CS 115, Fall 2002
Extra-credit Homework Assignment
Due: 11 November 2002

## Contents

Your task is a series of three hand evaluations. You may show all of the steps, or only show the steps right before each call to a one of the functions given (*i.e.*, just before each recursive call or call to a helper function).
As you do the hand evaluations, be sure to use cut and paste. Particularly useful are the alt-shift-arrow key sequences in DrScheme. Use alt-shift-right and alt-shift-left to select complete subexpressions and the copy and paste them as you go from step to step.
Do not attempt this homework on pencil and paper, especially if you plan to show all of the steps.
The grade for this assignment replaces your lowest homework grade, but the highest grade given will be a check.
The sample solutions contain every step.

## 1 Path-to-blue-eyes

A *family-tree* is either:

- 'unknown
- (*make-ft name eye-color mom dad*)
  where *name* and *eye-color* are symbols,
  and *mom* and *dad* are *family-trees*.

```
(define-struct ft (name eye-color mom dad))


;; path-to-blue-eyes : family-tree  list-of-symbols or #f
;; finds the path to a blue eyed ancestor
(define (path-to-blue-eyes ft)
  (cond
    [(eq? ft 'unknown) #f]
    [else
     (if (eq? (ft-eye-color ft) 'blue)
         '()
         (let ([mom-path (path-to-blue-eyes (ft-mom ft))]
               [dad-path (path-to-blue-eyes (ft-dad ft))])
           (cond
             [(and mom-path dad-path) (cons 'mom mom-path)]
             [(and mom-path (not dad-path)) (cons 'mom mom-path)]
             [(and dad-path (not mom-path)) (cons 'dad dad-path)]
             [else #f])))]))

(define tutu (make-ft 'emily 'brown 'unknown 'unknown))
(define opa (make-ft 'bruce 'blue 'unknown 'unknown))
(define mom (make-ft 'alice 'green tutu opa))
(define dad (make-ft 'bill 'brown 'unknown 'unknown))
(define me (make-ft 'robby 'hazel mom dad))
```

