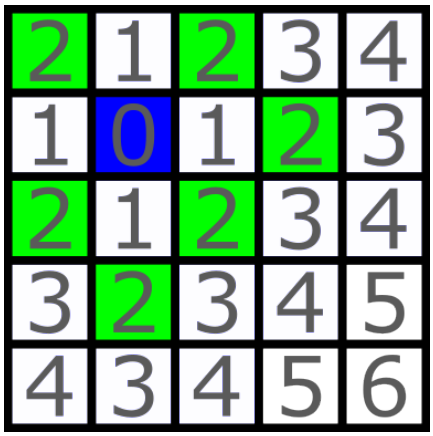
Then we ask three cells on that diagonal, and at the last query, we get the result 0, which means that a mine is found at the position $(2, 3)$.

The second mine was located at the position $(3, 2)$.

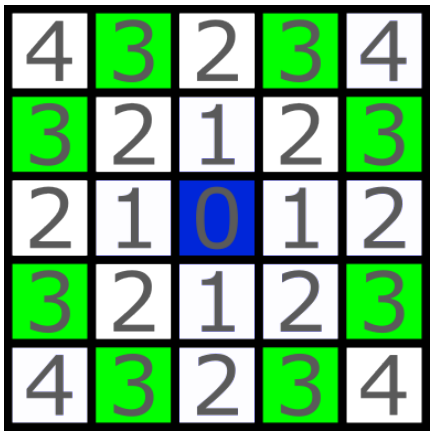
In the second test case, we start by asking the lower-right corner $(5, 5)$, and get the result 1, which means that one of the two neighbours contains a mine, let's call it mine 1.



Then we ask cell $(2, 2)$. We can see that these green cells don't intersect with the green cells from the first query, so they contain the other mine, let's call it mine 2.



Query 3 is cell $(3, 3)$. These cells contain mine 1, but we still don't know where exactly. Nevertheless, we can determine that the only possible cell for mine 2 is $(1, 1)$, because all other candidates are at a distance closer than 3 for this query.



,

D1. XOR Break — Solo Version

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

This is the solo version of the problem. Note that the solution of this problem may or may not share ideas with the solution of the game version. You can solve and get points for both versions independently.

You can make hacks only if both versions of the problem are solved.

Given an integer variable x with the initial value of n. A single break operation consists of the following steps:

- Choose a value y such that $0 < y < x$ and $0 < (x \oplus y) < x$.
- Update x by either setting $x = y$ or setting $x = x \oplus y$.

Determine whether it is possible to transform x into m using a maximum of 63 break operations. If it is, provide the sequence of operations required to achieve $x = m$.
You don't need to minimize the number of operations.

Here $\oplus$ denotes the bitwise XOR operation.

Input

The first line contains one positive integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of a single line containing two integers n and m ($1 \le m < n \le 10^{18}$) — the initial value of x and the target value of x.

Output

For each test case, output your answer in the following format.

If it is not possible to achieve m in 63 operations, print -1.

Otherwise,

The first line should contain k ($1 \le k \le 63$) — where k is the number of operations required.

The next line should contain $k + 1$ integers — the sequence where variable x changes after each break operation. The 1-st and $k + 1$-th integers should be n and m, respectively.

Standard Input	Standard Output
3 7 3 4 2 481885160128643072 45035996273704960	1 7 3 -1 3 481885160128643072 337769972052787200 49539595901075456 45035996273704960

Note

In the first test case $n = 7$, for the first operation $x = 7$ if we choose $y = 3$ then $(7 \oplus 3) < 7$, hence we can update x with 3 which is equal to m.

In the second test case $n = 4$, for the first operation $x = 4$.

If we choose:

- $y = 1$ then $(4 \oplus 1) > 4$
- $y = 2$ then $(4 \oplus 2) > 4$
- $y = 3$ then $(4 \oplus 3) > 4$

Hence we can't do the first operation and it is impossible to make $x = 2$.

,

D2. XOR Break — Game Version

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

This is an interactive problem.

This is the game version of the problem. Note that the solution of this problem may or may not share ideas with the solution of the solo version. You can solve and get points for both versions independently.

Alice and Bob are playing a game. The game starts with a positive integer n, with players taking turns. On each turn of the game, the following sequence of events takes place:

- The player having the integer p breaks it into two integers p_1 and p_2, where $0 < p_1 < p$, $0 < p_2 < p$ and $p_1 \oplus p_2 = p$.
- If no such p_1, p_2 exist, the player loses.
- Otherwise, the opponent does either select the integer p_1 or p_2.
- The game continues with the selected integer. The opponent will try to break it.

As Alice, your goal is to win. You can execute a maximum of 63 break operations. You have the choice to play first or second. The system will act for Bob.

Here $\oplus$ denotes the bitwise XOR operation.

Input

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

The only line of each test case contains a single integer n ($1 \leq n \leq 10^{18}$) — the number the game starts with.

Interaction

For each test case, the interaction begins by reading the integer n.

After reading n, print a single line containing either "`first`" or "`second`", denoting what you want to play as (as first or second correspondingly).

On Alice's turn, you are required to print two positive integers, p_1 and p_2 such that $0 < p_1 < p$, $0 < p_2 < p$ and $p_1 \oplus p_2 = p$. Here, p equals one of the two integers printed by Bob in the previous turn. If no turn has occurred previously, p is equal to n. If Alice cannot perform a break operation, print "`0 0`" to receive a `Wrong answer` verdict.

