CSCI 3310 – Semaphore Exercises

1. Consider two processes/threads, P1 and P2, that execute the code shown below. One possible execution order of P1 and P2 is shown as well. Fill in the table demonstrating the behavior of the semaphore. Assume that the starting value of the semaphore is 2 (i.e., S = new Semaphore(2)).

```
P1:  S.Wait();
     S.Wait();
     S.Signal();
     S.Signal();
```

```
P2:  S.Wait();
     S.Signal();
```

```
P1:  S->Wait();
P2:  S->Wait();
P1:  S->Wait();
P2:  S->Signal();
P1:  S->Signal();
P1:  S->Signal();
```

value	Queue	process state: execute or block P1	P2
2	empty	execute	execute

2. Suppose you have three threads A, B, and C. Thread A will execute **study_os()**, thread B will execute **drink_coffee()**, and thread C will execute **take_exam()**. Using a single semaphore, sketch code for each thread that will ensure that both **drink_coffee()** and **study_os()** happen before **take_exam()**. Remember to specify the starting value of the semaphore!

3. You and your roommate decide to have milk and cookies for dinner. Remembering the too much milk incident, you have decided to share the dinner prep by having you (Thread A) **buyMilk()**, while your roommate (Thread B) will **buyCookies()**. After both milk and cookies are purchased, both of you will **eatDinner()**. Write code using semaphores that will ensure that neither of you eats dinner until both the milk and cookies are acquired! Of course, you should still be able to purchase the milk and cookies concurrently.