Hand evaluate:

```
(path-to-blue-eyes me)
```

**Solution**

```
(path-to-blue-eyes
  (make-ft
    'robby
    'hazel
    (make-ft
      'alice
      'green
      (make-ft 'emily 'brown 'unknown 'unknown)
      (make-ft 'bruce 'blue 'unknown 'unknown))
    (make-ft 'bill 'brown 'unknown 'unknown)))
```

```
(cond
 ((eq?
   (make-ft
      'robby
      'hazel
      (make-ft
         'alice
         'green
         (make-ft 'emily 'brown 'unknown 'unknown)
         (make-ft 'bruce 'blue 'unknown 'unknown))
      (make-ft 'bill 'brown 'unknown 'unknown))
   'unknown)
  #f)
 (else
  (if (eq?
       (ft-eye-color
          (make-ft
             'robby
             'hazel
             (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
             (make-ft 'bill 'brown 'unknown 'unknown)))
       'blue)
      '()
      (let ((mom-path
              (path-to-blue-eyes
                 (ft-mom
                    (make-ft
                       'robby
                       'hazel
                       (make-ft
                          'alice
                          'green
                          (make-ft 'emily 'brown 'unknown 'unknown)
                          (make-ft 'bruce 'blue 'unknown 'unknown))
                       (make-ft 'bill 'brown 'unknown 'unknown)))))
            (dad-path
              (path-to-blue-eyes
                 (ft-dad
                    (make-ft
                       'robby
                       'hazel
                       (make-ft
                          'alice
                          'green
                          (make-ft 'emily 'brown 'unknown 'unknown)
                          (make-ft 'bruce 'blue 'unknown 'unknown))
                       (make-ft 'bill 'brown 'unknown 'unknown))))))
        (cond
         ((and mom-path dad-path) (cons 'mom mom-path))
((and mom-path (not dad-path)) (cons 'mom mom-path))
((and dad-path (not mom-path)) (cons 'dad dad-path))
(else #f))))))
```

```
(cond
 (#f #f)
 (else
  (if (eq?
       (ft-eye-color
          (make-ft
             'robby
             'hazel
             (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
             (make-ft 'bill 'brown 'unknown 'unknown)))
       'blue)
      '()
      (let ((mom-path
               (path-to-blue-eyes
                  (ft-mom
                     (make-ft
                        'robby
                        'hazel
                        (make-ft
                           'alice
                           'green
                           (make-ft 'emily 'brown 'unknown 'unknown)
                           (make-ft 'bruce 'blue 'unknown 'unknown))
                        (make-ft 'bill 'brown 'unknown 'unknown)))))
            (dad-path
               (path-to-blue-eyes
                  (ft-dad
                     (make-ft
                        'robby
                        'hazel
                        (make-ft
                           'alice
                           'green
                           (make-ft 'emily 'brown 'unknown 'unknown)
                           (make-ft 'bruce 'blue 'unknown 'unknown))
                        (make-ft 'bill 'brown 'unknown 'unknown))))))
        (cond
         ((and mom-path dad-path) (cons 'mom mom-path))
         ((and mom-path (not dad-path)) (cons 'mom mom-path))
         ((and dad-path (not mom-path)) (cons 'dad dad-path))
         (else #f))))))
```

```
(cond
 (else
  (if (eq?
       (ft-eye-color
         (make-ft
           'robby
           'hazel
           (make-ft
             'alice
             'green
             (make-ft 'emily 'brown 'unknown 'unknown)
             (make-ft 'bruce 'blue 'unknown 'unknown))
           (make-ft 'bill 'brown 'unknown 'unknown)))
       'blue)
      '()
      (let ((mom-path
              (path-to-blue-eyes
                (ft-mom
                  (make-ft
                    'robby
                    'hazel
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))
                    (make-ft 'bill 'brown 'unknown 'unknown)))))
            (dad-path
              (path-to-blue-eyes
                (ft-dad
                  (make-ft
                    'robby
                    'hazel
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))
                    (make-ft 'bill 'brown 'unknown 'unknown))))))
        (cond
         ((and mom-path dad-path) (cons 'mom mom-path))
         ((and mom-path (not dad-path)) (cons 'mom mom-path))
         ((and dad-path (not mom-path)) (cons 'dad dad-path))
         (else #f))))))
```

```
(if (eq?
     (ft-eye-color
       (make-ft
         'robby
         'hazel
         (make-ft
           'alice
           'green
           (make-ft 'emily 'brown 'unknown 'unknown)
           (make-ft 'bruce 'blue 'unknown 'unknown))
         (make-ft 'bill 'brown 'unknown 'unknown)))
     'blue)
    '()
    (let ((mom-path
            (path-to-blue-eyes
              (ft-mom
                (make-ft
                  'robby
                  'hazel
                  (make-ft
                    'alice
                    'green
                    (make-ft 'emily 'brown 'unknown 'unknown)
                    (make-ft 'bruce 'blue 'unknown 'unknown))
                  (make-ft 'bill 'brown 'unknown 'unknown)))))
          (dad-path
            (path-to-blue-eyes
              (ft-dad
                (make-ft
                  'robby
                  'hazel
                  (make-ft
                    'alice
                    'green
                    (make-ft 'emily 'brown 'unknown 'unknown)
                    (make-ft 'bruce 'blue 'unknown 'unknown))
                  (make-ft 'bill 'brown 'unknown 'unknown))))))
      (cond
       ((and mom-path dad-path) (cons 'mom mom-path))
       ((and mom-path (not dad-path)) (cons 'mom mom-path))
       ((and dad-path (not mom-path)) (cons 'dad dad-path))
       (else #f))))
```

```
(if (eq? 'hazel 'blue)
  '()
  (let ((mom-path
          (path-to-blue-eyes
            (ft-mom
              (make-ft
                'robby
                'hazel
                (make-ft
                  'alice
                  'green
                  (make-ft 'emily 'brown 'unknown 'unknown)
                  (make-ft 'bruce 'blue 'unknown 'unknown))
                (make-ft 'bill 'brown 'unknown 'unknown)))))
        (dad-path
          (path-to-blue-eyes
            (ft-dad
              (make-ft
                'robby
                'hazel
                (make-ft
                  'alice
                  'green
                  (make-ft 'emily 'brown 'unknown 'unknown)
                  (make-ft 'bruce 'blue 'unknown 'unknown))
                (make-ft 'bill 'brown 'unknown 'unknown))))))
    (cond
     ((and mom-path dad-path) (cons 'mom mom-path))
     ((and mom-path (not dad-path)) (cons 'mom mom-path))
     ((and dad-path (not mom-path)) (cons 'dad dad-path))
     (else #f))))
```

```
(if #f
  '()
  (let ((mom-path
         (path-to-blue-eyes
           (ft-mom
             (make-ft
               'robby
               'hazel
               (make-ft
                 'alice
                 'green
                 (make-ft 'emily 'brown 'unknown 'unknown)
                 (make-ft 'bruce 'blue 'unknown 'unknown))
               (make-ft 'bill 'brown 'unknown 'unknown)))))
        (dad-path
          (path-to-blue-eyes
            (ft-dad
              (make-ft
                'robby
                'hazel
                (make-ft
                  'alice
                  'green
                  (make-ft 'emily 'brown 'unknown 'unknown)
                  (make-ft 'bruce 'blue 'unknown 'unknown))
                (make-ft 'bill 'brown 'unknown 'unknown))))))
    (cond
     ((and mom-path dad-path) (cons 'mom mom-path))
     ((and mom-path (not dad-path)) (cons 'mom mom-path))
     ((and dad-path (not mom-path)) (cons 'dad dad-path))
     (else #f))))
```

```
(let ((mom-path
         (path-to-blue-eyes
           (ft-mom
             (make-ft
               'robby
               'hazel
               (make-ft
                 'alice
                 'green
                 (make-ft 'emily 'brown 'unknown 'unknown)
                 (make-ft 'bruce 'blue 'unknown 'unknown))
               (make-ft 'bill 'brown 'unknown 'unknown)))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (path-to-blue-eyes
          (make-ft
            'alice
            'green
            (make-ft 'emily 'brown 'unknown 'unknown)
            (make-ft 'bruce 'blue 'unknown 'unknown))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
       (cond
        ((eq?
          (make-ft
            'alice
            'green
            (make-ft 'emily 'brown 'unknown 'unknown)
            (make-ft 'bruce 'blue 'unknown 'unknown))
          'unknown)
         #f)
        (else
         (if (eq?
              (ft-eye-color
                (make-ft
                  'alice
                  'green
                  (make-ft 'emily 'brown 'unknown 'unknown)
                  (make-ft 'bruce 'blue 'unknown 'unknown)))
              'blue)
             '()
             (let ((mom-path
                     (path-to-blue-eyes
                       (ft-mom
                         (make-ft
                           'alice
                           'green
                           (make-ft 'emily 'brown 'unknown 'unknown)
                           (make-ft 'bruce 'blue 'unknown 'unknown)))))
                   (dad-path
                     (path-to-blue-eyes
                       (ft-dad
                         (make-ft
                           'alice
                           'green
                           (make-ft 'emily 'brown 'unknown 'unknown)
                           (make-ft 'bruce 'blue 'unknown 'unknown))))))
               (cond
                 ((and mom-path dad-path) (cons 'mom mom-path))
                 ((and mom-path (not dad-path)) (cons 'mom mom-path))
                 ((and dad-path (not mom-path)) (cons 'dad dad-path))
                 (else #f)))))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
(cond
 ((and mom-path dad-path) (cons 'mom mom-path))
 ((and mom-path (not dad-path)) (cons 'mom mom-path))
 ((and dad-path (not mom-path)) (cons 'dad dad-path))
 (else #f)))
```

```
(let ((mom-path
        (cond
         (#f #f)
         (else
          (if (eq?
                (ft-eye-color
                  (make-ft
                    'alice
                    'green
                    (make-ft 'emily 'brown 'unknown 'unknown)
                    (make-ft 'bruce 'blue 'unknown 'unknown)))
                'blue)
               '()
               (let ((mom-path
                       (path-to-blue-eyes
                         (ft-mom
                           (make-ft
                             'alice
                             'green
                             (make-ft 'emily 'brown 'unknown 'unknown)
                             (make-ft 'bruce 'blue 'unknown 'unknown)))))
                     (dad-path
                       (path-to-blue-eyes
                         (ft-dad
                           (make-ft
                             'alice
                             'green
                             (make-ft 'emily 'brown 'unknown 'unknown)
                             (make-ft 'bruce 'blue 'unknown 'unknown))))))
                 (cond
                  ((and mom-path dad-path) (cons 'mom mom-path))
                  ((and mom-path (not dad-path)) (cons 'mom mom-path))
                  ((and dad-path (not mom-path)) (cons 'dad dad-path))
                  (else #f)))))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
      (cond
       (else
        (if (eq?
             (ft-eye-color
               (make-ft
                 'alice
                 'green
                 (make-ft 'emily 'brown 'unknown 'unknown)
                 (make-ft 'bruce 'blue 'unknown 'unknown)))
             'blue)
            '()
            (let ((mom-path
                     (path-to-blue-eyes
                       (ft-mom
                         (make-ft
                           'alice
                           'green
                           (make-ft 'emily 'brown 'unknown 'unknown)
                           (make-ft 'bruce 'blue 'unknown 'unknown)))))
                  (dad-path
                    (path-to-blue-eyes
                      (ft-dad
                        (make-ft
                          'alice
                          'green
                          (make-ft 'emily 'brown 'unknown 'unknown)
                          (make-ft 'bruce 'blue 'unknown 'unknown))))))
              (cond
               ((and mom-path dad-path) (cons 'mom mom-path))
               ((and mom-path (not dad-path)) (cons 'mom mom-path))
               ((and dad-path (not mom-path)) (cons 'dad dad-path))
               (else #f)))))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (if (eq?
            (ft-eye-color
               (make-ft
                  ’alice
                  ’green
                  (make-ft ’emily ’brown ’unknown ’unknown)
                  (make-ft ’bruce ’blue ’unknown ’unknown)))
            ’blue)
           ’()
           (let ((mom-path
                   (path-to-blue-eyes
                      (ft-mom
                        (make-ft
                           ’alice
                           ’green
                           (make-ft ’emily ’brown ’unknown ’unknown)
                           (make-ft ’bruce ’blue ’unknown ’unknown)))))
                 (dad-path
                    (path-to-blue-eyes
                      (ft-dad
                        (make-ft
                           ’alice
                           ’green
                           (make-ft ’emily ’brown ’unknown ’unknown)
                           (make-ft ’bruce ’blue ’unknown ’unknown))))))
              (cond
               ((and mom-path dad-path) (cons ’mom mom-path))
               ((and mom-path (not dad-path)) (cons ’mom mom-path))
