Rescue Mission

Objective
Give practice with binary search in C.

Story
It seems that one of the ships did not heed your storm warning. You received a single transmission before your lost contact. They reported that the ship cannot change its speed. You know that the max speed is S knots (nautical miles per hour). The ship should be traveling in the positive x-direction at an integral speed (in the range of 0 to S). Based on the transmission quality you know that the ship transmitted from an integral x location in the range of 0 to L inclusive.

You will send a team to save them, but the rescue team will only be able to be sent once to a single location and takes 1 hour to arrive, so you must determine the exact location of the ship. Luckily you are able to borrow a satellite that passes over the ocean every hour. It can take a photo of only 1 nautical mile, but you can use this to determine if the ship has passed the location by picking up the traces of a wake, if the ship is at that location by picking up the picture of the ship, or if the ship has not yet reached that location by picking up nothing in the picture.

You want to ensure that the ship is not stranded for too long. The ship has to be rescued in 24 hours (at most 23 pictures). Keep in mind that after the last picture the ship will move for a full hour while you launch the rescue team. Finally, your first photo will be exactly 1 hour after the initial distress signal was sent.

Problem
Write a program that when given the maximum starting speed of the ship and the maximum x coordinate, can ask for information at specific x-coordinates at most 23 times, and can at any point after a query can report the current location of the ship.

Input
Input will begin with a line containing 2 integers, ***S*** and ***L*** ($1 \leq$ ***S***, ***L*** ≤ 100), representing the maximum possible speed of the ship in Knots and the maximum possible starting coordinate of the ship, respectively. Following this will be multiple lines. Each line contains a single response to one of your queries. The responses will be one of the following strings

- `Wake` - The ship has already passed by
- `No Wake` - The ship has not passed this location as of yet
- `Boat!` - The ship was captured in the photo

The testing system will allow you to read the response only after your program prints the query and flushes the standard output (use fflush(stdout)).

Output

To make the queries your program should print to standard output. To ask for a photo of a location your program should output “? x” where x is the location to take the photo. To send the rescue team your program should output “! x” where x is the location of the ship. Your program must flush the output (use fflush(stdout)) after each query and after your command to send the rescue team.

Sample Input	Sample Output

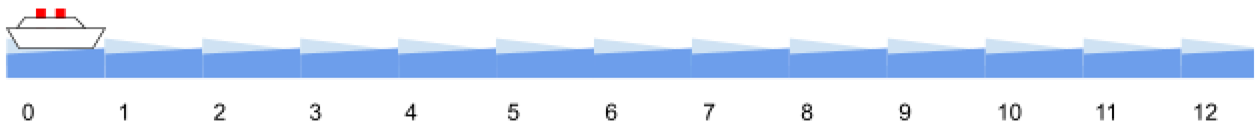
`3 3`
`No Wake`
`Wake`
`Wake`
`Wake`
`Boat!`
	`? 3`
	`? 1`
	`? 3`
	`? 5`
	`? 10`
	`! 12`

`5 2`
`Wake`
`Wake`
`Wake`
`Wake`
`Wake`
	`? 1`
	`? 5`
	`? 11`
	`? 18`
	`? 26`
	`! 32`

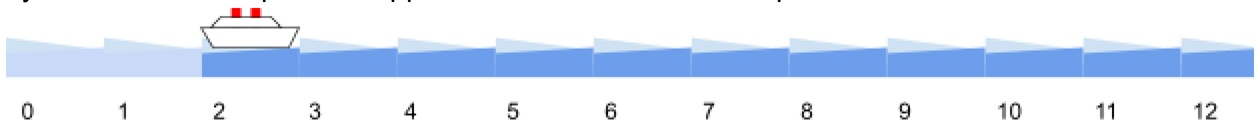
Explanation

Case 1

The boat has a speed of 2. Additionally the boat started at position 0.

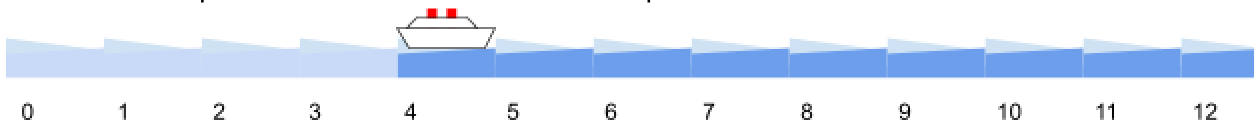


By the time the first picture happens the boat has moved to position 2.



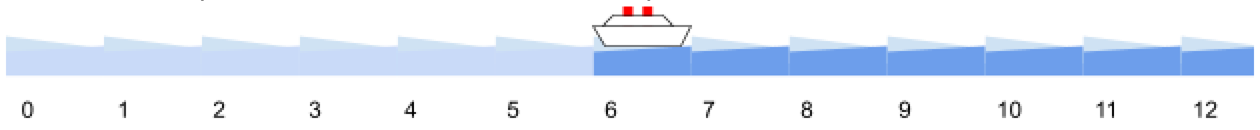
The sample took the picture at **position 3**. At that location there is no boat and **no wake**.

