LAB 2   Nabonita Saha
22301645

Task 1 :

(1)

Here, we used a nested loop approach to get a time complexity of $o(n^2)$. A flag was handled so we could determine the existence of the sum from the elements,

(02)

Here, we took the help of two pointer approach. Within a single loop we tried to navigate the position of sum by incrementing the index of the left one or decrementing the index of right one. In case right index turned out to be smaller than the left index, this is where the loop would no more be in repeat.

Task 02 :

(1)

Approach of merge sort has been used here, The function 'merge sort' worked on dividing the whole list into smallest forms of sublist whereas function 'merge' worked with arranging the numbers in ascending order comparing those sublists to each other. Merge Sort produces a time complexity of O(nlogn).

(02)

I have taken the approach of two pointer method in order to convert this task to a solution of o(n). I compared each element of both the lists once at a time and the number that was ~~sa~~ smaller was appended to a new list further incrementing the index for that list. Once the index number reached the length of the list, we worked on appending the next elements remaining within the list.

Task 03:

Here, for convinience I took the help of merge sort to arrange the list in ascending order based on their end time. Then within a loop I compared the last most element appended within the new list making sure the end time remains less than or equal to the starting time of the comparing element.

<u>Task 04</u> :

To implement this greedy algorithm we worked with rearranging the list based on the finishing time with the help of merge sort. Afterwards we considered looping where we set condition regarding what would be done for 2 persons or 3 person to do the maximum amount of tasks.