Lists defined inductively

\[ \text{LIST}(Z) \] is the smallest set satisfying this equation:

\[ \text{LIST}(Z) = \{ \text{'()} \} \cup \{ \text{cons} \ z \ zs \mid z \in Z, zs \in \text{LIST}(Z) \} \]

Equivalently, \( \text{LIST}(Z) \) is defined by these rules:

\[ \text{'}() \in \text{List}(Z) \]  \[ \text{(EMPTY)} \]

\[ z \in Z, zs \in \text{LIST}(Z) \]

\[ \text{(cons} \ z \ zs \text{)} \in \text{List}(Z) \]  \[ \text{(CONS)} \]
Equations and function for append

(append '() ys) == ys

(append (cons z zs) ys) == (cons z (append zs ys))

(define append (xs ys)
  (if (null? xs)
      ys
      (cons (car xs) (append (cdr xs) ys)))))
Naive list reversal

(define reverse (xs)
  (if (null? xs)
      '()
      (append (reverse (cdr xs))
               (list1 (car xs))))
)
Reversal by accumulating parameters

(define revapp (xs ys)
    (if (null? xs)
        ys
        (revapp (cdr xs)
            (cons (car xs) ys))))

(define reverse (xs) (revapp xs '()))