Back to the rumor mill
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)))]))
Design the function `rumor-chain` that takes a person image and a rumor mill and returns a list of person images representing everyone who must pass on the rumor for it to reach the given person; return `false` if the given person is never informed.
Pipes
More pipes

A pipeline has faucets (opened or closed, straight parts (copper or lead), and branches.
More pipes

A pipeline has faucets (opened or closed, straight parts (copper or lead), and branches.
More pipes

A pipeline has faucets (opened or closed, straight parts (copper or lead), and branches.
More pipes

A pipeline has faucets (opened or closed, straight parts (copper or lead), and branches.
More pipes

A pipeline has faucets (opened or closed, straight parts (copper or lead), and branches.
More pipes

A pipeline has faucets (opened or closed, straight parts (copper or lead), and branches.

;;; A Pipeline is either:
;;;   - Boolean
;;;   - (make-straight String Pipeline)
;;;   - (make-branch Pipeline Pipeline)
(define-struct straight [kind next])
(define-struct branch [next1 next2])
Example pipelines

;; A Pipeline is either:
;; - Boolean
;; - (make-straight String Pipeline)
;; - (make-branch Pipeline Pipeline)

#false
Example pipelines

;;; A Pipeline is either:
;;;   – Boolean
;;;   – (make-straight String Pipeline)
;;;   – (make-branch Pipeline Pipeline)

#true
Example pipelines

;;; A Pipeline is either:
;;; - Boolean
;;; - (make-straight String Pipeline)
;;; - (make-branch Pipeline Pipeline)

(make-straight "copper" #false)
Example pipelines

;;; A Pipeline is either:
;;; - Boolean
;;; - (make-straight String Pipeline)
;;; - (make-branch Pipeline Pipeline)

(make-straight "copper"
 (make-straight "lead" #false)
Example pipelines

;; A Pipeline is either:
;;  - Boolean
;;  - (make-straight String Pipeline)
;;  - (make-branch Pipeline Pipeline)

(make-branch
  (make-branch (make-straight "copper" #true) #false)
  (make-branch #false #false)
Programming with pipelines

;;; A Pipeline is either:
;;;  - Boolean
;;;  - (make-straight String Pipeline)
;;;  - (make-branch Pipeline Pipeline)
Programming with pipelines

;; A Pipeline is either:
;; – Boolean
;; – (make-straight String Pipeline)
;; – (make-branch Pipeline Pipeline)
Programming with pipelines

;; A Pipeline is either:
;; – Boolean
;; – (make-straight String Pipeline)
;; – (make-branch Pipeline Pipeline)

;; Pipeline -> ...
(define (fun-for-pipeline pl)
  (cond [(boolean? pl) ...]
        [(straight? pl)
         (... (straight-kind pl)
              (fun-for-pipeline (straight-next pl)))]
        [(branch? pl)
         (... (fun-for-pipeline (branch-next1 pl))
              (fun-for-pipeline (branch-next2 pl)))]))
Programming with pipelines

;; A Pipeline is either:
;; - Boolean
;; - (make-straight String Pipeline)
;; - (make-branch Pipeline Pipeline)

(define (fun-for-pipeline pl)
  (cond
   [(boolean? pl) ...
   [(straight? pl)
    (... (straight-kind pl)
     (fun-for-pipeline (straight-next pl)))]
   [(branch? pl)
    (... (fun-for-pipeline (branch-next1 pl))
     (fun-for-pipeline (branch-next2 pl)))]))
Pipeline examples

Design the function `water-running?` that takes a pipeline and determines whether any faucets are open.
Acknowledgments

This lecture incorporates material from:

Matthias Felleisen
Robert Bruce Findler
Matthew Flatt
Shriram Krishnamurthi
Marc Smith