```
(defunc rev (x)
  :ic (listp x)
  :oc (listp (rev x))
  (if (endp x)
      x
    (app (rev (rest x)) (list (first x)))))
```

```
(rev '(a b c d))
```

```
(app (rev '(b c d)) '(a)))     3 cons
(app (rev '(c d)) '(b))        2
(app (rev '(d)) '(c))          1
(app (rev nil) '(d))           0
```

$$\sum_{i=0}^{n-1} i \equiv \frac{(n-1)*n}{2}$$

$$\mathcal{O}(n^2)$$

```
(defunc rev-t (x acc)
  :ic (and (listp x) (listp acc))
  :oc (listp (rev-t x acc))
  (if (endp x)
      acc
      (rev-t (rest x) (cons (first x) acc))))

(defunc rev* (x)
  :ic (listp x)
  :oc (listp (rev* x))
  (rev-t x nil))
```

Φ_{rev} : (listp x) => (rev* x) = (rev x)

C1. (listp x)

LHS

(rev* x)

= { Def rev* }

(rev-t x nil)

= { L1 |((x x) (acc nil)) }

(app (rev x) nil)

= { phi_app-x-nil }

(rev x) QED

List x	acc	(rev-t x acc)	(rev x)
'(a b c d)	nil	'(d c b a)	'(d c b a)
'(b c d)	'(a)	'(d c b a)	'(d c b)
'(c d)	'(b a)	'(d c b a)	'(d c)
'(d)	'(c b a)	'(d c b a)	'(d)
nil	'(d c b a)	(d c b a)	nil

L1 (listp x) ∧ (listp acc) ⇒ (rev-t x acc) = (app (rev x) acc)

Prove L1 using I.S. of listp 2) $(consp\ x) \wedge \phi|_{((x\ (rest\ x)))} \Rightarrow \phi$

1) $\neg(consp\ x) \Rightarrow \phi$

Obligation 1:

C1. (listp x)
C2. (listp acc)
C3. (endp x)

(rev-t x acc)
= {Def rev-t, C3}
acc
= {Def app, C3}
(app x acc)
= {Def rev, C3}
(app (rev x) acc)

Obligation 2

C1. (listp x)
C2. (listp acc)
C3. $\neg$(endp x)
C4. (listp (rest x)) $\wedge$ (listp acc) $\Rightarrow$ ((rev-t (rest x) acc) = (app (rev (rest x)) acc))
C5. (listp (rest x)) {C3, C1, Def listp}
C6. ((rev-t (rest x) acc) = (app (rev (rest x)) acc) {MP, C5, C4, C2}

(rev-t x acc)

= {Def rev-t, c3}

(rev-t (rest x) (cons (first x) acc)) ⟵ Darn.

I.S. for rev-t:

1) $\neg((\text{listp } x) \wedge (\text{listp } acc)) \Rightarrow L1$ ✓

2) $(\text{listp } x) \wedge (\text{listp } acc) \wedge (\text{endp } x) \Rightarrow L1$

3) $(\text{listp } x) \wedge (\text{listp } acc) \wedge \neg(\text{endp } x) \wedge L1 \big|_{((x\ (\text{rest } x))\ (acc\ (\text{cons } (\text{first } x)\ acc)))} \Rightarrow L1$

C1. $\neg((\text{listp } x) \wedge (\text{listp } acc))$

C2. $(\text{listp } x) \wedge (\text{listp } acc)$

C3. nil {C1, C2}

Obligation 3

C1. (listp x)
C2. (listp acc)
C3. ¬(endp x)
C4. (listp (rest x)) ∧ (listp $\overset{\beta}{\text{(cons (first x) acc)}}$)
⇒ $\underbrace{\text{((rev-t (rest x) (cons (first x) acc)) = (app (rev (rest x)) (cons (first x) acc)))}}_{\alpha}$

C5. (listp (rest x)) {C1, C3, Def listp}
C6. (listp β) {Def listp, C2}
C7. α {C5, C6, C4, MP}

(rev-t x acc)
= {Def rev-t, C3}
(rev-t (rest x) (cons (first x) acc))
= {C7}
(app (rev (rest x)) (cons (first x) acc))

app?

