Lab 1

Problem Statement 1

A sequence of n distinct integers is said to be ***cyclically sorted*** if it was originally sorted in strictly increasing order and then rotated (shifted) some number of times. Formally, let a strictly increasing array be $A = [a_1, a_2, \ldots, a_n]$, such that $a_1 < a_2 < \cdots < a_n$. If we choose an index i $(1 \leq i \leq n)$, then the cyclically sorted version of A is: $B = [a_i, a_{i+1}, \ldots, a_n, a_1, \ldots, a_{i-1}]$. Given such an array B of n distinct integers, find the minimum element in the array. *You are guaranteed that the minimum element is unique.*

`Input`

- The first line contains an integer n, the number of elements in the array.
- The second line contains n integers representing the cyclically sorted array B.

`Output`: A single integer: the index of the smallest element in the array.

Example

`Input`

```
7
30 40 50 10 20 25 28
```

`Output`

```
3
```

Explanation

The array $[30, 40, 50, 10, 20, 25, 28]$ is a cyclic rotation of $[10, 20, 25, 28, 30, 40, 50]$. The smallest element is 10, which occurs at index 3.

Constraints

- Use an algorithm with $\mathcal{O}(\log n)$ time complexity.

Problem Statement 2

Given a sorted sequence of n **distinct** integers $a_1, a_2, \ldots, a_n$, determine whether there exists an index i such that: $a_i = i$. Here, the index i is considered to be **0-based** (i.e., $i \in \{0, 1, \ldots, n-1\}$). You are required to determine whether such an index exists, and if so, output it.

`Input`

- The first line contains an integer n, the number of elements in the array.
- The second line contains n integers representing the sorted array $a_0, a_1, \ldots, a_{n-1}$, with all elements distinct and in strictly increasing order.

`Output`

If there exists an index i such that $a_i = i$, output the index. If there are multiple such indices, output any one of them. If no such index exists, output `-1`.

Example

`Input`

```
5
-3 -1 1 3 5
```

`Output`

```
3
```

Explanation

At index 3, the element is 3, which satisfies $a_3 = 3$.

Constraints

- The array is sorted in strictly increasing order.
- All elements are distinct.
- Your algorithm should run in $\mathcal{O}(\log n)$ time.

Choclate Problem 1

You are given a sorted array of n distinct integers $A = [a_0, a_1, \ldots, a_{n-1}]$ in strictly increasing order, and a target value z. Your task is to determine the index of z in the array using a search strategy that mimics how we naturally search for a page number in a book estimating the likely position of the target using interpolation.

Understanding Interpolation Search Through a Book Analogy

Consider the way we open a book when we are searching for a certain page number. Say the page number is 200, and the book appears to have around 800 pages. Page 200 is therefore approximately at the one-fourth mark, and we use this knowledge as an indication of where to open the book. We will probably not hit page 200 on the first try; suppose instead that we land on page 250. We now reduce the search to the range from page 1 to page 250. The desired page (200) is about 80 percent of the way between the new range's endpoints. Thus, we try to go back about one-fifth of the way from page 250. We can continue this process until we get close enough to page 200 that we can flip one page at a time. This is exactly the idea behind **interpolation search**. Instead of cutting the search space by a fixed half (as in binary search), we cut it by an amount that seems most likely to succeed, based on the target's estimated position within the current range.

To do this efficiently, use this strategy:

- At each step, estimate the position of z using:

$$\text{pos} = \text{low} + \left\lfloor \frac{(z - A[\text{low}]) \cdot (\text{high} - \text{low})}{A[\text{high}] - A[\text{low}]} \right\rfloor$$

- Then update the search interval accordingly:
  - If $A[\text{pos}] == z$, return pos.
  - If $A[\text{pos}] < z$, search in the right subarray.
  - If $A[\text{pos}] > z$, search in the left subarray.
- Repeat the process while z lies within the current bounds.

If z does not exist in the array, return -1.

`Input`

- The first line contains two integers n and z, the number of elements in the array, and the target value.
- The second line contains n space-separated integers $a_0, a_1, \ldots, a_{n-1}$ in strictly increasing order.

`Output`: An index of z in the array if it exists, or -1 otherwise.

Example

```
Input

7 25
10 15 20 25 30 35 40

   Output

3
```