44. 4. Search in Rotated Sorted Array

There is an integer array nums sorted in ascending order (with distinct values). Prior to being passed to your function, nums is possibly rotated at an unknown pivot index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n�1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For example, [0,1,2,4,5,6,7] might be rotated at pivot index 3 and become [4,5,6,7,0,1,2]. Given the array nums after the possible rotation and an integer target, return the index of target if it is in nums, or -1 if it is not in nums. You must write an algorithm with O(log n) runtime complexity. Example 1: Input: nums = [4,5,6,7,0,1,2], target = 0 Output: 4

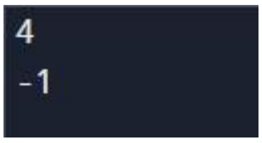
Example 2: Input: nums = [4,5,6,7,0,1,2], target = 3 Output: -1

AIM: To search an element in Rotated sorted array

PROGRAM:

```
def search(nums, target):

   left, right = 0, len(nums) - 1

   while left <= right:

      mid = (left + right) // 2

      if nums[mid] == target:

         return mid

      if nums[left] <= nums[mid]:

         if nums[left] <= target < nums[mid]:

            right = mid - 1

         else:

            left = mid + 1

      else:

         if nums[mid] < target <= nums[right]:

            left = mid + 1

         else:

            right = mid - 1

   return -1

nums1 = [4, 5, 6, 7, 0, 1, 2]

target1 = 0

print(search(nums1, target1))

nums2 = [4, 5, 6, 7, 0, 1, 2]

target2 = 3
```

```
print(search(nums2, target2))
```

OUTPUT: 

```
4
-1
```

TIME COMPLEXITY: O( log n)