= {Def app, first-rest axiom, def endp}
(app (rev (rest x)) (app (list (first x)) acc))
= {Assoc of app}
(app (app (rev (rest x)) (list (first x))) acc)
= {Def rev}

(app (rev x) acc)

QED

L1: (natp n) ∧ (natp acc) ⇒ (sumn-t n acc) = (+ acc (sumn n))

1) ¬((natp n) ∧ (natp acc)) ⇒ L1

2) (natp n) ∧ (natp acc) ∧ (n=0) ⇒ L1

3) (natp n) ∧ (natp acc) ∧ (n≠0) ∧ $L1|_{((n\ (-\ n\ 1))\ (acc\ (+\ n\ acc)))}$ ⇒ L1

(natp n) ⇒ (sumn* n) = (sumn n)

C1 (natp n)

(sumn* n)

= {Def sumn*}

(sumn-t n 0)

= $\{L1|_{((acc\ 0))}\}$

(+ 0 (sumn n))

= {Arith}

= (sumn n)

Obligation 1:

C1 IC

C2. ¬IC

C3. nil {C1, C2, PL}

Obligation 2

C1. (natp n)

C2. (natp acc)

C3. (n = 0)

LHS

(sumn-t n acc)

= {C3, Def sumn-t}

acc

RHS

(+ acc (sumn n))

= {Def sumn, C3}

(+ acc 0)

= {Arith}

acc

```
(defunc sumn-t (n acc)
  :ic (and (natp n) (natp acc))
  :oc (natp (sumn-t n acc))
  (if (equal n 0)
      acc
      (sumn-t (- n 1) (+ n acc))))
```

Obligation 3

C1. (natp n)
C2. (natp acc)
C3. (n ≠ 0)
C4. L1|((n (- n 1)) (acc (+ n acc)))

- - - - - - - - -

C5. (natp (- n 1)) {C3, Arith, C1}
C6. (natp (+ n acc)) {C2, C1, Arith}
C7. (sumn-t (- n 1) (+ n acc))
  = (+ (+ n acc) (sumn (- n 1)))
  {C5, C6, C4, MP}

LHS: (sumn-t n acc)

$\equiv$ {Def sumn-t, C3}

(sumn-t (- n 1) (+ n acc))

$\equiv$ {C7}

(+ (+ n acc) (sumn-t n))

RHS

(+ acc (sumn n))

$\equiv$ {Def sumn, C3}

(+ acc (+ n (sumn (- n 1))))

$\doteq$ {Arith}

(+ (+ n acc) (sumn (- n 1)))

LHS = RHS

QED

x	y	acc	(add-lists x y)	(add-lists-t x y acc)
'(1 2)	'(3 4 5)	nil	'(4 6 5)	'(4 6 5)
'(2)	'(4 5)	'(4)	'(6 5)	" "
nil	'(5)	'(6 4)	'(5)	" "
				" "
				" "

L1: $(\text{lorp } x) \wedge (\text{lorp } y) \wedge (\text{lorp acc}) \Rightarrow$ (add-lists-t x y acc)

= (app (rev acc) (add-lists x y))

$\varphi_{add\text{-}lists^*}$

C1. (lorp x)

C2. (lorp y)

(add-lists* x y)

≡ {Def add-lists*}

(add-lists-t x y nil)

≡ {L1}

~~(app (rev nil)~~(add-lists x y))

≡ {Def ~~rev~~ rev, def app, Def endp}

(add-lists x y)

Obligation 2 (from add-lists-t) ← alt or al

C1. (lorp x)
C2. (lorp y)
C3. (lorp acc)
C4. (endp x)

