Lab Assignment 02

Task-01

1. Yes, I can write an O(N^2) solution to solve the problem by using a nested loop. In nested loop, the first loop run for N times and the second loop run again for N*N times. To calculate the time complexity, we take the maximum number of time which is (N^2) and for this reason, the first problem of task 01 can be done with O(N^2).
2. I can solve this problem with the solution of O(N). To solve this problem, I create a function where I send the array, sum and length of the array. I take an empty dictionary and run a for loop where I calculate the remaining number from the sum by substituting the current index number value of the array from the sum. Then I check whether the value is in the dictionary or not. If in the dictionary then return the number which is in the dictionary and the current index, otherwise add the current index value and index in the dictionary. After all the process if we do not find the desired value and index then return "impossible".

Task-02

1. For this problem, I use the algorithm of merge sort which follows the divide and conquer procedure. In the divide and conquer procedure, we have to divide a large problem into a short problem which takes O(logn) time complexity and then we do the combine part where it takes O(n) time complexity. So, when the procedure is done properly it takes O(nlogn).
2. As I have two sorted lists, I can use them to create a one ascending order sorted list. In this case, I run a while loop where I compare less than value then other value and append in the new list. After completing the loop, I add the rest of the elements of the sorted lists if remaining.

Task-03

In this task, the question gives me an algorithm of a merge sort and so I just use the algorithm of the merge sort and complete the task. Merge sort algorithms follow the divide and conquer procedure. Furthermore, this sort has the time complexity of O(nlogn).

Task-04

The time complexity of my code is O(nlogn).

In this task, I use the divide and conquer procedure where I divide the array into small sizes by using a recursive method and conquer it, until it becomes a 1 length array in the findMax function. After it, I get 2 arrays, one is the left one and the other one is the right one. I send both arrays in the maxValue function where I find the max value in the array. So, in this process, I get the max value from an array.