Due time & date: Please check Carmen

Purpose: Learn how to use threads and how they synchronize using semaphores. For thread related function calls and semaphore operations you are allowed to use only those listed on Slides 9 & 10 Presentation G. For compilation commands see the first line of the program with threads you have been provided as a handout.

Assignment Part A: In this problem, there are four threads plus the starting thread (i.e. the one you start). The starting thread creates and initializes all semaphores, and also initializes 3 accounts with each account having $100,000, and creates three threads (Thread1, Thread 2 and Thread3). Then, the starting thread waits for the second thread it created to terminate and then it terminates.

In each of 10,000 iterations, Thread1 moves $200 from Account0 to Account1, Thread2 moves $200 from Account1 to Account2, while Thread3 moves 200 from Account2 to Account0. Note that it is allowed that an account gets in red (i.e. negative). After going through every 1,000 iterations each thread checks the total amount of money contained in the three accounts at that moment. If the sum is not $300,000, it must be some error in your program. Thread1 also creates Thread4, before starting to move money. Thread4 has the 50,000 iteration loop. Thread1, Thread2, Thread3 and Thread4 should synchronize 4 process (i.e. threads) 4 way synchronization before entering its loop. In each iteration, Thread4 reads contents of all 3 accounts. At the end, Thread4 prints a number of times it checked the sum and a number of times the sum was not equal 300,000. If your program is correct, the sum is always 300,000. Also, Thread4 should terminate using the exit system call.

You are given parts of code for Thread1, while the codes for other Thread2 and Thread3 are similar. You are supposed only to add some statements without significantly changing the existing code.

Thread1 Procedure

```c
int i, internal_reg;
/*synch. with Thread2, Thread3 & Thread4 (4 proc 4 way synch.)*/
for (i = 0, i < 10000; i++)
{
    internal_reg = Account[0];
    internal_reg = internal_reg - 200;
    Account[0] = internal_reg;
    /* same thing, except we're adding $200 to account1 now... */
    internal_reg = Account[1];
    internal_reg = internal_reg + 200;
    Account[1] = internal_reg;
    /* here add a code that outputs contents of each account and their sum after 1,000th, 2,000th, 3,000th, .... and 10,000th iteration*/
}
/* above include some wait and signal ops on semaphores. Do not over-synchronize.*/
```

/* above include some wait and signal ops on semaphores. Do not over-synchronize.*/
Also, Thread1 sleeps for 0.2 sec after 3,000<sup>th</sup>, 6,000<sup>th</sup>, and 9,000<sup>th</sup> iteration (total 0.6 sec), Thread2 sleeps 0.2 sec after 2,000<sup>th</sup>, 4,000<sup>th</sup>, 6,000<sup>th</sup>, and 9,000<sup>th</sup> iteration (total 0.8 sec), Thread3 slips 0.3 sec after 2,000<sup>th</sup> and 7,500<sup>th</sup> iteration, and Thread4 slips 0.05 sec after each of its 10,000 iterations (total 0.25 sec).

On/before due day/time:
1. electronically submit your source code on Carmen Cse2431 at Assignment Lab4a and
2. turn in hard copies of your source codes (indicate your compilation commands) and a hard copy of the output of your programs from the last executions.

Assignment Part B: This problem is “One Consumer and any of two Producers” problem with 3 threads plus the starting thread that creates those 3 threads.

The first thread, call it the A thread, places items into empty slots of its buffer (BufferX, with 20 slots and each slot with an integer (type int variable) and 5 characters (type char array[5]), but only one item in each iteration. This thread has 3000 iterations. Each item should include the item number (1, 2, 3, …, 299, 3000) for the integer and the characters “ABCaa” for even numbered items and the characters “XYZxx” for odd numbered items.

The second thread, call it the B thread, places items into empty slots of its buffer (BufferY, with 40 slots and each slot with an integer and 3 characters), but only one item in each iteration. ProcB has 4000 iterations each item should include the item number (1, 2, 3,…, 399, 4000) for the integer and the characters “MMM” for the first 2000 items and “NNN” for the second 2000 items.

In each iteration, the third thread, call it the C thread, takes one item from either buffer and prints them. This thread has 7000 iterations.

The A thread or the B thread waits if there is no empty slot for the current item, while the C thread waits only if all slots in both buffers are empty.

After putting each 500 items in its buffer the A thread sleeps for 0.1 sec, the B thread sleeps for 0.2 seconds after putting each 1000 items in its buffer, while the third thread sleeps for 0.3 seconds after taking each 2000 items.

The starting thread creates and initializes all necessary semaphores and then it waits for all other threads to terminate before it terminates. The A and the B threads are synchronized according to two way two process synchronization at the beginning of their codes.

Do not over-synchronize.

On/before due day/time:
1. electronically submit your source code on Carmen Cse2431 at Assignment Lab4b and
2. turn in hard copies of your source codes (indicate your compilation commands) and a hard copy of the output of your programs from the last executions.