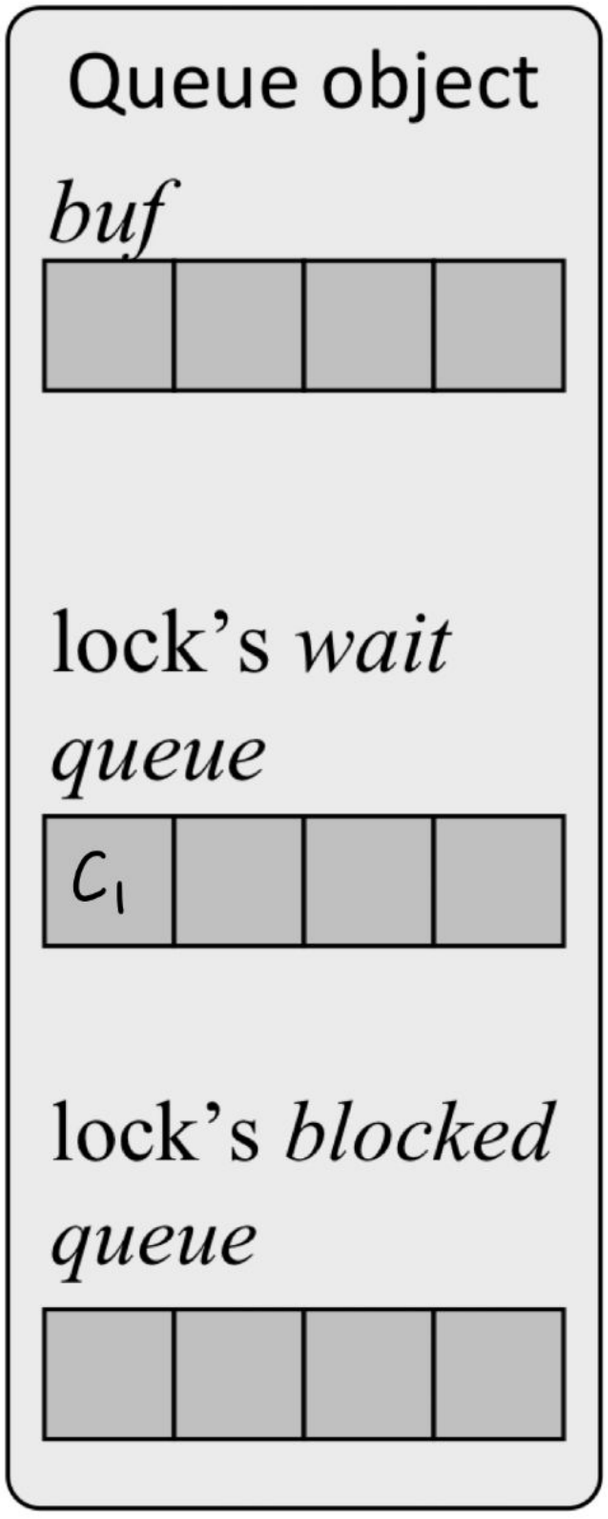
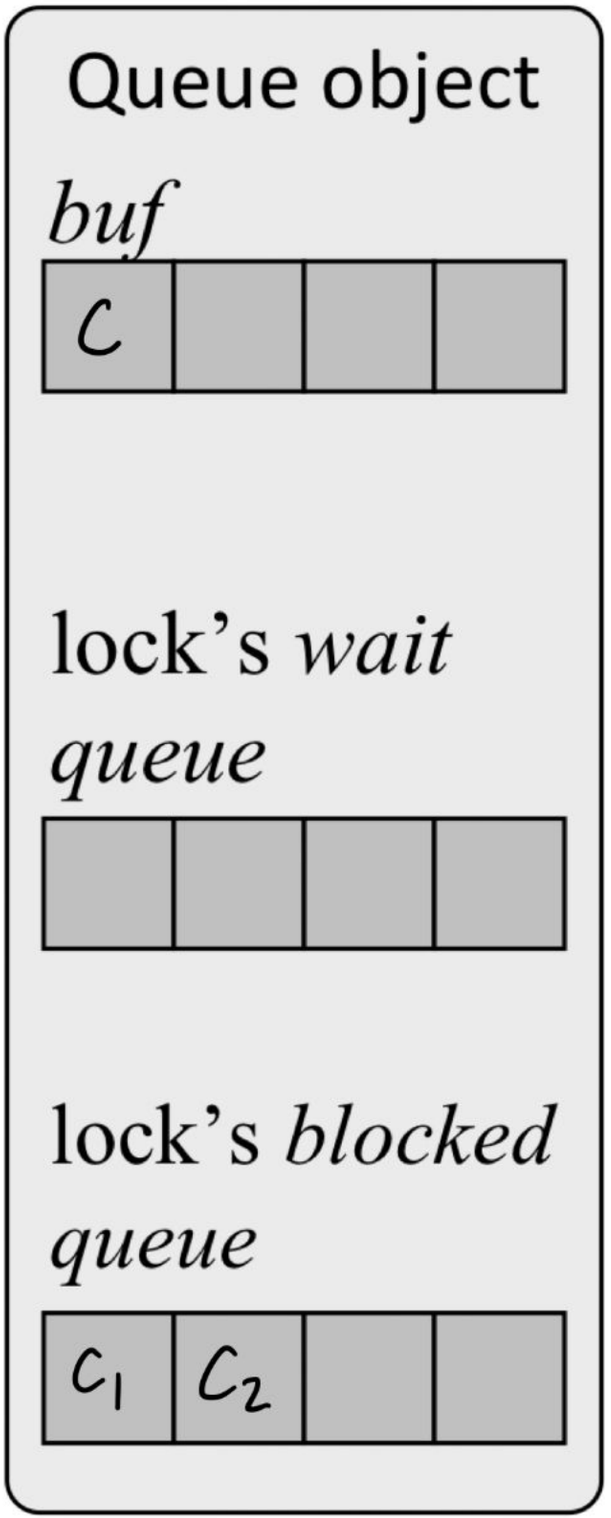


