1(a). The code contains nested while loops. The outer while loop runs for int(a[0]) iterations, where int(a[0]) is the value of the first element in the input file. The inner while loop also runs for int(a[0]) iterations. Within the inner loop, there is a constant time operation of checking if the sum of two elements is equal to the given sum. A flag variable is used to check if the condition does not meet for the whole input array. Here the dominating operation is the nested loops. Therefore, the overall time complexity of the code is O(n^2).

1(b). The code contains a while loop that iterates from l=0 to r=int(a[0])-1. The loop condition is l<r, which means the loop will run for n iterations, where n is the value of r. Inside the loop, there are constant time operations such as array element access, integer addition, and comparison. Therefore, the overall time complexity of the code is O(n), where n is the value of r.

2(a). The code reads the input file which has some lines of string. The code then splits the strings into arrays, which take O(n) time. The arrays are then concatenated, which takes O(n) time. Finally, the concatenated array is sorted using the sorted() function, which has a time complexity of O(nlogn). Therefore, the overall time complexity of the code snippet is O(nlogn).

2(b). The code contains a while loop that iterates a maximum of len1 + len2 times. Within the loop, there are constant time operations such as comparisons and appends. Therefore, the overall time complexity of the code is O(n), where n is the sum of len1 and len2.

3. The code snippet reads the input from a file and stores it in a list. Then, it sorts the list based on the second element of each sublist using the lambda function. Sorting a list of length n has a time complexity of O(nlogn). After sorting, it iterates through the sorted list once, checking if the first element of each sublist is greater than or equal to the end_time, than it increases task variable and updates end_time with the current end time and adds selected tasks to the seleted_task variable. This iteration has a time complexity of O(n). Therefore, the overall time complexity of the code snippet is O(nlogn) + O(n), which simplifies to O(nlogn).

4. The code contains two nested loops. The first loop iterates 'person' number of times, and the second loop iterates 'len(new)' number of times. Here, Every time the loop iterates it checks if 0 indexed task is selected for any person or not by checking in selected_task than if not selected it appends into the selected task and deletes from the main array. Then, the current main array is copied and sorted in order of the starting time. Then the next loop checks if the last selected task's ending time overlaps with the new one or not. If not it appends it in the selected tasks array and deletes it from the main array. At last, we get the final array with all selected tasks. Because two nested loops are used here the overall time complexity is O(n^2).