Spring 2017 **EECS 338 Assignment 7** **Due:** April 3rd, 2017 (Mon)

POSIX Thread Programming

G. Ozsoyoglu 100 points

In this assignment, you will use posix threads to implement the threaded version of the Baboon Crossing problem. You have seen the algorithmic solution to this problem in assignment #4.

Baboon Crossing Problem (semaphore-based process synchronization). There is a deep canyon somewhere in the Katavi National Park, Tanzania, and a single rope that spans the canyon from point A to point B. Baboons can cross the canyon by swinging hand-over-hand on the rope, but if two baboons going in opposite directions meet in the middle, they fight until one or both drop to their deaths. Furthermore, the rope is only strong enough to hold 5 baboons. If there are more baboons on the rope at the same time, it breaks.
In our environment, baboons (i.e., baboon processes) have become smarter and now use semaphores for peacefully using the crossing, instead of fighting and killing each other.
Assuming that the baboons can use semaphores, we would like to design a rope crossing synchronization scheme with the following properties:

- *Ordered crossing.* Baboons arriving point A (B) line up, and cross the rope to B (A) in the order they arrive.
- *Rope load.* There cannot be more than five baboons on the rope: the rope cannot handle the load.
- *Crossing guarantee.* Once a baboon process begins to cross the rope, it is guaranteed to get to the other side without running into another baboon going the other way. (Remember, baboons are smart now).
- *Streaming guarantee when only one side has baboons waiting to cross.* If multiple baboons arrive to point A, and, there are no baboons at point B, they can all cross in sequence, subject to the rope load and fairness restrictions.
- *Fairness (no starvation).* A continuing stream of baboons crossing in one direction should not bar baboons from crossing the other way indefinitely. So, after every 10 baboons complete their crossing from A to B (or vice versa), if baboons are waiting to cross from B to A, the crossing direction must switch into a B-to-A crossing.

Your implementation should consist of a fairly large number of threads, say, 20 A-to-B baboon threads, and 20 B-to-A baboon threads.

You should test your code with multiple execution scenarios; and your execution scenarios should clearly demonstrate the correctness of your implementation. For each action (e.g., an approaching A-to-B baboon, etc.), print a message to the screen containing the thread pid of the involved thread, the action, the current time, and the state of the baboon at the crossing and waiting to cross.

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 7.