Spring 2017

EECS 338 Assignment 5 Solutions
Concurrent Process Management with Semaphores

G. Ozsoyoglu 100 points

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 a monitor for a peaceful crossing, instead of fighting and killing each other.

Assuming that the baboons can use a monitor, 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.
- *Streaming guarantee when only one side has baboons waiting to cross.* If multiple baboons arrive to point A (or, B), and, there are no baboons at point B (or, A), 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 there are baboons waiting to cross from B to A (or, from A to B), the crossing direction must switch into a B-to-A (A-to-B) crossing.

Write a monitor-based solution (i.e., an algorithm) to the Baboon Crossing problem, together with the signal implementation alternative. Explain your algorithm, and explicitly specify any assumptions you make about the model.

```
type Baboon-Crossing= monitor;      //Type specification.
var XingCnt:int;  XedCnt:int; AtoBWaitCnt:int; BtoAWaitCnt:int;      //Xing:Crossing  Xed:Crossed
     toB:condition; toA:condition;   XingDrctn:Enumerated {None, AtoB, BtoA};

 procedure entry BaboonAtoB-Enter()
 if ((XingDrctn=AtoB or  XingDrctn=None) and XingCnt<4 and ((XedCnt+XingCnt)<10) or  BtoAWaitCnt=0))
       {XingDrctn:=AtoB; XingCnt++}
 else {AtoBWaitCnt++; toB.wait; AtoBWaitCnt−−; XingCnt++;  XingDrctn:=AtoB}

 procedure entry BaboonAtoB-Exit()
 { XedCnt++;   XingCnt−−;
  if (AtoBWaitCnt≠0 and ((XedCnt+XingCnt)<10) or BtoAWaitCnt=0)) toB.signal;
  else if (XingCnt=0 and (AtoBWaitCnt=0 or XedCount≥10) and BtoAWaitCnt≠0 )
            {XingDrctn:=BtoA; XedCnt:=0; toA.signal}
       else if (XingCnt=0 and AtoBWaitCnt=0 and BtoAWaitCnt=0)
            {XingDrctn:=None; XedCount:=0}}
```

procedure entry *BaboonBtoA-Enter()*
if ***(***(*XingDrctn=BtoA* ***or*** *XingDrctn=None)* ***and*** *XingCnt<4* ***and*** ***(***(*XedCnt+XingCnt)<10)* ***or*** *AtoBWaitCnt=0)***)**
{XingDrctn:=BtoA; XingCnt++}
else *{BtoAWaitCnt++; toA.wait; BtoAWaitCnt−−; XingCnt++; XingDrctn:=BtoA}*

procedure entry *BaboonBtoA-Exit()*
{ *XedCnt++; XingCnt−−;*
if ***(****BtoAWaitCnt≠0* ***and*** ***(****(XedCnt+XingCnt)<10****)*** ***or*** *AtoBWaitCnt=0)****)*** *toA.signal;*
else if ***(****XingCnt=0* ***and*** ***(****BtoAWaitCnt=0* ***or*** *XedCount≥10****)*** ***and*** *AtoBWaitCnt≠0* ***)***
{XingDrctn:=AtoB; XedCnt:=0; toB.signal}
else if ***(****XingCnt=0* ***and*** *BtoAWaitCnt=0* ***and*** *AtoBWaitCnt=0****)***
*{XingDrctn:=None; XedCount:=0}****}***

begin *XingCnt := XedCnt:=AtoBWaitCnt:=BtoAWaitCnt:=0; XingDrctn := None* ***end.***

Monitor Use: ***var*** *MyBaboon-Crossing:Baboon-Crossing;*

AtoB-Baboon: {//Come to rope;
BaboonAtoB-Enter(pid);
//CROSS;
BaboonAtoB-Exit(pid);
//Wonder in the park*}*

B-to-A-Baboon: {//Come to rope;
BaboonBtoA-Enter(pid);
//CROSS;
BaboonBtoA-Exit(pid);
//Wonder in the park*}*

Signaling/signaled process policy: *Signaling process continues to execute; and, signaled process is blocked until signaling process is blocked or leaves the monitor.*