Lab Assignment: Binary Search in a Rotated Sorted Array

Problem Statement

Given a rotated sorted array (i.e., an array that was initially sorted in ascending order but then rotated), write a function to search for a specific target element and return its index. If the target is not present in the array, return -1.

Examples

Example 1:

Input: arr = [4, 5, 6, 7, 0, 1, 2], target = 0

Output: Index of target: 4

Example 2:

Input: arr = [4, 5, 6, 7, 0, 1, 2], target = 3

Output: Index of target: -1

Explanation

In a rotated sorted array, at least one half (either left or right) of the array is always sorted. By identifying the sorted half and checking if the target lies within it, we can narrow the search space efficiently in each step.

Logic: Modified Binary Search

The key idea is that in a rotated sorted array:

- One half is always sorted.
- We can determine whether the target lies in the sorted half or the unsorted half.

By narrowing down the search space using these observations, we can maintain O(log n) complexity.

Function Signature (C Language)

```
int searchInRotatedArray(int arr[], int size, int target);
```

Task

Lab Assignment: Binary Search in a Rotated Sorted Array

- Implement the searchInRotatedArray function in C using binary search.
- The function should search for the target element and return its index.
- Do not use linear search. Aim for O(log n) complexity.
- Recursive or iterative implementation is allowed.