Exercise 3.16.

Ben Bitdiddle decides to write a procedure to count the number of pairs in any list structure. "It's easy," he reasons. "The number of pairs in any structure is the number in the `car` plus the number in the `cdr` plus one more to count the current pair." So Ben writes the following procedure:

```
(define (count-pairs x)
  (if (not (pair? x))
      0
      (+ (count-pairs (car x))
         (count-pairs (cdr x))
         1)))
```

Show that this procedure is not correct. In particular, draw box-and-pointer diagrams representing list structures made up of exactly three pairs for which Ben's procedure would return 3; return 4; return 7; never return at all.

Answer.

This procedure is not correct, for it fails to address the case of cycle as well as the case where both the `car` and the `cdr` of a pair can access the same subsequent pair. Figure 1 shows the case where Ben's

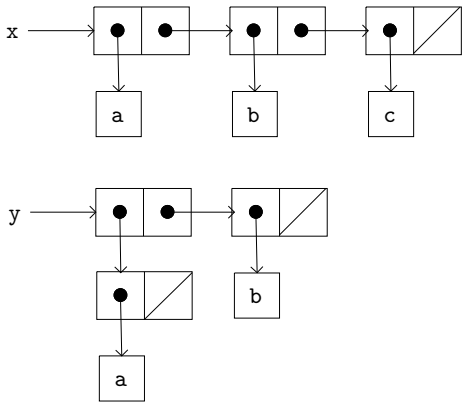

Figure 1. List structures consist of 3 pairs for which Ben's procedure would return 3.

procedure would correctly return 3 for list structures made up of exactly three pairs.

However, this illusion fall away when the lists admit sharing, Figure 2, fiigure 3 and figure 4 illustrate the case where list structures made up of exactly three pairs for which Ben's procedure would return 4; return 7 and never return at all respectively.

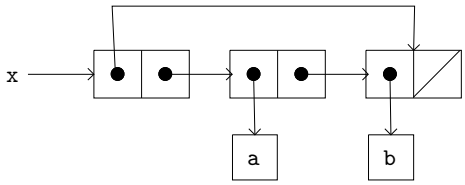

Figure 2. List structure consist of 3 pairs for which Ben's procedure would return 4.

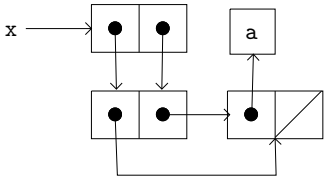

Figure 3. List structure consist of 3 pairs for which Ben's procedure would return 7.

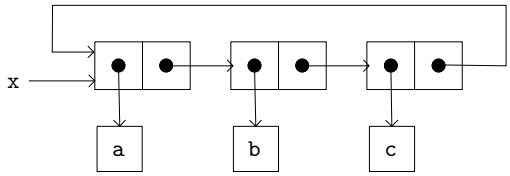

Figure 4. List structures consist of 3 pairs for which Ben's procedure would never return.