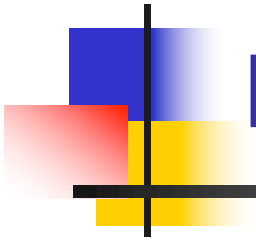
Announcements

- Accommodations and zone assignment out tomorrow
- Rules same as for Exam 1 with some tweaks; announcement tomorrow on Submitty
- Makeup date (tentative): Wed Nov.12, 5-8pm

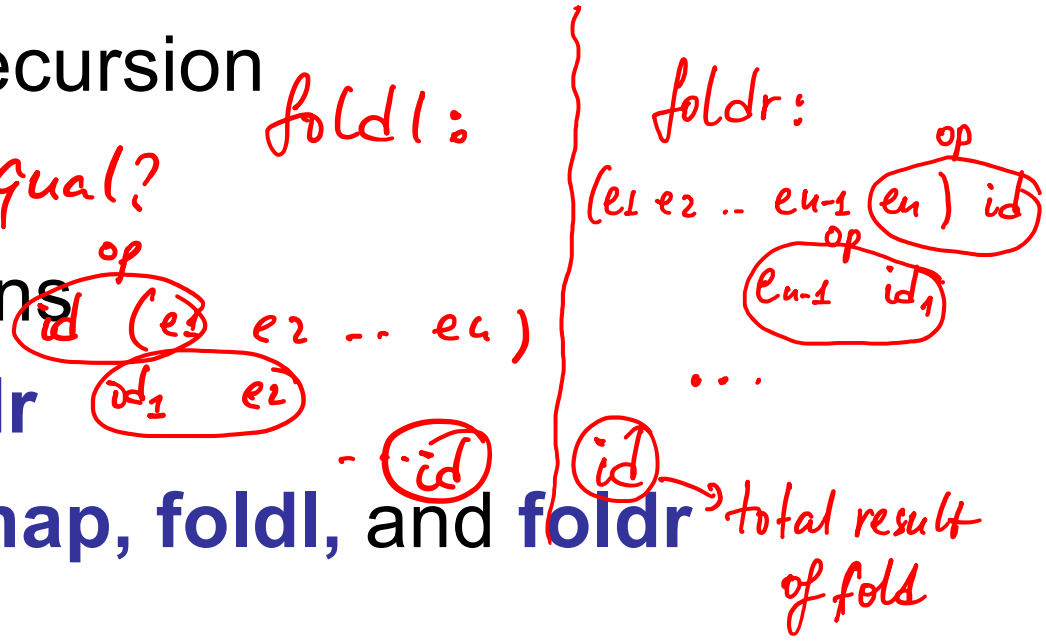
- Homework
  - Write your contracts
  - Adhere to syntax and functionality restrictions



Exam 2 Topics

Topics

- Scheme (Lectures 12 and 13, plus chapters)
  - S-expression syntax
  - Lists and recursion
  - Shallow and deep recursion
  - Equality
  - Higher-order functions
  - **map, foldl,** and **foldr**
  - Programming with **map, foldl,** and **foldr**
  - Tail recursion



Topics

- Scheme (Lecture 14, plus chapters)
  - Binding with **let, let*, letrec**
  - Scoping in Scheme
  - Closures and closure bindings

Topics

- Scoping, revisited (Lecture 14, plus chapters)
  - Static scoping
    - Reference environment
    - Functions as third-class values vs.
    - Functions as first-class values
  - Dynamic scoping
    - With shallow binding
    - With deep binding

Topics

- Lambda calculus (Lectures 15, 16, and 17)
  - Syntax and semantics
  - Free and bound variables
  - Substitution
  - Rules of the Lambda calculus: Alpha-conversion and Beta-reduction
  - Normal forms NF, HNF, WHNF
  - Reduction strategies: Normal order and Applicative order
  - Fixed-point combinator and recursion

Quiz 5

Question 1. (2pts) Scheme's scoping discipline is

Select one:

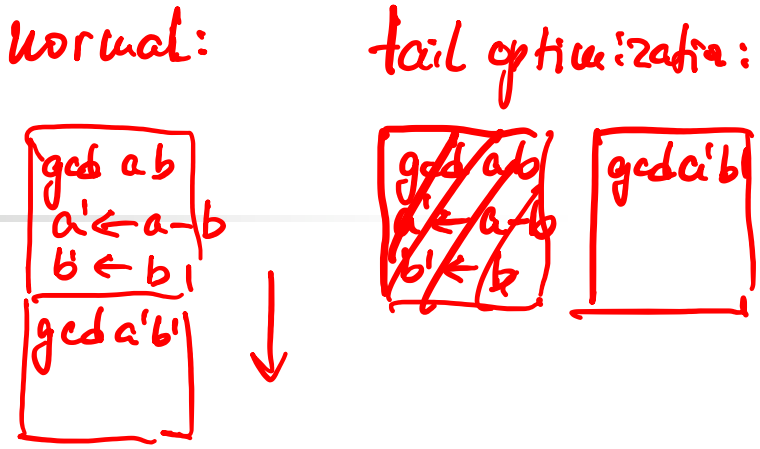
- static scoping
- dynamic scoping

Question 2. (2pts) Scheme's typing discipline is

Select one:

