Practice Problem Set

Please check that you have 3 problems that are spanned across 10 pages in total (including Korean translation and this cover page).

A. Card Game	(2+3 pages)	Korean translation available
B. Happy Point	(2 pages)	
C. Paper Folding	(2 pages)	

Practice A

Card Game

Time Limit: 0.5 Seconds

There are n $(2 \le n \le 10)$ cards, numbered 1 through n, lying out in a row on the floor. Each card is face up, with the integer 0 or 1 written on the back. You know that the integers written on the backs of cards 1 and n are 0 and 1, respectively, but you do not know what integers are written on the backs of the other cards.

You want to find two adjacent numbered cards that have 0 and 1 written on their backs, one after the other. Formally, you want to find any integer $1 \le i \le n-1$ such that the integer written on the back of card i is 0 and the integer on the back of card $i+1$ is 1.

To do this, you are allowed to request at most $n-2$ queries that flip a card over and check the value written on the back. Write a program to find any integer i that satisfies the above conditions, or to determine that no such i exists, by invoking a series of queries.

Interaction

This is an **interactive problem**. Your submitted program will interact with an **interactor** inside the grading server, which reads input from and writes output to your program.

Initially, an integer n representing the total number of cards is given as standard input on the first line. Subsequently, the interaction proceeds in rounds. In each round, your program must first write a line containing one of two types of requests as follows:

1. A query: "`?` x"
   - This query request starts with a character '`?`', followed by an integer x.
   - This asks to flip the card x over and check the value written on the back.
   - The value of x must be integer in the range $[1, n]$ inclusive.
   - This query request can be made up to $n-2$ times.
2. A final output: "`!` y"
   - This output request starts with a character '`!`', followed by an integer y.
   - This signals that your program has found any integer i that you are looking for, or determined that no such i exists.

After the request is asked, an input line is followed, containing the response of the interactor to your request, and the response is available on standard input. If your request is a query, the response will be an integer written on the back of card x. If your request is a final output, the response will be determined as follows:

- If $1 \le y \le n-1$, the interactor responds "`correct`" if the integers written on the backs of cards y and $y+1$ are 0 and 1, respectively.
- If $y = -1$, the interactor responds "`correct`" if such integer i does not actually exist.
- Otherwise, the interactor responds "`wrong`".

If your program violates the interaction protocol (e.g., issues a malformed request or makes more than $n-2$ queries), the interactor will immediately terminate the interaction. Also, your program must request

exactly one final output, and this must be the last request. After making the output request, your program must terminate gracefully. **Do not forget to flush the output** after each request.

The value of the integer written on the back of each card does not change throughout the interaction; the interactor is **not adaptive**.

The time and memory used by the interactor is also included in the calculation of your program's execution time and memory usage. You can assume that the maximum time used by the interactor is 0.1 second and the maximum amount of memory is 4 MiB.

Read (Interactor's Response)	**Sample Interaction 1**	**Write (Your Request)**
`6`		
		`? 1`
`0`		
		`? 4`
`1`		
		`? 3`
`0`		
		`! 3`
`correct`		

In this example, the integers written on the back of each card, in numerical order, were 0, 0, 0, 1, 0, and 1.

To flush, you need to do the following right after writing a request and a line break:

- `fflush(stdout)` in C;
- `std::cout << std::flush` in C++;
- `System.out.flush()` in Java;
- `sys.stdout.flush()` in Python.

A testing tool is provided to help you develop your solution, so it is not mandatory to use it. If you want to use it, you can download the attachment `testing_tool.py` from the *DOMjudge Problemset* page. You can run `python3 testing_tool.py -f input.in ./sol` to see how your program interacts for specific card settings. **This testing tool can be used regardless of the language of your program. It is recommended to read the comments in the source code of the testing tool for more detailed and important information**.

Practice A
Card Game
제한 시간: 0.5 초

1번부터 n번까지 번호가 붙어있는 n $(2 \le n \le 10)$ 장의 카드가 바닥에 일렬로 놓여있다. 각 카드는 앞면이 위를 향하게 놓여있으며, 뒷면에는 정수 0 또는 1이 적혀 있다. 여러분은 1번과 n번 카드의 뒷면에 적힌 정수가 각각 0과 1임을 알지만, 그 외의 카드의 뒷면에는 어떤 정수가 적혀 있는지 알지 못한다.

