Task 1a

Here we need to find the index to calculate a tentative sum We built a function and ~~eventually~~ check if sum of two elements ~~is~~ equals to tetative sum. Then, return the value in a list. Otherwise, we write impossible.

Task 1b

Here we can can use two pointers to avoid inner looping system. As the array is already sorted, to find the tentative sum we start one pointer from begining of ~~the~~ the array and another one from the end of it. We modify the iterators according to temorarily adding two pointer's values. ~~If~~ If two ~~sum~~ temorary sum is equals to tentative sum then we

return those it indecies. Otherwise, if the it two iterator becomes equals to each other or start becomes bigger than end, we terminte the loop and say Impossible as output.

Task 2a

Here we need to sort a merged array in nlogn time complexity. So we simply merge the two array and use .sort() to find our result.

Task 2b

We initiate two pointers for both of the arrays and another one for the merged array for which we used an empty array. We ~~change~~ compare the data with the iterators and change them accordingly, we insert those values after that in the empty array. Here

(P. T. O)

We are doing everything in one loop. So the time complexity becomes $O(n)$.

Task - 3

Firstly, we sort all the tasks using the end hour of every task in ascending order. We store a current pair from the task list, compare if the current pair's end hour is lesser or equals to the next task. If it is, we store the tasks and increase our count.

Task-4

Firstly, we sort all the tasks in ascending order according to the end time of all task. Then, we initialize an array with all the individual having their own task slot. Here we can not assign random tasks for all the individuals, we need to calculate the maximum one. To make a proper current pair array; Then we compare all the start time of list tasks with current pair's end hour. Eventually we get our maximum output.