;; Constants
;;
(define GINNY)

(define ROMILDA)

(define VINCENT)

(define PANSY)

(define DRACO)

(define CHO)
(define DOBBY
  (make-gossip Image RumorMill RumorMill))

;;; Data Definitions
;;;

;;; A RumorMill is either
;;; - '()
;;; - (make-gossip Image RumorMill RumorMill)
(define-struct gossip [who next1 next2])

#
;;; RumorMill -> ...
(define (fun-for-rumor-mill rm)
  (cond [[(empty? rm) ...]
    [(gossip? rm)
      (fun-for-rumor-mill (gossip-next1 rm))
      (fun-for-rumor-mill (gossip-next2 rm)))]))

;;; An example rumor mill for Ginny to hear about Harry's supposed hypogriff
;;; tattoo. This could be defined using the images directly, as in the slides
;;; and the starter file. We switch to using constants to make it easier to type.

(define GINNY-MILL
  (make-gossip GINNY '() '()))

(define ROMILDA-MILL
  (make-gossip ROMILDA '() GINNY-MILL))
(define VINCENT-MILL
  (make-gossip VINCENT '() '()))

(define DRACO-MILL
  (make-gossip DRACO ROMILDA-MILL VINCENT-MILL))

(define CHO-MILL
  (make-gossip CHO '() '()))

(define PANSY-MILL
  (make-gossip PANSY CHO-MILL DRACO-MILL))

;;
;; Functions
;;

;; A ListOfImageOrFalse is either
;; - ListOfImage
;; - #false

;; Any -> Boolean
;; Return #true if v is a list, whether empty or not (built-in!)
;; (define (list? v)
;;   (or (empty? v) (cons? v)))
(check-expect (list? '()) #true)
(check-expect (list? (cons 1 '())) #true)
(check-expect (list? #false) #false)

;; rumor-chain : RumorMill Image -> ListOfImageOrFalse
;; Return a list of everyone who must pass on a rumor for it to reach the
;; given person; return #false if the given person is never informed.
(define (rumor-chain rm who)
  (cond [(empty? rm) #false]
        [(gossip? rm)
         (cond [[(image=? who (gossip-who rm)) (list who)]
                [(list? (rumor-chain (gossip-next1 rm) who))
                 (cons (gossip-who rm)
                       (rumor-chain (gossip-next1 rm) who))]
                [(list? (rumor-chain (gossip-next2 rm) who))
                 (cons (gossip-who rm)
                       (rumor-chain (gossip-next2 rm) who))]
                [else #false]])))))

;; Can simplify by removing the (gossip? rm) check and moving the other
;; conditions to the outer `cond`.
(check-expect (rumor-chain PANSY–MILL GINNY)
  (cons PANSY (cons DRACO (cons ROMILDA '()))))
(check-expect (rumor-chain GINNY–MILL GINNY)
  '())
(check-expect (rumor-chain '() DRACO)
  #false)
(check-expect (rumor-chain ROMILDA–MILL DRACO)
  #false)
(check-expect (rumor-chain DRACO–MILL ROMILDA)
  (list DRACO))