               ((and dad-path (not mom-path)) (cons ’dad dad-path))
               (else #f)))))
      (dad-path
         (path-to-blue-eyes
           (ft-dad
              (make-ft
                ’robby
                ’hazel
                (make-ft
                   ’alice
                   ’green
                   (make-ft ’emily ’brown ’unknown ’unknown)
                   (make-ft ’bruce ’blue ’unknown ’unknown))
                (make-ft ’bill ’brown ’unknown ’unknown))))))
   (cond
    ((and mom-path dad-path) (cons ’mom mom-path))
    ((and mom-path (not dad-path)) (cons ’mom mom-path))
    ((and dad-path (not mom-path)) (cons ’dad dad-path))
    (else #f)))
```

```
(let ((mom-path
      (if (eq? ’green ’blue)
        ’()
        (let ((mom-path
              (path-to-blue-eyes
                (ft-mom
                  (make-ft
                    ’alice
                    ’green
                    (make-ft ’emily ’brown ’unknown ’unknown)
                    (make-ft ’bruce ’blue ’unknown ’unknown)))))
             (dad-path
               (path-to-blue-eyes
                 (ft-dad
                   (make-ft
                     ’alice
                     ’green
                     (make-ft ’emily ’brown ’unknown ’unknown)
                     (make-ft ’bruce ’blue ’unknown ’unknown))))))
           (cond
            ((and mom-path dad-path) (cons ’mom mom-path))
            ((and mom-path (not dad-path)) (cons ’mom mom-path))
            ((and dad-path (not mom-path)) (cons ’dad dad-path))
            (else #f)))))
     (dad-path
       (path-to-blue-eyes
         (ft-dad
           (make-ft
             ’robby
             ’hazel
             (make-ft
               ’alice
               ’green
               (make-ft ’emily ’brown ’unknown ’unknown)
               (make-ft ’bruce ’blue ’unknown ’unknown))
             (make-ft ’bill ’brown ’unknown ’unknown))))))
  (cond
   ((and mom-path dad-path) (cons ’mom mom-path))
   ((and mom-path (not dad-path)) (cons ’mom mom-path))
   ((and dad-path (not mom-path)) (cons ’dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (path-to-blue-eyes
                  (ft-mom
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown)))))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
          (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (path-to-blue-eyes (make-ft 'emily 'brown 'unknown 'unknown)))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
            (cond
             ((and mom-path dad-path) (cons 'mom mom-path))
             ((and mom-path (not dad-path)) (cons 'mom mom-path))
             ((and dad-path (not mom-path)) (cons 'dad dad-path))
             (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (cond
                 ((eq? (make-ft 'emily 'brown 'unknown 'unknown) 'unknown) #f)
                 (else
                  (if (eq?
                       (ft-eye-color (make-ft 'emily 'brown 'unknown 'unknown))
                       'blue)
                     '()
                     (let ((mom-path
                             (path-to-blue-eyes
                               (ft-mom
                                 (make-ft 'emily 'brown 'unknown 'unknown))))
                           (dad-path
                             (path-to-blue-eyes
                               (ft-dad
                                 (make-ft 'emily 'brown 'unknown 'unknown)))))
                        (cond
                         ((and mom-path dad-path) (cons 'mom mom-path))
                         ((and mom-path (not dad-path)) (cons 'mom mom-path))
                         ((and dad-path (not mom-path)) (cons 'dad dad-path))
                         (else #f)))))))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
           (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
      (let ((mom-path
            (cond
             (#f #f)
             (else
              (if (eq?
                   (ft-eye-color (make-ft 'emily 'brown 'unknown 'unknown))
                   'blue)
                  '()
                  (let ((mom-path
                         (path-to-blue-eyes
                           (ft-mom
                             (make-ft 'emily 'brown 'unknown 'unknown))))
                        (dad-path
                         (path-to-blue-eyes
                           (ft-dad
                             (make-ft 'emily 'brown 'unknown 'unknown)))))
                    (cond
                     ((and mom-path dad-path) (cons 'mom mom-path))
                     ((and mom-path (not dad-path)) (cons 'mom mom-path))
                     ((and dad-path (not mom-path)) (cons 'dad dad-path))
                     (else #f)))))))
           (dad-path
             (path-to-blue-eyes
               (ft-dad
                 (make-ft
                   'alice
                   'green
                   (make-ft 'emily 'brown 'unknown 'unknown)
                   (make-ft 'bruce 'blue 'unknown 'unknown))))))
        (cond
         ((and mom-path dad-path) (cons 'mom mom-path))
         ((and mom-path (not dad-path)) (cons 'mom mom-path))
         ((and dad-path (not mom-path)) (cons 'dad dad-path))
         (else #f))))
     (dad-path
       (path-to-blue-eyes
         (ft-dad
           (make-ft
             'robby
             'hazel
             (make-ft
               'alice
               'green
               (make-ft 'emily 'brown 'unknown 'unknown)
               (make-ft 'bruce 'blue 'unknown 'unknown))
             (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
      (let ((mom-path
             (cond
              (else
               (if (eq?
                    (ft-eye-color (make-ft ’emily ’brown ’unknown ’unknown))
                    ’blue)
                   ’()
                   (let ((mom-path
                          (path-to-blue-eyes
                            (ft-mom
                              (make-ft ’emily ’brown ’unknown ’unknown))))
                        (dad-path
                          (path-to-blue-eyes
                            (ft-dad
                              (make-ft ’emily ’brown ’unknown ’unknown)))))
                     (cond
                      ((and mom-path dad-path) (cons ’mom mom-path))
                      ((and mom-path (not dad-path)) (cons ’mom mom-path))
                      ((and dad-path (not mom-path)) (cons ’dad dad-path))
                      (else #f)))))))
           (dad-path
             (path-to-blue-eyes
               (ft-dad
                 (make-ft
                   ’alice
                   ’green
                   (make-ft ’emily ’brown ’unknown ’unknown)
                   (make-ft ’bruce ’blue ’unknown ’unknown))))))
        (cond
         ((and mom-path dad-path) (cons ’mom mom-path))
         ((and mom-path (not dad-path)) (cons ’mom mom-path))
         ((and dad-path (not mom-path)) (cons ’dad dad-path))
         (else #f))))
     (dad-path
       (path-to-blue-eyes
         (ft-dad
           (make-ft
             ’robby
             ’hazel
             (make-ft
               ’alice
               ’green
               (make-ft ’emily ’brown ’unknown ’unknown)
               (make-ft ’bruce ’blue ’unknown ’unknown))
             (make-ft ’bill ’brown ’unknown ’unknown))))))
  (cond
   ((and mom-path dad-path) (cons ’mom mom-path))
   ((and mom-path (not dad-path)) (cons ’mom mom-path))
   ((and dad-path (not mom-path)) (cons ’dad dad-path))
   (else #f)))
```

```
(let ((mom-path
       (let ((mom-path
              (if (eq?
                    (ft-eye-color (make-ft 'emily 'brown 'unknown 'unknown))
                    'blue)
                   '()
                   (let ((mom-path
                          (path-to-blue-eyes
                            (ft-mom
                              (make-ft 'emily 'brown 'unknown 'unknown))))
                         (dad-path
                          (path-to-blue-eyes
                            (ft-dad
                              (make-ft 'emily 'brown 'unknown 'unknown)))))
                     (cond
                      ((and mom-path dad-path) (cons 'mom mom-path))
                      ((and mom-path (not dad-path)) (cons 'mom mom-path))
                      ((and dad-path (not mom-path)) (cons 'dad dad-path))
                      (else #f)))))
             (dad-path
              (path-to-blue-eyes
                (ft-dad
                  (make-ft
                    'alice
                    'green
                    (make-ft 'emily 'brown 'unknown 'unknown)
                    (make-ft 'bruce 'blue 'unknown 'unknown))))))
         (cond
          ((and mom-path dad-path) (cons 'mom mom-path))
          ((and mom-path (not dad-path)) (cons 'mom mom-path))
          ((and dad-path (not mom-path)) (cons 'dad dad-path))
          (else #f))))
      (dad-path
       (path-to-blue-eyes
         (ft-dad
           (make-ft
             'robby
             'hazel
             (make-ft
               'alice
               'green
               (make-ft 'emily 'brown 'unknown 'unknown)
               (make-ft 'bruce 'blue 'unknown 'unknown))
             (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (if (eq? ’brown ’blue)
                   ’()
                   (let ((mom-path
                           (path-to-blue-eyes
                             (ft-mom
                                (make-ft ’emily ’brown ’unknown ’unknown))))
                         (dad-path
                           (path-to-blue-eyes
                             (ft-dad
                                (make-ft ’emily ’brown ’unknown ’unknown)))))
                      (cond
                       ((and mom-path dad-path) (cons ’mom mom-path))
                       ((and mom-path (not dad-path)) (cons ’mom mom-path))
                       ((and dad-path (not mom-path)) (cons ’dad dad-path))
                       (else #f)))))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                     (make-ft
                       ’alice
                       ’green
                       (make-ft ’emily ’brown ’unknown ’unknown)
                       (make-ft ’bruce ’blue ’unknown ’unknown))))))
           (cond
            ((and mom-path dad-path) (cons ’mom mom-path))
            ((and mom-path (not dad-path)) (cons ’mom mom-path))
            ((and dad-path (not mom-path)) (cons ’dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
             (make-ft
               ’robby
               ’hazel
               (make-ft
                  ’alice
                  ’green
                  (make-ft ’emily ’brown ’unknown ’unknown)
                  (make-ft ’bruce ’blue ’unknown ’unknown))
               (make-ft ’bill ’brown ’unknown ’unknown))))))
   (cond
    ((and mom-path dad-path) (cons ’mom mom-path))
    ((and mom-path (not dad-path)) (cons ’mom mom-path))
    ((and dad-path (not mom-path)) (cons ’dad dad-path))
    (else #f)))
```

```
(let ((mom-path
       (let ((mom-path
              (if #f
                  '()
                  (let ((mom-path
                         (path-to-blue-eyes
                           (ft-mom
                             (make-ft 'emily 'brown 'unknown 'unknown))))
                        (dad-path
                          (path-to-blue-eyes
                            (ft-dad
                              (make-ft 'emily 'brown 'unknown 'unknown)))))
                    (cond
                      ((and mom-path dad-path) (cons 'mom mom-path))
                      ((and mom-path (not dad-path)) (cons 'mom mom-path))
                      ((and dad-path (not mom-path)) (cons 'dad dad-path))
                      (else #f)))))
             (dad-path
               (path-to-blue-eyes
                 (ft-dad
                   (make-ft
                     'alice
                     'green
                     (make-ft 'emily 'brown 'unknown 'unknown)
                     (make-ft 'bruce 'blue 'unknown 'unknown))))))
         (cond
           ((and mom-path dad-path) (cons 'mom mom-path))
           ((and mom-path (not dad-path)) (cons 'mom mom-path))
           ((and dad-path (not mom-path)) (cons 'dad dad-path))
           (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
       (let ((mom-path
              (let ((mom-path
                     (path-to-blue-eyes
                       (ft-mom (make-ft 'emily 'brown 'unknown 'unknown))))
                    (dad-path
                      (path-to-blue-eyes
                        (ft-dad (make-ft 'emily 'brown 'unknown 'unknown)))))
                (cond
                 ((and mom-path dad-path) (cons 'mom mom-path))
                 ((and mom-path (not dad-path)) (cons 'mom mom-path))
                 ((and dad-path (not mom-path)) (cons 'dad dad-path))
                 (else #f))))
             (dad-path
               (path-to-blue-eyes
                 (ft-dad
                   (make-ft
                     'alice
                     'green
                     (make-ft 'emily 'brown 'unknown 'unknown)
                     (make-ft 'bruce 'blue 'unknown 'unknown))))))
         (cond
          ((and mom-path dad-path) (cons 'mom mom-path))
          ((and mom-path (not dad-path)) (cons 'mom mom-path))
          ((and dad-path (not mom-path)) (cons 'dad dad-path))
          (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (let ((mom-path (path-to-blue-eyes ’unknown))
                      (dad-path
                        (path-to-blue-eyes
                          (ft-dad (make-ft ’emily ’brown ’unknown ’unknown)))))
                   (cond
                    ((and mom-path dad-path) (cons ’mom mom-path))
                    ((and mom-path (not dad-path)) (cons ’mom mom-path))
                    ((and dad-path (not mom-path)) (cons ’dad dad-path))
                    (else #f))))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      ’alice
                      ’green
