CS 620: Homework 9

Carmen St. Jean

November 1, 2012

1. *(6) Question 6.39 of textbook. You are not permitted to look-up a solution to the Barber's problem on-line (or anywhere else).*

   A Barbershop consists of a waiting room with n chairs, and the barber room containing the barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are taken, then the customers will leave the shop. If the barber is busy, but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write pseudocode using semaphores to coordinate the barber and the customers.

   Important : Be sure to make variable declarations in your implementation and to specify initial values.

```
semaphore barberAwake = 0;       // 0 if barber is asleep, 1 if barber is awake
semaphore customerAvailable = 0; // 0 if no customers available, 1 if at least one available
semaphore modifyWaitingRoom = 1; // allows only one person to enter waiting room
int numberWaiting = 0;

Barber:
for (;;) {
    p(customerAvailable); // wait for a customer to be ready

    p(modifyWaitingRoom); // wait to be allowed to remove a customer
    numberWaiting--;      // take a customer from the waiting room

    // HAIRCUT

    v(modifyWaitingRoom); // allow others to enter waiting room

    v(barberAwake);       // signal you are awake and can do a haircut
}

Customer:
for (;;) {
    p(modifyWaitingRoom);      // wait for your turn to enter waiting room

    if (numberCustomersWaiting < n) { // enough seats in waiting room

        numberWaiting++;      // sit down in the waiting room
        v(modifyWaitingRoom); // signal you are done entering the waiting room

        v(customerAvailable); // signal you are available for haircut

        p(barberAwake);       // wait for the barber to wake up

        // HAIRCUT
```

```
    } else { // not enough seats in waiting room for you, leave shop
        v(modifyWaitingRoom); // signal you're done trying to enter the waiting room
    }
}
```

2. *(3) Suppose 3 threads, T1, T2, T3, are trying to access a shared resource using the following code:*

```
T_i
-----
for (;;) {
    <non critical section code>
    P(x)
    <critical section code>
    V(x)
    <non critical section code>
}
```

Initially, x = 1. Assume that xs semaphore queue is FIFO. Is starvation possible? If yes, show a scenario where one thread could starve. If no, prove that starvation is not possible.

Starvation is impossible because of the FIFO queue. The FIFO queue on x makes it so that all threads requesting a lock on x will eventually get x. In contrast, if the queue was LIFO, then one or two of the threads could easily starve will the remaining thread(s) continually gets access to the x variable.

3. *(4) Consider the following synchronization problem: process P1 adds 1 to a shared variable X, then process P2 adds 3 to X, then process P3 subtracts 4 from X, then P1 adds 1, then P2 adds 3, then P3 subtracts 4, ..., and so on. Processes P1, P2, and P3 are asynchronous processes that manipulate a shared variable X. Give the code for these three processes, using semaphores, to guarantee the above sequence of events.*

—

4. *Suppose there are several processors available (i.e., more than are needed). Therefore, several steps of a computation can be run in parallel. Semaphores are used for synchronizing code execution on the multiple processors, Each processor can perform one binary operation per time unit. Assume that each arithmetic operation takes 1 time step. Assume that the P() and V() operations take 0 time units to complete (i.e., they are very fast operations compared to the computation time).*

$$ANSWER = A + \frac{B}{C^D} * \frac{E-F}{G-H} * (I + J)$$

   (a) *(1) What is the minimum number of time units required to complete the computation on 1 CPU?*
   Nine time units are needed for one CPU.

   (b) *(4) What is the minimum number of time units required to complete the computation? For this minimum number of time units, what is the minimum number of processors required?*
   Five time units is the minimum time which is possible with at least four CPUs:

Thread	Time 1	Time 2	Time 3	Time 4	Time 5
T1	$C^D \rightarrow K$	$\frac{B}{K} \rightarrow P$	$Q * N \rightarrow R$	$P * R \rightarrow S$	$A + S \rightarrow ANSWER$
T2	$E - F \rightarrow L$	$\frac{L}{M} \rightarrow Q$			
T3	$G - H \rightarrow M$				
T4	$I + J \rightarrow N$				
Equation	$A + \frac{B}{K} * \frac{L}{M} * N$	$A + P * Q * N$	$A + P * R$	$A + S$	$ANSWER$

   (c) *(4) Consider your answer to the previous problem. Give the code for each processor, utilizing semaphores to handle any necessary synchronization.*

   —

5. *(3) Consider the following parallel program:*

```
THREAD x:
for (;;) {
    P(x)
    P(y)
    write "x"
    V(z)
}
```

```
THREAD y:
for (;;) {
    P(y)
    P(y)
    write "y"
}
```

```
THREAD z:
for (;;) {
    P(z);
    write "z"
    V(x);
    V(y);
}
```

Initially, x = 1; y = 2; z = 0. More than one output is possible. Describe the output format of this program.

If the y thread goes first, then the output will be "y" only. This is because the x thread requires a lock on the y variable to reach its write statement, and y does not release its locks on the y variable.

If the x or z threads go first, then the output will be "(xz)*y" (using regular expression terms, that is "xz" would be repeated by zero or more times and "y" would terminate all written messages). The z thread cannot write until the x thread writes because the x thread releases a lock on the z variable which z requires to enter its critical section. The x thread cannot write again until the z thread has written because it needs a lock on x and y which the z thread releases. The y thread terminates based on the reason described above.