(define (reverse L)

(if (null? L)

null

(append (reverse (rest L)) (cons (first L) null))

)

)

I assume the following properties of the append function

1. (and (list? x) (list? y)) → (list? (append x y))
2. (append null y) = y
3. x ≠ null → (first (append x y)) = (first x)
4. (append x null) = x
5. (length (append x y)) = (+ (length x) (length y))
6. (append x (append y z)) = (append (append x y) z)

Let (nth n L) be the function that returns the nth element of the list L.

If x is a list (reverse x) returns a list such that (nth n (reverse x)) = (nth (- (+ (length x) 1) n) x)

(define (list? L)

(if (or (null? L) (pair? L))

(list? (second L))

#f))

**1.** (list? L) 🡪 (list? (reverse L))

Base Case: L = null

(list? null) = #t

(list? (reverse null))

(reverse null) = ((if (null? null)

null

(append (reverse (rest null)) (cons (first null) null))

)

)

= null

(list? (reverse null)) = (list? null) = #t

#t 🡪 #t = #t

Inductive Hypothesis: (rest L)

(list? (rest L)) 🡪 (list? (reverse (rest L))

Inductive Proof: L

L = (cons x (rest L))

We must prove (list? L) 🡪 (list? (reverse L))

Assume (list? L) = #t

(list? L) = #t 🡪 (list? (rest L)) = #t 🡪 (list? (reverse (rest L))) = #t

(reverse L) = (append (reverse (rest L)) (cons (first L) null))

(list? (reverse L))

= (list? (append (reverse (rest L)) (cons (first L) null)))

**PROPERTY 1 of APPEND is 🡪 not 🡨 !!!**

(list? (append x y)) 🡪 (and (list? x) (list? y))

= (list? (cons (first L) null))

= (list? null) = #t

(list? L) 🡪 (list? (reverse L))

**2.** (length (reverse x)) = (length x)

Base Case: x = null

(length (reverse null)) = (length null) = 0

Inductive Hypothesis: x = (rest L)

(length (reverse (rest L))) = (length (rest L)) = N

Inductive Proof: x = L = (cons z (rest L))

(length (reverse L))

(reverse L) = (append (reverse (rest L)) (cons (first L) null))

(length (append (reverse (rest L)) (cons (first L) null))) =

(+ length (reverse (rest L)) (length (cons (first L) null)))

(length (cons (first L) null) = 1

(length (reverse (rest L)) = N

= (+ N 1) = N + 1

(length L) = (length (cons z (rest L))) = (+ N 1) = N + 1

N + 1 = N + 1

**3.** (reverse (append x y)) = (append (reverse y) (reverse x))

Base Case: x = null

(reverse (append null y)) = (reverse y)

(append (reverse x) (reverse null)) = (append (reverse null) (reverse y)) = (reverse y)

Inductive Hypothesis: (rest x)

(reverse (append (rest x) y)) = (append (reverse y) (reverse (rest x)))

Inductive Proof: x

(append x y)

= (cons (first x) (append (rest x) y))

(reverse (cons (first x) (append (rest x) y)))

L = (cons (first x) (append (rest x) y))

(rest L) = (append (rest x) y)

(first L) = (first x)

(reverse L) = (append (reverse (rest L)) (cons (first x) null))

= (append (reverse (append (rest x) y)) (cons (first x) null))

(reverse (append (rest x) y)) = (append (reverse y) (reverse (rest x)))

= (append (append (reverse y) (reverse (rest x))) (cons (first x) null))

(append (append A B) C) = (append A (append B C))

A = (reverse y)

B = (reverse (rest x))

C = (cons (first x) null)

(append B C) = (append (reverse (rest x)) (cons (first x) null)) = (reverse x)

= (append (reverse y) (reverse x))

**4.** (reverse (reverse x)) = x

Base Case: x = null

(reverse (reverse null)) = (reverse null) = null

Inductive Hypothesis: (rest x)

(reverse (reverse (rest x))) = (rest x)

Inductive Proof: x

(reverse (reverse x))

(reverse x) = (append (reverse (rest x)) (cons (first x) null))

(reverse (append (reverse (rest x)) (cons (first x) null)))

A = (reverse (rest x))

B = (cons (first x) null))

(reverse (append A B)) = (append (reverse B) (reverse A))

(append (reverse (cons (first x) null)) (reverse (reverse (rest x))))

(reverse (cons (first x) null)) = (cons (first x) null)

(append (cons (first x) null) (reverse (reverse (rest x))))

(reverse (reverse (rest x))) = (rest x)

(append (cons (first x) null) (rest x)) = (cons (first x) (rest x)) = x

**5.** (nth n (reverse L)) = (nth (- (+ (length L) 1) n) L)