When the next picture is taken the boat moves to position 4.



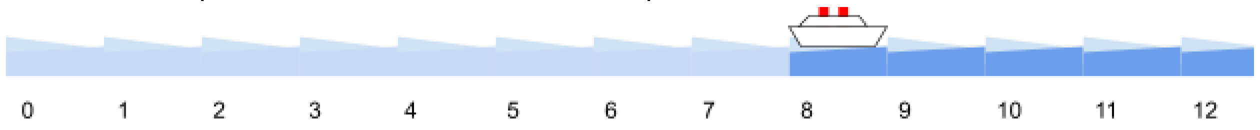
This picture is taken at **position 1**. There is no boat, but there is a **wake**.

When the next picture is taken the boat moves to position 6.



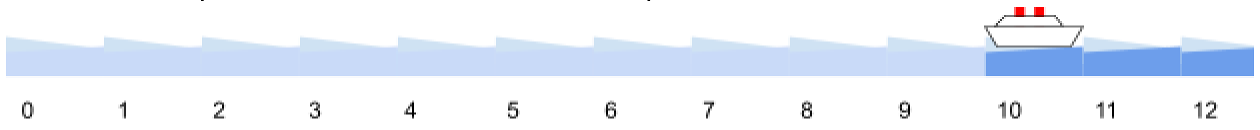
The next picture is taken at **position 3**. At that location there is no boat, but there is now a **wake**.

When the next picture is taken the boat moves to position 8.



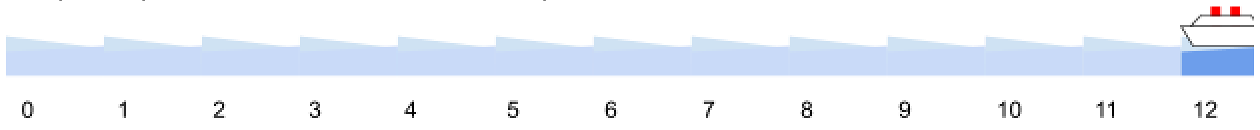
The next picture is taken at **position 5**. At that location there is no boat, but there is now a **wake**.

When the next picture is taken the boat moves to position 8.



The final picture is taken at **position 10**. At that location there is a **boat**.

The rescue team is now sent out. At this point the boat has moved to position 12. Luckily, the sample requests to send the team to that position.



Case 2

The boat started at position 2 with a speed of 5.

The ship will be at location 7 during the first picture. The picture is taken at location 1. There is a wake at that point.

The ship will be at location 12 during the second picture. The picture is taken at location 5. There is a wake at that point.

The ship will be at location 17 during the third picture. The picture is taken at location 11. There is a wake at that point.

The ship will be at location 22 during the fourth picture. The picture is taken at location 18. There is a wake at that point.

The ship will be at location 27 during the fifth picture. The picture is taken at location 26. There is a wake at that point.

The ship will be at location 32 when the sample sends out the rescue team. The rescue team is sent to location 32. There ship is at that location.

Hints

FLUSH Your Output: After every printf remember to fflush(stdout).

Track ALL Possibilities: Keep track of all the possible boats that exist. There should be 1 possibility for every combination of speed and starting location. When your program takes a photo and gets feedback. Eliminate all the boats that would have generated a different output.

Possibility Storage: You can store the possibilities in a 2D grid, indexed by the initial parameters of the ship, that contains a 0/1 indicating whether that possibility is still valid.

The Midpoint: Your program should guess the "midpoint". The midpoint is a bit unusual in this problem. At each time the photo is taken there is a list of possible locations that the ship could be (based on the speed/location combinations that have not been invalidated).

The midpoint should be the ship location that roughly splits the "search space" (the location of all remaining valid combinations) in half.

Midpoint Candidates: Don't guess random valid ship locations. Instead loop through the list of ship locations and check if that ship location works as a "midpoint". The midpoints should only be considered from the still valid ships.

Extension: This problem was made easier from an alternative version where the max speed and location could be up to 1000. If you finish the version of the problem for homework try to consider how you could efficiently solve the problem for larger bounds.

Grading Criteria

- Good comments, whitespace, and variable names
  - 15 points
- Use standard input/output
  - 5 points
- No extra input output (e.g. input prompts, “Please enter the number of friends:”)
  - 5 points
- Flush your output every time your print
  - 5 points
- Track the valid ship combinations
  - 10 points
- Check the effectiveness of a possible midpoint
  - 5 points
- “Move” the ships after each photo
  - 5 points
- Programs will be tested on <u>10 cases</u>
  - 5 points <u>each</u>
  - 6 of these cases will have fixed values
  - 4 of these cases will be adversarial
    - Without being certain of the location of the ship you will get the case wrong

No points will be awarded to programs that do not compile using “gcc -std=gnu11 -lm”.

Sometimes a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem use a binary search. ***<u>Without this programs will earn at most 50 points!</u>***

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 10 seconds}) will also be treated as wrong.

<u>No partial credit will be awarded for an incorrect case.</u>