Exercise 2.4.

Here is an alternative procedural representation of pairs. For this representation, verify that `(car (cons x y))` yields `x` for any objects `x` and `y`.

```
(define (cons x y)
  (lambda (m) (m x y)))

(define (car z)
  (z (lambda (p q) p)))
```

What is the corresponding definition of `cdr`? (Hint: To verify that this works, make use of the substitution model of section 1.1.5.)

Answer.

By looking at its definition, we know that value returned by `(cons x y)` is a procedure which takes one argument and applies it to `x` and `y`. `(car z)` is accordingly defined to apply `z` to a procedure which itself takes an order pair and returns its latter element. Hence, using the substitution model, the process generated in evaluating `(car (cons x y))` would be:

```
(car (cons x y))
((cons x y) (lambda (p q) p))
((lambda (m) (m x y)) (lambda (p q) p))
((lambda (p q) p) x y)
x
```

Therefore, we have shown that `(car (cons x y))` yields `x` for any objects `x` and `y`.

The way `(car z)` being evaluating may surely inspires us the corresponding definition of `cdr`:

```
(define (cdr z)
  (z (lambda (p q) q)))
```

Similarly, the validation of `cdr` can also be checked by means of substitution model:

```
(cdr (cons x y))
((cons x y) (lambda (p q) q))
((lambda (m) (m x y)) (lambda (p q) q))
((lambda (p q) q) x y)
y
```