CS3523 Spring 2018 Final Exam
Extending Producer Consumer Problem: The Hungry Tribe problem

Description

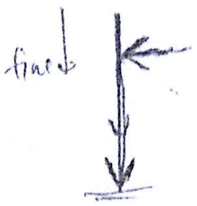
A tribe consisting of **K** tribesmen eat communal dinners from a large pot that can hold **M** servings of cooked food. There is a single cook who cooks the food. When a tribesman (or tribeswoman) wants to eat, he helps himself from the pot, unless it is empty. If the pot is empty, the tribesman wakes up the cook and then waits until the cook has refilled the pot.

A tribesman performs two things: Either eat or perform community service. Any number of tribesman threads run the following code for tribesmen. An unsynchronized tribesman code is shown below:

```
while (true) {
        getServingFromPot();          // hungry phase
        eat();
        community-service();          // working phase
}
```

And one cook thread runs this code. An unsynchronized cook code:

```
while (true) {
        putServingsInPot(M);          //
        cook-sleep();                 // Cooks & then goes to sleep
}
```



The synchronization constraints are:

- Tribesmen cannot invoke getServingFromPot if the pot is empty. Initially the pot has **M** servings of the food.
- The cook can invoke putServingsInPot only if the pot is empty.

You have to implement this solution using **semaphores.**

A total of number of **K** threads are created in the system. Each thread corresponding to a tribesman performs only two activities: Eating or doing Community Service. A tribesman thread eats & does community service for units time that is exponentially distributed with an average of **α** and **β** milliseconds respectively.

The cook thread executes two activities: Put **M** servings into the Pot or cooking (& sleeping). Again the time taken for these activities are exponentially distributed with an average of **λ** and **μ** milliseconds respectively.

Your program terminates once all the **K** tribesmen threads have finished eaten **n** times.

Input

The input the program is a file that contains all the parameters, in this format:

K = **50**
M = **10**
n = **20**
α = **15**
β = **20**

$\lambda = 10$
$\mu = 5$

Output

The output file is a log of all the actions performed by all the threads. It is of them form – Time:Action. A sample of the output is here:

```
time0: K threads created.
time1: T1 becomes hungry
time2: T2 becomes hungry
time3: T1 eats from the Pot
      .
      .
      .
time30: T1 has eaten n times. Hence, exits.
time31: T2 has eaten n times. Hence, exits.
      .
      .
      .
time50: Last thread exits
      .
      .
      .
```

In addition to the output, you have to compute these two metrics:

- Average waiting time to eat: The metric 'average waiting time' is average of the waiting times taken by all the threads present in the system to read.

Submission Instructions

Please submit the following documents:

- a readme file, describing how to execute your program
- the program to execute
- a design document describing the design of your program

Please zip all these documents and name it as: <Roll-No.>-Tribe.zip. Then upload it on the google classroom to be created later.