Explanation

Lab 7

ID: 19201089
Section: 9

## Task 1

Here, first we insert all the tasks into a list and then sort them based on ascending order of ending time. Then, we iterate through this list. If the list is empty, the first task will be added to a separate list called 'p'. Once this is done, the next task will be judged in this way:- difference = current task's starting time - previous task's ending time. If difference is positive, the task is added to 'p' list. This check is done to ensure that the tasks don't clash with one another.

## Task 2

We get the number of people, p from the input file. So, we create p number of empty lists inside a larger list called ~~tasks~~ 'total'. The tasks are inside a list tasks and sorted based on their ending times. Now, we iterate over the tasks. The very first task is sent to the first empty list in 'total'. The other tasks check if the inner lists are not empty iteratively. The difference is calculated in like task 1. If the list is empty, the difference is set to a negative number as it is not going to be used. The goal is to find the most optimal person to give the task to so that the difference is the least but non-negative. So, we have another array called appe pos where we insert the pair (person, difference) to keep track of who has the most

optimal situation. If this pos array was not empty, we find the pair with the least difference and assign the task to that person/ inner list. However, if not positive difference exists (clashes everywhere); if there is an empty array, the task will be sent there. We are also keeping count of how many tasks are being assigned. Pos is reset at the end of each loop.

## Task 3

We have a getParent function that recursively finds the parent of all nodes. Initially, the parent of each node is themselves. We iterate through every connection and first get both node's parents. If their parents are different, it means they aren't friends. so, we set the parent of node 1 to the parent of node 2. and also node 2. We have also kept a list for friends of all nodes. so, friends of parent 1 will be incremented by friends of parent 2. The output is parent 1's friends at every step.