Lab Assignment: Counting Rotations in a Circularly Sorted Array

Problem Statement

Given a circularly sorted array of integers with no duplicates, determine how many times the array has been rotated in the anti-clockwise direction.

A rotation shifts each element to the left by one position, and the first element moves to the end. The number of rotations is equal to the index of the smallest element in the rotated array.

Examples

Example 1:

Input: nums = [8, 9, 10, 2, 5, 6]

Output: The array has been rotated 3 times

Example 2:

Input: nums = [2, 5, 6, 8, 9, 10]

Output: The array has been rotated 0 times

Explanation

In a circularly sorted array, the smallest element indicates the rotation point. Its index gives the number of rotations.

Logic: Modified Binary Search

To solve the problem efficiently in O(log n) time, we adapt the binary search approach.

The key observation is that the smallest element is the only one whose previous and next elements are both greater than itself. By comparing the middle element with its neighbors and the boundaries of the current search interval, we can determine the position of the smallest element.

Function Signature (C Language)

```
int countRotations(int arr[], int size);
```

Lab Assignment: Counting Rotations in a Circularly Sorted Array

Task

- Implement the countRotations function in C using the binary search logic.
- The function should return the number of rotations in the array.
- Do not use linear search. Aim for O(log n) complexity.