Task1

The function here takes in the list of integers, the length of that list, the targetted value, and also the output file. The algorithm goes through the list to find two integers that add up to the targetted sum. This is done through nested looping, with one loop selecting the first integer, while the second loop looking for a match, with the positions of the pair being written onto the output file if a pair is found. If no pair is found, the 'Impossible' is written to the output file

Task2

The function here merges in two sorted lists of varying lengths into a single sorted list. It initiates an empty list called 'merg', where it stores the result. After one list is run out, it appends the other list into 'merg', with the merg file being then shipped off to the output file.

Task 3

This function organises tasks based on intervals of time. ~~by at~~ To begin with it sorts out a the list provided through the input text based on their end times, and then goes through this sorted list using ~~at~~ a loop, adding tasks to an empty list if their start time is greater than or equal to the previous end times, and if the duration of the task is less than 4.

After going through the entire loop, it counts the number of tasks which it writes onto the output file, followed by the selected intervals.

Task 4

This function is basically a greedy algorithm that specifies how many tasks can be completed by a specific number of people available. It takes in the number of people, tasks, the list of task intervals and the output file as input. It sorts through the list of tasks based on ~~the~~ their end-times, and then goes through the sorted list, assigning each task to the first available person, taking count of completed tasks. The total count is then loaded off to the output file.