

### Exercise 3.12.

The following procedure for appending lists was introduced in section 2.2.1:

```
(define (append x y)
  (if (null? x)
      y
      (cons (car x) (append (cdr x) y))))
```

`Append` forms a new list by successively `consing` the elements of `x` onto `y`. The procedure `append!` is similar to `append`, but it is a mutator rather than a constructor. It appends the lists by splicing them together, modifying the final pair of `x` so that its `cdr` is now `y`. (It is an error to call `append!` with an empty `x`.)

```
(define (append! x y)
  (set-cdr! (last-pair x) y)
  x)
```

Here `last-pair` is a procedure that returns the last pair in its argument:

```
(define (last-pair x)
  (if (null? (cdr x))
      x
      (last-pair (cdr x))))
```

Consider the interaction

```
(define x (list 'a 'b))

(define y (list 'c 'd))

(define z (append x y))

z
(a b c d)

(cdr x)
<response>

(define w (append! x y))

w
(a b c d)

(cdr x)
<response>
```

What are the missing `<response>`s? Draw box-and-pointer diagrams to explain your answer.


### Answer.

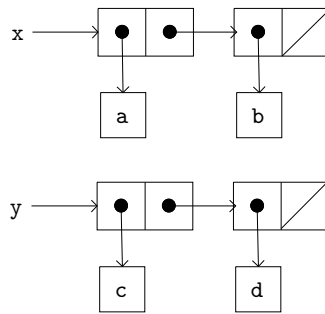
Figure 1 shows the box-and-pointer diagrams before appending the lists. Evaluating the expression `(define z (append x y))` extend the list `x` by `y` to set up a new list, namely `z`. Observe that both list `x` and `y` remain invariant in the process. The result of the operation is shown in figure 2. Thus, when we send the query `(cdr x)` to the interpreter, it would response us with

(b)

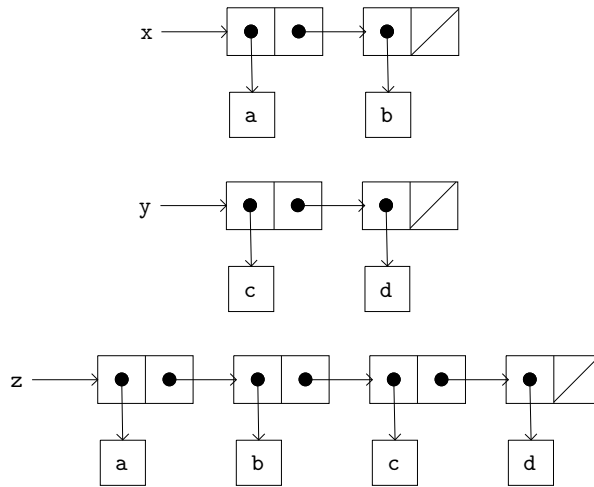
The expression `(define w (append! x y))` however, perform rather different operations on the lists. It constructed the list `w` by concatenating list `x` and `y` together. Figure 3 illustrate the operations in the process in a rather intuitive way. Hence, the interpreter would response the query `(cdr x)` with

---

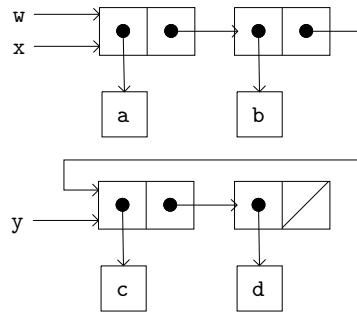
\*. Creative Commons  2013, Lawrence X. Amlord (颜世敏, aka 颜序).



**Figure 1.** Lists x: (a b) and y: (c d).



**Figure 2.** Effect of `(define z (append x y))` on the list in figure 1.



**Figure 3.** Effect of `(define w (append! x y))` on the list in figure 1.

(b c d)