D Destabilized Drone

Time limit: 2s

Your brand new drone company is planning to beat the competition with an amazing new piece of software, called the Bank And Pitch Controller. This software will make sure the drone is always horizontal, a must have feature for high end drones. In order to do so, it needs to measure the *bank* and *pitch* of the drone. Since the drone already has a front facing camera, this will be used to measure these numbers.

Given a single frame (image) from this camera, the software runs a highly advanced machine learning model to determine whether each pixel in the frame is sky, sea, or exactly on the horizon. The machine learning model is rather slow and can process only 900 pixels before the next video frame comes in. To stabilize the drone quickly enough, you need to create an efficient algorithm that can find the horizon by querying at most 900 pixels. Using this information, the rest of the BAPC will be able to compute the bank and pitch.

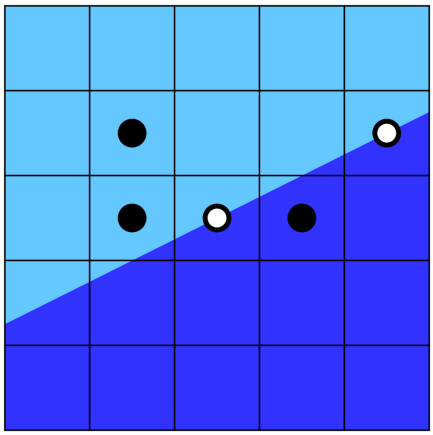


Figure D.1: Visualisation of Sample 1 showing the queried pixels, including two pixels on the horizon marked in white.

It is given that the horizon can be modelled by an exact straight line, and that at least two pixels in the image will be classified as horizon. Furthermore, the drone is usually flying roughly horizontal, so you may assume that the top row of the picture is always sky and that the bottom row of the picture only contains sea pixels.

A visualisation of the first sample can be seen in Figure D.1.

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor first sends a line containing two integers w and h ($3 \leq w, h \leq 1000$), the width and height of the image.

Then, your program should make at most 900 queries to determine the horizon. Each query is made by printing a line of the form "`? x y`" ($1 \le x \le w$, $1 \le y \le h$), where x is the column of the pixel, counting from the left, and y is the row of the pixel, counting from the bottom. In response to each query, the interactor will print one of: "`sky`", "`sea`", or "`horizon`", indicating whether the pixel is above, below, or on the horizon respectively.

When you have determined the horizon, print a single line of the form "`! x1 y1 x2 y2`" ($1 \le x_1, x_2 \le w$, $1 \le y_1, y_2 \le h$) containing an exclamation mark, followed by the coordinates of two distinct pixels on the horizon. This line does not count as one of your queries. Following this, your submission should exit and not read any more input.

If there are multiple valid solutions, you may output any one of them.

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Sample Interaction 1

Read	Write
`5 5`	
	`? 2 4`
`sky`	
	`? 4 3`
`sea`	
	`? 5 4`
`horizon`	
	`? 2 3`
`sky`	
	`? 3 3`
`horizon`	
	`! 5 4 3 3`

Sample Interaction 2

Read	Write
`1000 1000`	
	`? 999 999`
`horizon`	
	`? 2 3`
`horizon`	
	`! 2 3 999 999`