Task-01:
Opening the input file, no of tasks is stored in N and tasks with starting and ending times are stored in tasks list. Then the tasks are sorted according to their ending time in ascending order.

Schedule list is initialised and the first task is assigned to temp. Temp is appended to schedule and then all the tasks are **compared with temp whether the starting time of the tasks are greater than the ending time of temp. If found, temp is updated and the task is appended to schedule**. The length of temp is assigned to task_count.

The task count is written in the output file and the tasks in schedule are written in the output file as well.

Task-2:
Opening the input file, no of tasks is stored in N, no of people in M and tasks with starting and ending times are stored in tasks list. Then the tasks are sorted according to their ending time in ascending order.

A nested list, schedule is initialized with an empty list for each person and a list, end is initialized with a 0 for each person. The first task is assigned to variable, st and it is appended to the first list in schedule and the end time for first person is updated to the end time of first task.
Inside a for-loop with range equal to tasks, a sentinel value is assigned to difference and then inside a for-loop with range equal to number of persons, **the difference between starting time of next task and ending time of the last task assigned to each person is stored in interval variable. If the ending time of the person's last task is smaller than the starting of next task and interval is smaller than the difference, person variable is updated and the difference is updated with the current interval.** Outside the for loop with range of persons, the task is appended to the list of the last updated person with the **least interval** and the person's ending time is updated with the ending time of the task. Thus, all the possible tasks are assigned to the lists of all the people.

For each element in schedule, the task count is found and written in the output file.

Task-03:
Opening the input file, the number of people in village is stored in N and the number of friendships is stored in K and a list, Q is created with the connections.

A list, X is created with each person as single elements individual lists. A find function is written which takes a nested list and an integer and returns the list in the nested list which has the integer as an element.

For each of the connections, calling the **find function** with the connection elements as integers and X as nested list, the lists are returned and compared. **If they are not equal, the two lists of the two elements are concatenated and becomes a new list, new and the previous individual lists are removed from X and the new is appended to X**. The length of new is written in the output file each time. If the lists for the two elements of the connection is same, the length of that list is written in the output file.