Fall 2015 **EECS 338 Assignment 5** **Due:** November 5, 2015

G. Ozsoyoglu *Posix Thread programming*

The Readers-Writers Problem with Starving Writers; 100 points

In this assignment you will use Posix threads to implement the readers-writers problem where "writers may starve" in the sense that a stream of readers can arrive to "read" (as long as there is already a reader reading) even when there is a writer waiting to write. Note that this problem was covered in the class, and you can use the semaphore-based solution in *slide 38* of the slide set *3.2.a.BasicAlgorithmsAndSemaphores.2015.9.17.pdf.*

Your solution consists of one process with sufficiently many streaming reader/writer threads arriving to read and write randomly. For each action within the threads of the process, print a message to the screen containing the thread_id tid of the involved thread and describing the action involved.

You should test your code with multiple execution scenarios; and your execution scenarios should clearly demonstrate the correctness of your implementation. For each action, print a message to the screen containing the thread_id of the involved thread, the action, the current time, and the state of the thread. Also, do not forget to conform to the assignment grading policy requirements listed at http://art.case.edu/338.F15/grpolicy.html

Requirements and Hints:

- Make sure that your shared variables are mutually exclusively modified and/or accessed.
- You will need the *rand()* and *srand()* functions (seed *srand()* with the current time).

Run your program in the script environment, just like in the previous assignments. Remember to error-check all system calls: check return values for success, and use *perror* when possible on failure.

On the due date, submit your code, Makefile, and program output to Blackboard as Assignment 5.