1 Here in the mergesort function we split the array in half and give it to the merge function that gives us a sorted list and finally when all the small merged sorted ~~lists~~ arrays are merged it gives us one merged array / and prints in output file.

2 Here we split the array and give it to merge function (modified) that gives us the largest value of the given small arrays and finally gives us the largest number by comparing the data returned by the first split's max's and output it in the output file.

3 Here we used merging , counting and sorting algorithum, Whenever I found a swap I added a count. [I < j] (Inversion) And after adding all the swaps together we find the total swaps we did.

④ Here I started a while loop from 1 to n as we are using 1-based indexing. And Traversed through the array as $i, j=i+1$ until n. and saved the maximum value while $i<j<=n$ and whenever we hit $j=n$ we reset $i+=1$ and j to $j=i+1$ and run it again. And finally returned the max value to output file.

⑤ Here we used the Quicksort algorithm and took the last value as pivot. And whenever we sort we traverse and sort according to pivot by putting all the smaller values of pivot to left then pivot then larger values and continue this unless we have only 1 element left in both smaller arrays. (subarray) Here we sort according to one pivot at a time by making their position right.

6 Here we use the partitioning part of Quicksort method and choose a pivot. As we are doing 1 based indexing if we find k-th smallest value directly if the length of the left sublist is k-1, so Kth value = pivot. And if k-1 is larger than len(left) then recursively apply the same thing on the right array and if k-1 is smaller then it means it would be on the left array and apply the method on left array recursively. And do not operate on the unnecessary arrays as we just want to find the k-1th value and not sort the array.