



C:



kate



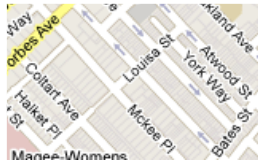
hugo



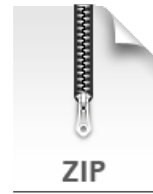
escape.rkt



map.pdf



snacks



lotto.zip

4 8 15 16 23 42

;; A Pittsburgh road is
...

Iterative Refinement

- Start simple
- When the problem needs more:
 - expand your data definition
 - adjust your functions

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  
; - string
```

```
; A file-or-directory is either  
; - file  
; - directory
```

```
; A LOFD is either  
; - empty  
; - (cons file-or-directory LOFD)
```

Initial Data Definitions

```
; A directory is
; - (make-dir string LOFD)
(define-struct dir (name content))

; A file is
; - string

; A file-or-directory is either
; - file
; - directory

; A LOFD is either
; - empty
; - (cons file-or-directory LOFD)
```

Initial Implementation

```
; find : directory string -> bool
(define (find dir n)
  ... (dir-name dir)
  ... (lofd-find (dir-content dir) n) ...)

; file-find : file string -> bool
(define (file-find file n)
  ... file ...)

; fod-find : file-or-directory string -> bool
(define (fod-find fod n)
  (cond
    [(string? fod) ... (file-find fod n) ...]
    [(dir? fod) ... (find fod n) ...]))

; lofd-find : lofd string -> bool
(define (lofd-find lofd n)
  (cond
    [(empty? lofd) ...]
    [(cons? lofd)
     ... (fod-find (first lofd) n)
     ... (lofd-find (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 file-or-directory is either  
; - file  
; - directory
```

```
; A LOFD is either  
; - empty  
; - (cons file-or-directory LOFD)
```


Revised Implementation

```
; find : directory string -> bool
(define (find dir n)
  ... (dir-name dir)
  ... (lofd-find (dir-content dir) n) ...)
```

```
; file-find : file string -> bool
(define (file-find file n)
  ... (file-name file)
  ... (file-size file) ...)
```

```
; fod-find : file-or-directory string -> bool
(define (fod-find fod n)
  (cond
    [(file? fod) ... (file-find fod n) ...]
    [(dir? fod) ... (find fod n) ...]))
```

```
; lofd-find : lofd string -> bool
(define (lofd-find lofd n)
  (cond
    [(empty? lofd) ...]
    [(cons? lofd)
     ... (fod-find (first lofd) n)
     ... (lofd-find (rest lofd) n) ...]))
```