Clicker Questions for December 4
Is this tail recursive?

(define fib (lambda (n)
  (cond
    [(= 0 n) 1]
    [(= 1 n) 1]
    [else (let ([a (fib (- n 1))][b (fib (- n 2))])
      (+ a b))])))

A. Yes
B. Yes, but it gives the wrong answer
C. No, because the let should be a letrec
D. No, because it does something with the values of the recursive calls.
Answer D: No, because it does something with the values of the recursive calls.
I want to look for ways to escape from a function when you are buried in a recursion. First of all, what does this function do?

```scheme
(define count (lambda (a lat)
    (cond
        [(null? lat) 0]
        [(eq? a (car lat)) (+ 1 (count a (cdr lat)))]
        [else (count a (cdr lat))])))
```

A. It counts stuff
B. It counts the number of instances of symbol 'a in lat.
C. It counts the number of instances of its first argument in its second.
D. It crashes
Answer C: It counts the number of instances of its first argument in its second.

For example (count 'x '(a x b x a a c a x)) is 3
Now, I want to change count so that if it ever finds a 'b in \textit{lat} it prints an error message. Consider this:

\begin{verbatim}
(define count (lambda (a lat)
    (cond
      [(null? lat) 0]
      [(eq? 'b (car lat)) "Oh no! I'm afraid of b's"]
      [(eq? a (car lat)) (+ 1 (count a (cdr lat)))]
      [else (count a (cdr lat))]]))
\end{verbatim}

What is \texttt{(count 'a '(a b c d c b a))}?

A. 2
B. An error
C. The "Oh no!" string
D. Something else
Answer B: An error, because it tries to add 1 to the "Oh no!" string.