Scheme's Equality Operators:

(= a b) compares numbers and is unreliable for other comparisons.

(equal? a b) compares structures:

(equal? '(2 3) (cdr '(1 2 3))) => #t

but (equal? 4 (+ 2 2)) => #f

eq? and eqv? compare memory locations rather than structures.

(eql? a b) and (eqv? a b) both return #t if a and b are lists stored at the same location.

If a and b are numbers

(eqv? a b) => (= a b)

(eql? a b) is implementation-dependent.

(eqv? (/ 10 3) (/ 20 6)) => #t, since eqv? is the same as = for numbers.

(eq? (/10 3) (/ 20 6)) => #f on our system.

Moral:

- Use = for numeric comparisons
- Use equal? if you want to know if two lists are structurally identical.
- Use eqv? if you want to know if two lists are stored at the same location.

What does this function do?

```
((define a (lambda (v1 v2)
        (cond
          [(null? v1) v2]
          [else (cons (car v1) (a (cdr v1) v2))])))
```

Examples on flat lists:

lat = list of atoms

(nth n lat) returns the nth element of lat
(rember a lat) removes the first occurrence of a from lat
(rember-all a lat) removes every occurrence of a from lat
(rember-2 a lat) removes the second occurrence of a from lat
(index a lat) returns the 0-based index of a in lat, or -1 if a is not in lat.
(remove-numbers lat) removes all of the numbers from lat