CS 140 Lab Report 6 Bonus Item

Dale Sealtiel T. Flores
2023-11373
THX/WXY

7. Regarding Section 3.7, answer the following
   (a) Explain how the semaphore-based barrier is able to ensure that no more than `N` threads are able to pass through the barrier. Make reference to semaphore values and calls to `wait` and `post`.

   Answer: After initializing the barrier from `init_barrier(&b)`, it will go and create the threads and then go to the `wait_barrier(&b, NTHREADS)` where it will call `sem_wait(&b->mutex)`. Since in the initializiation we ran `sem_init(&b->mutex, 0, 1)` where we set the semaphore integer to 1, it will allow the thread to continue and then decrement it to 0. It will then increment the counter because the line `b->count += 1` where it will then run `sem_post(&b->mutex)`, which increments the semaphore integer back to 1. It will then check if the counter is equal to `N` from the line `if(b->count == n)`, where it will be false. After that, it will run `sem_wait(&b->barrier)` where the thread will be stuck since in the initializiation we have `sem_init(&b->barrier)` where the semaphore integer was set to 0. This pattern will run again and again from Thread 1 until the `N`th thread. Where eventually it will be true in the conditional `if(b->count == n)` where it will run `sem_post(&b->barrier)`, where it will increment the semaphore integer of the barrier. This effectively lets the 1st thread pass through, and then decrements the semaphore integer back to 0. The `N`th thread will then run `sem_wait(&b->barrier)` where it will get stuck. Since the first thread was able to pass through, it will then be able to run `(A)` which is `sem_post(&b->barrier)` where it will increment the semaphore integer of barrier let a thread pass through (in this case thread 2), where it will repeat the same process of letting a thread through and then "locking" the barrier, where that thread will then "unlock" the barrier and let the next thread pass through. This repeats until the `N`th thread will run `(A)` where the semaphore integer of barrier will stay with a value of 1 since there is no thread to call `sem_wait`.

   (b) Illustrate with a concrete example how commeneting out the line labeled `(A)` in Code Block 7 may potentially cause a *deadlock*

Answer: Suppose that we have 2 threads let's name them `T1` and `T2`.

Thread Running	Line Executed	Notes
T1	`sem_wait(&b->mutex)`	
T1	`b->count += 1`	
T1	`sem_post(&b-mutex)`	
T1	`if (b->count == n){...}`	This is false
T1	`sem_wait(&b->barrier)`	`T1` will be stuck here
T2	`sem_wait(&b->mutex)`	
T2	`b->count += 1`	
T2	`sem_post(&b-mutex)`	
T2	`if (b->count == n){...}`	This is True
T2	`sem_post(&b->mutex)`	will let `T1` pass through
T2	`sem_wait(&b->barrier)`	`T2` will be stuck here

Since `T1` has been allowed to continue execution, it will then finish its execution immediately since `(A)` has been removed. This results in `T2` being stuck in `sem_wait(*b->barrier)` where it will be permanently blocked thus creating a *deadlock*.