**Task 1:**

The code reads tasks with start and end times from 'input1.txt'. It sorts the tasks by their end times and selects a set of non-overlapping tasks using a greedy approach and append the selected task in to list1 finally the length of list1 will be written to 'output1.txt'. Because len(list1) will tell us how many task a person can take without overlapping

**Task 2:**

This code reads task intervals and the number of available workers from 'input2.txt'. It sorts the tasks by their end times and assigns tasks to workers in a way that maximizes the number of tasks performed. It then writes the count of assigned tasks to 'output2.txt'. The goal is to efficiently assign tasks to workers so that tasks with earlier start times are prioritized and each worker handles at most one task without overlap.

**Task 3:**

This code reads the number of people and the number of queries from 'input3.txt'. It processes queries to build and track groups of friends. It uses a union-find data structure to manage friend groups efficiently. For each query, it finds or creates the friend group and writes the size of the group to 'output3.txt'. The code calculates and records the sizes of friend groups based on the provided queries, aiming to maximize the group sizes as friends are connected.