SFWRENG 3BB4 – Software Design III
Concurrent System Design

Assignment 2

Due date: Friday, October 28, 2016, 11:59pm (EDT)

Instructions: Please read the questions carefully. Solutions must be submitted in the Avenue Dropbox created for the assignment. Note that the deadline is strictly enforced. Avenue tracks the exact time that submissions are uploaded and late submissions may be rejected.

Question 1

Part A Using only basic threads and synchronization primitives, implement a Java program that outputs a random number between `0` and `N`.

Part B Model the Java program of the previous part so as to increase the level of confidence on its correctness.

Question 2

This problem is taken from Allen Downey's *The Little Book of Semaphores* (which is, in turn, an adaptation from a problem presented in Tanenbaum's *Operating Systems: Design and Implementation*):

> "There is a deep canyon somewhere in Kruger National Park, South Africa, and a single rope that spans the canyon. 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 will fight and 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 will break."

Assuming baboons can use semaphores, design a synchronization scheme with the following properties: (i) Once a baboon has begun to cross, it is guaranteed to get to the other side without running into a baboon going the other way. (ii) There are never more than 5 baboons on the rope. (iii) A continuing stream of baboons crossing in one direction should not bar baboons going the other way indefinitely (no starvation).

Part A Model the problem presented above in FSP in a way such that it satisfies the requirements (i) – (iii).

Part B Implement the FSP model of Part A in Java.

Question 3

Consider the following Java implementation:

```
1 public class Main {
2   public static void main(String[] args) {
3     final Object rs1 = new Object( );
4     final Object rs2 = new Object( );
5
6     // Here's th1: It tries to lock rs1, then rs2.
7     Thread th1 = new Thread() {
8       public void run() {
9         while (true) {
10          // th1 tries to lock rs1.
11          synchronized (rs1) {
12            // After locking rs1, th1 does something that takes time.
13            // Observe that during this period th2 may be created and put to run.
14            ...
15            // th1 tries to lock rs2.
16            synchronized (rs2) { ... }
17          }
18        }
19      }
20    };
21
22    // Here's th2:  It tries to lock rs2, then rs1.
23    Thread th2 = new Thread() {
24      public void run() {
25        while (true) {
26          // th2 tries to lock rs2.
27          synchronized (rs2) {
28            // After locking rs2, th2 does something that takes time.
29            // Observe that during this period th1 may be created and put to run.
30            ...
31            // th2 tries to lock rs1.
32            synchronized (rs1) { ... }
33          }
34        }
35      }
36    };
37
38    // Start.
39    th1.start();
40    th2.start();
41  }
42 }
```

Part A Model the Java program above in FSP and use this model to show that this program may deadlock.

Part B Solve the deadlock situation of Part A.