# The Magic Snake

During the CEOI excursion to Sächsische Schweiz an invisible magic snake whispers to you, "I will give you points if you can guess my length". You do not see the snake, so that initially you know nothing about its location. But you hear the snake again: "I am moving away from you on a straight line that starts at your current position. But I remain within your snake hearing distance." Luckily, you know that this distance measures 12 122 length units. Therefore, you decide to divide the line into 12 122 units, numbered from 0 to 12 121, 0 being your position.

Now, the snake speaks to you again: "You may ask me questions, two at a time. Each question must be of the following form:

> Does your body cover unit $U$?"

For each of your questions, the snake promises to give you one of the following responses:

> Yes, some part of my body covers unit $U$ at the moment.
>
> OR
>
> No, I have not reached unit $U$ yet.
>
> OR
>
> No, I have already moved beyond unit $U$ ☺.

To make matters worse, the snake may move **forward** by exactly $K$ units just before it answers your question pair. Its movements do **not** have to follow a pattern; it might especially move in a way that annoys you.

When you have gained enough information, you can estimate its length. It might not be possible to determine its length exactly, but you are awarded points if the snake's actual length **differs by no more than $K$ units** from your guess.

## Example

In Figure 1 we find a magic snake of length 5 units. You know that its "speed value" $K$ is 1 unit. You do not know that it initially covers units 4 to 8, and that it decided to move at every second pair of questions.

Now it is time to ask questions. In your first question pair, you ask the snake whether it covers units 0 and 11, respectively. It answers that it has already moved beyond unit 0, but it has not reached unit 11 yet.

Of course, you need to ask another pair of questions. This time, you ask about units 2 and 10. The snake moves 1 unit and answers that it has already moved beyond unit 2, but it has not reached unit 10 yet. At this moment, you are certain that the snake's length is between 1 and 7 units, and its body is located between unit 3 and unit 9, inclusively.

True to its plan, the snake does not move before answering your third pair of questions, which ask about units 4 and 9. The snake answers that some part of its body covers unit 9, but it has moved beyond unit 4. Now you can conclude that the head of the snake is at most at unit

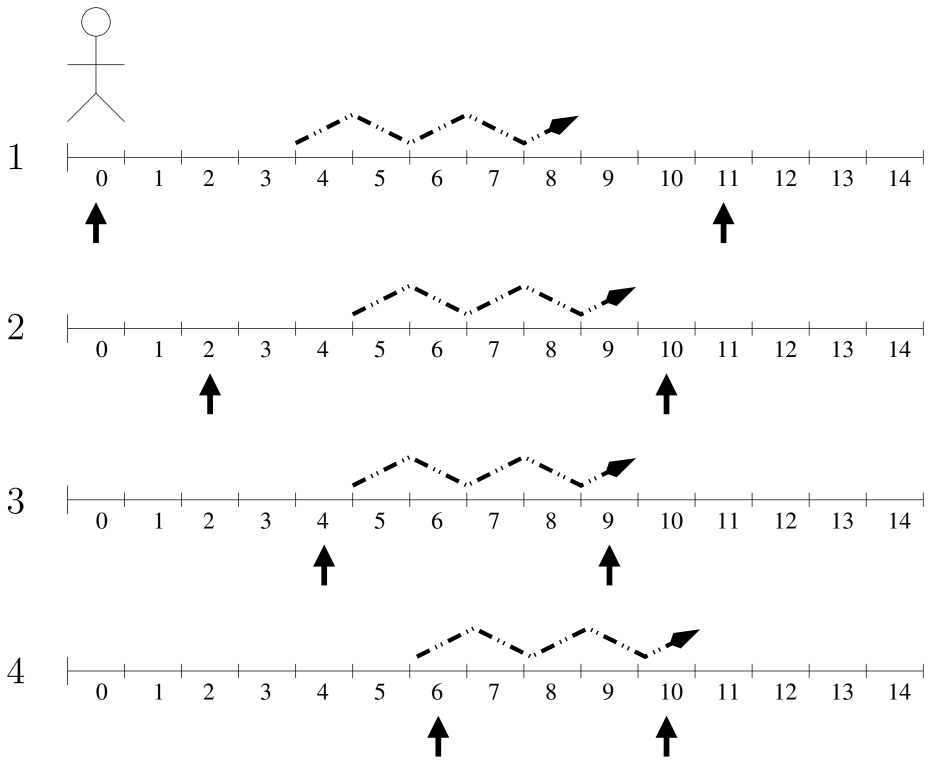

Figure 1: A possible scenario with a magic snake of length 5.

10 (remember that you do not know when the snake moves), and we can also deduce that the snake's length is between 1 and 6 units.

Since the snake can move 1 unit away, you decide to ask at your fourth turn if the snake covers units 6 and 10. The snake moves 1 unit away and then replies that some parts of its body cover unit 6 and unit 10. You conclude that the snake may be 5 or 6 units long. (The possible locations of the snake's body are 5 to 10, 6 to 10 and 6 to 11. It is impossible to have the entire snake's body covering units 5 to 11, since it would have answered in the previous query that a part of its body covered unit 4.) So now you have a sufficiently good estimate of the snake's length.

