Explanation

Task 1a:

In this task, I first took the total loop time and targeted value. Later I used nested loop to check two integers value if they add up to the targeted value and inserted their index in a list. If list is empty it means no numbers can add up to be the targeted value. Otherwise it would print the index of the numbers whose sum is equal to target.

Task 1b:

For this code, the given time complexity is O(N). Therefore, I cannot use nested loop. Rather I used 2 pointer inside a while loop. 1 pointer started from the begining and the other one from the end. If the sum is bigger than the targeted value then the ending pointer is decreased otherwise the beginner pointer is increased.

Task 2a:

In this task, I took 2 sorted list and appended them in a new list. Later, I used sort function to sort them.

Task 2b:

In this task, the time complexity is O(n). So, I cannot use sort function anymore. Rather, I have to use two pointers for two sorted list and compare the values. Smaller values will get inside the new list first and then the larger values. If any items are left they will be iterated again and put into the new list as the old list is sorted properly.

Task 3:

For this task, I took the starting time in a list and ending time in another list. Later I used a pointer and sorted the ending time first and later the starting time accordingly. Then I used greedy algorithm and took the first pair of number. Later, I had to check if my previous work's ending time and next job's starting time clashed or not. If they didn't clash, I took on the job and put them in a list otherwise I skipped it.

Task 4:

In this task, I put the starting time and ending time in seperate lists and sorted the ending time and the starting time after that accordingly. Later, I used three loops to distribute work. I checked if a task's ending time clashed with other job's starting time or else I assigned the work to him and lastly counted the total work that can be done.