6.090
Building Programming Experience

Lecture 6

1/18/2007

Outline

- Sorting
- Searching

Sorting

- Classing Computer Science Problem
- Put the list (3 4 5 1 3 2 1) in order
  - (1 1 2 3 3 4 5)
- Break down into several steps

insert

- Write a procedure **(insert** itm lst) that takes an already sorted list and inserts *itm* into it
- Example:
  - (insert 1 (list 3 4 5)) → (1 3 4 5)
  - (insert 3 (list 1 2 4 5)) → (1 2 3 4 5)

New Special form: cond

- Cond can replace a set of nested if expressions:
- (**cond** (*test1 result1*)
- (*test2 result2*)
- ….
- (**else** *value*))

insert

```
(define (insert itm lst)
  (cond ((null? lst) (list itm))
        ((< itm (car lst))
         (cons itm lst))
        (else
         (cons (car lst)
               (insert itm (cdr lst)))
         )))
```

Sorting

- Can we use insert to sort an unsorted list?
- (define (insert-sort lst)

Insert-sort

```
(define (insert-sort lst)
 (if (null? lst) lst
     (insert (car lst)
             (insert-sort (cdr lst)))))
```

Searching

- Another classic problem
- Is 5 in the list (1 2 3 6 7)
- What's the simple way of solving this?
- (define (contains? elm lst) ….

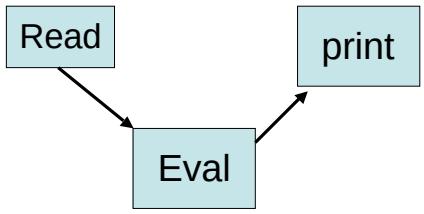
Contains?

```
(define (contains? elm lst)
  (cond ((null? lst) #f)
        ((= (car lst) elm) #t)
        (else (contains? elm (cdr lst)))))
```

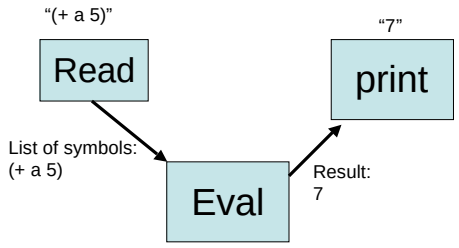
Searching for words

- What if we want to see if a list contains a certain word?
- Scheme has two options:
  - Strings: ("foo" "bar" "baz")
  - Symbols: (foo bar baz)

What's a symbol?



What's a symbol?



Creating Symbols

- New special form: (**quote** *expr*)
- Examples:
- (quote a) → a
- (quote 5) → 5
- (quote (+ a 5)) → (+ a 5)
- Shortcut for writing:
  - '(+ a 5) → (quote (+ a 5))
  - 'a → (quote a)
  - Use ' not ` (backquote)

Operations on symbols

- Test if two symbols are equal:
  - (eq? 'a 'b) → #f
  - (eq? 'a 'a) → #t
- eq? works on small integers, but not other numbers
  - Use = for other numbers

Searching for words

- Is the word bar in the list '(foo bar baz)

```
(define (contains? elm lst)
  (cond ((null? lst) #f)
        ((eq? (car lst) elm) #t)
        (else (contains? elm (cdr lst)))))
```

(contains? 'bar '(foo bar baz)) → #t

How many times?

- How many times does foo appear in a list '(foo bar foo baz quux)?
- Use higher order procedures
- (define (how-many elm lst) …

Filter

- Like **map**, **filter** is a common HOP
- Remove all elements from lst where pred is false
- (define (filter pred lst) …)

Filter

```
(define (filter pred lst)
  (cond ((null? lst) null)
        ((pred (car lst))
         (cons (car lst)
               (filter pred (cdr lst))))
        (else (filter pred (cdr lst)))))
```

Back to how-many

- (how-many ‘a ‘(a b a b c d a)) → 3

Back to how-many

- (how-many ‘a ‘(a b a b c d a)) → 3
- If we use filter, we can get just the a’s:

```
(filter (lambda (x) (eq? x ‘a))
        ‘(a b a b c d a))
   ;Value: (a a a)
```

All together

```
(define (how-many e lst)
  (length (filter
           (lambda (x) (eq? x e))
           lst)))
```

Quiz

- Quiz Tomorrow (Friday)
  - Intended to be like 6.001 quiz 1
  - Open notes (but no computers)
  - Graded like problem sets
    - Feedback provided, but not used for 6.090 grade