

Clicker Questions

September 11

```
(define f
  (lambda (x)
    (lambda (y) (if (< y x) x y))))
```

What is (f 3)?

A. An error.

B. #t

C. A function; if we give this function an argument y it returns the smaller of y and 3. So ((f 3) 1) is 1 and ((f 3) 10) is 3

D. A function; if we give this function an argument y it returns the larger of y and 3. So ((f 3) 1) is 3 and ((f 3) 10) is 10

```
(define f  
  (lambda (x)  
    (lambda (y) (if (< y x) x y))))
```

What is (f 3)?

Answer D: A function; if we give this function an argument y it returns the larger of y and 3. So ((f 3) 1) is 3 and ((f 3) 10) is 10

What is the difference between:

```
(define f  
  (lambda (n)  
    (let ([base 1])  
      (if (< n base)  
          base  
          (* n (f (- n 1))
```

```
(define g  
  (let ([base 1])  
    (lambda (n)  
      (if (< n base)  
          base  
          (* n (g (- n 1))
```

- A. f is the factorial function, g gives an error.
- B. g is the factorial function, f gives an error
- C. Both are correct, f is more efficient
- D. Both are correct, g is more efficient

Answer D: Both are correct, g is more efficient

What is the difference between the two expressions

```
( (lambda (x y) (* (+ x 2) y)) 3 4)
```

```
(let ([x 3] [y 4] )  
    (* (+ x 2) y))
```

- A. The first evaluates to 18 and the second to 20.
- B. Both evaluate to 20 but the first is more efficient.
- C. Both evaluate to 20 but the second is more efficient.
- D. There is no difference

Answer D: There is no difference

Both expressions are evaluated by extending the current environment with bindings of x to 3 and y to 4, and then evaluating $(* (+ x 2) y)$ in this extended environment.