(add-lists-t x y acc)
≡ {Def alt, C4}
(app (rev acc) y)
≡ {Def al, C4}
(app (rev acc (al x y))

(al x y)

Obligation 4

C1. (lorp x)
C2. (lorp y)
C3. (lorp acc)
C4 ¬(endp x)
C5. ¬(endp y)
C6 L1 ((x (rest x)) (y (rest y)) (acc (cons (+ (first x) (first y)) acc)))

α = (+ (first x) (first y))

C7. (lorp (rest x) { — }
C8. (lorp (rest y) { — }
C9. (lorp α { — }

C10 (alt (rest x)(rest y) α) = (app (rev α)(al (rest x)(rest y)))
{C7, C8, C9, C6, MP}

(alt x y acc)
= {Def}
(alt (rest x)(rest y) α)
= {C10}
(app (rev α) (al (rest x)(rest y)))
= {Def rev, cons-p axiom}
(app (app (rev acc)(list (+ (first x)(first y)))) (al (rest x)(rest y)))
= {Assoc of app, Def app, Def list}
(app (rev acc) (cons (+ (first x)(first y)) (al (rest x)(rest y))))
= {Def al}
(app (rev acc)(al x y))

"QED"

```
(defun less* (e lr)
  :ic (and (rationalp e) (lorp lr))
  :oc (lorp (less* e lr))
  (rev (less-t e lr nil)))
```

```
(defun less-t (e lr acc)
  :ic (and (rationalp e) (lorp lr) (lorp acc))
  :oc (lorp (less-t e lr acc))
  (cond ((endp lr) acc)
        ((< (first lr) e) (less-t e (rest lr) (cons (first lr) acc)))
        (t                (less-t e (rest lr) acc))))
```

```
(defun less (e lr)
  :ic (and (rationalp e) (lorp lr))
  :oc (lorp (less e lr))
  (cond ((endp lr) lr)
        ((< (first lr) e) (cons (first lr)
                                (less e (rest lr))))
        (t   (less e (rest lr)))))
```

C1. (lorp lr)
C2. (rationalp e)

(less* e lr)
= {Def less*}
(rev (less-t e lr nil))
= {L1}
(rev (app (rev (less e lr)) nil))
= {$\phi_{app.X\_nil}$}
(rev (rev (less e lr)))
= {ϕ_{rev_rev}}
(less e lr) QED

Just this for time

I.S. for less-t

① ¬IC ⇒ L1
② IC ∧ (endp lr) ⇒ L1
③ IC ∧ ¬(endp lr) ∧ L1$|_{((lr\ (rest\ lr))(acc\ (cons\ (first\ lr)\ acc)))}$ ∧ (< (first lr) e) ⇒ L1
④ IC ∧ ¬(endp lr) ∧ ¬(< (first lr) e) ∧ L1$|_{((lr\ (rest\ lr)))}$ ⇒ L1

Obligation 3

C1. (lorp lr)
C2. (lorp acc)
C3. (rationalp e)
C4. (< (first lr) e)
C5. ¬(endp lr)
C6. L1$|_{((lr\ (rest\ lr))(acc\ (cons\ (first\ lr)\ acc)))}$
C7. (lorp (rest lr)) {C1, C5, Def lorp}

C8. (lorp (cons (first lr) acc)) {Def lorp, C2, C5}

C9. (less-te (rest lr) (cons (first lr) acc)) = (app (rev (less e (rest lr))) (cons (first lr) acc))

{C7, C8, C6, MP}

LHS

(less-t e lr acc)

≡ {Def less-t, C5, C4}

(less-t e (rest lr) (cons (first lr) acc)))

= {C9}

(app (rev (less e (rest lr))) (cons (first lr) acc))

(app (rev (less e (rest lr))) (cons (first lr) acc))

QED

RHS

(app (rev (less e lr)) acc)

≡ {Def less, C5, C4}

(app (rev (cons (first lr) (less e (rest lr)))) acc)