                      (make-ft ’emily ’brown ’unknown ’unknown)
                      (make-ft ’bruce ’blue ’unknown ’unknown))))))
           (cond
            ((and mom-path dad-path) (cons ’mom mom-path))
            ((and mom-path (not dad-path)) (cons ’mom mom-path))
            ((and dad-path (not mom-path)) (cons ’dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              ’robby
              ’hazel
              (make-ft
                ’alice
                ’green
                (make-ft ’emily ’brown ’unknown ’unknown)
                (make-ft ’bruce ’blue ’unknown ’unknown))
              (make-ft ’bill ’brown ’unknown ’unknown))))))
   (cond
    ((and mom-path dad-path) (cons ’mom mom-path))
    ((and mom-path (not dad-path)) (cons ’mom mom-path))
    ((and dad-path (not mom-path)) (cons ’dad dad-path))
    (else #f)))
```

```
(let ((mom-path
       (let ((mom-path
              (let ((mom-path
                     (cond
                      ((eq? 'unknown 'unknown) #f)
                      (else
                       (if (eq? (ft-eye-color 'unknown) 'blue)
                           '()
                           (let ((mom-path
                                   (path-to-blue-eyes (ft-mom 'unknown)))
                                 (dad-path
                                   (path-to-blue-eyes (ft-dad 'unknown))))
                             (cond
                              ((and mom-path dad-path) (cons 'mom mom-path))
                              ((and mom-path (not dad-path))
                               (cons 'mom mom-path))
                              ((and dad-path (not mom-path))
                               (cons 'dad dad-path))
                              (else #f)))))))
                    (dad-path
                      (path-to-blue-eyes
                        (ft-dad (make-ft 'emily 'brown 'unknown 'unknown)))))
                (cond
                 ((and mom-path dad-path) (cons 'mom mom-path))
                 ((and mom-path (not dad-path)) (cons 'mom mom-path))
                 ((and dad-path (not mom-path)) (cons 'dad dad-path))
                 (else #f))))
             (dad-path
               (path-to-blue-eyes
                 (ft-dad
                   (make-ft
                     'alice
                     'green
                     (make-ft 'emily 'brown 'unknown 'unknown)
                     (make-ft 'bruce 'blue 'unknown 'unknown))))))
         (cond
          ((and mom-path dad-path) (cons 'mom mom-path))
          ((and mom-path (not dad-path)) (cons 'mom mom-path))
          ((and dad-path (not mom-path)) (cons 'dad dad-path))
          (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
((and mom-path dad-path) (cons 'mom mom-path))
((and mom-path (not dad-path)) (cons 'mom mom-path))
((and dad-path (not mom-path)) (cons 'dad dad-path))
(else #f)))
```

```
(let ((mom-path
       (let ((mom-path
              (let ((mom-path
                     (cond
                      (#t #f)
                      (else
                       (if (eq? (ft-eye-color 'unknown) 'blue)
                           '()
                           (let ((mom-path
                                  (path-to-blue-eyes (ft-mom 'unknown)))
                                 (dad-path
                                  (path-to-blue-eyes (ft-dad 'unknown))))
                             (cond
                              ((and mom-path dad-path) (cons 'mom mom-path))
                              ((and mom-path (not dad-path))
                               (cons 'mom mom-path))
                              ((and dad-path (not mom-path))
                               (cons 'dad dad-path))
                              (else #f)))))))
                    (dad-path
                     (path-to-blue-eyes
                      (ft-dad (make-ft 'emily 'brown 'unknown 'unknown)))))
                (cond
                 ((and mom-path dad-path) (cons 'mom mom-path))
                 ((and mom-path (not dad-path)) (cons 'mom mom-path))
                 ((and dad-path (not mom-path)) (cons 'dad dad-path))
                 (else #f))))
             (dad-path
              (path-to-blue-eyes
               (ft-dad
                (make-ft
                 'alice
                 'green
                 (make-ft 'emily 'brown 'unknown 'unknown)
                 (make-ft 'bruce 'blue 'unknown 'unknown))))))
         (cond
          ((and mom-path dad-path) (cons 'mom mom-path))
          ((and mom-path (not dad-path)) (cons 'mom mom-path))
          ((and dad-path (not mom-path)) (cons 'dad dad-path))
          (else #f))))
      (dad-path
       (path-to-blue-eyes
        (ft-dad
         (make-ft
          'robby
          'hazel
          (make-ft
           'alice
           'green
           (make-ft 'emily 'brown 'unknown 'unknown)
           (make-ft 'bruce 'blue 'unknown 'unknown))
          (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
((and mom-path dad-path) (cons 'mom mom-path))
((and mom-path (not dad-path)) (cons 'mom mom-path))
((and dad-path (not mom-path)) (cons 'dad dad-path))
(else #f)))
```

```
(let ((mom-path
       (let ((mom-path
              (let ((mom-path #f)
                    (dad-path
                      (path-to-blue-eyes
                        (ft-dad (make-ft 'emily 'brown 'unknown 'unknown)))))
                  (cond
                   ((and mom-path dad-path) (cons 'mom mom-path))
                   ((and mom-path (not dad-path)) (cons 'mom mom-path))
                   ((and dad-path (not mom-path)) (cons 'dad dad-path))
                   (else #f))))
             (dad-path
               (path-to-blue-eyes
                 (ft-dad
                   (make-ft
                     'alice
                     'green
                     (make-ft 'emily 'brown 'unknown 'unknown)
                     (make-ft 'bruce 'blue 'unknown 'unknown))))))
           (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
      (let ((mom-path
            (let ((mom-path #f) (dad-path (path-to-blue-eyes 'unknown)))
              (cond
               ((and mom-path dad-path) (cons 'mom mom-path))
               ((and mom-path (not dad-path)) (cons 'mom mom-path))
               ((and dad-path (not mom-path)) (cons 'dad dad-path))
               (else #f))))
           (dad-path
             (path-to-blue-eyes
               (ft-dad
                 (make-ft
                   'alice
                   'green
                   (make-ft 'emily 'brown 'unknown 'unknown)
                   (make-ft 'bruce 'blue 'unknown 'unknown))))))
        (cond
         ((and mom-path dad-path) (cons 'mom mom-path))
         ((and mom-path (not dad-path)) (cons 'mom mom-path))
         ((and dad-path (not mom-path)) (cons 'dad dad-path))
         (else #f))))
     (dad-path
       (path-to-blue-eyes
         (ft-dad
           (make-ft
             'robby
             'hazel
             (make-ft
               'alice
               'green
               (make-ft 'emily 'brown 'unknown 'unknown)
               (make-ft 'bruce 'blue 'unknown 'unknown))
             (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (let ((mom-path #f) (dad-path #f))
                  (cond
                   ((and mom-path dad-path) (cons 'mom mom-path))
                   ((and mom-path (not dad-path)) (cons 'mom mom-path))
                   ((and dad-path (not mom-path)) (cons 'dad dad-path))
                   (else #f))))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
          (cond
           ((and mom-path dad-path) (cons 'mom mom-path))
           ((and mom-path (not dad-path)) (cons 'mom mom-path))
           ((and dad-path (not mom-path)) (cons 'dad dad-path))
           (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (cond
                 ((and #f #f) (cons 'mom #f))
                 ((and #f (not #f)) (cons 'mom #f))
                 ((and #f (not #f)) (cons 'dad #f))
                 (else #f)))
              (dad-path
                 (path-to-blue-eyes
                   (ft-dad
                      (make-ft
                        'alice
                        'green
                        (make-ft 'emily 'brown 'unknown 'unknown)
                        (make-ft 'bruce 'blue 'unknown 'unknown))))))
           (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
           (ft-dad
             (make-ft
                'robby
                'hazel
                (make-ft
                  'alice
                  'green
                  (make-ft 'emily 'brown 'unknown 'unknown)
                  (make-ft 'bruce 'blue 'unknown 'unknown))
                (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
       (let ((mom-path
              (cond
               (#f (cons 'mom #f))
               ((and #f (not #f)) (cons 'mom #f))
               ((and #f (not #f)) (cons 'dad #f))
               (else #f)))
             (dad-path
              (path-to-blue-eyes
               (ft-dad
                (make-ft
                 'alice
                 'green
                 (make-ft 'emily 'brown 'unknown 'unknown)
                 (make-ft 'bruce 'blue 'unknown 'unknown))))))
         (cond
          ((and mom-path dad-path) (cons 'mom mom-path))
          ((and mom-path (not dad-path)) (cons 'mom mom-path))
          ((and dad-path (not mom-path)) (cons 'dad dad-path))
          (else #f))))
      (dad-path
       (path-to-blue-eyes
        (ft-dad
         (make-ft
          'robby
          'hazel
          (make-ft
           'alice
           'green
           (make-ft 'emily 'brown 'unknown 'unknown)
           (make-ft 'bruce 'blue 'unknown 'unknown))
          (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (cond
                 ((and #f (not #f)) (cons 'mom #f))
                 ((and #f (not #f)) (cons 'dad #f))
                 (else #f)))
               (dad-path
                 (path-to-blue-eyes
                   (ft-dad
                     (make-ft
                       'alice
                       'green
                       (make-ft 'emily 'brown 'unknown 'unknown)
                       (make-ft 'bruce 'blue 'unknown 'unknown))))))
            (cond
             ((and mom-path dad-path) (cons 'mom mom-path))
             ((and mom-path (not dad-path)) (cons 'mom mom-path))
             ((and dad-path (not mom-path)) (cons 'dad dad-path))
             (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (cond
                 ((and #f #t) (cons 'mom #f))
                 ((and #f (not #f)) (cons 'dad #f))
                 (else #f)))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
            (cond
             ((and mom-path dad-path) (cons 'mom mom-path))
             ((and mom-path (not dad-path)) (cons 'mom mom-path))
             ((and dad-path (not mom-path)) (cons 'dad dad-path))
             (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (cond
                 ((and #f #t) (cons 'mom #f))
                 ((and #f (not #f)) (cons 'dad #f))
                 (else #f)))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
            (cond
             ((and mom-path dad-path) (cons 'mom mom-path))
             ((and mom-path (not dad-path)) (cons 'mom mom-path))
             ((and dad-path (not mom-path)) (cons 'dad dad-path))
             (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path
                (cond
                 (#f (cons 'mom #f))
                 ((and #f (not #f)) (cons 'dad #f))
                 (else #f)))
               (dad-path
                 (path-to-blue-eyes
                   (ft-dad
                     (make-ft
                       'alice
                       'green
                       (make-ft 'emily 'brown 'unknown 'unknown)
                       (make-ft 'bruce 'blue 'unknown 'unknown))))))
           (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
       (dad-path
         (path-to-blue-eyes
           (ft-dad
             (make-ft
               'robby
               'hazel
               (make-ft
                 'alice
                 'green
                 (make-ft 'emily 'brown 'unknown 'unknown)
                 (make-ft 'bruce 'blue 'unknown 'unknown))
               (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
        (let ((mom-path (cond ((and #f (not #f)) (cons 'dad #f)) (else #f)))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
            (cond
              ((and mom-path dad-path) (cons 'mom mom-path))
              ((and mom-path (not dad-path)) (cons 'mom mom-path))
              ((and dad-path (not mom-path)) (cons 'dad dad-path))
              (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
        (let ((mom-path (cond ((and #f #t) (cons 'dad #f)) (else #f)))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
          (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
        (let ((mom-path (cond (else #f)))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
          (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
        (let ((mom-path #f)
              (dad-path
                (path-to-blue-eyes
                  (ft-dad
                    (make-ft
                      'alice
                      'green
                      (make-ft 'emily 'brown 'unknown 'unknown)
                      (make-ft 'bruce 'blue 'unknown 'unknown))))))
          (cond
           ((and mom-path dad-path) (cons 'mom mom-path))
           ((and mom-path (not dad-path)) (cons 'mom mom-path))
           ((and dad-path (not mom-path)) (cons 'dad dad-path))
           (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path #f)
              (dad-path
                (path-to-blue-eyes (make-ft 'bruce 'blue 'unknown 'unknown))))
           (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
        (let ((mom-path #f)
              (dad-path
                (cond
                 ((eq? (make-ft 'bruce 'blue 'unknown 'unknown) 'unknown) #f)
