Project: Count Inversions

Problem Description

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

Example: The sequence 2, 4, 1, 3, 5 has three inversions (2, 1), (4, 1), (4, 3).

Requirement

- Design should be based on divide and conquer.
- Running time should NOT be worse than $\Theta(n \log n)$.
- Must use recursion to solve the subproblems.

Input Specification

Your program should read from **an input file**, which starts with a line indicating the number of test cases. Each of the following lines indicates a test case, and each test case includes a sequence of numbers (separated by spaces) for which you need to count inversions.

Sample Input

```
3
2 4 1 3 5
1 2 4 8 9 3 5 6
1 20 6 4 5
```

Output Specification

For each test case, print one line with a format of `The sequence has ? inversions` in **console**.

Sample Output

Below is the correct output for the previous sample input.

```
The sequence has 3 inversions.
The sequence has 7 inversions.
The sequence has 5 inversions.
```

Hint

- The design can be based on merge sort.
- How to get number of inversions in merge()?

Happy coding!