여러분은 뒷면에 각각 차례대로 0과 1이 적혀 있는 번호가 인접한 두 카드를 찾고 싶다. 엄밀하게 이야기하자면, i번 카드의 뒷면에는 0, $i+1$번 카드의 뒷면에는 1이 적혀 있는, 그러한 정수 $1 \le i \le n-1$을 아무거나 하나 찾고자 한다.

이를 위해, 여러분은 한 장의 카드를 뒤집어서 뒷면에 적힌 값을 확인하는 질의를 최대 $n-2$번 할 수 있다. 이러한 질의를 적절히 수행하여, 위의 조건을 만족하는 정수 i를 아무거나 하나 찾거나, 그러한 i가 존재하지 않음을 알아내는 프로그램을 작성하시오.

인터랙션

이 문제는 **인터랙티브 문제**이다. 여러분이 제출한 프로그램은 채점 서버 내부의 **인터랙터**와 상호작용(인터랙션)하게 된다. 인터랙터는 여러분의 프로그램의 출력을 읽으며, 인터랙터의 출력은 여러분의 프로그램의 입력으로 주어지게 된다.

여러분의 프로그램이 실행되면, 처음에 카드의 총 개수를 나타내는 정수 n이 표준입력으로 첫 줄에 주어진다. 이후의 인터랙션은 라운드로 진행된다. 각 라운드에서 여러분의 프로그램은 먼저 다음과 같이 두 가지 유형의 요청 중 하나를 형식에 맞게 한 줄에 출력해야 한다:

1. 질의: "? x"
   - 이 질의 요청은 문자 '?'와 정수 x로 구성된다.
   - 이 요청은 x번 카드를 뒤집어서 뒷면에 적힌 값을 확인하라는 것을 의미한다.
   - 정수 x의 값은 1 이상 n 이하여야 한다.
   - 이 질의 요청은 최대 $n-2$번 할 수 있다.
2. 최종 답안: "! y"
   - 이 답안 요청은 문자 '!'와 정수 y로 구성된다.

- 이 요청은 여러분이 찾고자 하는 정수 i를 아무거나 하나 찾았거나, 그러한 i가 존재하지 않음을 알아냈음을 의미한다.
- 정수 y의 값은 1 이상 $n-1$ 이하이거나 -1이여야 한다.

요청을 출력한 후에는 그 요청에 대한 인터랙터의 응답이 표준입력으로 주어진다. 질의 요청의 경우, 인터랙터는 x번 카드의 뒷면에 적혀 있는 정수를 응답한다. 최종 답안 요청의 경우, 인터랙터의 응답은 다음과 같이 결정된다:

- $1 \le y \le n-1$인 경우, y번과 $y+1$번 카드의 뒷면에 각각 차례대로 0과 1이 적혀 있다면, 인터랙터는 "`correct`"를 응답한다.
- $y = -1$ 인 경우, 찾고자 하는 정수 i가 실제로도 존재하지 않는다면, 인터랙터는 "`correct`"를 응답한다.
- 그 외의 경우, 인터랙터는 "`wrong`"을 응답한다.

여러분의 프로그램이 인터랙션 프로토콜을 위반하는 경우 (예를 들어, 조건이나 형식에 맞지 않는 요청을 출력하거나, $n-2$번보다 더 많은 질의 요청을 하는 경우), 인터랙터는 즉시 인터랙션을 종료한다. 또한, 여러분의 프로그램은 **정확히 하나의 최종 답안 요청**을 출력해야 하며, 이것이 **마지막 요청**이어야 한다. 답안 요청을 한 후에는 프로그램이 정상적으로 종료되어야 한다. 각 요청 직 후 **반드시 출력 버퍼를 플러시해야 함**에 유의하라.

각 카드의 뒷면에 적혀 있는 값은 인터랙션 내내 변하지 않는다. 즉, 인터랙터는 **적응적이지 않다**.