                 (else
                  (if (eq?
                      (ft-eye-color (make-ft 'bruce 'blue 'unknown 'unknown))
                      'blue)
                     '()
                     (let ((mom-path
                             (path-to-blue-eyes
                               (ft-mom
                                 (make-ft 'bruce 'blue 'unknown 'unknown))))
                           (dad-path
                             (path-to-blue-eyes
                               (ft-dad
                                 (make-ft 'bruce 'blue 'unknown 'unknown)))))
                        (cond
                         ((and mom-path dad-path) (cons 'mom mom-path))
                         ((and mom-path (not dad-path)) (cons 'mom mom-path))
                         ((and dad-path (not mom-path)) (cons 'dad dad-path))
                         (else #f))))))))
           (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
       (let ((mom-path #f)
           (dad-path
             (cond
              (#f #f)
              (else
               (if (eq?
                   (ft-eye-color (make-ft 'bruce 'blue 'unknown 'unknown))
                    'blue)
                  '()
                  (let ((mom-path
                         (path-to-blue-eyes
                           (ft-mom
                             (make-ft 'bruce 'blue 'unknown 'unknown))))
                       (dad-path
                         (path-to-blue-eyes
                           (ft-dad
                             (make-ft 'bruce 'blue 'unknown 'unknown)))))
                    (cond
                     ((and mom-path dad-path) (cons 'mom mom-path))
                     ((and mom-path (not dad-path)) (cons 'mom mom-path))
                     ((and dad-path (not mom-path)) (cons 'dad dad-path))
                     (else #f))))))))
         (cond
          ((and mom-path dad-path) (cons 'mom mom-path))
          ((and mom-path (not dad-path)) (cons 'mom mom-path))
          ((and dad-path (not mom-path)) (cons 'dad dad-path))
          (else #f))))
     (dad-path
       (path-to-blue-eyes
         (ft-dad
           (make-ft
             'robby
             'hazel
             (make-ft
               'alice
               'green
               (make-ft 'emily 'brown 'unknown 'unknown)
               (make-ft 'bruce 'blue 'unknown 'unknown))
             (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path #f)
             (dad-path
                (cond
                 (else
                  (if (eq?
                      (ft-eye-color (make-ft 'bruce 'blue 'unknown 'unknown))
                      'blue)
                     '()
                     (let ((mom-path
                             (path-to-blue-eyes
                                (ft-mom
                                  (make-ft 'bruce 'blue 'unknown 'unknown))))
                           (dad-path
                              (path-to-blue-eyes
                                (ft-dad
                                   (make-ft 'bruce 'blue 'unknown 'unknown)))))
                        (cond
                         ((and mom-path dad-path) (cons 'mom mom-path))
                         ((and mom-path (not dad-path)) (cons 'mom mom-path))
                         ((and dad-path (not mom-path)) (cons 'dad dad-path))
                         (else #f)))))))
           (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
           (ft-dad
              (make-ft
                'robby
                'hazel
                (make-ft
                   'alice
                   'green
                   (make-ft 'emily 'brown 'unknown 'unknown)
                   (make-ft 'bruce 'blue 'unknown 'unknown))
                (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
       (let ((mom-path #f)
             (dad-path
              (if (eq?
                    (ft-eye-color (make-ft 'bruce 'blue 'unknown 'unknown))
                    'blue)
                   '()
                   (let ((mom-path
                          (path-to-blue-eyes
                            (ft-mom (make-ft 'bruce 'blue 'unknown 'unknown))))
                         (dad-path
                           (path-to-blue-eyes
                             (ft-dad
                               (make-ft 'bruce 'blue 'unknown 'unknown)))))
                     (cond
                      ((and mom-path dad-path) (cons 'mom mom-path))
                      ((and mom-path (not dad-path)) (cons 'mom mom-path))
                      ((and dad-path (not mom-path)) (cons 'dad dad-path))
                      (else #f))))))
         (cond
          ((and mom-path dad-path) (cons 'mom mom-path))
          ((and mom-path (not dad-path)) (cons 'mom mom-path))
          ((and dad-path (not mom-path)) (cons 'dad dad-path))
          (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (let ((mom-path #f)
              (dad-path
                (if (eq? 'blue 'blue)
                   '()
                   (let ((mom-path
                           (path-to-blue-eyes
                              (ft-mom (make-ft 'bruce 'blue 'unknown 'unknown))))
                         (dad-path
                           (path-to-blue-eyes
                              (ft-dad
                                 (make-ft 'bruce 'blue 'unknown 'unknown)))))
                      (cond
                       ((and mom-path dad-path) (cons 'mom mom-path))
                       ((and mom-path (not dad-path)) (cons 'mom mom-path))
                       ((and dad-path (not mom-path)) (cons 'dad dad-path))
                       (else #f))))))
           (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f))))
      (dad-path
        (path-to-blue-eyes
           (ft-dad
              (make-ft
                'robby
                'hazel
                (make-ft
                   'alice
                   'green
                   (make-ft 'emily 'brown 'unknown 'unknown)
                   (make-ft 'bruce 'blue 'unknown 'unknown))
                (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
        (let ((mom-path #f) (dad-path '()))
          (cond
           ((and mom-path dad-path) (cons 'mom mom-path))
           ((and mom-path (not dad-path)) (cons 'mom mom-path))
           ((and dad-path (not mom-path)) (cons 'dad dad-path))
           (else #f))))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
          (cond
           ((and #f '()) (cons 'mom #f))
           ((and #f (not '())) (cons 'mom #f))
           ((and '() (not #f)) (cons 'dad '()))
           (else #f)))
        (dad-path
          (path-to-blue-eyes
            (ft-dad
              (make-ft
                'robby
                'hazel
                (make-ft
                  'alice
                  'green
                  (make-ft 'emily 'brown 'unknown 'unknown)
                  (make-ft 'bruce 'blue 'unknown 'unknown))
                (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path
        (cond
         (#f (cons 'mom #f))
         ((and #f (not '())) (cons 'mom #f))
         ((and '() (not #f)) (cons 'dad '()))
         (else #f)))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (cond
         ((and #f (not '())) (cons 'mom #f))
         ((and '() (not #f)) (cons 'dad '()))
         (else #f)))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path
        (cond
          ((and #f #f) (cons 'mom #f))
          ((and '() (not #f)) (cons 'dad '()))
          (else #f)))
       (dad-path
         (path-to-blue-eyes
           (ft-dad
             (make-ft
               'robby
               'hazel
               (make-ft
                 'alice
                 'green
                 (make-ft 'emily 'brown 'unknown 'unknown)
                 (make-ft 'bruce 'blue 'unknown 'unknown))
               (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path (cond ((and '() (not #f)) (cons 'dad '())) (else #f)))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path (cond ((and '() #t) (cons 'dad '())) (else #f)))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (cond
         ((eq? (make-ft 'bill 'brown 'unknown 'unknown) 'unknown) #f)
         (else
          (if (eq?
                (ft-eye-color (make-ft 'bill 'brown 'unknown 'unknown))
                'blue)
              '()
              (let ((mom-path
                        (path-to-blue-eyes
                          (ft-mom (make-ft 'bill 'brown 'unknown 'unknown))))
                     (dad-path
                       (path-to-blue-eyes
                         (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
                (cond
                 ((and mom-path dad-path) (cons 'mom mom-path))
                 ((and mom-path (not dad-path)) (cons 'mom mom-path))
                 ((and dad-path (not mom-path)) (cons 'dad dad-path))
                 (else #f))))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (cond
         (#f #f)
         (else
          (if (eq?
                (ft-eye-color (make-ft 'bill 'brown 'unknown 'unknown))
                'blue)
               '()
               (let ((mom-path
                        (path-to-blue-eyes
                          (ft-mom (make-ft 'bill 'brown 'unknown 'unknown))))
                      (dad-path
                        (path-to-blue-eyes
                          (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
                 (cond
                  ((and mom-path dad-path) (cons 'mom mom-path))
                  ((and mom-path (not dad-path)) (cons 'mom mom-path))
                  ((and dad-path (not mom-path)) (cons 'dad dad-path))
                  (else #f))))))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (cond
         (else
          (if (eq?
               (ft-eye-color (make-ft 'bill 'brown 'unknown 'unknown))
               'blue)
              '()
              (let ((mom-path
                      (path-to-blue-eyes
                        (ft-mom (make-ft 'bill 'brown 'unknown 'unknown))))
                    (dad-path
                      (path-to-blue-eyes
                        (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
                (cond
                 ((and mom-path dad-path) (cons 'mom mom-path))
                 ((and mom-path (not dad-path)) (cons 'mom mom-path))
                 ((and dad-path (not mom-path)) (cons 'dad dad-path))
                 (else #f))))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (if (eq? (ft-eye-color (make-ft 'bill 'brown 'unknown 'unknown)) 'blue)
          '()
          (let ((mom-path
                  (path-to-blue-eyes
                    (ft-mom (make-ft 'bill 'brown 'unknown 'unknown))))
                (dad-path
                  (path-to-blue-eyes
                    (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
            (cond
              ((and mom-path dad-path) (cons 'mom mom-path))
              ((and mom-path (not dad-path)) (cons 'mom mom-path))
              ((and dad-path (not mom-path)) (cons 'dad dad-path))
              (else #f))))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (if (eq? 'brown 'blue)
            '()
            (let ((mom-path
                     (path-to-blue-eyes
                       (ft-mom (make-ft 'bill 'brown 'unknown 'unknown))))
                  (dad-path
                     (path-to-blue-eyes
                       (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
              (cond
               ((and mom-path dad-path) (cons 'mom mom-path))
               ((and mom-path (not dad-path)) (cons 'mom mom-path))
               ((and dad-path (not mom-path)) (cons 'dad dad-path))
               (else #f))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (if #f
            '()
            (let ((mom-path
                     (path-to-blue-eyes
                        (ft-mom (make-ft 'bill 'brown 'unknown 'unknown))))
                   (dad-path
                     (path-to-blue-eyes
                        (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
               (cond
                ((and mom-path dad-path) (cons 'mom mom-path))
                ((and mom-path (not dad-path)) (cons 'mom mom-path))
                ((and dad-path (not mom-path)) (cons 'dad dad-path))
                (else #f))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (let ((mom-path
                 (path-to-blue-eyes
                   (ft-mom (make-ft 'bill 'brown 'unknown 'unknown))))
               (dad-path
                 (path-to-blue-eyes
                   (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
          (cond
           ((and mom-path dad-path) (cons 'mom mom-path))
           ((and mom-path (not dad-path)) (cons 'mom mom-path))
           ((and dad-path (not mom-path)) (cons 'dad dad-path))
           (else #f)))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (let ((mom-path (path-to-blue-eyes 'unknown))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
          (cond
           ((and mom-path dad-path) (cons 'mom mom-path))
           ((and mom-path (not dad-path)) (cons 'mom mom-path))
           ((and dad-path (not mom-path)) (cons 'dad dad-path))
           (else #f)))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (let ((mom-path #f)
              (dad-path
                (path-to-blue-eyes
                  (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
          (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f)))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (let ((mom-path #f) (dad-path #f))
          (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f)))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (cond
          ((and #f #f) (cons 'mom #f))
          ((and #t (not #f)) (cons 'mom #f))
          ((and #f (not #f)) (cons 'dad #f))
          (else #f))))
   (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

