```
;; Simple file systems, model 2:

;; File is String
;; interp. the string represents the file's name

(define-struct dir (n c))
;; Directory (Dir) is (make-dir String ListOfFileOrDir)
;; interp. n is the name of the directory, c is its contents

;; ListOfFileOrDir is one of:
;;  - empty
;;  - (cons File ListOfFileOrDir)
;;  - (cons Dir ListOfFileOrDir)
;; interp. a list of potentially intermixed files and directories

;
 PRE-LAB EXERCISES: There are 3 pre-lab exercises below here.


;
 (1) Define at least 2 examples of File and 4 examples of Dir here.
     As usual, start with a base case, make sure to have examples
     at least 2 deep and 2 wide.


(define F0 "Abc")
(define F1 "Xyz")
(define F2 "Ijk")

(define D0 (make-dir "Foo" empty))
(define D1 (make-dir "Bar" (list F0)))
(define D2 (make-dir "Baz" (list F0 F1)))
(define D3 (make-dir "Qux" (list D2 F2)))
(define D4 (make-dir "Quux" (list "Uvw" D3 "Def")))

;
 (2) Write the template for Dir here. Be sure to include templates for
     all types involved in mutual reference cycles. Then encapsulate
     the templates with a local expression.


(define (fn-for-dir d)
  (local [(define (fn-for-lofod lfd)
            (cond [(empty? lfd) (...)]
                  [(string? (first lfd))
                   (... (fn-for-file (first lfd))
                        (fn-for-lofod (rest lfd)))]
                  [else
                   (... (fn-for-dir (first lfd))
                        (fn-for-lofod (rest lfd)))]))

          (define (fn-for-file f)
            (... f))]
    (... (dir-n d)
         (fn-for-lofod (dir-c d)))))
```

```
;; Simple file systems, model 3:

(define-struct file (n s c))
;; File is (make-file String Natural String)
;; interp. n is file name, s is size, c represents the contents

(define-struct dir (n dirs files))
;; Directory (Dir) is (make-dir String ListOfDir ListOfFile)
;; interp. n is the name of the directory
;;         dirs is the sub-directories
;;         files is the files in the directory

;; ListOfDir is one of:
;;  - empty
;;  - (cons Dir ListOfDir)

;; ListOfFile is one of:
;;  - empty
;;  - (cons File ListOfFile)

(define F0 (make-file "ABC" 128 "asdf"))
(define F1 (make-file "DEF" 256 "asdfghjk"))
(define F2 (make-file "GHI" 512 "asdfghjkl;qwerty"))

(define D0 (make-dir "Foo" empty empty))
(define D1 (make-dir "Bar" empty (list F0 F1 F2)))
(define D2 (make-dir "Baz" (list D1) empty))
(define D3 (make-dir "Qux" (list D1 D2) (list F2)))
(define D4 (make-dir "Quux" (list D2 D3) (list F0 F1)))
(define D5 (make-dir "Corge" (list D1 D2 D3 D4) (list F0 F1 F2)))

;
 (2) Write the template for Dir here. Be sure to include templates for
     all types involved in mutual reference cycles. Then encapsulate
     the templates with a local expression.


(define (fn-for-dir d)
  (local [(define (fn-for-lod lod)
            (cond [(empty? lod) ...]
                  [else
                   (... (fn-for-dir (first lod))
                        (fn-for-lod (rest lod)))]))

          (define (fn-for-lof lof)
            (cond [(empty? lof) ...]
                  [else
                   (... (fn-for-file (first lof))
                        (fn-for-lof (rest lof)))]))

          (define (fn-for-file f)
            (... (file-n f)
                 (file-s f)
                 (file-c f)))]
    (... (dir-n d)
         (fn-for-lod (dir-dirs d))
         (fn-for-lof (dir-files d)))))
```