인터랙터가 사용한 시간과 메모리도 여러분의 프로그램의 실행 시간 및 메모리 사용량에 포함된다. 인터랙터가 사용하는 최대 시간은 0.1 초, 최대 메모리 양은 4MiB 라고 가정해도 좋다.

인터랙션 예시 1

표준입력 (인터랙션의 응답)	**표준출력 (여러분의 요청)**
`6`	
	`? 1`
`0`	
	`? 4`
`1`	
	`? 3`
`0`	
	`! 3`
`correct`	

이 예시에서, 각 카드의 뒷면에 적혀 있는 정수의 값은 순서대로 0, 0, 0, 1, 0, 1이었다.

출력 버퍼를 플러시하기 위해서는, 요청과 개행 문자를 출력한 직후 다음을 수행해야 한다:

- C 의 경우, `fflush(stdout)`
- C++의 경우, `std::cout << std::flush`
- Java 의 경우, `System.out.flush()`
- Python 의 경우, `sys.stdout.flush()`

풀이 코드를 작성하는 데에 도움이 되는 테스트 도구를 제공하지만, 반드시 사용할 필요는 없다. 테스트 도구를 사용하고 싶다면, *DOMjudge Problemset* 페이지에서 첨부 파일 `testing_tool.py` 을 다운로드한다. `python3 testing_tool.py -f input.in ./sol` 을 실행하여 특정 카드 세팅에 대해 여러분의 프로그램이 어떻게 인터랙션하는지 확인할 수 있다. 이 테스트 도구는 여러분의 프로그램의 언어에 관계없이 사용할 수 있다. **더 상세하고 중요한 정보는 테스트 도구의 소스 코드에 있는 주석을 통해 확인하라**.

Practice B
Happy Point
Time Limit: 1 Second

Jim, a boy in a gloomy mood after the COVID-19 pandemic, has a feeling that all the burden in the world is in his back. He wants to measure the happiness of his friends. One method he devised is to analyze the text messages from his friends to determine if the friends are happy. If a character in the message is included in the word "HAPPY," it is a happy character; if a character is in the word "SAD," it is a gloomy one. The character 'A' is included in both words, and it is a happy and gloomy character. The character 'B' is included in neither words, and it is neither happy nor gloomy.

The degree of happiness of the message is calculated based on the number of happy and gloomy characters of the message. If a character is happy, the happy point P_H is incremented by one; if a character is gloomy, the gloomy point P_G is incremented by one. The degree of happiness H of a message is calculated as follows:

$$H = \frac{P_H}{P_H + P_G}$$

If a character is neither happy nor gloomy, it is not counted as either P_H or P_G. Assume that the degree of happiness is 0.5 if both the happy point and the gloomy point are zero ($P_H = P_G = 0$).

For example, given the message from a friend as follows:

"SAD MOVIES ALWAYS MAKE ME CRY"

the happy point of this message can be calculated as 6 since A and Y appear four and two times, respectively: for A, one in SAD, two in ALWAYS, and one in MAKE; for Y, one in each of ALWAYS and CRY. The gloomy point, however, is 8 since S, A, and D appear three, four, and one time, respectively: for S, one in each of SAD, MOVIES, and ALWAYS; for A, it is calculated as same as that of the happy point; for D, one in SAD. Since $P_H = 6$ and $P_G = 8$, the degree of happiness is $H = \frac{6}{14} = 0.4286$, that is 42.86%. Therefore, the friend is in a gloomy mood since the degree of happiness is less than 50%.

Given a text message from Jim's friend, write a program to calculate the degree of happiness of the friend based on the message.

Input
Your program is to read from standard input. The input consists of a line containing the message from your friend. The message consists of alphabetic words only. The words are separated by a space. Every word consists of capital letters only. The maximum length of a word is 20 and the maximum number of words is 80.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the degree of happiness calculated. The degree of happiness should be printed in percent without the percent symbol. The degree of happiness should be rounded to two decimal places after the decimal point. Beware that the degree of happiness is 0.5 if both the happy point (P_H) and the gloomy point (P_G) are zero.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
SAD MOVIES ALWAYS MAKE ME CRY	42.86