- static typing
- dynamic typing

Quiz 5



Questions 3 and 4 refer to the following Scheme function:

```
(define (fun a b)
  (cond ((= a b) a)
        ((> a b) (fun (- a b) b))
        (else (fun a (- b a)))))
```

Question 3. (2pts) What does `fun` compute? Note: you may assume that the arguments are positive integers.

GCD

Question 4. (2pts) `fun` is tail-recursive.

Select one:

- true
- false

Quiz 5

(define (atom? x)
(not (pair? x)))

Question 5. (2pts) Function `atomcount`, defined below, attempts to count the number of atoms nested in a list. Recall predicate `atom?` that we wrote in class — it returns true if given an object that is *not a pair* (i.e., an object we cannot take the `car` or the `cdr` of).

```
(define (atomcount lis)
  (cond ((atom? lis) 1)
        ((null? lis) 0)
        (else (+ (atomcount (car lis)) (atomcount (cdr lis))))))
```

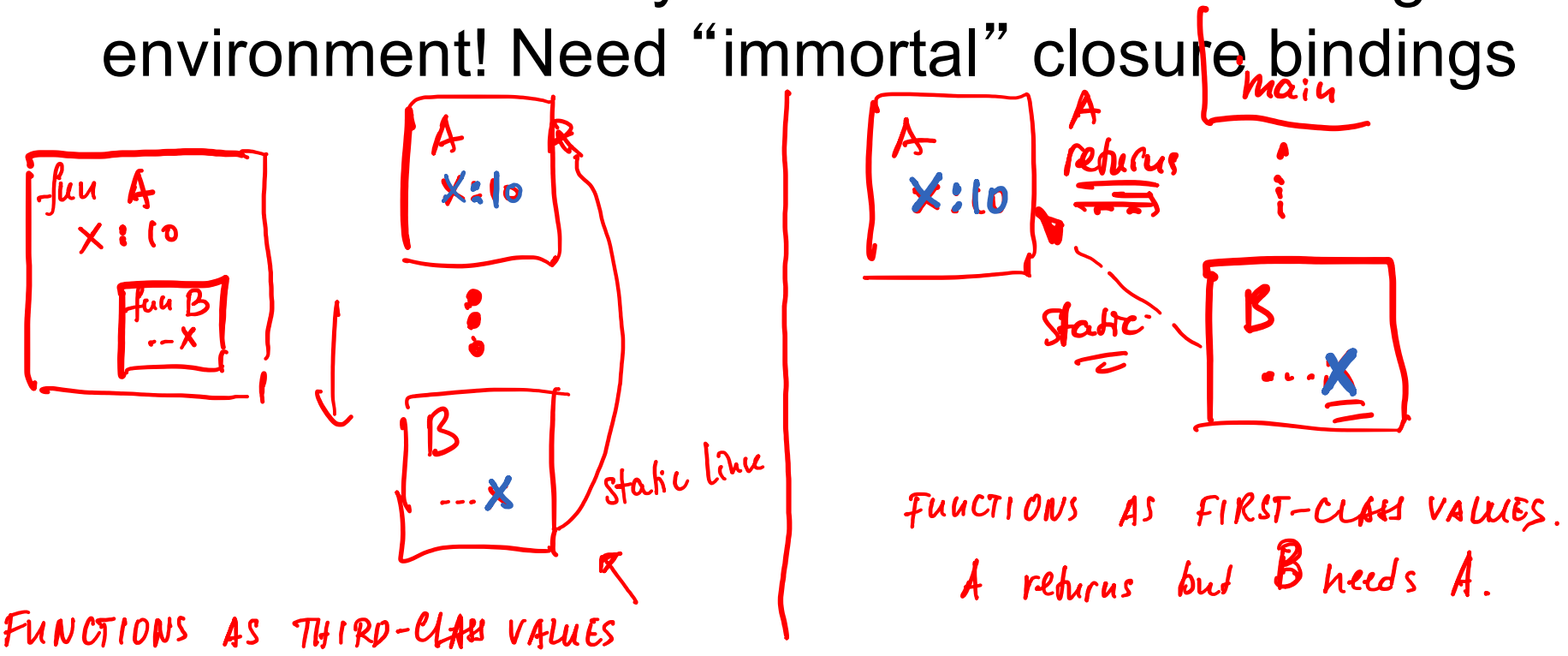
`(atomcount '(1 (2 (3))))` yields

6

(atom? 'a) → #t
(atom? '(a)) → #f
(atom? '()) → #t

Scoping with First-Class Functions

- When functions are first-class values, static scoping gets more involved
  - Function value may outlive static referencing environment! Need "immortal" closure bindings



