Cse660 **Lab 4. Assignment** February 18, 2011.

Due date*:* Friday, February 25, 2011.

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 Slide 9 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 main thread. The main thread creates 3 accounts with $10000 in each, and three threads. Then, the main thread waits for the first thread it created to terminate and then it terminates.

In each of 1000 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. After going through 100 iterations every thread prints the total amount of money contained in the three accounts at that moment. If the sum is not $30000, it must be some error in your program. Thread1 also creates Thread4, before starting to move money for an account to another. Thread4 has the 5000 iteration loop and it should synchronize with other 3 processes in 4 process 4 way synchronization before entering its loop. In each iteration, contents of all 3 accounts are read and if the sum is not equal 30000, that is printed. At the end, Thread4 prints a number of times it checked the sum and a number of times the sum was not equal 3000. Also, Thread4 should terminate last (use the exit system call).

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

Thread1 Procedure

```
  {int i, internal_reg;
/*synch. with Thread2, Thread3 & Thread4 (4 proc 4 way synch.)*/
    for (i = 0, i < 1000; i++)
      {
       internal_reg = Account [0];
       internal_reg = internal_reg - 200;
       Account[0] = internal_reg;
         /* same thing, except we're adding $100 to account1 now... */
       internal_reg = Account [1];
       internal_reg = internal_reg + 200;
       Account[1] = internal_reg;
        /* here add a code that prints contents of each account and
           their sum after 100th, 200th, 300th, .... and 1000th iteration*/
       }
/* above include some wait and signal ops on semaphores. Do not over-synchronize.*/
```

Thread1 sleeps for 1 sec after 500 iterations, Thread2 slips 1 sec after 550 iterations, Thread3 slips 1 sec after 650 iterations, and Proc4 slips 1 sec after 2500 iterations.

Submit your source code file using command:

submit c660aa lab4a ThreadA.cpp

Also, provide a hard copy of your source codes and a hard copy of the output of your programs from one of executions, plus a compilation command.

Assignment Part B: This problem is 'One Consumer and both of two Producers' problem with 3 threads plus the main thread that creates those 3 threads.

Each of 3 created threads has 500 iterations.

The first thread places items into empty slots of its buffer (BufferA with 20 slots and each slot with an integer and 2 characters), but only one item in each iteration. Each item should include the item number (1, 2, 3,..., 499, 500) for the integer and characters 'xx'.

The second thread places items into empty slots of its buffer (BufferB, with 30 slots and each slot with 3 characters and an integer), but only one item in each iteration. Each item should include characters 'YYY' followed by the item number (1, 2, 3, ..., 499, 500) for the integer.

The third thread takes, in each iteration, one item from each buffer (total of two) and prints them. If the threads are properly synchronized, your output will be:

1xx YYY1 2xx YYY2 3xx YYY3 4xx YYY4 5xx YYY5 6xx YYY6 ... 498xx YYY498 499xx YYY499 500xx YYY500

The first or the second thread waits if there is no empty slot for the current item. The third thread waits if all slots in any buffer are empty.

After putting every 50 items in its buffer the first thread sleeps for 1 sec, the second thread sleeps for 2 seconds after putting every 75 items in its buffer, while the third thread sleeps for 1 sec after taking every 100 items.

The main thread, besides creating 3 threads, also creates all necessary semaphores and then it terminates. The first and second threads are synchronized according to two way two process synchronization at the beginning of their codes.

Use semaphores as the only synchronization mechanism and do not over-synchronize.

Submit your source code files using command:

submit c660aa lab4b ThreadB.cpp

Also, provide a hard copy of your source code and a hard copy of the output of your program from one of executions, plus a compilation command.