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? 2.5 (/ 10 4)) => #f

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

(eq? 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 in Dr. Racket

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.
- Use eq? if you are only comparing atoms.

What does this function do? You can assume it will be called with arguments for v1 and v2 that are lists.

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

Examples on flat lists: we'll write these in class

lat = list of atoms

(same? lat1 lat2) returns #t if the lats have the same atoms in the same order

The rest of these aren't especially about equality

(rev lat) reverses lat.
(remove-numbers lat) removes all of the numbers from lat
(remove-stuff pred lat) removes any element from lat that satisfies pred.
(remover pred) returns a procedure that takes a lat and removes elements that satisfy pred