CSCI 460: Operating Systems—**Assignment 3 (8 marks)**

This assignment is on priority inversion (page 458-460), well, a simplified version of it. You could use any of the common languages C, C++, Java, Python, etc. Assume the following:

a. There are three kinds of jobs with priorities $P(T_1) = 3$, $P(T_2) = 2$ and $P(T_3) = 1$. A job with a higher priority means it can preempt another one with a lower priority immediately, except that T_1 cannot preempt T_3 as they share a common devise.

b. T_1 shares a small buffer of length 3 with T_3, with an initial value of $< 0, 0, 0 >$ at time 0. When T_1 gets hold of it, it changes the content to $< 1, 1, 1 >$ and print out a new line: T_1`111`T_1`.` When T_3 gets hold of it, it changes the content to $< 3, 3, 3 >$ and prints out a new line: T_3`333`T_3`.` Each of T_1 and T_3 takes 3ms (1ms = 1 slice) to run, and you can assume that the enforcement of mutex, etc, takes no time.

c. T_2 shares nothing with T_1 and T_3. When T_2 runs to completion with no interruption, it prints out a new line: T_2`NNNNNNNNNN`T_2`.`, where 'N' means "normal". Each T_2 takes 10ms to run. When T_1 preempts T_2, T_2 will print out the number of N's according to the actual time it has run (in the number of slices). For instance, T_2 runs at time 0, then there is a request at time 6 for T_1, then T_2 should print out: T_2`NNNNNN`T_2`.` Then, T_1 takes over at time 6. Same rule applies to T_3, when it is preempted by T_2.

d. The input sequence of jobs are in the form $\langle arrival\_time, j \rangle$, where $j = 1, 2, 3$ is the index for T_j. When you have a sequence of three inputs: $< 1, 1 >, < 3, 2 >, < 6, 3 >$, the output should be like:

`time 1,` T_1`111`T_1`.`
`time 4,` T_2`NNNNNNNNNN`T_2`.`
`time 14,` T_3`333`T_3`.`

(1) To make the debugging easier, you could use random sequences of at most 10 job requests. Implement the version with priority inversion, i.e., a higher priority job can always interrupt a lower priority job (except that T_1 cannot preempt T_3 as they share a common buffer). Print out one scenario when the priority inversion occurs.

(2) For the following sequence of input print out your output: $< 1, 3 >, < 3, 2 >, < 6, 3 >, < 8, 1 >, < 10, 2 >, < 12, 3 >, < 26, 1 >$.

Date Due: **11:30pm on Monday, November 28, 2016 (on or before 11:30pm, Nov 28, 2016).** Load your source code and output as two separate files on D2L in the folder **Assignment 3**, preferably in the form of **family_name − 3.java** and **family_name − 3.output** (assuming that you are using Java).