Task 1

(a) The problem is to find the index of two values whose sum is equals to k. So, at first, in first loop from 0 to before last value we have held every value and compared with all the values after that value. As, we have to visit every value 2 times the time complexity is $O(n^2)$.

(b) Now, we have stored the values and their corresponding index in a dictionary as (key, value) pair. Now, If we find (k-key) for any key then we can collect their index. as the answer. As, we have to traverse the value of array one time to store in dictionary and then traverse one time again in dictionary for checking. So, time complexity is $O(n)+O(n)$ or $O(n)$.

Task 2

Alice and Bob have two sorted list. we have to concade them and sort them.

(a) To do this in O(nlogn). we first concade the two list. Then we used merge sort as we know merge sorts time complexity is O(nlogn). It first divide the array into two. then make them sort and then merge them from small to large in O(n) time.

(b) We know that, the ~~lists~~ lists are already sorted. ~~There for~~ Therefore, before concading. if we compare them which one is smaller, then we can sort it in O(n).

Task - 3

We know merge sorting is deviding the list into half, then, sord them individual. Then, merge them from smallest to largest from two of the sorted list. In my code, I have done exactly the same thing - I have ~~merged~~ divide it until it's ~~before~~ length become 1. Then, if ; compare and merge, then we will have sorted list using merge sord.

Task 4

To find the max value in a list. We divided the list into half. we keep doing it until its length become 1. then if we find two values, we will compare which one is greater. At last, we will find max value. The comparison's ~~the~~ time complexity is O(1). and ~~to do~~ the time complexity to divide in half is O(logn). Therefore, total time complexity is O(logn)