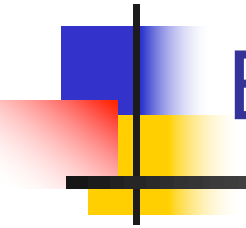

Exam 2 Topics

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 15, 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 and 16)
  - Syntax and semantics
  - Free and bound variables
  - Substitution
  - Rules of the Lambda calculus
    - Alpha-conversion
    - Beta-reduction
  - Normal forms
  - Reduction strategies
    - Normal order
    - Applicative order

Quiz 5

- Question 1. Scheme's scoping discipline is
  - (a) static scoping
  - (b) dynamic scoping

Quiz 5

- Question 2. Scheme's typing discipline is

  (a) static typing

  (b) dynamic typing

Quiz 5

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

- Question 3. What does fun compute?

- Question 4. fun is tail-recursive.

  (a) true

  (b) false

Quiz 5

- Question 5. Function atomcount attempts to count the number of atoms nested in a list

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

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

(a) 6

(b) 3

(c) 4