B. Searching Rectangles

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

Filya just learned new geometry object — rectangle. He is given a field consisting of $n \times n$ unit cells. Rows are numbered from bottom to top with integer from 1 to n. Columns are numbered from left to right with integers from 1 to n. Cell, located at the intersection of the row r and column c is denoted as (r, c). Filya has painted two rectangles, such that their sides are parallel to coordinate axes and each cell lies fully inside or fully outside each of them. Moreover, no cell lies in both rectangles.

Later, hedgehog Filya became interested in the location of his rectangles but was unable to find the sheet of paper they were painted on. They were taken by Sonya and now she wants to play a little game with Filya. He tells her a query rectangle and she replies with the number of initial rectangles that lie **fully inside** the given query rectangle. The query rectangle should match the same conditions as initial rectangles. Rectangle lies fully inside the query if each o its cells lies inside the query.

Filya knows Sonya really well, so is sure that if he asks more than 200 questions she will stop to reply.

Input

The first line of the input contains an integer n ($2 \le n \le 2^{16}$) — size of the field.

For each query an integer between 0 and 2 is returned — the number of initial rectangles that lie fully inside the query rectangle.

Output

To make a query you have to print "$? \ x_1 \ y_1 \ x_2 \ y_2$" (without quotes) ($1 \le x_1 \le x_2 \le n$, $1 \le y_1 \le y_2 \le n$), where (x_1, y_1) stands for the position of the bottom left cell of the query and (x_2, y_2) stands for the up right cell of the query.
You are allowed to ask no more than 200 queries. After each query you should perform "flush" operation and read the answer.

In case you suppose you've already determined the location of two rectangles (or run out of queries) you should print "$! \ x_{11} \ y_{11} \ x_{12} \ y_{12} \ x_{21} \ y_{21} \ x_{22} \ y_{22}$" (without quotes), where first four integers describe the bottom left and up right cells of the first rectangle, and following four describe the corresponding cells of the second rectangle. You can print the rectangles in an arbitrary order. After you have printed the answer, print the end of the line and perform "flush". Your program should terminate immediately after it print the answer.

Interaction

To flush you can use (just after printing an integer and end-of-line):

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

You will get the `Wrong Answer` verdict if you ask more than 200 queries, or if you print an incorrect coordinates.

You will get the `Idleness Limit Exceeded` verdict if you don't print anything (but you should) or if you forget about flushing the output (more info below).

Hacking.

The first line should contain an integer n ($2 \le n \le 2^{16}$).

The second line should contain four integers x_1, y_1, x_2, y_2 ($1 \le x_1 \le x_2 \le n$, $1 \le y_1 \le y_2 \le n$) — the description of the first rectangle.

The third line contains the description of the second rectangle in the similar way.

Example

input

```
5
2
1
0
1
1
1
0
1
```

output

```
? 1 1 5 5
? 1 1 3 3
? 1 1 3 1
? 2 2 2 2
? 3 3 5 5
? 3 3 3 5
? 3 3 3 4
? 3 4 3 5
! 2 2 2 2 3 4 3 5
```