Scoping with First-Class Functions

- Dynamic scoping gets more involved as well
  - Shallow binding vs. deep binding

- Dynamic scoping with shallow binding
  - Reference environment for function/routine is not created until the function is called
    - All non-local references are resolved using the most-recent-frame-on-stack rule --- just follow dynamic links

Scoping with First-Class Functions

- Dynamic scoping with deep binding
  - When a function/routine is passed as an argument, the code that passes the function/routine has a particular reference environment (the current one!) in mind. It passes this reference environment along with the function value (it passes a closure).

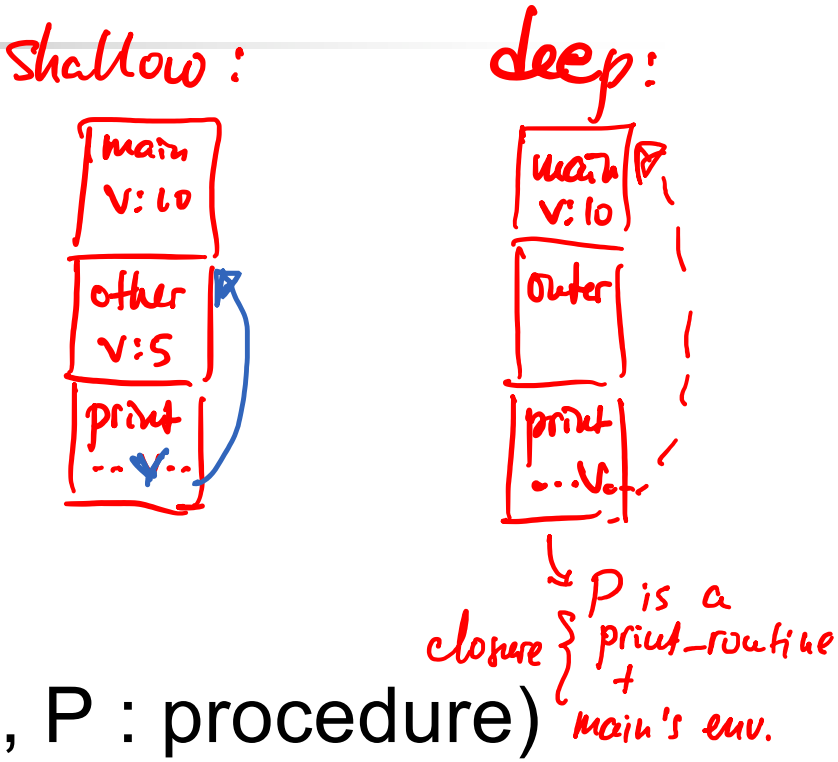
Example

```
v : integer := 10
people : database

print_routine (p : person)
  if p.age > v
     write_person(p)

other_routine (db : database, P : procedure)
    v : integer := 5
    foreach record r in db
        P(r)

other_routine(people, print_routine) /* call in main */
```



Quiz 6

Questions 1-3 below refer to the following Scheme function:

```
(define A
   (lambda ()
      (let* ((x 2)
             (C (lambda (P) (let ((x 4)) (P) )))
             (D (lambda () x))
             (B (lambda () (let ((x 3)) (C D)))))
        (B))))
```

Question 1. (1pts) What gets printed when we call the function in the interpreter?

2

D is "constant 2" function

Question 2. (2pts) What would get printed if Scheme used *dynamic scoping with shallow binding*?

4

D is just λ().x function.
x is resolved following dynamic link

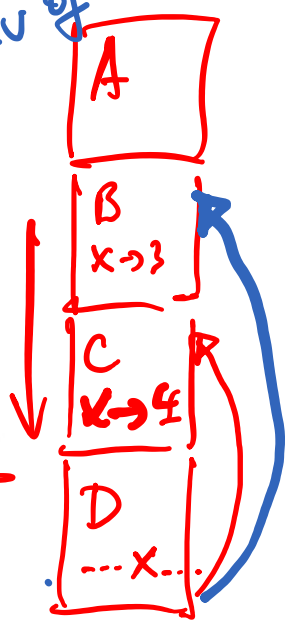
Question 3. (2pts) What would get printed if Scheme used *dynamic scoping with deep binding*?

