In *The Mythical Man-Month* (1975), Turing award recipient Fred Brooks writes:
“The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures. Yet the program construct, unlike the poet’s words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself… One types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were nor could be… It prints results, draws pictures, produces sounds, moves arms. The magic of myth and legend has come true in our time…”
“The computer resembles the magic of legend in this respect, too. If one character, one pause, of the incantation is not strictly in proper form, the magic doesn’t work. Human beings are not accustomed to being perfect, and few areas of human activity demand it. Adjusting to the requirement for perfection is, I think, the most difficult part of learning to program.”
A long time ago in a class exactly, exactly here....
(make-gossip
  (make-gossip
    (make-gossip
      '()
      '()))
  (make-gossip
    (make-gossip
      '()
      (make-gossip
        '()' '()))
    (make-gossip
      '()' '()))
Programming with rumors

;;; A RumorMill is either
;;; - '()
;;; - (make-gossip Image RumorMill RumorMill)

;;; RumorMill -> ...
(define (fun-for-rumor-mill rm)
  (cond [(empty? rm) ...]
        [(gossip? rm)
           (... (gossip-who rm)
                (... (gossip-next1 rm))
                (... (gossip-next2 rm)))]))
A more realistic rumor mill
In our rumor mill, we restricted each person to spread gossip to at most two other people.

This isn’t very realistic; some gossips talk to lots of people!
Let each gossip talk to any number of people:

- Pansy
- Draco
- Cho
- Romilda
- Vincent
- Ginny
How do we represent an arbitrary number of gossip connections?
How do we represent an arbitrary number of gossip connections?

;; A ListOfGossip is either
;;   - '()
;;   - (cons Gossip ListOfGossip)

;; A Gossip is
;;   (make-gossip Image ListOfGossip)
(define-struct gossip [who nexts])
Programming with revised rumor mills

;; A ListOfGossip is either
;;   - '()
;;   - (cons Gossip ListOfGossip)

;; A Gossip is
;;  (make-gossip Image ListOfGossip)
(define-struct gossip [who nexts])
Programming with revised rumor mills

;; A ListOfGossip is either
;; - '()
;; - (cons Gossip ListOfGossip)

;; A Gossip is
;; (make-gossip Image ListOfGossip)
(define-struct gossip [who nexts])
Programming with revised rumor mills

;; A ListOfGossip is either
;;   - '()
;;   - (cons Gossip ListOfGossip)

;; A Gossip is
;; (make-gossip Image ListOfGossip)
(define-struct gossip [who nexts])
Programming with revised rumor mills

;;; A ListOfGossip is either
;;;  - '()
;;;  - (cons Gossip ListOfGossip)

;;; A Gossip is
;;;  (make-gossip Image ListOfGossip)
(define-struct gossip [who nexts])
(define (fun-for-log l)
  (cond [[(empty? l) ...]
         [(cons? l)
          (... (fun-for-gossip (first l)) ...
               (fun-for-log (rest l)) ...)]]))

(define (fun-for-gossip g)
  (... (gossip-who g) ...
       (fun-for-log (gossip-nexts g)) ...))
Examples for revised rumor mills
Examples for revised rumor mills

Design **count-gossips** which takes a gossip and returns the number of people informed by the gossip (including the starting person).
Examples for revised rumor mills

Design **count-gossips** which takes a gossip and returns the number of people informed by the gossip (including the starting person).

Design **informed?**, which takes a person image and a gossip and determines whether the person is part of the rumor mill.
Examples for revised rumor mills

Design **count-gossips** which takes a gossip and returns the number of people informed by the gossip (including the starting person).

Design **informed?**, which takes a person image and a gossip and determines whether the person is part of the rumor mill.

Design **remove-person-from-gossip**, which takes a person image and a gossip and returns a gossip where the given person is uninformed.
Examples for revised rumor mills

Design **count-gossips** which takes a gossip and returns the number of people informed by the gossip (including the starting person).

Design **informed?**, which takes a person image and a gossip and determines whether the person is part of the rumor mill.

Design **remove-person-from-gossip**, 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.
Acknowledgments

This lecture incorporates material from:

Matthias Felleisen
Robert Bruce Findler
Matthew Flatt
Shriram Krishnamurthi
Marc Smith