**Lisp**

CSCI 334
Stephen Freund

---

**Factorial**

- Two cases:
  - base case: $n$ is 0
  - recursive case: $n \times \text{fact}(n-1)$

- \(\text{(defun fact (n)}\)
  \(\text{(cond ((eq n 0) 1)}\)
  \(\text{(t (* n (fact (- n 1))))})\)

- \(\text{(fact 3)} \rightarrow 6\)

---

**List Length**

- Two cases
  - base case: empty list
  - recursive case: process "car", recurse on "cdr"

- \(\text{(defun length (l)}\)
  \(\text{(cond ((eq l nil) 0)}\)
  \(\text{(t (+ 1 (length (cdr l))))})\)

---

**Membership**

- Does a list contain a specific atom?

- \(\text{(contains 'A '(C A M P)) \rightarrow t}\)
- \(\text{(contains 'B '(C A M P)) \rightarrow nil}\)

- \(\text{(defun contains (x l)}\)
  \(\text{(cond ((eq x nil) nil)}\)
  \(\text{((eq x (car l)) t)}\)
  \(\text{(t (contains x (cdr l))))})\)
Lisp (2)

CSCI 334
Stephen Freund

Recap
• Expressions
  - (+ 1 2 3) -> 6
• Lists
  - nil
  - (cons 'A '(B C)) -> (A B C)
  - (car '(A B C)) -> A
  - (cdr '(A B C)) -> (B C)
  - (car (cdr '(A B C))) -> B
  - also have length, append, reverse

Recap (2)
• Conditional
  - (cond (test1 result1) ... (testn resultn))
  - tests:
    • (eq x y)
    • (< x y)
    • (atom x) (will want to use in problem set)
• Functions
  - (defun cube (x) (* x x x))
  - [sheep]
• (division)
• (wrap up insertion-sort)

Notes
• Book uses slightly different dialect of Lisp
• Use class notes as a reference, or the Lisp tutorial on the web site.
• Major differences
  - use T instead of true
  - use (defun ...) instead of (define ...)
  - use (mapcar #'f l) instead of (maplist f l)

Insert
• Insert a number into a sorted list
• (insert 4 '(1 3 7)) -> (1 3 4 7)
• (defun insert (x l)
    (cond ((eq l nil) (cons x nil))
          ((< x (car l)) (cons x l))
          (t (cons
            (car l)
            (insert x (cdr l)))))))

Encoding Trees

Cow
BADGER
Eel
Aardvark Donkey Fox
**Encoding Trees**

Cow  
Badger  Eel  
Aardvark  Donkey  Fox

'(Cow (Badger (Aardvark nil nil) nil)  
(Eel (Donkey nil nil) (Fox nil nil)))

**Encoding Records**

class Book {  
    String author;  
    String title;  
    int year;  
} ...

**Encoding Book Records**

List Form: (Author Title Year)  
ex: (McCarthy Lisp 1960)  
(defun author (book) (car book) )  
(defun title (book)  (car (cdr book)) )  
(defun year (book)   (car (cdr (cdr book))))  

(author '(McCarthy Lisp 1960)) -> McCarthy  
title '(McCarthy Lisp 1960)) -> Lisp  
(year '(McCarthy Lisp 1960)) -> 1960

**Lisp Memory Model**

- Cons cell:  
  - Address  
  - Decrement

- Atom:  
  - Atom  
  - value

- (cons 'A (cons 'B (cons 'C nil)))

**Sharing**

(a)  
(b)  

- Both structures could be printed as (A.B)(A.B)  
- Which is result of evaluating  
  (cons (cons 'A 'B) (cons 'A 'B)) ?
Garbage Collection

Clear Tags

Mark Reachable Cells

Mark Reachable Cells
Mark Reachable Cells

Free Unreachable Cells

Programs As Data

Genetic Programming

Mark Reachable Cells

Programs As Data

Genetic Programming
Genetic Programming

```
(defun move (...)  
  (cond (test ...)  
    (t (m ...))))

(defun move (...)  
  (f (+ ...)  
    (...)))

(defun move (...)  
  (g (h ...)  
    (...)))
```

Mutate