3

D is a closure { λ().x + env. of B

with shallow: D is just λ().x

with deep: { λ().x ref env of B



Quiz 6

Question 4. (2pts) Willy Wazoo wrote a function `f`. What does it do?

```
(define (f lis)
  (foldl (lambda (x y) (if (and (number? x) (> x 5)) (cons x y) y)) lis '()))
```

Select one:

- selects all numerical elements in lis greater than 5
- selects all numerical elements in lis less or equal than 5
- neither; Willy's function has a bug

Quiz 6

Question 5. (1pt) Consider the lambda term $(\lambda v.v)\ ((\lambda v.v)\ (\lambda x.(\lambda x.x)\ z))$. There are this many reducible expressions in this term:

3

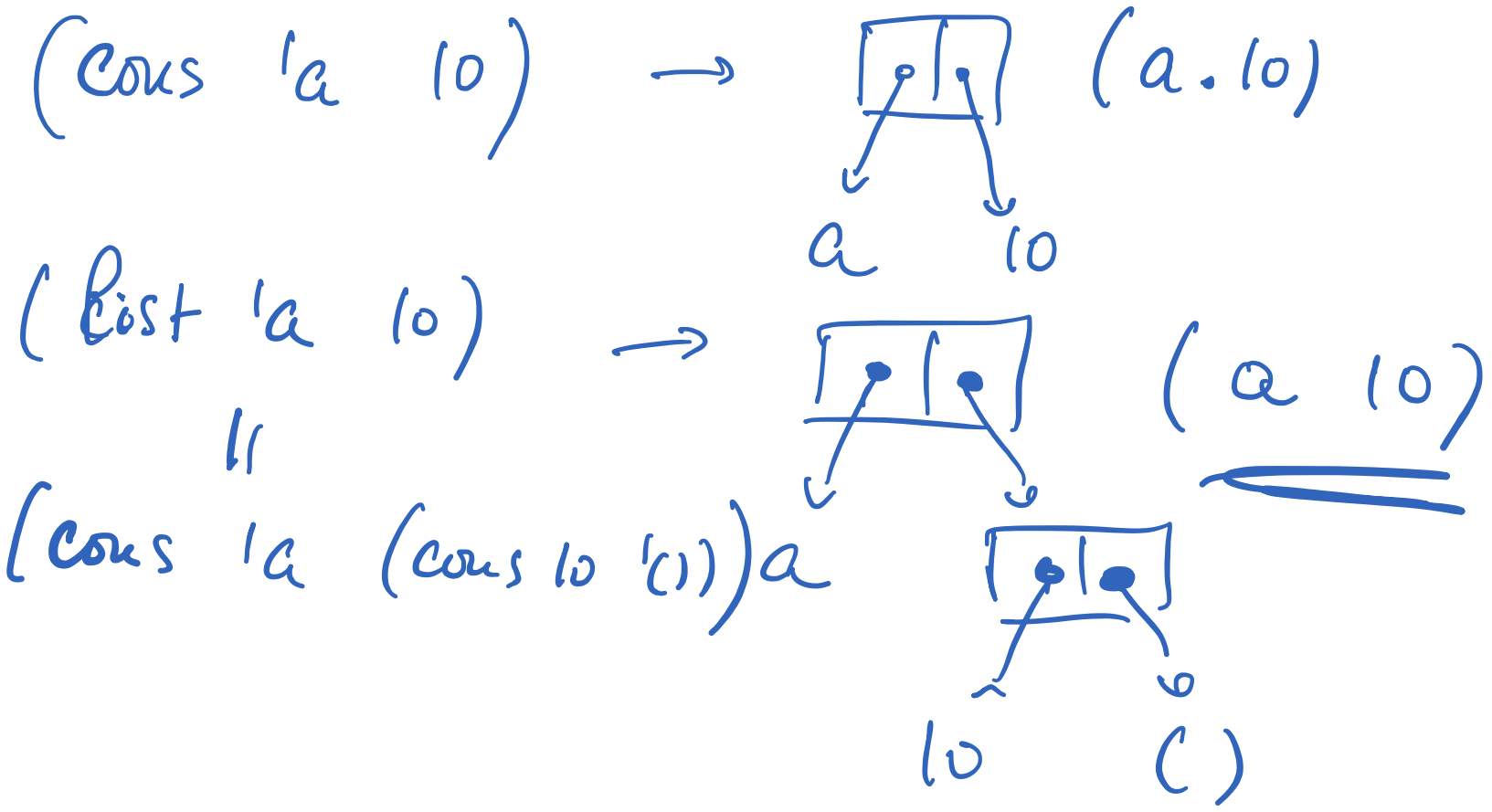
Question 6. (1pt) Is lambda term $x\ ((\lambda y.y)\ z)$ in Weak Head Normal Form (WHNF)?

Yes

Question 7. (1pt) Is term $x\ ((\lambda y.y)\ z)$ in Normal Form (NF)?

No

Questions?



The End