Nabonita Saha
22301645

Lab - 03

Task 01 : We took the approach of merge sort in lab 2 Task 2 . It follow the divide and conquer approach following a time complexity of $o(n \log n)$.

Task 2 : Here the divide and conquer approach has been set where we mapped the string numbers into list of integers named 'alice'. This list/array divided itself recursively and in its smallest form of single term it continued to return the current maximum value through comparing it through a condition.

The recursive call faces $o(\log n)$ whereas based on the input size the input size it has to face $o(n)$. Since $o(n)$ is the dominant one, we can say the time complexity to be $o(n)$.

Task 03 :

Here, we took the help of 'count' function that counted the number of times there was a smaller number in the right side of a given number recursively. We returned that number as the result and to do so we took the help of Try - except blocks which basically handled the errors that might had been faced due to I/O errors, value errors etc.

Task 04 :

Here the "maximum" function works on finding max 1 ~~that~~ and max 2 through looping finding the greatest and second most greatest number in the list 'a' and returning arr[max] (or greatest num)] ** 2 arr[max 2 (second most greatest num)] ; further errors ~~and~~ are controlled through try and error block.

Task 05:

Based on the code snippet we implemented quickSort. It further worked with "partition" function where based on conditioning the elements were swapped alligning to ascending order.

Task 6:

Here, quick select algorithm has been in use, which basically works to find the k^{th} smallest element within an unsorted array. Quick select is a bit different from Quicksort as it only focuses on the partition that contains the k-th element.

This function focuses on finding numbers or elements smaller than the pivot, equal to the pivot and greater than the pivot. Based on the recursive conditioning, it works to find the k^{th} smallest term.