Administrative Notes

Homework

- Short homework will be available later today
- Due Wednesday
- Worth five points

Exams

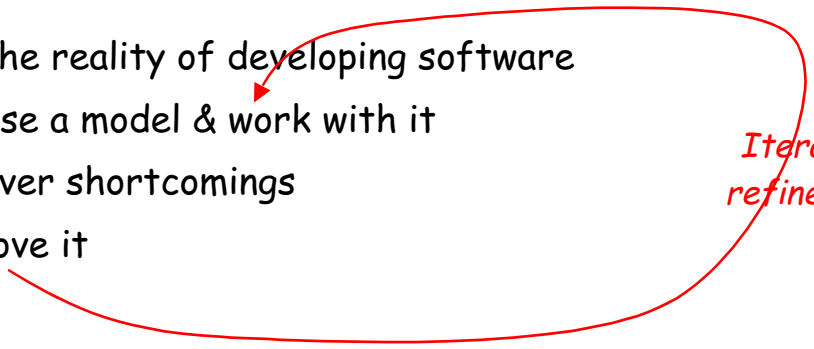
- Graded, will hand back at end of class

Lessons from Family Trees

1. Started with intuitive, child-centric model
   - → Built some programs, answered some questions
   - → Discovered some shortfalls
2. Formulated another model, the parent-centric tree
   - → Built some programs, answered some questions
   - → Discovered some shortfalls

This is the reality of developing software

- Propose a model & work with it
- Discover shortcomings
- Improve it



Lessons from Family Trees

Mutually-recursive data structures

- Parent-centric trees had two parts
  - → Parent and list-of-parent
  - → Relationship shown in the arrows & in the templates
- This is an important and recurring pattern
  - → One program per data structure or data item
  - → Calls between them to reflect the relationships in the data

Remember, in COMP 210, the structure of the data dictates the structure of the program

Files and Directories

File System

- Notion of files is ubiquitous in computing
- File is a named collection of data, typically stored on some non-volatile media
- Files have names to help us work with them
- DrScheme lives in a file, these notes live in a file, …
- Files are organized in some hierarchical fashion

Notion of directory or folder

- Critical for information hiding, naming, and understanding
- Common notion in most file systems

Files and Directories

Characterizing a file system

- Two kinds of objects: files and directories

Questions

- Can directory hold a directory?
- How deep can they go?
- How long can a file's name be?
- How many directories can hold a given file?
- How many directories can hold a given directory?

Files and Directories

Simple model of a file system

→ Files represented as symbols

→ Directory is a list of its contents

```
;; a directory is a structure
;;   (make-dir  name  contents)
;; where name is a symbol and
;;   contents is a list of files and directories
(define-struct dir  (name contents))

;; a lofd (list-of-files-and-directories) is one of
;;   – empty, or
;;   – (cons f r)  where f is a file and r is an lofd, or
;;   – (cons f r)  where f is a dir  and r is an lofd
```

;; a file is a symbol

Files and Directories

Template for File System

```
(define (f  a-file …)  …)                ;; simple template for file

(define (g a-dir …)                      ;; structure template for dir
   (   …  (dir-name a-dir) …
      …  (h  (dir-contents a-dir) …) … )

(define (h a-lofd …)                     ;; list of several types for lofd
  (cond
    [(empty? a-lofd)          … ]
    [(symbol? (first a-lofd))
          … (f  (first a-lofd) … ) …
          … (h  (rest a-lofd) … ) … ]
    [(dir? (first a-lofd))
          … (g  (first a-lofd) … ) …
          … (h  (rest a-lofd) … ) … ]
   ))
```

Write the program

```
;; Count-files : dir -> number
;; Purpose: return # of files in tree
(define (count-files  a-dir) … )
```


Files and Directories

Count-files

```
;; count-files: dir -> number
(define (count-files a-dir)              ;; if directories don't count as
    (count-lofd  (dir-contents a-dir)) ) ;; files …

;; count-lofd: lofd -> number
(define (count-lofd a-lofd)
   (cond
     [(empty? a-lofd)          0]
     [(symbol? (first a-lofd))
           (add1 (count-lofd  (rest a-lofd)) ) ]
     [(dir? (first a-lofd))
           (+  (count-files  (first a-lofd))
               (count-lofd  (rest  a-lofd)) ) ]
    ))
```

Files and Directories

Count-files

```
;; count-files: dir -> number
(define (count-files a-dir)                ;; if directories do count
     (add1 (count-lofd  (dir-contents a-dir)) ) )

;; count-lofd: lofd -> number
(define (count-lofd a-lofd)
   (cond
     [(empty? a-lofd)          0]
     [(symbol? (first a-lofd))
           (add1 (count-lofd  (rest a-lofd)) ) ]
     [(dir? (first a-lofd))
           (+  (count-files  (first a-lofd))
               (count-lofd  (rest  a-lofd)) ) ]
    ))
```


Files and Directories

Depth-dir

→ Write a program that consumes a dir and produces a number indicating how many levels of nested directories are in the tree

```
;; depth-dir: dir -> number
(define (depth-dir a-dir)
     (add1 (depth-lofd  (dir-contents a-dir)) ) )

;; depth-lofd: lofd -> number
(define (depth-lofd a-lofd)
   (cond
     [(empty? a-lofd)          0]
     [(symbol? (first a-lofd))
           (depth-lofd  (rest a-lofd)) ]
     [(dir? (first a-lofd))
           (max  (depth-dir  (first a-lofd))
                 (depth-lofd  (rest  a-lofd)) ) ]
    ))
```

Results of the First Exam

Statistics

- Range of grades: 16 to 99 points (out of 100)
- Average: 77.5
- Standard deviation 21

Histogram

- → 90-100: 14
- → 80-89: 9
- → 70-79: 7
- → 16-69: 8

Remember:

This exam is 10% of your grade, or about two homeworks.

Next exam is 20% of your grade.

Resul

You can recover from a bad performance on this first exam.

The second and third exams have more weight.

Homework is 50% of your grade.

Advice

- 90-99: Keep up the good work
- 80-89: More attention to details
  - → Contracts, templates, recursion, syntax
- 70-79: More practice
  - → Do your homework independently
  - → Check it with your homework partner
- 16-69: Deeper problems
  - → Some time pressure, some fundamental misunderstandings
    - Syntax, templates
  - → Work extra problems from the book
    - Exercises at the end of each section
    - Make them run in DrScheme
  - → Go to office hours

Attend class & lab lectures