```
(let ((mom-path (cons 'dad '())) (dad-path #f))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

```
(cond
  ((and (cons 'dad '()) #f) (cons 'mom (cons 'dad '())))
  ((and (cons 'dad '()) (not #f)) (cons 'mom (cons 'dad '())))
  ((and #f (not (cons 'dad '()))) (cons 'dad #f))
  (else #f))
```

```
(cond
  (#f (cons 'mom (cons 'dad '())))
  ((and (cons 'dad '()) (not #f)) (cons 'mom (cons 'dad '())))
  ((and #f (not (cons 'dad '()))) (cons 'dad #f))
  (else #f))
```

```
(cond
  ((and (cons 'dad '()) (not #f)) (cons 'mom (cons 'dad '())))
  ((and #f (not (cons 'dad '()))) (cons 'dad #f))
  (else #f))
```

```
(cond
 ((and (cons 'dad '()) #t) (cons 'mom (cons 'dad '())))
 ((and #f (not (cons 'dad '()))) (cons 'dad #f))
 (else #f))
```

```
(cond
 (#t (cons 'mom (cons 'dad '())))
 ((and #f (not (cons 'dad '()))) (cons 'dad #f))
 (else #f))
```

```
(cons 'mom (cons 'dad '()))
```

## 2 Union-nodup

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (let ((mom-path (path-to-blue-eyes 'unknown))
              (dad-path
                (path-to-blue-eyes
                  (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
          (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f)))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

Hand Evaluate:

