CS3L:
Introduction to Symbolic Programming

Lecture 15:
Procedures as Arguments

Summer 2008 Colleen Lewis
colleenL@berkeley.edu

Today

- Homework
  - **Mon:** Mini-project 2 due Thursday
- Use the Modeler
- Procedures as Arguments
- Procedures returned from Procedures
- Higher Order Functions (HOF)
- & Recursion Patterns

REALLY IMPORTANT! Use the Modeler!

- The exercises that use the modeler are really helpful!
- If you don't know how to do them – ASK!

Procedures can be taken as arguments…

```
(define (math-function? func)
  (or (equal? func +)
      (equal? func -)
      (equal? func *)
      (equal? func /)))

STk> (math-function +)
#t
STk> (math-function '+)
#f
```

…and procedures can be returned from procedures

```
(define (choose-func name)
   (cond ((equal? name 'plus) +)
         ((equal? name 'minus) -)
         ((equal? name 'divide) /)
         (else 'sorry)))


STk> ((choose-func 'plus) 3 5)
8
STk> ((choose-func 'minus) 3 5)
-2
```

Higher order function (HOFs)

- A HOF is a function that takes a function as an argument.

```
(define (do-math f arg1 arg2)
  (if (and (equal? arg2 0)
           (equal? f /))
      '(error – divide by zero)
      (f arg1 arg2)))

  (do-math – 10 3)
  ;; should I quote the "-"?
```

Patterns for simple recursions

- Most recursive functions that operate on a sentence fall into:
  - Mapping `<- every`
  - Counting
  - Finding
  - Filtering `<- keep`
  - Testing
  - Combining `<- accumulate`

Common HOFs

- There are three main ones that work with words and sentences:
  - **`every`** - do something to each element
  - **`keep`** - return only certain elements
  - **`accumulate`** - combine the elements

```
STk> (every last '(hi how are you))
(i w e u)
```