≡ {Def rev, consp axiom, Def endp}

(app (app (rev (less e (rest lr))) (list (first lr))) acc)

≡ {Assoc of app}

(app (rev (less e (rest lr))) (app (list (first lr)) acc))

≡ {Def app (twice), Def endp, consp axiom}

```
(defunc ssort (l)
  :ic (lorp l)
  :oc (lorp (ssort l))
  (if (endp l)
      l
      (cons (min-l l) (ssort (del (min-l l) l)))))
```

```
(defunc ssort* (l)
  :ic (lorp l)
  :oc (lorp (ssort* l))
  (ssort-t l nil))
```

```
(defunc ssort-t (l acc)
  :ic (and (lorp l) (lorp acc))
  :oc (lorp (ssort-t l acc))
  (if (endp l)
      (rev acc)
      (ssort-t (del [e] l) (cons e acc))))
```

e = ~~(max-l l)~~
(min-l l)

L1. (lorp l) ∧ (lorp acc) ⇒ (ssort-t l acc) = (app (rev acc) (ssort l))

Proof

C1. (lorp l)

(ssort* l)
= {Def ssort*}
(ssort-t l ~~acc~~ nil)
= {L1}
(app (rev nil) (ssort l))
= {Def rev, Def endp}
(app nil (ssort l))
= {Def app}
(ssort l)

QED

I.S. for ssort-t

1) ¬((lorp l) ∧ (lorp acc)) ⇒ L1

2) (lorp l) ∧ (lorp acc) ∧ (endp l) ⇒ L1

3) (lorp l) ∧ (lorp acc) ∧ ¬(endp l) ∧ L1|((l (del (min-l l) l))
(acc (cons (min-l l) acc))) ⇒ L1

m

L1. (lorp l) ∧ (lorp acc) ⇒ (ssort-t l acc)
≡ (app (rev acc) (ssort l))

Obligation 1

C1. ¬((lorp l) ∧ (lorp acc))
C2. ((lorp l) ∧ (lorp acc))
C3. nil {C1, C2, PL}

Obligation 2

C1. (lorp l)
C2. (lorp acc)
C3 (endp l)

(ssort-t l acc)
= {Def ssort-t, C3}
(rev acc)
≡ {App-x-nil theorem}
(app (rev acc) nil)
≡ {C3}
(app (rev acc) l)
≡ {Def ssort, C3}
(app (rev acc) (ssort l))

Obligation 3

C1. (lorp l)
C2. (lorp acc)
C3. ¬(endp l)
C4. L1 |((l (del m l))
(acc (cons m acc)))

- - - - - - - - - - - - - -

C5 (lorp (del m l)) {Del thm, C1}
C6. (lorp (cons m acc)) {Contract theorem min-l, Def lorp, C2}
C7. (ssort-t (del m l)(cons m acc)) = (app (rev (cons m acc))(ssort (del m l)))

RHS (app (rev acc)(ssort l))
= { Def ssort, C3 }
(app (rev acc) (cons m (ssort (del m l))))

≡ {Def of app, Def endp}
(app (rev acc) (cons m (app nil (ssort (del m l)))))

≡ {Def app, Def list, Def endp, consp axiom}
(app (rev acc) (app (list m) (ssort (del m l))))

≡ {Assoc of app}
(app (app (rev acc) (list m)) (ssort (del m l)))

≡ {Def rev, first-rest axiom, Def endp, consp axiom}
(app (rev (cons m acc)) (ssort (del m l)))

≡ {C7}
(ssort-t (del (min-l l) l) (cons (min-l l) acc))

≡ {Def ssort-t, C3}
(ssort-t l acc)

QED

hash map
binary tree
List
Array list
set
stack
queue
deque
heap
priority queue
red-black tree

graph

$$\frac{max + min}{2}$$

stack: push, pop, peek

(+ (* 3 4) 7)