33. Search in Rotated Sorted Array

APPRACH=>

=> MAKE 2 CASE FOR PIVOT (MID>MID+1)->MID.).   MID<MID-1=> RETURN MID-1

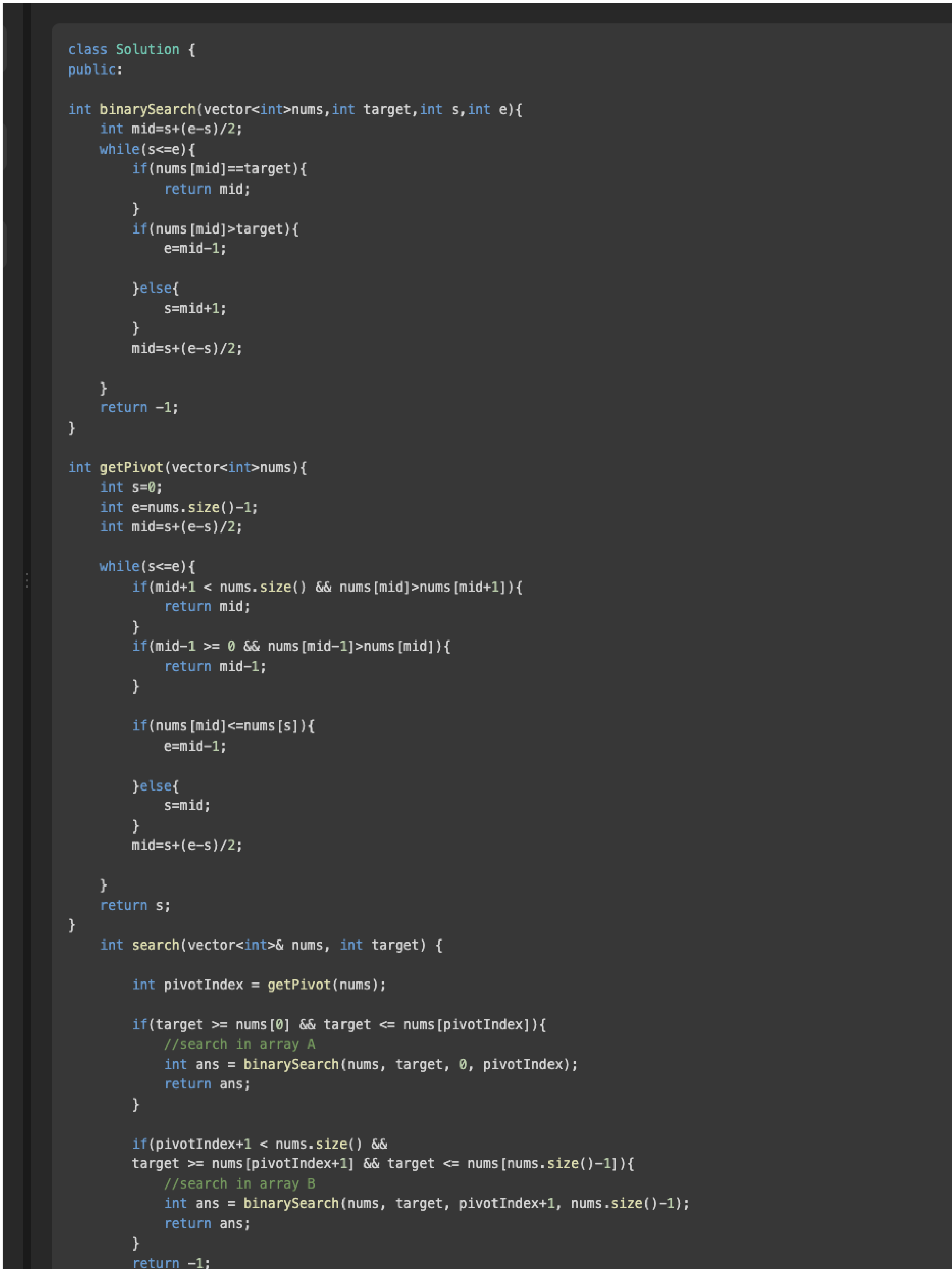
=> 1 CASE FOR MOVING LEFT.   (MID>=S. ->S=MID;ELSE E=MID-1;

=> CASE FOR MOVING RIGHT

/// EDGE CASE

-> IF ONLY ELEMNT MEANS S==E

THEN RETURN S OR E COZ 1 ELEMNT MEANS IT IS THE PIVOT;



```cpp
class Solution {
public:

int binarySearch(vector<int>nums,int target,int s,int e){
    int mid=s+(e-s)/2;
    while(s<=e){
        if(nums[mid]==target){
            return mid;
        }
        if(nums[mid]>target){
            e=mid-1;

        }else{
            s=mid+1;
        }
        mid=s+(e-s)/2;

    }
    return -1;
}

int getPivot(vector<int>nums){
    int s=0;
    int e=nums.size()-1;
    int mid=s+(e-s)/2;

    while(s<=e){
        if(mid+1 < nums.size() && nums[mid]>nums[mid+1]){
            return mid;
        }
        if(mid-1 >= 0 && nums[mid-1]>nums[mid]){
            return mid-1;
        }

        if(nums[mid]<=nums[s]){
            e=mid-1;

        }else{
            s=mid;
        }
        mid=s+(e-s)/2;

    }
    return s;
}
    int search(vector<int>& nums, int target) {

        int pivotIndex = getPivot(nums);

        if(target >= nums[0] && target <= nums[pivotIndex]){
            //search in array A
            int ans = binarySearch(nums, target, 0, pivotIndex);
            return ans;
        }

        if(pivotIndex+1 < nums.size() &&
        target >= nums[pivotIndex+1] && target <= nums[nums.size()-1]){
            //search in array B
            int ans = binarySearch(nums, target, pivotIndex+1, nums.size()-1);
            return ans;
        }
        return -1;
```