There are two kinds of threads -- consumer threads *C1, C2*, ..., that remove characters from *buf*, and producer threads *P1, P2*, ..., that add characters to *buf*. For our purposes, *buf.size( )* returns the number of characters in the buffer.

The buffer *buf* is initially empty.

1. Consumer *C1* enters the synchronized block for the *get* method
2. *buf.size( ) == 0* is true
3. *wait( )* is executed, placing *C1* on the lock's *wait queue*

Show the status of *buf*, lock's *wait queue* and lock's *blocked queue.*

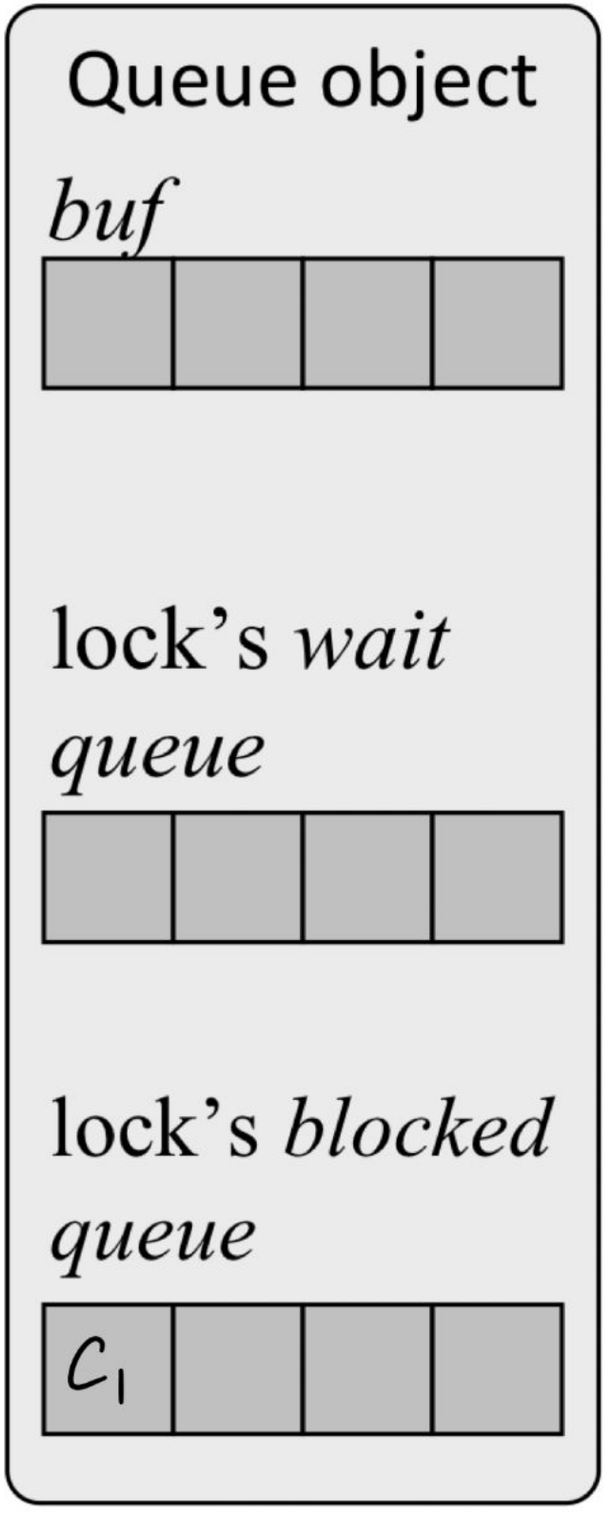


1. Consumer 2 (*C2*) is just about to enter the synchronized block for the *get* method, but has not acquired the lock.

2. Producer *P1* enters the synchronized method *put*, acquires the lock, places the character "*c*" into *buf*, and calls *notify( )*.

3. *C1* is woken up by the notify and must reacquire the lock before proceeding. Thus both *C1* and *C2* are competing for the lock.

Show the status of *buf*, lock's *wait queue* and lock's *blocked queue.*



1. One of *C1* and *C2* is non-deterministically chosen to get the lock. Let's say *C2* gets the lock. It gets to enter the method since *C1* is awake it is put on the *blocked queue*, not back on the *wait queue*.

2. *C2* gets the character and releases the lock which is then acquired by *C1*.

Is there a character in *buf* for *C1* to get?

