Iterative Refinement

Lecture 14
30 October 2019
Updates
A Pre-Halloween Warning:

This classroom was constructed on the site of the creepy, haunted attic of Sanders Physics.

😱
Asprey Lecture

Tile: Building Devices to Help you Find Your Belongings

3:15–4:15 p.m. Wednesday, 30 October SP 105

Jossie Haines, Platform Engineering Director at Tile, will explain how they built a secure location finding network that supports getting 1.5 billion unique location updates per day. This is the world’s largest lost and found community spanning over 230 countries and territories and locating more than 5 million unique items every day. She will cover the whole stack from building a secure Bluetooth transport layer to developing a scalable backend service on AWS.

Refreshments will be served
Assignment 3 Extra Credit

The assignment requires you to make the balls bounce off the sides of the window.

For extra credit, make the balls bounce when they collide with each other.

Only try this if you have everything else done. There’s no extra credit for doing this if the base assignment isn’t Satisfactory or Excellent.
Academic honesty in CS
What’s appropriate and inappropriate when you’re programming may not always feel clear to you, so I want to go over it.
If you’re told you can work in a pair, as we’ve done on labs, it’s fine to collaborate.

You must acknowledge your collaborators, e.g., by putting both of your names in a comment.
Homework assignments and exams must be completed individually.

The only *people* you should consult are the coaches for this section of 101 – Jonathan, Max, and Norman – and me.

Don’t ask for help from other students in this class, from other sections of 101, from people who took the class before, from people online, etc.
Written resources you’re always free to consult:

The textbook ([htdp.org](htdp.org))

The Racket documentation (from DrRacket’s Help menu or by going to [racket-lang.org](racket-lang.org))

Slides, handouts, previous solutions, etc. on our course website ([cs.vassar.edu/~cs101/5](cs.vassar.edu/~cs101/5))
Consulting other sources puts you in danger of academic dishonesty (cheating):

On the Internet you might find solutions from other sections of 101, either this semester or in the past courses at other schools random people

Using code and ideas from these sources in your assignments or on an exam is equivalent to copying sentences from someone else’s book, essay, etc. and claiming them as your own.
Our homework assignments are intended for you to think through the problems and, if you get stuck, to talk them through with the coaches or me.

The coaches and I will do our best to help you in a way that leads you to understand **what** to do, rather than **giving you the answers**. Other people and other resources won’t be careful about that, and using someone else’s code or ideas undermines this learning process.
The course syllabus states:

**Academic Integrity**

Please don’t cheat. Read [Going to the Source](#). Since this course is concerned with composing code, the guidelines that apply to writing in general apply equally to the writing of computer programs. Copying someone else’s code without attribution amounts to plagiarism. So give proper attribution for the help you receive. Quoting from Chapter X of Going to the Source, “In suspected cases of plagiarism, the instructor prepares a written statement of complaint to the Academic Panel.” Please don’t put yourself or your professor in that position. *When in doubt, ask your professor or a coach before seeking any help from another source.*
Iterative refinement
home
home

kate

escape.rkt

;; A Pittsburgh road is ...

map.pdf

hugo

snacks

lotto.zip

4 8 15 16 23 42
A file system is a tree structure where each directory can have an arbitrary number of children. This is a lot like the "more realistic rumor mill" we saw last class! These are both kinds of \textit{n-ary trees}. 
Iterative refinement

Start simple
Iterative refinement

Start simple

When the problem needs more:

- Expand your data definition
- Adjust your functions
Iterative refinement

;; find : Directory String → Boolean
Iterative refinement

;; find : Directory String -> Boolean

A Directory has:

  a name
  files and other directories

A File has:

  a name
Initial data definitions

;; A Directory is
;;  (make-dir String LOFD)
(define-struct dir [name content])

;; A File is a String

;; A FileOrDirectory is either:
;;  - File
;;  - Directory

;; A LOFD is either:
;;  - '()
;;  - (cons FileOrDirectory LOFD)
Initial data definitions

;;; A Directory is
;;; (make-dir String LOFD)
(define-struct dir [name content])

;;; A File is a String

;;; A FileOrDirectory is either:
;;; - File
;;; - Directory

;;; A LOFD is either:
;;; - '()  // Reference × 4
;;; - (cons FileOrDirectory LOFD) // Self-reference
Initial implementation

;;; find : Directory String -> Boolean
(define (find dir n)
  (... (dir-name dir) ...
    (find-for-lofd (dir-content dir) n) ...))

;;; find-for-file : File String -> Boolean
(define (find-for-file file n)
  (... file ...))

;;; find-for-fod : FileOrDirectory String -> Boolean
(define (find-for-fod fod n)
  (cond [(string? fod) (... (find-for-file fod n) ...)]
        [(dir? fod) (... (find fod n) ...)]))

;;; find-for-lofd : LOFD String -> Boolean
(define (find-for-lofd lofd n)
  (cond [(empty? lofd) ...]
        [(cons? lofd)
          (... (find-for-fod (first lofd) n) ...
               (find-for-lofd (rest lofd) n) ...)])
New problem

Compute how much space is used on the disk:

;; du : Directory -> Number
Revised data definitions

;;; A Directory is
;;; (make-dir String LOFD)
(define-struct dir [name content])

;;; A File is
;;; (make-file String Number)
(define-struct file [name size])

;;; A FileOrDirectory is either:
;;; – File
;;; – Directory

;;; A LOFD is either:
;;; – '()
;;; – (cons FileOrDirectory LOFD)
Revised implementation

;; find : Directory String → Boolean
(define (find dir n)
  (... (dir-name dir) ...
      (find-for-lofd (dir-content dir) n) ...))

;; find-for-file : File String → Boolean
(define (find-for-file file n)
  (... (file-name file) ...
       (file-size file) ...))

;; find-for-fod : FileOrDirectory String → Boolean
(define (find-for-fod fod n)
  (cond [(file? fod) (... (find-for-file fod n) ...)]
        [(dir? fod) (... (find fod n) ...)]))

;; find-for-lofd : LOFD String → Boolean
(define (find-for-lofd lofd n)
  (cond [(empty? lofd) ...]
        [(cons? lofd)
         (... (find-for-fod (first lofd) n) ...
              (find-for-lofd (rest lofd) n) ...)])
Acknowledgments

This lecture incorporates material from:

- Matthias Felleisen
- Robert Bruce Findler
- Matthew Flatt
- Shriram Krishnamurthi
- Marc Smith