Sample Input 2	Output for the Sample Input 2
ICPC PROGRAMMING	75.00

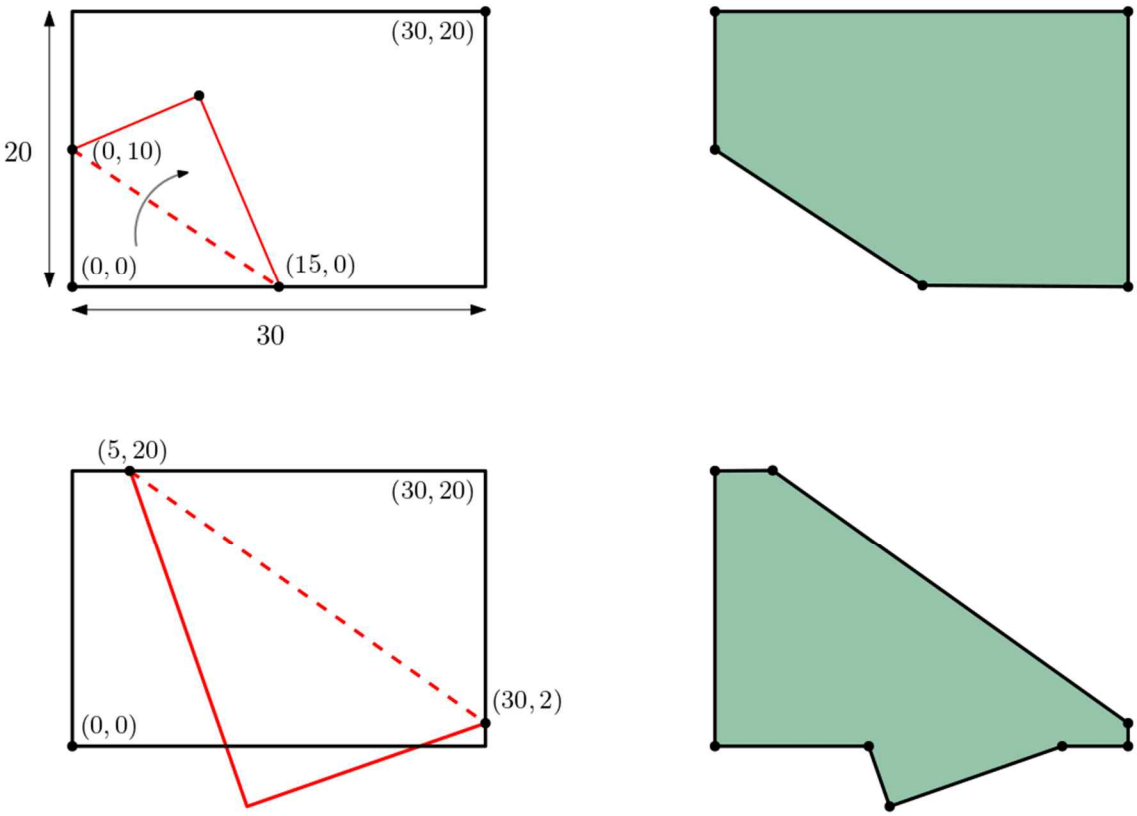
Sample Input 3	Output for the Sample Input 3
WRITING	50.00

Practice C

Paper Folding

Time Limit: 1 Second

A rectangular paper is placed in the first quadrant with its vertex at the origin as shown in the figures below. We will fold this paper along a line which is specified by two points on the different edges of the paper. You are asked to compute the area of the 2D polygonal shape of the folded paper. In the following figures, you can see two papers and the corresponding fold lines (red dotted) and the folded paper (green polygon).



Input

Your program is to read from standard input. The first line contains two integers w and h, $10 \leq w, h \leq 1{,}000$ which represent the width and height of a rectangular paper, respectively. The coordinates of the two points that determine the folding line will be given at the second line and the third line. Note that two folding points are on the two different edges of the rectangle.

Output

Your program is to write to standard output. Print exactly one integer that is the integer part of the area of the folded polygon. For example, if the area of the folded polygon is 56.678, then you should print 56.

The following shows sample input and output for two test cases illustrated above.

Sample Input 1	Output for the Sample Input 1
30 20 0 10 15 0	525

Sample Input 2	Output for the Sample Input 2
30 20 5 20 30 2	397