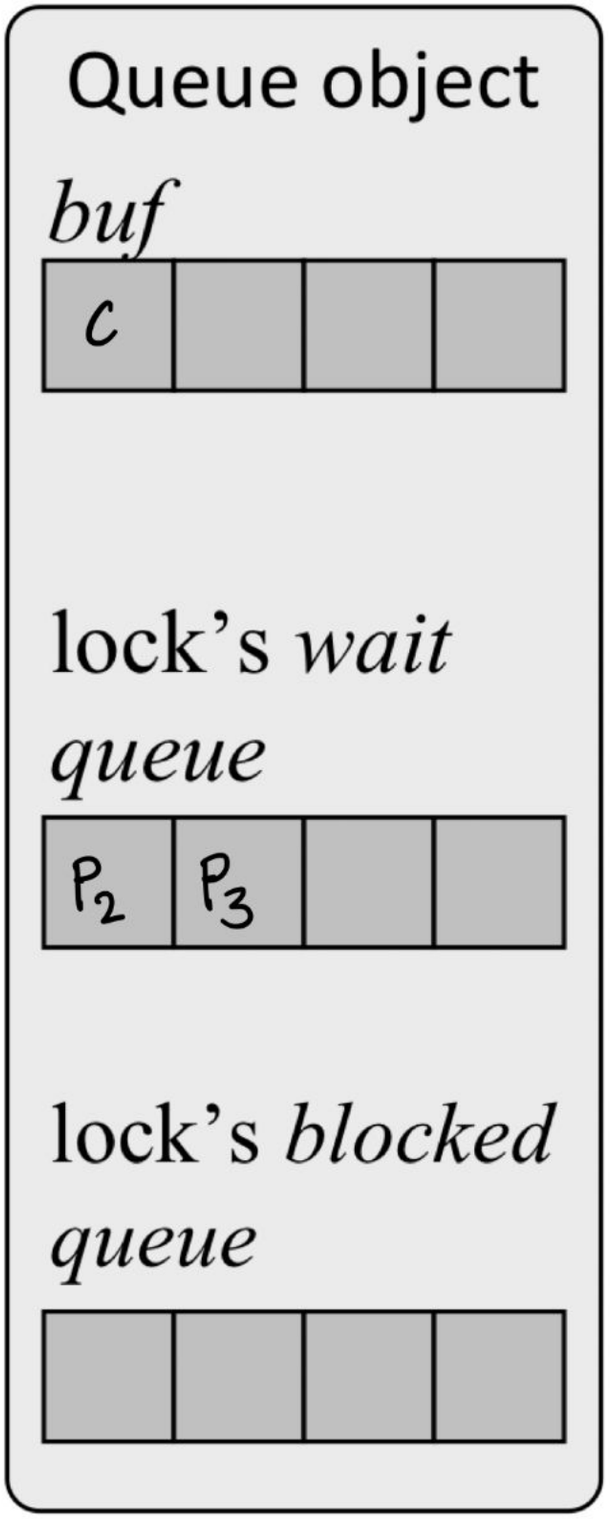
No

What will happen in the program as written?

Buffer is empty so C_1 will still be kept on hold

What would have happened if the *while* loop was not in the *get( )* method?

If there was no while loop then C_1 will try to get a value out of the buffer which will result in getting an index out of bounds error

