Task 8: Circular Queue Binary Search

Consider a circular queue (implemented using a fixed-size array) where the elements are sorted but have been rotated at an unknown index. Describe an approach to perform a binary search for a given element within this circular queue.

1. Initialize Pointers: Set two pointers, low and high, to the start and end of the array respectively.
2. Binary Search Loop:
   - While low is less than or equal to high:
     - Calculate the mid-point mid as low + (high - low) / 2.
     - If the element at mid is the target, return mid.
     - Determine if the left half or the right half is properly sorted:
       - Left Sorted: If the element at low is less than or equal to the element at mid:
         - If the target is within the range [low, mid], move high to mid - 1.
         - Otherwise, move low to mid + 1.
       - Right Sorted: If the element at mid is less than the element at high:
         - If the target is within the range [mid, high], move low to mid + 1.
         - Otherwise, move high to mid - 1.
3. Return Not Found: If the loop ends without finding the target, return an indication that the target is not in the array (e.g., -1).

Implementation -

```
public class CircularQueueBinarySearch {
    public static int search(int[] nums, int target) {
        int low = 0;
        int high = nums.length - 1;

        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (nums[mid] == target) {
                return mid;
            }

            // Determine if the left half is sorted
            if (nums[low] <= nums[mid]) {
                // Target is in the left half
                if (nums[low] <= target && target < nums[mid]) {
                    high = mid - 1;
                } else {
                    low = mid + 1;
            }
        } else {
            // The right half is sorted
            if (nums[mid] < target && target <= nums[high]) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
    }

    return -1; // Target not found
}

public static void main(String[] args) {
    int[] circularQueue = {10, 15, 20, 0, 5};
    int target = 5;
    int result = search(circularQueue, target);
    System.out.println("Index of target " + target + ": " + result); // Output should be 4
}
}
```