Problem 1 With a Little Bit of Luck (4%)

This is an interactive task.

Eliza's father holds an arithmetic progression (等差級數) of length n, where $2 \leq n \leq 10^9$. He asks you to guess the common difference (公差) of the sequence he holds so that he can finish the parade to his wedding ceremony.

Usually this is an easy task. However, Eliza's father shuffled the sequence and now it is in random order.

> " Tell me the index you want to know during the parade, one at a time, I'll tell you the number in that position. "

On the parade to Eliza's father's wedding ceremony, you only have 8 chances to ask him and reveal the secret common difference. Use your chances well.

Interaction

The following describes the interaction between your program and Eliza's father (the jury program).

- In the beginning of the interaction, Eliza's father will tell you the length of the sequence he has, an integer n between 2 and 10^9.
- To make an inquiry on the i^{th} element of the sequence during the parade, print a line containing

  ```
  ?  %d
  ```

  That is, a question mark followed by a space and the integer i to query.

  To ensure no I/O delay, flush the I/O buffer immediately with

  ```
  fflush(stdout);
  ```

  After printing your inquiry (to the `STDOUT` stream), read the response of the jury program from the `STDIN` stream. The response will be of the following forms:

  1. " `%lld` "

     In response to your inquiry, provided that the index you give is valid, Eliza's father will print the element in that position in a line. Note that it is an integer of type `long long int`.

     Note that it contains one string token consisting of that number. Also note that, if the index you query is not valid, the response will be an arbitrary integer.

Problem 1 continued on next page. . .

2. “ `Attempt Limits Exceeded!` ”

   If your program has made 8 queries and tries to make one more query, the judge will respond you this message. (Note that it contains 3 string tokens in total.)

   In this case, your program should terminate immediately, and you will get the ”Wrong-Answer” verdict.

- When your program is ready to answer the secret common difference, print a line

  ```
  !  %d
  ```

  That is, a exclamation mark followed by a space and the integer to answer.

  To ensure no I/O delay, flush the I/O buffer immediately with

  ```
  fflush(stdout);
  ```

  After printing your answer, read the response of the jury program from the `STDIN` stream. The response will be of the following forms:

  1. “ `Correct.` ”

     When you have made the correct guess, the judge will respond this message.

     Note that it contains one string token. In this case, your program should terminate immediately and you will get the ”Correct” verdict.

  2. “ `Wrong Answer.` ”

     If your program gives a wrong guess, the judge will respond you this message. (Note that it contains 2 string tokens in total.)

     In this case, your program should terminate immediately, and you will get the ”Wrong-Answer” verdict.

Notes

Note that, the interaction is made via the standard I/O streams `STDIN` and `STDOUT`.

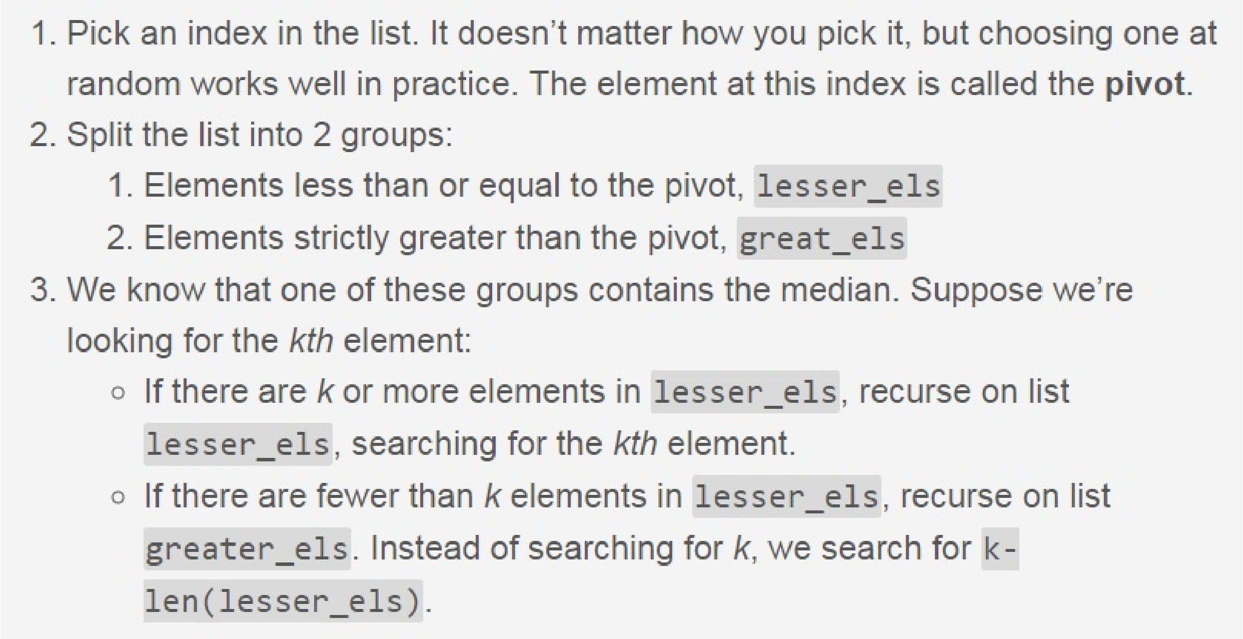
It is guaranteed that Eliza’s father does hold an arithmetic progression of length n, and he will respond honestly according to the rule described above.

Problem 2 Quick-Select (6%)

Read the following description, and implement the *quick-select* algorithm.

Quick-Select

To select the k^{th} element from an *un-ordered* sequence in expected linear time.



1. Pick an index in the list. It doesn't matter how you pick it, but choosing one at random works well in practice. The element at this index is called the **pivot**.
2. Split the list into 2 groups:
    1. Elements less than or equal to the pivot, `lesser_els`
    2. Elements strictly greater than the pivot, `great_els`
3. We know that one of these groups contains the median. Suppose we're looking for the *kth* element:
    - If there are *k* or more elements in `lesser_els`, recurse on list `lesser_els`, searching for the *kth* element.
    - If there are fewer than *k* elements in `lesser_els`, recurse on list `greater_els`. Instead of searching for *k*, we search for `k-len(lesser_els)`.

Input

The first line contains an integer n, $(2 \leq n \leq 10^7)$, the length of the sequence.

The second line contains n integers, the elements in the sequence separated by a space.

The third line contains an integer k, $(1 \leq k \leq n)$, the order of the element to select.

Output

Print the k^{th} smallest integer of the sequence.

Example

Input

```
7
3 2 1 6 5 2 8
4
```

Output

```
3
```