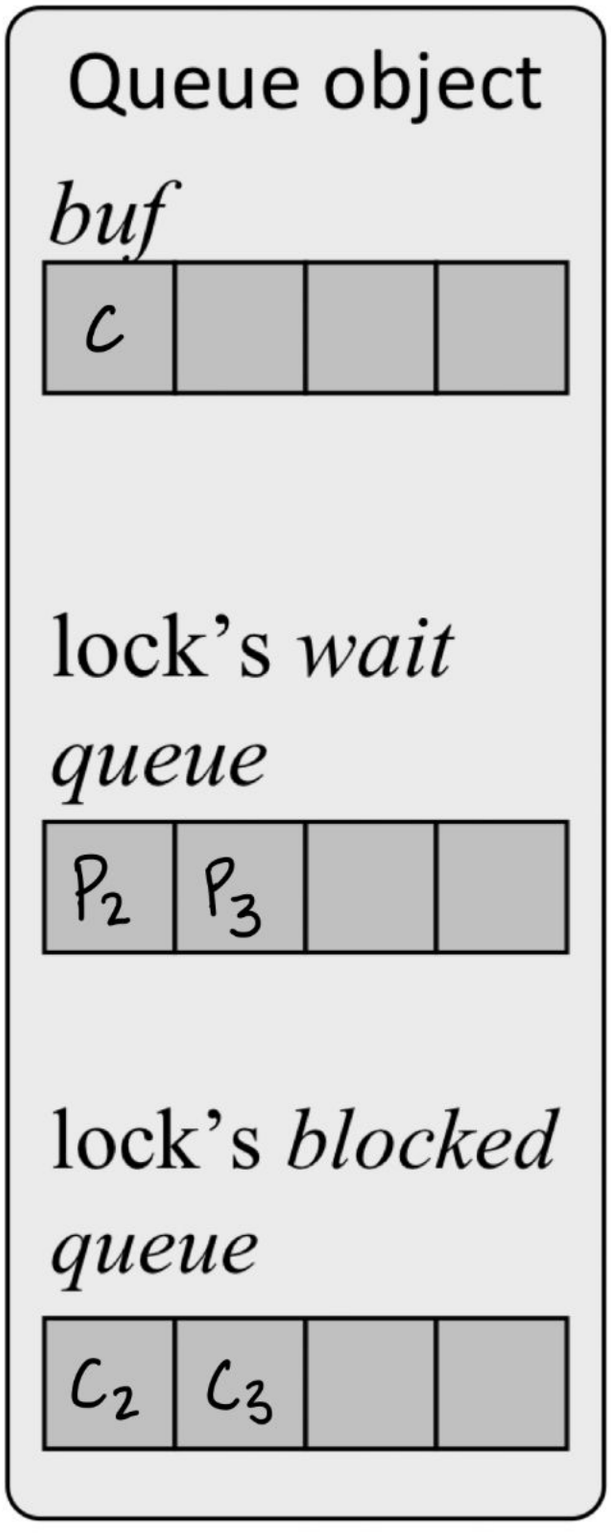


Let's look at a scenario that shows the need for notifyAll instead of notify in the code.

To make this easy, assume a buffer size of *1*. Producer and consumer threads are named as before. *buf* is initially empty.

1. *P1* puts a "c" into the buffer.
2. *P2* attempts a put, checks the while loop and performs a *wait( )*
3. *P3* attempts a put, checks the *while* loop and performs a *wait( )*

