beCP
2026

# Task 1.1: The Rotated Locks (rotation)

Author: Bruno Ploumhans
Time limit: 1 s Memory limit: 16 MB

*Note: This task is interactive. Please look at the special instructions to implement, compile and test your program. Don't hesitate to ask the staff for help.*

In the ancient walled city of Khiva, Uzbekistan, archaeologists have discovered a vault with $N$ locks arranged on a circle, numbered from 1 to $N$ around the ring. Now they are asking for your help. Here is the information that they have gathered. Long ago, each lock was assigned a distinct secret value. The values are simply the numbers 1 through $N$, but they have been rotated; that is, the sequence 1, 2, 3, ..., $N$ was placed on the circle and then rotated by some unknown offset $k$. So lock number 1 holds the value $k+1$, lock number 2 holds $k+2$, and so on, wrapping around back to 1 after $N$...

The vault opens when you identify the offset $k$ by which the circle was rotated. Unfortunately, local guardians don't allow you to inspect any lock's value directly, as they claim you might damage the mechanism. Instead, you can only ask a guardian to answer comparison queries: you give him two lock positions $i$ and $j$, and he tells you which of the two holds the smaller value. The guardians have limited patience, so using a limited number of queries your task is to find by how much the circle was rotated.

## Interaction Protocol

You need to implement a function `solve(int N)` which will be called by the grader. Your `solve` function will receive the integer $N$ as an argument. Within this function, you can use the following function provided by the interaction library:

- **`int ask_guardian(int position1, int position2)`**.
  - Call this function with two positions. The two positions must be between 1 and $N$, inclusive, and different from each other.
  - The function returns one of two positions given to it: the one of the lock which holds the smaller value.
  - You may call this function at most **60** times.

Once you have determined the offset $k$ by which the numbers on the locks were rotated, your `solve` function must return it to report your answer. You must ensure $0 \leq k < N$.

## General limits

- $2 \leq N \leq 100,000,000$
- $0 \leq k < N$
- You may call `ask_guardian` at most 60 times.

## Additional constraints

| Subtask | Points | Constraints |
|---|---|---|
| A | 27 | $2 \leq N \leq 60$. |
| B | 24 | $2 \leq N \leq 100$. |
| C | 49 | No additional constraint. |

## Sample Interaction

Here is an example interaction for $N = 5$ and $k = 4$. In this case, the locks have the following values:

| **Lock Position** | **Lock Value** |
|---|---|
| 1 | 5 |
| 2 | 1 |
| 3 | 2 |
| 4 | 3 |
| 5 | 4 |

Say we decide to perform the following series of calls:

| **Your Call** | **Return Value** | **Guardian's Report** |
|---|---|---|
| `ask_guardian({1, 2})` | `2` | The smallest of the two locks is at position 2. |
| `ask_guardian({2, 3})` | `2` | The smallest of the two locks is at position 2. |
| `ask_guardian({3, 4})` | `3` | The smallest of the two locks is at position 3. |
| `ask_guardian({2, 5})` | `2` | The smallest of the two locks is at position 2. |

From this interaction, we deduce that since the lock at position 2 is smaller than all the other ones it must have a value of 1. We can thus deduce that the numbers were rotated by $k = 4$.

## Local Testing with Sample Grader

### How to Use

1. Save the `rotation.h` and `grader.cpp` files in the same directory as your solution file (which should be named `rotation.cpp`).
2. Compile the sample grader along with your solution using:

   ```
   g++ -std=gnu++11 -O2 grader.cpp rotation.cpp -o rotation
   ```

3. Run the compiled program:

   ```
   ./rotation
   ```

4. The sample grader will prompt you to enter the value of $N$ and $k$ (e.g., `5 4`). Enter these values separated by spaces.
5. Your `solve` function will then be called and will interact with the sample grader's implementation of `ask_guardian()`. The sample grader will output the result of your submission.
6. Alternatively, you can use the sample file:

   ```
   ./rotation < sample1.in
   ```

**Important Notes:**

- The sample grader is for local testing only and might not enforce all the constraints of the official grader (e.g., the exact query limit).
- You should only submit your `rotation.cpp` file to the CMS.