Question :

Leet code for Search in Rotated sorted Array

Solution:

```
class Solution {

    public int search(int[] nums, int target) {

         int l = 0;

    int r = nums.length - 1;



    while (l <= r) {

      final int m = (l + r) / 2;

      if (nums[m] == target)

        return m;

      if (nums[l] <= nums[m]) { // nums[l..m] are sorted.

        if (nums[l] <= target && target < nums[m])

          r = m - 1;

        else

          l = m + 1;

      } else { // nums[m..n - 1] are sorted.

        if (nums[m] < target && target <= nums[r])

          l = m + 1;

        else

          r = m - 1;

      }

    }



    return -1;


    }}
```

Question : Leet code for find-first-and-last-position-of-element-in-sorted-array.

Solution:

```
class Solution {

    public int[] searchRange(int[] nums, int target) {

    final int l = firstGreaterEqual(nums, target);

    if (l == nums.length || nums[l] != target)

      return new int[] {-1, -1};

    final int r = firstGreaterEqual(nums, target + 1) - 1;

    return new int[] {l, r};

  }



  private int firstGreaterEqual(int[] A, int target) {

    int l = 0;

    int r = A.length;

    while (l < r) {

      final int m = (l + r) / 2;

      if (A[m] >= target)

        r = m;

      else

        l = m + 1;

    }

    return l;


    }

}
```