

## Synchronization (Coding)

"To the left, to the left, everything you own in a box to the left..."

While out shopping with her girlfriends one day, Beyoncé discovers that her boyfriend has been cheating on her. Beyoncé heads home to throw her cheating boyfriend out.

When Beyoncé arrives home, she tells her (now) ex to pack all of his things into boxes (on the left, naturally) and waits for him to do so. The waiting is longer than she had hoped, though, because her ex first has to acquire boxes (he needs two!), before he can pack them. Fortunately, he finds a pile of seven in the alleyway. When he is finished packing, Beyoncé calls him a cab.

Once the cab has been summoned, Beyoncé storms off to do Beyoncé things and her ex waits for the cab. When the cab arrives he loads his things into the cab and then leaves with the driver.

Model this situation using both semaphores and monitors. Write three functions:

```
void beyonce()      //function executed by Beyonce, who arrives home from
                    //shopping and tells her boyfriend to pack his things.
                    //Once he is finished packing, she calls him a cab and
                    //then storms off.

void ex()           //function executed by boyfriend, who waits for
                    //Beyonce to arrive home, acquires boxes, packs,
                    //waits for the cab, loads in his things, and rides away

void cab_driver()   //function executed by cab drivers, who wait for someone
                    //to need a cab, drive to the correct location, wait
                    //for the cab to be loaded, and then drive to the
                    //destination
```

1. [8pts] Solve the problem using semaphores.
2. [12pts] Solve the problem using monitors.

Be certain to follow coding conventions discussed in class and to write your solutions so that they can be run by many instances of Beyoncé, Beyoncé boyfriends, and cab drivers at the same time. You may indicate any non-synchronization or mutual exclusion activities using the angle bracket convention we used in class (e.g., <pack> or <drive>). You may assume the box pile is periodically refreshed.

Clearly indicate your solutions to parts (a) and (b).

Sena buy-home = 9  
 Sena ex-pack = 0  
 Sena boxes = 7

buy-price()  
 < arrives here >  
 buy-home → up()

ex-pack → down()  
 call-cab → up()

Car driver  
 ch + buy-home = 0  
 buy-home CV  
 Lock lock;

buy-price()  
 lock → acquire()  
 buy-home ++  
 buy-home CV → signal()

Sena call-cab = 0  
 Sena cab-arrive = 0  
 Sena cab-loaded = 0

ex()

buy-home → down()  
 2x boxes → down()

ex-pack → <sup>up</sup>down()  
 cab-arrive → down()  
 cab-loaded → up()

Cab-driver()  
 call-cab → down()  
 cab-arrive → up()  
 cab-loaded → down()

ex()  
 lock → acquire()  
 while (buy-home ≤ 0)  
 buy-home CV → wait()  
 buy-home --

Cab-driver()  
 lock → acquire()

2x { while (boxes < 0)  
 boxes CV → wait()  
 boxes --

lock → release()

lock → release()

lock → release()