**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. **Identify the Rotation Index:**
   - First, we need to find the rotation index, which represents the point where the sorted array has been rotated. This can be done using a modified binary search algorithm.
2. **Perform Binary Search:**
   - After identifying the rotation index, we can treat the circular queue as two sorted arrays.
   - We can then perform a binary search on both halves of the circular queue separately.

## JAVA code

```
public class CircularQueueBinarySearch {
   public static int search(int[] nums, int target) {
   int rotationIndex = findRotationIndex(nums);
   int left = binarySearch(nums, target, 0, rotationIndex - 1);
   int right = binarySearch(nums, target, rotationIndex, nums.length - 1);
      return (left != -1) ? left : (right != -1) ? right : -1;    }
   private static int findRotationIndex(int[] nums) {
      int left = 0;
      int right = nums.length - 1;
      while (left < right) {
         int mid = left + (right - left) / 2;

         if (nums[mid] > nums[right]) {
            left = mid + 1;
         } else {
            right = mid;
         }
      }
      return left;
 }
```

```
    private static int binarySearch(int[] nums, int target, int left, int right) {
        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return -1; // Element not found
    }

    public static void main(String[] args) {
        int[] nums = {4, 5, 6, 7, 0, 1, 2};
        int target = 0;

        int index = search(nums, target);
        System.out.println("Index of " + target + " in the circular queue: " + index);
    }
}
```

# Explanation of the Code

1. **Find Rotation Index Method:**
   - This method finds the rotation index of the circular queue using a modified binary search algorithm.
   - It compares the middle element with the last element of the array to determine whether the rotation point lies to the left or right of the middle.
   - The rotation index is returned when the search space is narrowed down to a single element.
2. Binary Search Method:
   - This method performs a standard binary search on a sorted array.
   - It searches for the target element within a specified range of the array.
3. **search Method:**
   - This method coordinates the search process.
   - It finds the rotation index of the circular queue and then performs binary searches on the two halves of the queue separately.
   - If the target is found in either half, its index is returned; otherwise, -1 is returned.
4. **Main Method:**
   - Demonstrates the usage of the search method with a sample circular queue (nums) and target element (target).