Divide & Conquer QUESTIONS

Question 1 : Apply Merge sort to sort an array of Strings. (Assume that all the characters in all the Strings are in lowercase). (**EASY**)

Sample Input 1 : arr = { "sun", "earth", "mars", "mercury" }
Sample Output 1 : arr = { "earth", "mars", "mercury", "sun"}

Question 2 : Given an array nums of size n, return the majority element. (**MEDIUM**)

The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Sample Input 1 : nums = [3,2,3]
Sample Output 1 : 3

Sample Input 2 : nums = [2,2,1,1,1,2,2]
Sample Output 2 : 2

Constraints (extra Conditions):

- n == nums.length
- 1 <= n <= 5 * 104
- -109 <= nums[i] <= 109

Question 3 : Given an array of integers. Find the Inversion Count in the array. (**HARD**)

Inversion Count: For an array, inversion count indicates how far (or close) the array is from being sorted. If the array is already sorted then the inversion count is 0. If an array is sorted in the reverse order then the inversion count is the maximum.
Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.

Sample Input 1 : N = 5, arr[ ] = {2, 4, 1, 3, 5}
Sample Output 1 : 3, because it has 3 inversions - (2, 1), (4, 1), (4, 3).

Sample Input 2 : N = 5, arr[ ] = {2, 3, 4, 5, 6}
Sample Output 2 : 0, because the array is already sorted

Sample Input 3 : N = 3, arr[] = {5, 5, 5}
Sample Output 3 : 0, because all the elements of the array are the same & already in a sorted manner.

(**Hint** : A sorting algorithm will be used to solve this question.)
Note - This question is important. Even if you are not able to come up with the approach, please understand the solution.