On Bob's turn, you should read two integers, p_1 and p_2 such that $0 < p_1 < p$, $0 < p_2 < p$ and $p_1 \oplus p_2 = p$. Here, p equals one of the two integers printed by Alice in the previous turn. If no turn has occurred previously, p is equal to n. If Bob cannot perform a break operation $p_1 = 0$ and $p_2 = 0$ in which case you should proceed to the next test case.

If any break operation performed by Alice is invalid, the interactor prints "`-1 -1`" and your code should promptly exit to receive a `wrong answer` verdict.

If Alice performs 63 turns and Bob can still execute a break operation on the current integers, the interactor prints "`-1 -1`", and your code should promptly exit to receive a `wrong answer` verdict.

After printing a query, do not forget to output the 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 the documentation for other languages.

In this problem, hacks are disabled.

Standard Input	Standard Output
4 1 0 0 3 0 0 13 3 4 0 0 777777770001 0 0	 second first 2 1 first 10 7 1 2 first 777777770000 1

Note

Explanation for the interaction.

Interactor / Bob	Alice	Explanation
4		t
1		n for the first test case
	second	Alice chooses to go second
0 0		Bob says he cannot break $p = 1$
3		n for the second test case
	first	Alice chooses to go first
	1 2	Alice breaks $p = 3$ into $p_1 = 1$ and $p_2 = 2$
0 0		Bob says he cannot break $p = 1$ or $p = 2$
13		n for the third test case
	first	Alice chooses to go first
	10 7	Alice breaks $p = 13$ into $p_1 = 10$ and $p_2 = 7$
3 4		Bob breaks $p = 7$ into $p_1 = 3$ and $p_2 = 4$
	1 2	Alice breaks $p = 3$ into $p_1 = 1$ and $p_2 = 2$
0 0		Bob says he cannot break $p = 1$ or $p = 2$

777777770001		n for the fourth test case
	first	Alice chooses to go first
	777777770000 1	Alice breaks $p = 777\,777\,770\,001$ into $p_1 = 777\,777\,770\,000$ and $p_2 = 1$
0 0		Bob says he cannot perform break operation.

This table is for explanation only and does not reflect the actual behavior of the interactor.

Note that in the last test case Bob could choose p_1 and perform a break operation but he gave up.

,

E. Weird LCM Operations

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

Given an integer n, you construct an array a of n integers, where $a_i = i$ for all integers i in the range $[1, n]$. An operation on this array is defined as follows:

- Select three distinct indices i, j, and k from the array, and let $x = a_i$, $y = a_j$, and $z = a_k$.
- Update the array as follows: $a_i = \operatorname{lcm}(y, z)$, $a_j = \operatorname{lcm}(x, z)$, and $a_k = \operatorname{lcm}(x, y)$, where lcm represents the least common multiple.

Your task is to provide a possible sequence of operations, containing at most $\lfloor \frac{n}{6} \rfloor + 5$ operations such that after executing these operations, if you create a set containing the greatest common divisors (GCDs) of all subsequences with a **size greater than** 1, then all numbers from 1 to n should be present in this set.
After all the operations $a_i \leq 10^{18}$ should hold for all $1 \leq i \leq n$.

We can show that an answer always exists.

Input

The first line contains one integer t ($1 \leq t \leq 10^2$) — the number of test cases. The description of the test cases follows.

The first and only line of each test case contains an integer n ($3 \leq n \leq 3 \cdot 10^4$) — the length of the array.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^4$.

Output

The first line should contain an integer k ($0 \leq k \leq \lfloor \frac{n}{6} \rfloor + 5$) — where k is the number of operations.

The next k lines should contain the description of each operation i.e. 3 integers i, j and k, where $1 \leq i, j, k \leq n$ and all must be distinct.

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

Note

In the third test case, $a = [1, 2, 3, 4, 5, 6, 7]$.

First operation:

$i = 3, j = 5, k = 7$

$x = 3, y = 5, z = 7$.

$a = [1, 2, \operatorname{lcm}(y, z), 4, \operatorname{lcm}(x, z), 6, \operatorname{lcm}(x, y)]$ = $[1, 2, 35, 4, 21, 6, 15]$.

Second operation:

$i = 5, j = 6, k = 7$

$x = 21, y = 6, z = 15.$

$a = [1, 2, 35, 4, \text{lcm}(y, z), \text{lcm}(x, z), \text{lcm}(x, y)]$ = $[1, 2, 35, 4, 30, 105, 42]$.

Third operation:

$i = 2, j = 3, k = 4$

$x = 2, y = 35, z = 4.$

$a = [1, \text{lcm}(y, z), \text{lcm}(x, z), \text{lcm}(x, y), 30, 105, 42]$ = $[1, 140, 4, 70, 30, 105, 42]$.

Subsequences whose GCD equal to i is as follows:

$\gcd(a_1, a_2) = \gcd(1, 140) = 1$

$\gcd(a_3, a_4) = \gcd(4, 70) = 2$

$\gcd(a_5, a_6, a_7) = \gcd(30, 105, 42) = 3$

$\gcd(a_2, a_3) = \gcd(140, 4) = 4$

$\gcd(a_2, a_4, a_5, a_6) = \gcd(140, 70, 30, 105) = 5$

$\gcd(a_5, a_7) = \gcd(30, 42) = 6$

$\gcd(a_2, a_4, a_6, a_7) = \gcd(140, 70, 105, 42) = 7$