Show the status of *buf*, lock's *wait queue* and lock's *blocked queue.*



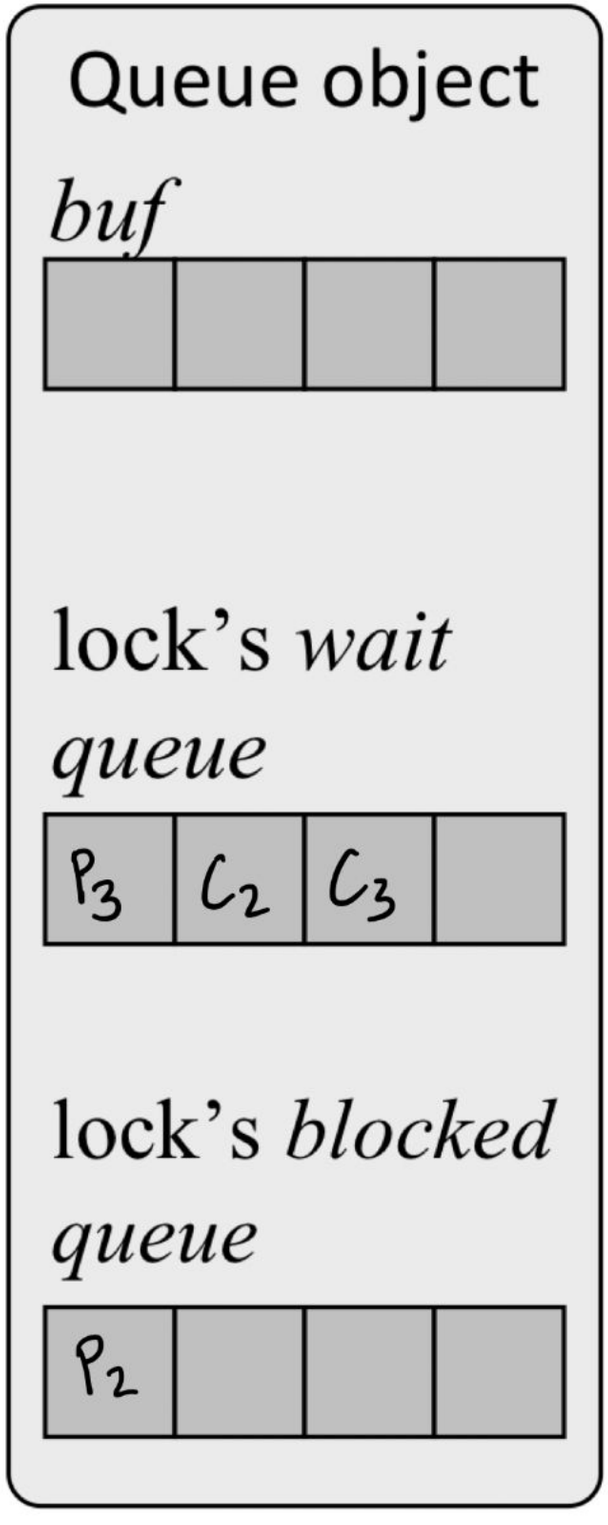
4. The following happen at time step 4:

   a. *C1* attempt to get 1 character and enters the *get* method;

   b. *C2* attempts to get 1 character but blocks on entry to the *get* method;

   c. *C3* attempts to get 1 character but blocks on entry to the *get* method;

