CSCI 301, Lab # 3
Winter 2017

Due: Your program, named `lab03.rkt`, must be submitted to Canvas before midnight, Tuesday, Jan 31.

Background: It's pretty easy to write a procedure to return the shorter of two lists using only `car`, `cdr` and `null?`, if we first compute the length of each. For example, we can use:

```
(define length
  (lambda (ls)
    (if (null? ls)
        0
        (+ 1 (length (cdr ls))))))

(define shorter
  (lambda (ls1 ls2)
    (if (<= (length ls1) (length ls2))
        ls1
        ls2)))
```

Note: `length` is actually builtin to DrRacket, but we show a definition here to make a point.

However, this is very inefficient if the lists are vastly different in length. For example, if we had one list with length over a billion, and the other with only 5 things in it, this procedure would traverse *both* lists to the end, only to return the second list. Clearly we could have found out which was shorter by only examining the first few of the items in each list.

Programming: Rewrite the `shorter` procedure, so that it returns the same value as the procedure above, but does not use `length`, and does not traverse both of the lists to the end when one is longer than the other. You can write as many procedures as you like, but make sure:

- you use only `car`, `cdr` and `null?` as builtin procedures,
- your procedure returns the shorter input list (or the first if they are the same length),
- your procedure does not copy any lists,
- traverses the minimum number of items to determine which list is shorter, and
- uses functional style, no assignment statements!