CPC '21 Contest 1 P5 - AQT and Modern Art

Time limit: 3.0s **Memory limit:** 256M

For his art project final, AQT had to create an abstract art by placing many rectangles on the Cartesian plane, sometimes stacking rectangles on top of other rectangles. However, one day before the project was due, an unnamed wrong-doer told him to `rm -rf --no-preserve-root /` his computer, and he lost his beautiful art painting!

Luckily, even though his files were deleted, remnants of the data that once existed were still present on his computer, and a technician was able to send queries to the computer to help AQT recover his painting. Each query is able to return the number of rectangles fully covered in a rectangular area. However, the data remaining on the computer is unstable, so the technician can only send so many queries before the data is too corrupted to use. Unfortunately, the technician was unable to devise a method to recover AQT's rectangles in few enough queries, and has thus turned to you for help.

Formally, AQT needs to recover a painting consisting of N rectangles on the coordinate plane such that the i^{th} one covers the area from $x_{i,1}$ to $x_{i,2}$ on the x-axis and from $y_{i,1}$ to $y_{i,2}$ on the y-axis. He is allowed to send queries a, b, c, d that return the number of rectangles i such that $a \leq x_{i,1} \leq x_{i,2} \leq b$ and $c \leq y_{i,1} \leq y_{i,2} \leq d$. **The maximum number of queries he may make is $61\,000$.**

Constraints

For all subtasks:

$1 \leq N \leq 500$

$1 \leq x_{i,1}, x_{i,2}, y_{i,1}, y_{i,2} \leq 10^9$

$x_{i,1} \leq x_{i,2}$

$y_{i,1} \leq y_{i,2}$

Subtask 1 [10%]

$1 \leq N \leq 10$

$1 \leq x_{i,1}, x_{i,2}, y_{i,1}, y_{i,2} \leq 10$

Subtask 2 [10%]

$1 \leq N \leq 10$

Subtask 3 [25%]

$y_{i,1} = y_{i,2} = 1$ for all $1 \leq i \leq N$

Subtask 4 [55%]

No additional constraints.

Interaction

The first and only line of input contains N, the number of rectangles originally on AQT's painting. You will then begin interaction.

To send a query, output 4 integers a, b, c, d in the following format: `? a b c d`. The grader will then output the number of rectangles i such that $a \leq x_{i,1} \leq x_{i,2} \leq b$ and $c \leq y_{i,1} \leq y_{i,2} \leq d$. If at any point you make an invalid query (it does not satisfy the constraints $1 \leq a \leq b \leq 10^9, 1 \leq c \leq d \leq 10^9$) or make too many queries, the grader will output `-1` before exiting.

Once you are done interaction, you must report the N rectangles. First, output `!` on a single line, and then N more lines of 4 space-separated integers each denoting the N rectangles you recovered. While you may output the rectangles in any order, the coordinates of rectangle i should be output in the order $x_{i,1}$, $x_{i,2}$, $y_{i,1}$, and $y_{i,2}$, separated by spaces. It is required that $1 \leq x_{i,1} \leq x_{i,2} \leq 10^9$ and $1 \leq y_{i,1} \leq y_{i,2} \leq 10^9$ for all $1 \leq i \leq N$.

Note that if there are duplicate rectangles, each rectangle must be outputted individually (i.e. if there are three rectangles with the same coordinates, you should output those coordinates three times).

If at any point you attempt an invalid query (or make too many queries), `-1` will be outputted (followed by a `\n` character of course). When this happens, you should terminate your program to avoid reading from a closed input stream, and you will receive a verdict of `Wrong Answer`. Otherwise, the verdict may be undefined.

Please note that you may need to flush `stdout` after each query or after answering a test case. In C++, this can be done with `fflush(stdout)` or `cout << flush` (depending on whether you use `printf` or `cout`). In Java, this can be done with `System.out.flush()`. In Python, you can use `sys.stdout.flush()`.

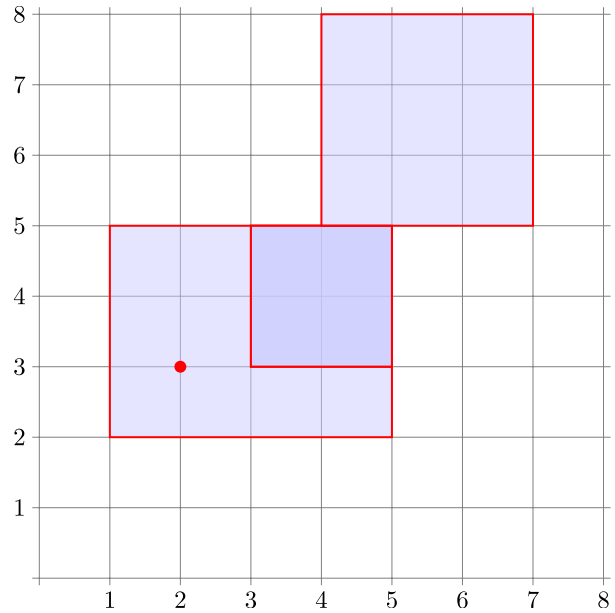
Note that the grader is not adaptive, so the rectangles are fixed at the start of every test case.

Sample Interaction

`>>>` denotes your output, do not print this out.

```
5
>>> ? 1 5 1 5
4
>>> ? 2 2 3 3
1
>>> ? 3 5 3 5
2
>>> ? 1 5 3 4
1
>>> ? 3 7 4 8
1
>>> !
>>> 2 2 3 3
>>> 4 7 5 8
>>> 3 5 3 5
>>> 3 5 3 5
>>> 1 5 2 5
```

Below are the rectangles in the sample interaction in diagram form. Note that the point at $(2, 3)$ represents the rectangle with 0 area.



And here they are as a list of 4-tuples in the format $(x_{i,1}, x_{i,2}, y_{i,1}, y_{i,2})$:

- $(1, 5, 2, 5)$
- $(2, 2, 3, 3)$
- $(3, 5, 3, 5)$
- $(3, 5, 3, 5)$
- $(4, 7, 5, 8)$

Note that the rectangles may be output in any order.