RECITATION: **103**
NUMBER OF HOURS TO COMPLETE PS: **4**

1. *(10 pt) Describe and explain each of the following (use examples):*

   (a) *The concept of interleaving instructions of two threads*

   - Interleaving instructions of two threads involves running instructions for both threads and context switching between the two. This gives the appearance of running simultaneously but in reality they are swapping quickly between the two threads.

   (b) *Each of the requirements for deadlock (give examples). Explain why the first three are necessary but not sufficient for deadlock.*

   - The first condition is mutual exclusion, which means the resources involved are not shareable and only one process can be using a resource at any given time. The second condition is that one of the processes involved has one of the resources involved and is trying to get more. An example of this is the philosophers problem where one of the diners has one chopstick and is waiting for another. The third condition is no preemption, so processes cannot be forced to give up resources. The final condition is that each process is waiting for a resource that is being held by another process. The first three conditions are not sufficient to cause deadlock alone because if there is no resource contention there would be no waiting on other resources. Eventually one of the involved processes would have all of the resources it needed to complete and then it would release the resource.

2. *(10 pt) Compare and contrast the following (make sure to define each of the items):*

   (a) *Mode switch vs process switch*

   - A process switch is done by the operating system scheduler. It saves the state of the process running, and then swaps a new one into running. A mode switch switches between user mode and kernel mode. A process calls a mode switch when it needs to access system resources. The two are different in that a process switch changes the active process in the os and a mode switch allows the process to access system resources.

   (b) *Semaphore vs conditional variable*

- A semaphore is an atomic data structure that is used for mutual exclusion, which a conditional variable also can do. The main difference here is that a process will spin on a semaphore but will not on a conditional variable because the conditional variable will sleep the process until it is signaled.

3. *(20 pt) Given the code below answer the following questions (explain your reasoning):*

```
int temp;
void swap(int *y, int *z)
{
    int local;
    local = temp;
    temp = *y;
    *y = *z;
    *z = temp;
    temp = local;
}
```

(a) *Is the function thread safe?*

- This function is not thread safe because there is a global variable that is being updated inside of the process. When multiple threads are running this process if one runs the `temp = *y` and then another thread runs the `temp = local` then there is no way to know what will be assigned to `*z` or to `local`.

(b) *Is the function reentrant?*

- The function itself is reentrant because none of the internal variables, aside from the `int temp`, are going to be changed by reentering the function.

4. *(30 pt) Barrier Synchronization: A **barrier** is a tool for synchronizing the activity of a number of threads. When a thread reaches a **barrier point**, it cannot proceed until other threads have reached this point as well. When the last thread reaches the barrier point, all threads are released and can resume concurrent execution. You have a number of threads processing data and each piece of data takes a different amount of time to process. When a thread has completed its processing, it will ask for more work. However, the data requires that 4 workers all start together on the next 4 pieces of data. This means we want to gather 4 threads before having them begin processing. Please provide a solution (pseudocode), explain how it works, show that it is starvation free and deadlock free.*

```
barrier point = 4;
\\ Do preliminary processing
point.wait;
\\ Do next processing stuff
```

   - First you set up the barrier with the required number of processes that need to reach that point before it continues. Then we do all of our processing for thre first data points. When done we go wait on the barrier we set up before. This will add to the count of processes at the barrier and when we get to four processes they will all continue. This is starvation free because none of the processes will need to use resources being used by other processes so something will always be running. This is deadlock free for the same reason and therefore does not meet the fourth condition for deadlock.

5. *(30 pt) CU wants to show off how politically correct it is by applying the U.S. Supreme Courts Separate but equal is inherently unequal doctrine to gender, ending its long-standing practice of gender-segregated bathrooms on campus. However, as a concession to tradition, it decrees that when a woman is in the bathroom, other women may enter, but no men, and vice versa. Also, due to fire code, at most N (N ¿ 1) individuals may use the bathroom at any time. Your task is to write two functions: `man_use_bathroom()` and `woman_use_bathroom()`. Provide a monitor-based solution that manages access to the bathroom. Your solution should be fair, starvation free and deadlock free.*

   -