More Realistic Rumor Mill

Let each gossip talk to any number of people:

- Lindsey
- Derrick
- Amir
- Seiichi
- Joe
- Mike
Representing Revised Rumor Mills

How do we represent an arbitrary number of gossip connections?
Representing Revised Rumor Mills

How do we represent an arbitrary number of gossip connections?

; A list-of-gossip is either
;   - '()
;   - (cons gossip list-of-gossip)

; A gossip is
;   (make-gossip image list-of-gossip)
(define-struct gossip (who nexts))
Programming with Revised Rumor Mills

; A list-of-gossip is either
;   - '()
;   - (cons gossip list-of-gossip)

; A gossip is
;   (make-gossip image list-of-gossip)
Programming with Revised Rumor Mills

; A list-of-gossip is either
;   - ()
;   - (cons gossip list-of-gossip)

; A gossip is
;   (make-gossip image list-of-gossip)
Programming with Revised Rumor Mills

; A list-of-gossip is either
;   - '()
;   - (cons gossip list-of-gossip)

; A gossip is
;   (make-gossip image list-of-gossip)
Programming with Revised Rumor Mills

; A list-of-gossip is either
;   - '()
;   - (cons gossip list-of-gossip)

; A gossip is
;   (make-gossip image list-of-gossip)
Programming with Revised Rumor Mills

; A list-of-gossip is either
;   - '()
;   - (cons gossip list-of-gossip)

; A gossip is
;   (make-gossip image list-of-gossip)

(define (func-for-log l)
  (cond
   [(empty? l) ...]
   [(cons? l)
     ... (func-for-gossip (first l))
     ... (func-for-log (rest l))]]))

(define (func-for-gossip g)
  ... (gossip-who g)
  ... (func-for-log (gossip-nexts g)) ...)

Examples for Revised Rumor Mills

Implement `count-people`, which takes a gossip and returns the number of people informed by the gossip (including the starting person)

Implement the function `informed?` which takes a person image and a gossip and determines whether the person is part of the rumor mill

Implement `remove-person`, which takes a person image and a gossip and returns a gossip where the given person is uninformed

... and any other function for the old rumor mills