Goal:

- Learning multi-threaded workload distribution
- Multi-Threaded Communication

Description:

In this problem, you will be given a variable *K (1<=K<=100)* denoting the number of work servers. Then, you will be given *K* lines of information. In each line, you will be given two values a and b denoting the name of the server *N* (a string containing English small letters, |N|<=50), and an integer value *T (1<=T<=10000)* denoting the time in milliseconds, respectively. The value of T denotes that, this server *N*, can take any task *x* that requires at most *T* amount of time. If *t* denotes the completion time of task *x*, then t<=T. You should use, wait () or sleep () to implement this feature of processing a request. If any task *x* requires time *t* where t > T, then the server *N* can not be designated with this task.

After giving the information of the servers, you will be given *X(1<=X<=1000)* tasks. Each task is associated with two information *C* and *D*, denoting the ID of the task and the required time to process this task in milliseconds, respectively. As already stated, you can attach a task *x (1<=x<=X)* with a server N, if and only if, the maximum processing time of *N* matches or exceeds *x*'s processing time.

Your main challenge in this problem is that, given the tasks, you must attach them to the servers. Each server can be considered as a thread. Also, periodically, you need to print some information denoting the last completed tasks for each server along with currently ongoing tasks for each server. A sample input format can be found below.

Submission Format:

I would highly encourage you to write the codes using as many files as possible for the clarity of the coding. But, while submitting the solution, please compile all the code in one file, so that, I can run on my machine through a single command (Javac Roll_9.java, Java Roll_9). Also, please name the main class (Public class) as public class Roll_x, where x denotes your actual roll.

Penalty/Reward:

This problem evaluates your knowledge of understanding the threaded communication between multiple threads. So, it is important to understand, how dynamically threads are created, controlled, resumed, and suspended. This assignment sheds light on these. So, while implementing, please focus on these.

3 a 500 b 45678 c 34 10 1 100 2 6789 3 12 4 9087 5 1234 6 5 7 1234 8 9876 9 87 10 32	Outputs will vary based on your design to distribute the works. You are open to work as per your aesthetics.