CPSC 313 Introduction to Computer Systems
Summer 2016 Homework 2
Carsten Hood
UIN: 922009787

1. What is the role of the contentionscope for pthreads? How does it affect the execution of pthreads?
Contention scope determines whether the thread uses process-level or system-level resources. It affects control of thread execution, where threads using the process contention scope are scheduled by the process and its libraries and threads using the system contention scope are dependent on control and resource availability of the operating system.

2. Assume you have a system that does not provide a usleep(unsigned long usec) call to suspend the execution of the thread for a given amount of time, say in μsecs. How would you implement this function using condition variables? (Describe your implementation in form of a C/C++ code snippet.)

```
void custom_sleep(int delay) {
    struct timeval start_time;
    gettimeofday(&start_time, NULL);
    int microsecs_in_sec = 1000000;
    int seconds = delay/microsecs_in_sec;
    int microsecs = delay%microsecs_in_sec;

    struct timespec end_time;
    end_time.tv_sec = start_time.tv_sec + seconds;
    end_time.tv_nsec = start_time.tv_usec + microsecs;

    pthread_mutex_lock(&mutex1);
    pthread_cond_timedwait(&cond1, &mutex1, &end_time);
    pthread_mutex_unlock(&mutex1);
}
```

3. [1] Which of the scheduling algorithms listed below could result in starvation?
   (a) First-come, first-served (YES)
   (b) Shortest job first (YES)
   (c) Round robin (NO)
   (d) Fixed priority (YES)

4. [1] Servers can be designed to limit the number of open connections. For example, a server may wish to have only N connections active at any time. As soon as N connections are established, the server will not handle more connections until at last one existing connection is closed. How would you use semaphores to limit the number of concurrent connections? Assume that you have a function called HandleNextIncomingConnection() and one called CloseExistingConnection(), which are called before and at the end of handling a connection. How would you add the necessary synchronization code to limit the number of concurrent connections to at most N?

```
Semaphore server(N);
HandleNextIncomingConnection() {
    server.P();
    // handle next connection here
}
CloseExistingConnection() {
    // close existing connection here
    server.V();
}
```

5. [3] Ten processes share a critical section implemented by using a semaphore x. Nine of these processes use the code x.P(); <critical section>; x.V(). However, one process erroneously uses the code

x.V(); <critical section>; x.P(). What is the maximum number of processes that can be in the critical section at the same time?

2 processes (assuming max allowed is 1)

6. You just started a new position at Banana,Inc., and your new boss knows that you have done well in CSCE 313. You first assignment therefore is to design the entry protocol for Banana, Inc.'s unisex bathroom. Your boss specifies the following two requirements:
(a) There cannot be men and women in the bathroom at the same time.
(b) There should never be more than three employees wasting company time in the bathroom.
In the interest of both Banana, Inc. and of its employees, your solution should avoid deadlocks.
Describe the implementation of your solution by adding the necessary synchronization to the female code listed below. (The male code should be similar.)

```
// declare and initialize semaphores here
Semaphore * empty(1);
Semaphore * shared_multiplex(3);  // to avoid starvation

Semaphore * women_mutex(1);
Semaphore * women_multiplex(3);
int women_counter = 0;

Semaphore * men_mutex(1);
Semaphore * men_multiplex(3);
int men_counter = 0;
for(;;) { // forever
  // this code is for female employees. male code is similar.
  // <here the employee is at his/her desk>

  // (here comes the code for the employee to access the bathroom.)
P(shared_mutex);    // only semaphore common to men & women
  P(women_mutex);
    women_counter++;
    if (women_counter == 1)
      P(empty);     // mark as not empty on first comer
  V(women_mutex);
V(shared_mutex);

  P(women_multiplex);
  // <here the employee is in the bathroom>
  V(women_multiplex);

  // (here comes the code for the employee to leave the bathroom.)
  P(women_mutex)
    women_counter--;
    if (women_counter == 0)
      V(empty)      // mark as empty on last leaver
  V(women_mutex)
}
```

Hint: I would help to have the following semaphores: empty to denote that the bathroom is empty, and either gender can enter, and women multiplex and men multiplex to limit the number of users in the bathroom.

Bonus: It is likely that your solution does not address starvation. A long line of men can come and enter while a woman is waiting, and vice versa. You get 5 BONUS POINTS if your solution does not allow this to happen.

(I think the starvation issue is addressed by `shared_mutex` above)