CS3L:
Introduction to Symbolic Programming

Lecture 18:
HOF

Summer 2008 Colleen Lewis
colleenL@berkeley.edu

Announcements

- Midterm two Tuesday July 29th
  - This one will probably be harder than the first
- Colleen will be out of town Thurs-Monday
- Homework
  - Miniproject 3 starts Today! Due Friday July 25th

Today

- **lambda**
- **every**
- **keep**
- **Accumulate**
- Global variables
- Tick-tack-toe (ttt)

lambda

(lambda (*arg1 arg2* …) body…)

```
(define (square x) (* x x))

(define square (lambda (x)(* x x))
THESE ARE THE SAME!
```

```
(define (square y) (lambda (x) (* x x)))
(define square (lambda (y) (lambda (x) (* x x)))
These are the same! Called: ((square 1) 4)➔ 16
```

every

(**every** *procedure sent*)

- *procedure*
  - a procedure that takes in **one argument**
  - a procedure that returns a word or a sentence
- *sent*
  - a sentence with 0 or more words
  - a word with 0 or more letters

every example

```
(define (change-1st-char char sent)
  (every
    (lambda (wd) (word char (bf wd)))
    sent))
> (change-1st-char 'x'(eat get let))
```

```
(se
  (lambda ('eat) (word 'x 'eat)   (lambda ('get) (word 'x 'get)   (lambda ('let) (word 'x 'let)
    'xat                            'xet                            'xet
```

keep

(**keep** *procedure sent*)

- *procedure*
  - a procedure that takes in **one argument**
  - a procedure that returns #t or #f
- *sent*
  - a sentence with 0 or more words
  - a word with 0 or more letters

keep example

```
(define (keep-equal? wd sent)
  (keep
   (lambda (one-wd)
     (equal? wd one-wd))
   sent))

> (keep-equal? 'cat '(car cat con))
```

```
(se
  (lambda ('car)
  (equal? 'cat 'car))
      '()
  (lambda ('bar)
  (equal? 'cat 'cat))
      'cat
  (lambda ('con)
  (equal? 'cat 'con))
      '()
```

accumulate

(**accumulate** *procedure sent*)

- *procedure*
  - a procedure that takes in **two arguments**
  - a procedure that combines things together
- *sent*
  - a sentence with 1 or more words
  - a word with 1 or more letters

accumulate

```
(accumulate (lambda () …) '(cat dog hat tag sag)
```

```
'cat  'dog  'hat  'tag  'sag
 ↓     ↓    ↓      ↓
                    '?
                '??
        '???
'????
```

How to use global variables

```
(define  (pi)  3.1415)
(define circumference
     (lambda (diameter)
               (* diameter (pi))))
```

```
(define pi 3.1415)
(define circumference
     (lambda (diameter)
               (* diameter pi)))
```

The board

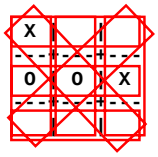
```
 X |   |
---+---+---
 O | O | X
---+---+---
   |   |
```

```
"X _ _"

"O O X"  ──────→  "X _ _ O O X _ _ _"

"_ _ _"
```

Triples (another representation of a board)



"X _ _ O O X _ _ _"

(x23 oox 789 xo7 2o8 3x9 xo9 3o7)