Exercise 3.14.

The following procedure is quite useful, although obscure:

```
(define (mystery x)
  (define (loop x y)
    (if (null? x)
        y
        (let ((temp (cdr x)))
          (set-cdr! x y)
          (loop temp x))))
  (loop x '()))
```

`Loop` uses the "temporary" variable `temp` to hold the old value of the `cdr` of `x`, since the `set-cdr!` on the next line destroys the `cdr`. Explain what `mystery` does in general. Suppose `v` is defined by `(define v (list 'a 'b 'c 'd))`. Draw the box-and-pointer diagram that represents the list to which `v` is bound. Suppose that we now evaluate `(define w (mystery v))`. Draw box-and-pointer diagrams that show the structures `v` and `w` after evaluating this expression. What would be printed as the values of `v` and `w`?

Answer.

The procedure `mystery` takes as its argument a list `L` and produce another list which is in the inverse order of `L`. Figure 1 shows the box-and-pointer diagram that represent the list to which `v` is bound.

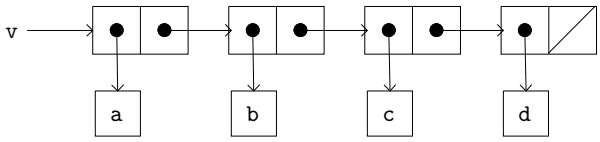

Figure 1. List `v: (a b c d)`.

Figure 2 shows the structure v and w after evaluating the expression `(define w (mystery v))`. When

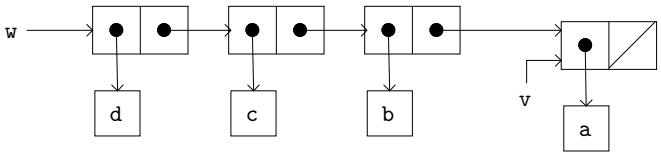

Figure 2. The structure `v` and `w` after evaluating the expression `(define w (mystery v))`.

we query the interpreter the value of `v` and `w`, it would response by printing:

```
v
(a)

w
(d c b a)
```