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (let ((mom-path #f)
              (dad-path
                (path-to-blue-eyes
                  (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
          (cond
            ((and mom-path dad-path) (cons 'mom mom-path))
            ((and mom-path (not dad-path)) (cons 'mom mom-path))
            ((and dad-path (not mom-path)) (cons 'dad dad-path))
            (else #f)))))
  (cond
    ((and mom-path dad-path) (cons 'mom mom-path))
    ((and mom-path (not dad-path)) (cons 'mom mom-path))
    ((and dad-path (not mom-path)) (cons 'dad dad-path))
    (else #f)))
```

**Solution**

```
(cond
 ((null? (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))
 (else
  (let ((un (union-nodup (cdr (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))))
```

```
(cond
 (#f (cons 1 (cons 2 '())))
 (else
  (let ((un (union-nodup (cdr (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))))
```

```
(cond
 (else
  (let ((un (union-nodup (cdr (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))))
```

```
(let ((un (union-nodup (cdr (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
    un
    (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un (union-nodup (cons 3 '()) (cons 1 (cons 2 '())))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (cond
         ((null? (cons 3 '())) (cons 1 (cons 2 '())))
         (else
          (let ((un (union-nodup (cdr (cons 3 '())) (cons 1 (cons 2 '())))))
             (if (number-in-set? (car (cons 3 '())) un)
                un
                (cons (car (cons 3 '())) un)))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (cond
         (#f (cons 1 (cons 2 '())))
         (else
          (let ((un (union-nodup (cdr (cons 3 '())) (cons 1 (cons 2 '())))))
            (if (number-in-set? (car (cons 3 '())) un)
                un
                (cons (car (cons 3 '())) un)))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (cond
         (else
          (let ((un (union-nodup (cdr (cons 3 '())) (cons 1 (cons 2 '())))))
            (if (number-in-set? (car (cons 3 '())) un)
                un
                (cons (car (cons 3 '())) un)))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un (union-nodup (cdr (cons 3 '())) (cons 1 (cons 2 '())))))
          (if (number-in-set? (car (cons 3 '())) un)
              un
              (cons (car (cons 3 '())) un)))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un (union-nodup '() (cons 1 (cons 2 '())))))
          (if (number-in-set? (car (cons 3 '())) un)
              un
              (cons (car (cons 3 '())) un)))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un
                (cond
                 ((null? '()) (cons 1 (cons 2 '())))
                 (else
                  (let ((un (union-nodup (cdr '()) (cons 1 (cons 2 '())))))
                     (if (number-in-set? (car '()) un)
                        un
                        (cons (car '()) un))))))))
            (if (number-in-set? (car (cons 3 '())) un)
               un
               (cons (car (cons 3 '())) un)))))
    (if (number-in-set? (car (cons 2 (cons 3 '())))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un
                (cond
                 (#t (cons 1 (cons 2 '())))
                 (else
                  (let ((un (union-nodup (cdr '()) (cons 1 (cons 2 '())))))
                     (if (number-in-set? (car '()) un)
                        un
                        (cons (car '()) un))))))))
           (if (number-in-set? (car (cons 3 '())) un)
              un
              (cons (car (cons 3 '())) un)))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un (cons 1 (cons 2 '()))))
           (if (number-in-set? (car (cons 3 '())) un)
              un
              (cons (car (cons 3 '())) un)))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? (car (cons 3 '())) (cons 1 (cons 2 '())))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '())))) un)
      un
      (cons (car (cons 2 (cons 3 '())))) un)))
```

```
(let ((un
        (if (number-in-set? 3 (cons 1 (cons 2 '())))
           (cons 1 (cons 2 '()))
           (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
             ((null? (cons 1 (cons 2 '()))) #f)
             (else
              (or (= 3 (car (cons 1 (cons 2 '()))))
                   (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
             (#f #f)
             (else
              (or (= 3 (car (cons 1 (cons 2 '()))))
                   (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
     un
     (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
             (else
              (or (= 3 (car (cons 1 (cons 2 '()))))
                  (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or (= 3 (car (cons 1 (cons 2 '()))))
                 (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or (= 3 1) (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or #f (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? 3 (cdr (cons 1 (cons 2 '()))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? 3 (cons 2 '()))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
       (if (cond
             ((null? (cons 2 '())) #f)
             (else
              (or (= 3 (car (cons 2 '())))
                   (number-in-set? 3 (cdr (cons 2 '()))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
     un
     (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
             (#f #f)
             (else
              (or (= 3 (car (cons 2 '())))
                   (number-in-set? 3 (cdr (cons 2 '()))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
              (else
               (or (= 3 (car (cons 2 '())))
                   (number-in-set? 3 (cdr (cons 2 '()))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '())))))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or (= 3 (car (cons 2 '()))) (number-in-set? 3 (cdr (cons 2 '()))))
           (cons 1 (cons 2 '()))
           (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or (= 3 2) (number-in-set? 3 (cdr (cons 2 '()))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or #f (number-in-set? 3 (cdr (cons 2 '()))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? 3 (cdr (cons 2 '())))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? 3 '())
           (cons 1 (cons 2 '()))
           (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
              ((null? '()) #f)
              (else (or (= 3 (car '())) (number-in-set? 3 (cdr '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '())))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
              (#t #f)
              (else (or (= 3 (car '())) (number-in-set? 3 (cdr '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if #f
           (cons 1 (cons 2 '()))
           (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un (cons (car (cons 3 '())) (cons 1 (cons 2 '())))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
    un
    (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un (cons 3 (cons 1 (cons 2 '())))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(if (number-in-set? (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '()))))
  (cons 3 (cons 1 (cons 2 '())))
  (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cons 3 (cons 1 (cons 2 '()))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      ((null? (cons 3 (cons 1 (cons 2 '())))) #f)
      (else
       (or (= 2 (car (cons 3 (cons 1 (cons 2 '())))))
           (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '()))))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (#f #f)
      (else
       (or (= 2 (car (cons 3 (cons 1 (cons 2 '())))))
           (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '()))))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (else
       (or (= 2 (car (cons 3 (cons 1 (cons 2 '())))))
           (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '())))))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 (car (cons 3 (cons 1 (cons 2 '()))))))
        (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '())))))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 3) (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '())))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or #f (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '())))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '())))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cons 1 (cons 2 '())))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cons 1 (cons 2 '())))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      ((null? (cons 1 (cons 2 '()))) #f)
      (else
       (or (= 2 (car (cons 1 (cons 2 '()))))
           (number-in-set? 2 (cdr (cons 1 (cons 2 '()))))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (#f #f)
      (else
       (or (= 2 (car (cons 1 (cons 2 '()))))
           (number-in-set? 2 (cdr (cons 1 (cons 2 '()))))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (else
       (or (= 2 (car (cons 1 (cons 2 '()))))
           (number-in-set? 2 (cdr (cons 1 (cons 2 '())))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 (car (cons 1 (cons 2 '()))))
        (number-in-set? 2 (cdr (cons 1 (cons 2 '())))))
  (cons 3 (cons 1 (cons 2 '())))
  (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 1) (number-in-set? 2 (cdr (cons 1 (cons 2 '())))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or #f (number-in-set? 2 (cdr (cons 1 (cons 2 '())))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cdr (cons 1 (cons 2 '()))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cons 2 '()))
  (cons 3 (cons 1 (cons 2 '())))
  (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      ((null? (cons 2 '())) #f)
      (else
       (or (= 2 (car (cons 2 '()))) (number-in-set? 2 (cdr (cons 2 '()))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (#f #f)
      (else
       (or (= 2 (car (cons 2 '()))) (number-in-set? 2 (cdr (cons 2 '()))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (else
        (or (= 2 (car (cons 2 '()))) (number-in-set? 2 (cdr (cons 2 '()))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 (car (cons 2 '()))) (number-in-set? 2 (cdr (cons 2 '()))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 2) (number-in-set? 2 (cdr (cons 2 '()))))
  (cons 3 (cons 1 (cons 2 '())))
  (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or #t (number-in-set? 2 (cdr (cons 2 '()))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if #t
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

(*cons* 3 (*cons* 1 (*cons* 2 ’())))

## 3 Union-sort

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (path-to-blue-eyes
          (ft-dad
            (make-ft
              'robby
              'hazel
              (make-ft
                'alice
                'green
                (make-ft 'emily 'brown 'unknown 'unknown)
                (make-ft 'bruce 'blue 'unknown 'unknown))
              (make-ft 'bill 'brown 'unknown 'unknown))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

Hand Evaluate:

```
(let ((mom-path (cons 'dad '()))
      (dad-path
        (cond
         ((eq? (make-ft 'bill 'brown 'unknown 'unknown) 'unknown) #f)
         (else
          (if (eq?
                (ft-eye-color (make-ft 'bill 'brown 'unknown 'unknown))
                'blue)
              '()
              (let ((mom-path
                       (path-to-blue-eyes
                         (ft-mom (make-ft 'bill 'brown 'unknown 'unknown))))
                     (dad-path
                       (path-to-blue-eyes
                         (ft-dad (make-ft 'bill 'brown 'unknown 'unknown)))))
                (cond
                 ((and mom-path dad-path) (cons 'mom mom-path))
                 ((and mom-path (not dad-path)) (cons 'mom mom-path))
                 ((and dad-path (not mom-path)) (cons 'dad dad-path))
                 (else #f)))))))))
  (cond
   ((and mom-path dad-path) (cons 'mom mom-path))
   ((and mom-path (not dad-path)) (cons 'mom mom-path))
   ((and dad-path (not mom-path)) (cons 'dad dad-path))
   (else #f)))
```

