Name: Pranto Roy Lab - 3. ID: 22301261

Task - 1

Here I used merge sort to sort the given array of numbers. Firstly in mergeSort() function I divided the list in such a way that we get a single element then I send the last 2 list values to merge() function compared each values using left, right pointers and then insert them in a new list.

Task - 2

In this task I used find_max() function which is a modification of merge_sort. Then from that function we call max() in this function we find the max number that was send from find_max() recursively. Here the time complexity of the over all code is O(n log n) logn for find_max() and n for max function.

Task-3

Here in task 3 used divide and conquer method to find the total number of pairs. Here the list is divided into 2 parts or 2 halves and then is a counter which counts the number of valid pairs in each halves recursively using the merge sort function and the merge function merges the elements keeping a constant counting. So by modifing merge sort a little I accomplish the task by divide and conquer method.

Task-4

Here I also used the divide and conquer method i.e the merge sort but modified version. In the mergesort function(). after dividing when there are 2 values I take the sum and store if from both side then the list is sent to Calculations(). Then we find the max value of left list and right list by using abs() to get rid of (-) and append the reqired value

in another list and find their max and finally we get a value as designated in the question $A[i] + A[j]^2$. Here time complexity is $O(n \log n)$

Task - 5

In this task I just converted the algorithm into coding language. Its a a code of quick sort. Here I set the last number as pivot and the numbers are changed on the basis of the pivots.

Task - 6

In this task I used a modified ~~final~~ version of quick sort to find the k^{th} smallest number in a list for each query. Here I select a pivot element from the array and partitioning the other elements into two sub arrays

according to whether they are less than or greater than the pivot. The pivot is their its final sorted position and if k equals to pivots position, we return the pivot. And then I get the following outputs.