Show the status of *buf*, lock's *wait queue* and lock's *blocked queue.*



5. The following happen at time step 5.

   a. *C1* is executing the *get* method, gets the character, calls *notify* and exits the method (releasing the lock and giving *C2* and *C3* a chance to acquire it);

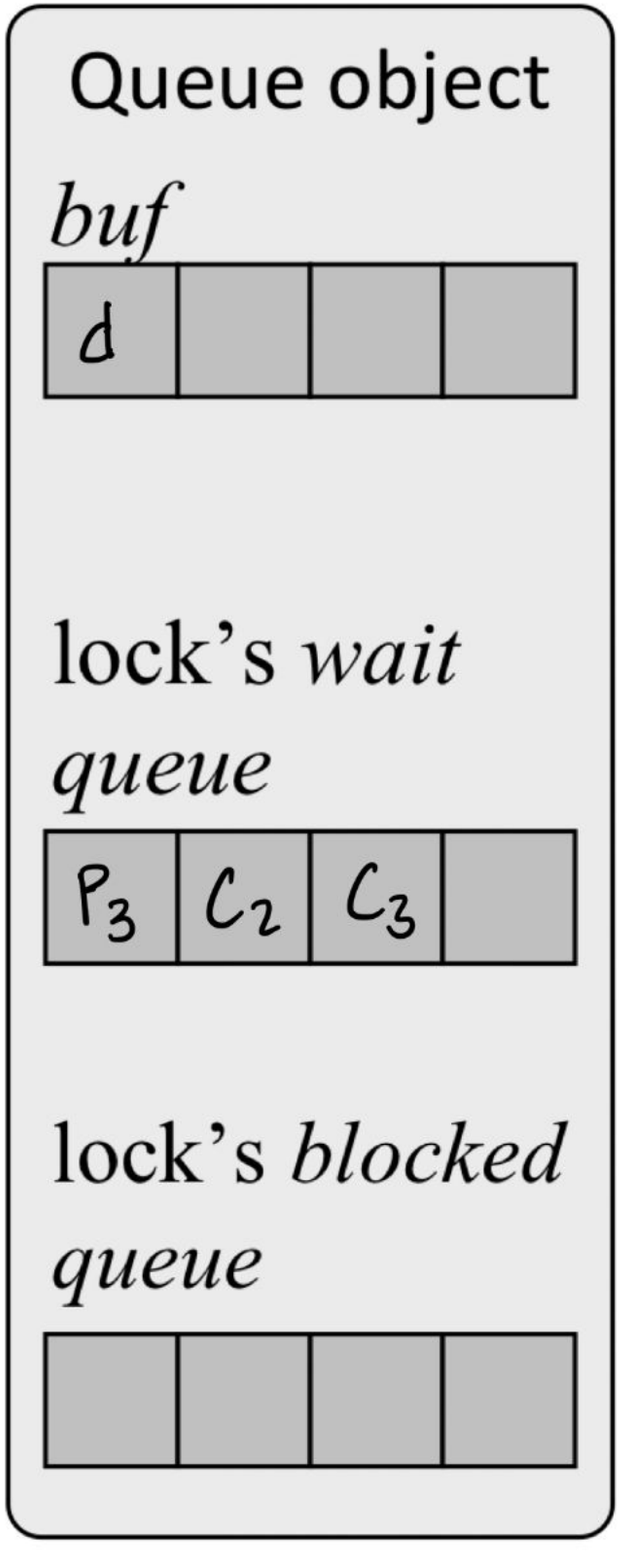
   b. The *notify* wakes up *P2*

   c. BUT, *C2* enters the method before *P2* can (*P2* must reacquire the lock), so *P2* blocks on entry to the put method;

   d. *C2* checks the wait loop, sees there are no more characters in the buffer and so it waits (releasing the lock in the process)

   e. *C3* enters the method after *C2*, but before *P2*, checks the wait loop, sees there are no more characters in the buffer, and so it waits