**Solution**

```
(cond
 ((null? (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))
 (else
  (let ((un (union-nodup (cdr (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))))
```

```
(cond
 (#f (cons 1 (cons 2 '())))
 (else
  (let ((un (union-nodup (cdr (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))))
```

```
(cond
 (else
  (let ((un (union-nodup (cdr (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))))
```

```
(let ((un (union-nodup (cdr (cons 2 (cons 3 '()))) (cons 1 (cons 2 '())))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
    un
    (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un (union-nodup (cons 3 '()) (cons 1 (cons 2 '())))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
    un
    (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (cond
         ((null? (cons 3 '())) (cons 1 (cons 2 '())))
         (else
          (let ((un (union-nodup (cdr (cons 3 '())) (cons 1 (cons 2 '())))))
             (if (number-in-set? (car (cons 3 '())) un)
                un
                (cons (car (cons 3 '())) un)))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (cond
         (#f (cons 1 (cons 2 '())))
         (else
          (let ((un (union-nodup (cdr (cons 3 '())) (cons 1 (cons 2 '())))))
            (if (number-in-set? (car (cons 3 '())) un)
                un
                (cons (car (cons 3 '())) un)))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
     un
     (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
       (cond
        (else
         (let ((un (union-nodup (cdr (cons 3 '())) (cons 1 (cons 2 '())))))
           (if (number-in-set? (car (cons 3 '())) un)
               un
               (cons (car (cons 3 '())) un)))))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un (union-nodup (cdr (cons 3 '())) (cons 1 (cons 2 '())))))
          (if (number-in-set? (car (cons 3 '())) un)
              un
              (cons (car (cons 3 '())) un)))))
    (if (number-in-set? (car (cons 2 (cons 3 '())))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un (union-nodup '() (cons 1 (cons 2 '())))))
          (if (number-in-set? (car (cons 3 '())) un)
              un
              (cons (car (cons 3 '())) un)))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un
                (cond
                 ((null? '()) (cons 1 (cons 2 '())))
                 (else
                  (let ((un (union-nodup (cdr '()) (cons 1 (cons 2 '())))))
                     (if (number-in-set? (car '()) un)
                        un
                        (cons (car '()) un)))))))
            (if (number-in-set? (car (cons 3 '())) un)
               un
               (cons (car (cons 3 '())) un)))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un
                (cond
                 (#t (cons 1 (cons 2 '())))
                 (else
                  (let ((un (union-nodup (cdr '()) (cons 1 (cons 2 '())))))
                     (if (number-in-set? (car '()) un)
                        un
                        (cons (car '()) un)))))))
            (if (number-in-set? (car (cons 3 '())) un)
               un
               (cons (car (cons 3 '())) un)))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (let ((un (cons 1 (cons 2 '()))))
          (if (number-in-set? (car (cons 3 '())) un)
              un
              (cons (car (cons 3 '())) un)))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? (car (cons 3 '())) (cons 1 (cons 2 '())))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? 3 (cons 1 (cons 2 '())))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
             ((null? (cons 1 (cons 2 '()))) #f)
             (else
              (or (= 3 (car (cons 1 (cons 2 '()))))
                  (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
              (#f #f)
              (else
                (or (= 3 (car (cons 1 (cons 2 '()))))
                    (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))))
             (cons 1 (cons 2 '()))
             (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
              (else
               (or (= 3 (car (cons 1 (cons 2 '()))))
                   (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or (= 3 (car (cons 1 (cons 2 '()))))
                 (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or (= 3 1) (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or #f (number-in-set? 3 (cdr (cons 1 (cons 2 '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? 3 (cdr (cons 1 (cons 2 '()))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? 3 (cons 2 '()))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
             ((null? (cons 2 '())) #f)
             (else
              (or (= 3 (car (cons 2 '())))
                   (number-in-set? 3 (cdr (cons 2 '()))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
     un
     (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
              (#f #f)
              (else
                (or (= 3 (car (cons 2 '())))
                    (number-in-set? 3 (cdr (cons 2 '()))))))
             (cons 1 (cons 2 '()))
             (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
              (else
               (or (= 3 (car (cons 2 '())))
                   (number-in-set? 3 (cdr (cons 2 '()))))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '())))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
     un
     (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or (= 3 (car (cons 2 '()))) (number-in-set? 3 (cdr (cons 2 '()))))
           (cons 1 (cons 2 '()))
           (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or (= 3 2) (number-in-set? 3 (cdr (cons 2 '()))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (or #f (number-in-set? 3 (cdr (cons 2 '()))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? 3 (cdr (cons 2 '())))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
        un
        (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (number-in-set? 3 '())
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
              ((null? '()) #f)
              (else (or (= 3 (car '())) (number-in-set? 3 (cdr '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
       un
       (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if (cond
              (#t #f)
              (else (or (= 3 (car '())) (number-in-set? 3 (cdr '())))))
            (cons 1 (cons 2 '()))
            (cons (car (cons 3 '())) (cons 1 (cons 2 '())))))))
    (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un
        (if #f
           (cons 1 (cons 2 '()))
           (cons (car (cons 3 '())) (cons 1 (cons 2 '()))))))
   (if (number-in-set? (car (cons 2 (cons 3 '())))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un (cons (car (cons 3 '())) (cons 1 (cons 2 '())))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
    un
    (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(let ((un (cons 3 (cons 1 (cons 2 '())))))
  (if (number-in-set? (car (cons 2 (cons 3 '()))) un)
      un
      (cons (car (cons 2 (cons 3 '()))) un)))
```

```
(if (number-in-set? (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '()))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cons 3 (cons 1 (cons 2 '()))))
  (cons 3 (cons 1 (cons 2 '())))
  (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      ((null? (cons 3 (cons 1 (cons 2 '())))) #f)
      (else
       (or (= 2 (car (cons 3 (cons 1 (cons 2 '())))))
           (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '()))))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (#f #f)
      (else
       (or (= 2 (car (cons 3 (cons 1 (cons 2 '())))))
           (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '()))))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (else
       (or (= 2 (car (cons 3 (cons 1 (cons 2 '())))))
           (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '()))))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 (car (cons 3 (cons 1 (cons 2 '())))))
        (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '()))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 3) (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '())))))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or #f (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '())))))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cdr (cons 3 (cons 1 (cons 2 '())))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cons 1 (cons 2 '())))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cons 1 (cons 2 '())))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
     ((null? (cons 1 (cons 2 '()))) #f)
     (else
      (or (= 2 (car (cons 1 (cons 2 '()))))
          (number-in-set? 2 (cdr (cons 1 (cons 2 '())))))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (#f #f)
      (else
       (or (= 2 (car (cons 1 (cons 2 '()))))
           (number-in-set? 2 (cdr (cons 1 (cons 2 '()))))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (else
       (or (= 2 (car (cons 1 (cons 2 '()))))
           (number-in-set? 2 (cdr (cons 1 (cons 2 '())))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 (car (cons 1 (cons 2 '()))))
        (number-in-set? 2 (cdr (cons 1 (cons 2 '())))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 1) (number-in-set? 2 (cdr (cons 1 (cons 2 '())))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or #f (number-in-set? 2 (cdr (cons 1 (cons 2 '())))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cdr (cons 1 (cons 2 '()))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (number-in-set? 2 (cons 2 '()))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      ((null? (cons 2 '())) #f)
      (else
       (or (= 2 (car (cons 2 '()))) (number-in-set? 2 (cdr (cons 2 '()))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (#f #f)
      (else
       (or (= 2 (car (cons 2 '()))) (number-in-set? 2 (cdr (cons 2 '()))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (cond
      (else
        (or (= 2 (car (cons 2 '()))) (number-in-set? 2 (cdr (cons 2 '()))))))
    (cons 3 (cons 1 (cons 2 '())))
    (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 (car (cons 2 '()))) (number-in-set? 2 (cdr (cons 2 '()))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or (= 2 2) (number-in-set? 2 (cdr (cons 2 '()))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if (or #t (number-in-set? 2 (cdr (cons 2 '()))))
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(if #t
   (cons 3 (cons 1 (cons 2 '())))
   (cons (car (cons 2 (cons 3 '()))) (cons 3 (cons 1 (cons 2 '())))))
```

```
(cons 3 (cons 1 (cons 2 '())))
```