Problem 1 (Order Statistics) **YASIN AYDIN**

(a) The best asymptotic running time complexity of comparison based algorithms is $O(nlogn)$. We can use the $Merge - Sort$ algorithm since it gives the $O(nlogn)$. After sorting, we need to iterate over the numbers from the smallest to the k th item so we do not need any order-statistics algorithm for that. This operation will cost $O(k)$ so in total the running time complexity is $O(nlogn) + O(k)$.

(b) The order-statistics algorithms I would use is $SELECT$ since it has complexity of $O(n)$. So, using $SELECT(k)$ we can obtain the k th smallest element and the partition that has elements that are smaller than the k th smallest number. After that, we can use $Merger - Sort$ to sort the partitioned list with $O(klogk)$ running time complexity. In total, we have complexity of $O(n) + O(klogk)$.

(I used $Merge - Sort$ for both sub questions because there was no constraints about the space complexity.)

I would use the second method because when we are using the $SELECT$ order-statistics algorithm we don't have to deal with the values bigger then k th value which are decreasing the run-time and the necessary space for the $Merge - Sort$, especially when n is very large number.

Problem 2 (Linear-time Sorting) **YASIN AYDIN**

(a) $Assumption$: The characters(letters) in the strings are all uppercase letters so that we can use ASCII values to compare characters without any problem.

We can use the array of arrays instead of just one array. Let us call the array of arrays the $mainArray$. Each array in $mainArray$ will be ASCII values of corresponding letters in the strings (e.g. "YASIN" = [89, 65, 83, 73, 78]). The size of arrays will be same and this size will be determined by the longest strings' size. The empty spaces left in the remaining shorter strings will be filled with the smallest ASCII value of the English alphabet (e.g. "YASIN" will become "YASINA" = [89, 65, 83, 73, 78, 65]) to maintain the lexicographic order (e.g. "A" should be left of the "AZ" because of the length). Other than that the algorithm will behave same as $Radix - Sort$ so we will compare the ASCII values starting from the right to left, ordering them in ascending order.

(b) The longest strings are "VEYSEL" and "ZEYNEP" which are length 6 so the size of the arrays will be 6. With this property the shorter strings such as "ALI" will become [65, 76, 73, 65, 65, 65]. At each iteration the algorithm will sort the $mainArray$ based on the current index which will be from 5 to 0. Performing these operations will give us the sorted string array as shown below.
[86, 69, 89, 83, 69, 76] - VEYSEL
[65, 76, 73, 65, 65, 65] - ALI
[83, 69, 76, 73, 78, 65] - SELIN
[89, 65, 83, 73, 78, 65] - YASIN
[90, 69, 89, 78, 69, 80] - ZEYNEP
Start: [VEYSEL, ALI, SELIN, YASIN, ZEYNEP]
Iteration 1: [ALI, SELIN, YASIN, VEYSEL, ZEYNEP]
Iteration 2: [ALI, VEYSEL, ZEYNEP, SELIN, YASIN]
Iteration 3: [ALI, SELIN, YASIN, ZEYNEP, VEYSEL]
Iteration 4: [ALI, SELIN, YASIN, ZEYNEP, VEYSEL]
Iteration 5: [YASIN, SELIN, ZEYNEP, VEYSEL, ALI]
Iteration 6: [ALI, SELIN, VEYSEL, YASIN, ZEYNEP]

(c) Given the $k =$ characters in English alphabet , $d =$ the length of the longest string , $n =$ the number of strings in the $mainArray$ (the size of $mainArray$).
The asymptotic running time of the algorithm is $\Theta(d(n + k))$ if the stable sort it uses takes $\Theta(n + k)$.