Clicker Questions
October 30
The Lab 6 directions say to add symbols *True* and *False* to the initial environment. What will we bind *True* to?

A. #t
B. 'True
C. 1
D. (primproc 'True)
Answer b: 'true
The Lab6 directions also say you should add *equals?* to the list of primitive procedures. How will you compute (equals? a b)?

A. (eq? a b)
B. (eqv? a b)
C. (if (eq? a b) 'True 'False)
D. (if (eqv? a b) 'True 'False)
Answer D: (if (eqv a b) 'True 'False)

You want (equals? a b) to return 'True or 'False, not #t or #f.
In the standard Scheme if statement (if A B C), if A evaluates to #f the value of C is returned; if A evaluates to anything else the value of B is returned. We will parse an if-expression into a tree that holds 3 subtrees: the parse tree for the condition, the tree for the true-branch and the tree for the false-branch. Which pseudocode evaluates an if-expression in environment env?

A.  (if (eval-exp condition env) (eval-exp true-branch env) (eval-exp false-branch env))

B.  (let ([c (eval-exp condition env)])
    (if c (eval-exp true-branch env) (eval-exp false-branch env))

C.  (let ([c (eval-exp condition env)])
    (if (eq? c 'True) (eval-exp true-branch env) (eval-exp false-branch env))

D.  (let ([c (eval-exp condition env)])
    (if (eq? c 'False) (eval-exp false-branch env) (eval-exp true-branch env))
D. (let ([c (eval-exp condition env)])
  (if (eq? c 'False) (eval-exp falsle-branch env) (eval-exp true-branch env)))
In standard Scheme, what does (if (= x 0) 'bad (/ 100 x)) evaluate to when x is 2?
A. An error
B. bad
C. 50
D. +inf.0
Answer C: 50
In standard Scheme, what does \((\text{if } (= x 0) \text{ 'bad } (/ 100 x))\) evaluate to when \(x\) is 0?

A. An error
B. 'bad
C. 50
D. +inf.0
Answer B: 'bad
Consider the following rather goofy function:
(define iffy (lambda (a b c) (if a b c)))
What does (iffy (= x 0) 'bad (/ 100 x)) evaluate to when x is 0?

A. An error
B. 'bad
C. 50
D. +inf.0
Answer A: an error

iffy turns an if-expression into a function; when functions are called they always evaluate all of their arguments. So when the third argument (/ 100 x) is evaluated we get a divide-by-zero error.