Show the status of *buf*, lock's *wait queue* and lock's *blocked queue.*

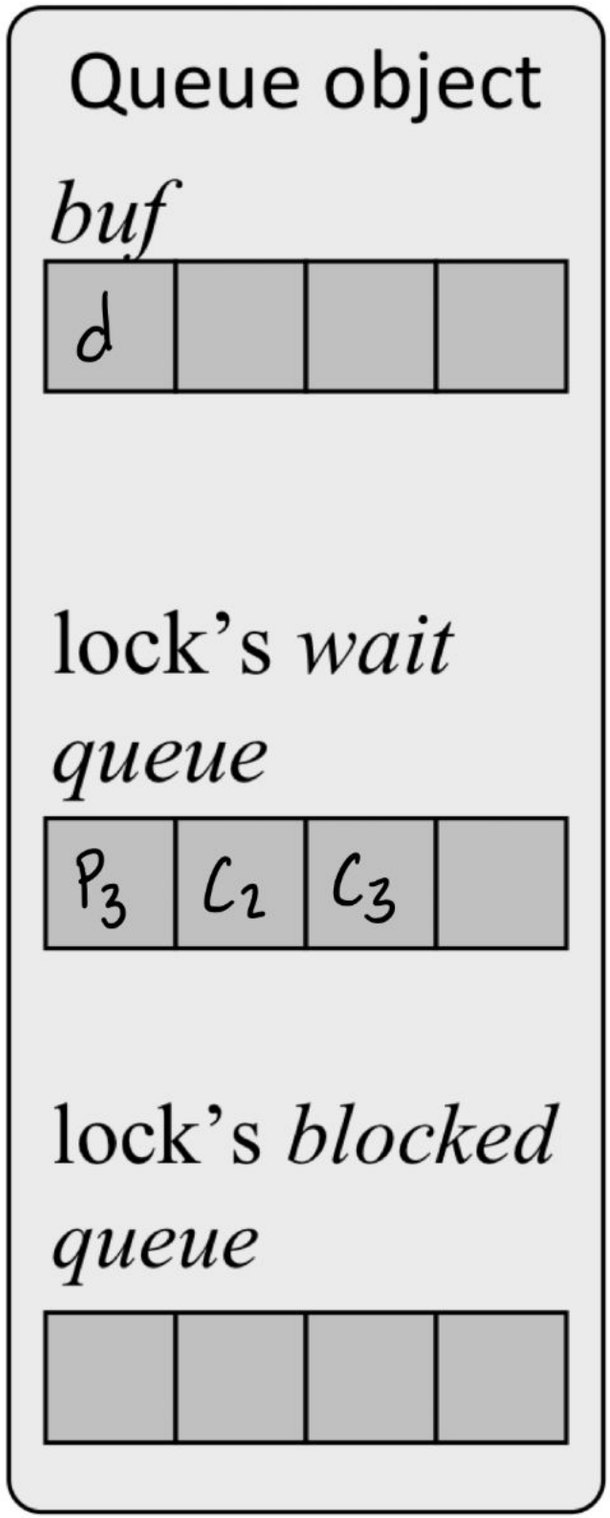


6. The following happen at time step 6.

   a. Now *P3*, *C2* and *C3* are all waiting!

   b. *P2* acquires the lock, puts a "*d*" in the buffer, calls *notify* and exits the method

Show the status of *buf*, lock's *wait queue* and lock's *blocked queue.*



7. The following happens at time step 7.

   a. *P2*'s notification wakes up *P3* (any thread can be woken up)

   b. *P3* checks the wait loop condition. There is already a character ("*d*") in the buffer and so it waits.

Show the status of *buf*, lock's *wait queue* and lock's *blocked queue.*

Is it possible for any thread to be woken up by another notify? No because all three threads are in the lock's wait queue

What would have happened if in 6b a notifyAll( ) was called?

Would've released the other 3 waiting threads