CS431 Lab 2 Fall 2013 Burris

Due: Friday, November 1, 2013. Do not procrastinate. Additional labs will be assigned during this period.

First implement a counting semaphore in Ada using the rendezvous (or protected types) or Java using synchronized methods. You may use “C” if desired but may not use a counting semaphore from a library (you must implement the semaphore yourself)! A good start on using Ada to implement a semaphore is in the class notes. Implementation using a “protected type” (monitor) is preferred. Now implement the landing officer of Lab 1 as a “task type” and shuttles as a “task type” (or classes extending thread in Java) accepting their names and other appropriate information via discriminates, constructor, or other appropriate initialization protocol.

Create a fleet of 10 shuttles and 3 landing officers as arrays of task resources (Java threads). Each shuttle must take off and land 5 times prior to terminating. Shuttles do not require assistance to takeoff. Again, a landing officer may only work with one shuttle at a time. Each shuttle must use your counting semaphore to gain access to a landing officer. Shuttles and landing officers must clearly identify themselves and client each time they generate output. A race condition results in destruction of the entire fleet. Shuttles and landing officers are subject to the constraints in Lab 1.

**“C” Option:**

Successfully complete the mission allowing your pool of shuttles to compete for the attention of 2 landing officers.

**“B” Option:**

First implement the “C” Option. Now successfully complete the mission allowing your pool of shuttles to compete for the attention of 3 landing officers. Your program must prompt the user for the number of shuttles and landing officers desired and create them dynamically at runtime (preferably in the stack but the heap will be allowed). You may use this code to generate output for the “C” Option if desired.

**“A” Option:**

First implement the “B” Option. Now solve the “B” option requirements using a monitor (protected type/entry points in Ada or synchronized methods in Java or “C” obeying the specified restrictions). You may not use memory interlock semaphores or other low level synchronization methods.

**ALL GRADING OPTIONS:**

For credit above the “C” Option, you must make arrangements outside of class to demonstrate you solution works. In addition, for each grading option you must submit the code followed by the output. The “C” option code and results must appear first followed sequentially by the “B” Option code and results then the “A” Option code and results.

You may use Windows, Unix (Linux), or Apple as the OS. Just remember you must physically demonstrate your solution to meet lab requirements above the “C” Option. You may not use reptiles (Python), drugs (PHP), or gems (Perl) as implementation languages. If you have another language choice, convenience me it is appropriate.