Subject: Task 5  
Posted by jdwastrack on Sun, 03 Apr 2016 21:07:00 GMT  
View Forum Message <> Reply to Message

Is there a way to construct multiple binary semaphores in scam?

Subject: Re: Task 5  
Posted by jarobinson3 on Mon, 04 Apr 2016 02:46:44 GMT  
View Forum Message <> Reply to Message

No. You will have to build your own.

Subject: Re: Task 5  
Posted by lusth on Tue, 05 Apr 2016 20:45:51 GMT  
View Forum Message <> Reply to Message

Scam has but a single semaphore. You can build a myriad of concurrency control structures off of that single semaphore.

Subject: Re: Task 5  
Posted by jmbeach1 on Mon, 18 Apr 2016 06:37:30 GMT  
View Forum Message <> Reply to Message

What should happen if the mutex runs out of resources. I am under the impression that we are initializing the mutex with n resources. Aka (define m (mmutex 3)) initializes a mutex with three resources. If I call ((m'p)) three times to acquire three resources back from it and then call ((m'p)) one more time, shouldn't I return a 'FORBIDDEN symbol? Or do you literally always return 'ACQUIRED when the user calls ((m'p))? 

Subject: Re: Task 5  
Posted by jzarob on Tue, 19 Apr 2016 04:01:19 GMT  
View Forum Message <> Reply to Message

I think the best thing to do would be to go into a wait loop where you keep checking the availability in the thread that's trying to request the mutex.

As soon as one of the threads that has one of the mutex in m and releases it by calling ((m'v)), the one that's waiting can acquire (and then the call ((m'p)) would return 'ACQUIRED)
I'm having trouble seeing how to do this with only one binary semaphore. I'm under the impression that this task would require two binary semaphores. One to protect the semaphore count in the function bodies. And one to prevent deadlocks on the other semaphore. Surely I'm missing something?

You only need a single semaphore.

Deadlock on what other semaphore? There is only a single semaphore.

Let a and b be binary semaphores.

```
(define (signal)
 (b 'wait)
 (update-semaphore-in-signal)
 (if (should-signal-a-in-signal)
   (a 'signal)
   (b 'signal))
)

(define (wait)
 (a 'wait)
 (b 'wait)
 (update-semaphore-in-wait)
 (if (should-signal-a-in-wait)
   (a 'signal)
   (b 'signal))
)
```

So are we supposed to replace one of those with busy waiting? Makes me feel kinda dirty.
Subject: Re: Task 5  
Posted by jarobinson3 on Sat, 23 Apr 2016 05:32:54 GMT  
View Forum Message <> Reply to Message

In scam there is only a single binary semaphore, where did you get the other? Also I have no clue what your "code" is doing.

Subject: Re: Task 5  
Posted by tscrompton on Sat, 23 Apr 2016 06:01:07 GMT  
View Forum Message <> Reply to Message

I'm not sure if we're on the same page. The code that I provided is pseudocode for the standard implementation of counting semaphores via binary semaphores. I purposely left out details from it to avoid providing too much of the answer to this assignment. My question was how we should go about solving this with just that single binary semaphore. I was asking if substituting busy waiting for one of the binary semaphores in the pseudocode above is the appropriate approach.

Subject: Re: Task 5  
Posted by jarobinson3 on Mon, 25 Apr 2016 22:16:40 GMT  
View Forum Message <> Reply to Message

I am not sure how you can replace a binary semaphore with busy waiting. Here is an idea that should work: Have a set of variables that hold the state of the two binary semaphores. Whenever you need to update one just lock, perform operation, then unlock.