Task 1a:

For this task we traverse the whole array using two loops, checking each of the combinations, if we find a match we immediately make the flag false, print the output and break the inner loop. As the flag is false, the outer loop will also break. If we don't find any match after the end of two loops we print "Impossible". The complexity of this code is $O(n^2)$.

Task 1b: For ~~this soft~~ solving the task in O(nlogn) time, we used binary search, so we sorted the array. We checked each of the value binary searching. slicing the array without the value and (sum-value) as k of the key, when we find the match, we compare the original array ~~the~~ and getting indexes from the original array and printed it.

Task 2a: For nlogn time complexity we used merge sort, we concatenate the two list the sorted the whole list with merge sort. The complexity of merge sort is O(nlogn)

Task 2b: For o(n) solution we just the two array and comparing the two array with each other and sorting them, putting them into a array and returning the array, also we used another loop just in case for any leftout values, as there are not nested loop used. The complexity is o(n).

Task 3:

We just sorted the given list with a merge sort algorithm which follow devide and conquer approach, we devide the array using recursion and conquering while sorting the array with merge() function.

Task 4: Using the devide and conquer method for finding the max value, we devide the ~~value~~ array ~~with recursion~~ using recursion and conquering while returning the maximum value and at the end we find max value. The time complexity for this code is : $O(\log_2 n)$.