TASK1(A)

At first, we read from the given text file. Then separate the size, target sum and list. Then perform a for loop until the size, take an empty list to append the index, and take the flag as false. And then we took another for loop from i+1 to size. In this loop, we are measuring the sum of the immediate two values of the array. If the sum matches the target sum, then append its index+1 to the list and change the flag into true. And if the flag remains false, then print ‘impossible’. Finally the index is written in the output text file. We are performing two loops so the time complexity for this code is O(n^2).

TASK1(B)

At first, we read from the given text file. Then separate the size, target sum and array. Then take an empty list , two pointers i=0 and j=size-1. Perform a while loop in the list to measure the sum of the two immediate values with multiple conditions. If i and j are equal, then print ‘impossible’, if the measured sum is equal to the target sum, then append its index+1 to the list, if the measured sum is less than the target sum , then increase i and if it's greater than the target sum, then decrease j. Finally the index is written in the output text file. We are performing one loop so the time complexity for this code is O(n).

TASK2(A)

We started by reading from the provided text file . Next, separate the two lists and the size. Initially, we added two lists to combine them.

Run a loop until the combined lists length is reached, then save the results as an integer in a variable. Sort them using the sort function, then print the sorted list. Finally, the output text file contains the sorted list. This code's time complexity is O(nlogn).

TASK2(B)

We started by reading from the provided text file. Next separate the two lists and the size. First , take an empty list and two pointers i and j. Next run a while loop to go through each value in the two lists while applying multiple conditions so that the values can be compared and added to the list in sorted order.Finally, the output text file contains the sorted list. This code's time complexity is O(n).

TASK3

We divided the input into two parts after reading it from the provided text file. Sort the tasks according to their end times, ascending. Next, take a variable and an empty list. Verify the person's start and end times and continue the for loop until the size is reached. It checks whether the user's previous tasks' end time is greater than the start time of any of the remaining tasks. Count the people who are available. Finally the output text file contains the counted people.

TASK4

We started by reading from the provided text file. First loop iterates over the whole list, second loop iterates the list that contains the counter of the end time of the last task, third loop checks if the end time of the last task is equal to the remaining tasks start time. If so, the task is then assigned to that person and the task index is appended to another list, which contains all the completed tasks index and counter is increased. In the second loop, it is checked if the end time of the last task is greater than any of the remaining task start time. If so, the task is then assigned to that person and the task index is appended in another list and the counter is increased.