1. WAP that takes 2 sorted (in non-increasing order) integer arrays as input, then merges the two arrays into one array sorted in non-increasing order in O(n+m).

Sample input	Sample output
4 5 3 2 1 5 7 6 3 2 1	7 6 5 3 3 2 2 1 1
5 6 4 3 2 1 3 6 4 0	6 6 4 4 3 2 1 0

2. WAP that takes n integer numbers, sorts them using Merge sort.

Sample input	Sample output
5 6 2 3 3 5	2 3 3 5 6
6 5 6 7 8 0 1	0 1 5 6 7 8

3. WAP that takes n integer numbers, and counts the number of inversions in the array using divide and conquer algorithm in O(nlogn). Two elements a[i] and a[j] form an inversion if i < j and a[i] > a[j].

Sample input	Sample output
4 6 1 3 2	4
6 5 6 7 8 0 1	8

In sample 1, the inversions are
Index (0,1) -> (6,1) because 0<1 and 6>1.
Index (0,2) -> (6,3) because 0<2 and 6>3.
Index (0,3) -> (6,2) because 0<3 and 6>2.
Index (2,3) -> (3,2) because 2<3 and 3>2.
Total 4 inversions.

Note: If you are unable to solve the 3rd problem here's a link to help you. But first please try hard on your own and think about what you can do. How to divide the problem and how to calculate inversion for the current array.

https://www.interviewbit.com/blog/count-inversions-of-an-array/ Look approach 2.

4. WAP that takes n integer numbers and an integer k, and checks if there are two numbers in the array which sums to k. You have to do it inside the Merge Sort function, divide and conquer fashion in O(nlogn).

Sample input	Sample output
4 6 1 3 2 5	YES
6 5 6 7 8 0 1 16	NO

In the first sample, k = 5, a[2] + a[3] = 3 + 2 = 5.
In the second sample, k = 16, No pair of elements sum to 16. It is not allowed to use the same index twice. For example a[3] + a[3] = 8 + 8 = 16 but we cannot use index 3 twice.