Administrative Notes

Homework

- Handed out today, due Friday, 2/15/2002
- Anyone without a homework partner, please contact me

First Exam

- Covers Sectons 1-12 of the book
  - → Not family trees
  - → Includes natural numbers (lab lecture + Friday)
- Covers class lectures, lab lectures, homework through Friday

In-family? (version 1)

```
;; in-family?: ftn  symbol → boolean
;; Purpose: determine if the symbol is in the ftn
;;              return true if found and false otherwise
(define  (in-family? a-ftn  kin)
   (cond
      [(symbol?   a-ftn)   (symbol=?  a-ftn  kin)]
      [(ftn?         a-ftn)
       (or
          (symbol=?   (ftn-name  a-ftn)  kin)
          (in-family?  (ftn-mother a-ftn)  kin )
          (in-family?  (ftn-father   a-ftn)  kin )
          ) ]   ) )
```

The or combines the results of the expressions that are its arguments. It returns true if any argument evaluates to true

In-family? (version 2)

```
;; in-family?: ftn  symbol → boolean
;; Purpose: determine if the symbol is in the ftn
;;              return true if found and false otherwise
(define  (in-family? a-ftn  kin)
   (cond
      [(symbol?   a-ftn)   (symbol=?  a-ftn  kin)]
      [(ftn?         a-ftn)
       (cond
          [(symbol=?   (ftn-name  a-ftn)  kin)    true]
          [(in-family?  (ftn-mother a-ftn)  kin )  true]
          [(in-family?  (ftn-father   a-ftn)  kin )  truie]
          [else                                    false]
          ) ]   ) )
```


In-family? (version 3)

- On more complex **ftns**

```
;;  in-family? :  ftn  symbol -> boolean
;; Purpose:  returns true if symbol is in the family tree
(define  (in-family? a-ftn name )
    (cond
      [(empty?  a-ftn)   false  ]
      [(ftn?       a-ftn)
            (or
           (symbol=? (ftn-name  a-ftn)  name )
           (in-family?  (ftn-mother   a-ftn)  name)
           (in-family?  (ftn-father     a-ftn)  name) )
         ]
   ) )
```

Change is in this case.
(was symbol=?**)**

Count-female ancestors (version 1)

```
;; count-female-ancestors: ftn -> num
;; Purpose: consumes a ftn and returns the number of
;;    female ancestors
(define (count-female-ancestors a-ftn)
   (cond
      [(empty? a-ftn)     0]
      [else
         (cond
          [(empty?  (ftn-mother a-ftn)
            (count-female-ancestors (ftn-father a-ftn))]
          [else (+ 1
                (count-female-ancestors (ftn-mother a-ftn))
                (count-female-ancestors (ftn-father   a-ftn)) )]
           )]
   ))
```


Count-female ancestors (version 1)

```
;; count-female-ancestors: ftn -> num
;; Purpose: consumes a ftn and returns the number of
;;    female ancestors
(define (count-female-ancestors a-ftn)
   (cond
      [(empty? a-ftn)     0]
      [else
         (cond
          [(empty?  (ftn-mother a-ftn)
            (count-female-ancestors (ftn-father a-ftn))]
          [else (+ 1
                (count-female-ancestors (ftn-mother a-ftn))
                (count-female-ancestors (ftn-father   a-ftn)) )]
           )]
   ))
```

This case is a mess!

Count-female ancestors (version 2)

```
;; count-mother: ftn -> num
;; Pupose: how many ancestors to add for current mother
(define (count-mother a-ftn)
   (cond
      [(empty?    a-ftn)    0]
      [else                 1]
    ))

;; count-female-ancestors: ftn -> num
;; Purpose: consumes a ftn and returns the number of female ancest
(define (count-female-ancestors a-ftn)
   (cond
      [(empty? a-ftn)    0]
      [else
         (+ (count-mother (ftn-mother a-ftn)
           (count-female-ancestors (ftn-mother a-ftn))
           (count-female-ancestors (ftn-father   a-ftn)) )]
   ))
```


Count-blue-eyed-female-ancestors

```
;; count-if-blue-eyes:  ftn -> num
;; Purpose:  returns 1 if the ftn has blue eyes, 0 otherwise
(define (count-if-blue-eyes a-ftn)
   (cond
      [(symbol=?  'blue  (ftn-eyes a-ftn))   1]
      [else                                  0]
    ))
```

Only change is in the helper functions

```
;; count-mother: ftn -> num
;; Pupose: determine how many ancestors to add for current mother
(define (count-mother a-ftn)
   (cond
      [(emtpy?     a-ftn)     0]
      [else                   (count-if-blue-eyes a-ftn)]
    ))
```