## Interaction

Your program must use a library to interact with the snake. The library provides the following functionality:

- `function get_speed() : longint` / `int get_speed()` — it returns the constant distance $K$ the snake may move before it answers your question pair.
- `procedure ask_snake(U1, U2 : longint, var A1 : char, var A2 : char)` / `void ask_snake(int U1, int U2, char *A1, char *A2)` — Your program may ask questions to the snake by calling this function. `U1` is the unit parameter for the first question.

U2 is the unit parameter for the second question. A1 is set to one of the following values as the answer to the first question.

- ‘s’ if a part of the body of the snake covers that unit.
- ‘f’ (for *front*) if the snake has not yet reached unit U1.
- ‘b’ (for *behind*) if the snake has already moved beyond unit U1.

A2 is set to the answer to the second question in the same way.

- `procedure tell_length(L : longint)` / `void tell_length(int L)` — Call this function to tell the snake your estimate of its length L.

Your program must call the library functions in the following order:

1. First, it calls `get_speed`. This function may be called only once.
2. Then, it calls `ask_snake` as many times as it needs to gather information (possibly 0 times).
3. Finally, it calls the function `tell_length` to tell the snake your guess of its length. This call will terminate your program.

Your program must not read or write any files, and it must not use standard input/output nor try to access any memory outside your program.

## Programming and Compile Instructions

If you submit Pascal source code, the source code must contain the statement

```
uses snakelib;
```

Use the following command to compile your program

```
ppc386 -O2 -dCONTEST -XS snake.pas
```

If you submit C/C++ source code, the source code must contain the line

```
#include "snakelib.h"
```

Use one of the following commands to compile your program

```
gcc -std=c99 -O2 -DCONTEST -static snakelib.c snake.c -o snake
g++ -std=c++98 -O2 -DCONTEST -static snakelib.c snake.cpp -o snake
```

Please note that the grading system uses the compiler options mentioned on the overview sheet.

## Mock Library and Sample Program

To let you experiment with your solution, you are given a sample library, the sources of which are in `snakelib.pas`, `snakelib.c`, and `snakelib.h` in your home directory. You can modify this library, but we suggest you do not change its interface.

The mock library behaves as follows:

- When `get_speed` is called for the first time, the library opens the file `snake.in` and reads the data of the snake. `snake.in` is formatted as follows: In its only line, it contains three space separated integers: $Len$, $Tail$, and $K$, which denote the actual length of the snake, the closest unit covered by the snake, and the constant possible displacement of the snake, respectively. These integers are checked against the constraints given in the end of this problem statement.
- When `ask_snake` is called, the library answers as specified before, and at every second pair of questions, the snake moves.
- The library halts the program when `tell_length` is called, and it judges whether your estimate length is good enough.
- The library prints an error when a function is called out of order and terminates the program.
- The library outputs debug output to `snake.log`.

You are also provided with two simple programs (`snake.pas` and `snake.cpp`) and a sample input file (`snake.in`) that illustrate the library usage.

## Example Run

| **Function Call** | **Return Value(s) and Explanation** |
|---|---|
| `get_speed();` | Returns `1`. The snake may move 1 unit immediately before it answers a pair of questions. |
| Pascal<br>`ask_snake(0, 11, A1, A2);`<br>C or C++<br>`ask_snake(0, 11, &A1, &A2);` | `A1='b', A2='f'`. You ask if the snake covers units 0 and 11, and the snake answers it has already moved beyond unit 0, and it has not reached unit 11 yet. |
| Pascal<br>`ask_snake(2, 10, A1, A2);`<br>C or C++<br>`ask_snake(2, 10, &A1, &A2);` | `A1='b', A2='f'`. You ask if the snake covers units 2 and 10, and the snake answers it has already moved beyond unit 2, and it has not reached unit 10 yet. |
| Pascal<br>`ask_snake(4, 9, A1, A2);`<br>C or C++<br>`ask_snake(4, 9, &A1, &A2);` | `A1='b', A2='s'`. You ask if the snake covers units 4 and 9, and the snake answers it has already moved beyond unit 4, but part of its body covers unit 9. |
| Pascal<br>`ask_snake(6, 10, A1, A2);`<br>C or C++<br>`ask_snake(6, 10, &A1, &A2);` | `A1='s', A2='s'`. You ask if the snake covers units 6 and 10, and the snake answers parts of its body cover units 6 and 10. |
| `tell_length(6);` | You guess the length of the snake to be 6 units. Because its actual length is 5, your estimate is sufficient. |

## Constraints

$0 \leq K \leq 10$. There are 12 122 units, numbered from 0 to 12 121. The snake moves only forward, but it never leaves the hearing range. For its length $Len$ it holds that $1 \leq Len \leq 12\,122$.

## Scoring

You get full score for a test if your estimate guess differs at most $K$ units from the real length, and your program calls the `ask_snake` function at most 13 times. If your program uses more than 13 calls, but less than 42 calls to arrive at a good estimate, then the snake awards you only half of the allocated score for that test. Otherwise, the snake gives you nothing. If your program violates the interaction protocol during an execution of a test, you will receive 0 points for that test.