Exercise 3.40.

Give all the possible values of `x` that can result from executing

```
(define x 10)

(parallel-execute (lambda () (set! x (* x x)))
                  (lambda () (set! x (* x x x))))
```

Which of these possiblities remain if we instead use serialized procedures:

```
(define x 10)

(define s (make-serializer))

(parallel-execute (s (lambda () (set! x (* x x))))
                  (s (lambda () (set! x (* x x x)))))
```

Answer.

Executing the procedures

```
(define x 10)

(parallel-execute (lambda () (set! x (* x x)))
                  (lambda () (set! x (* x x x))))
```

creates two concurrent processes—P_1, which sets `x` to the square of itself, and P_2, which computes and set `x` to its cube. After the execution, `x` will be left with one of five possible values, dued to the interleaving of the events of P_1 and P_2:

1000000:	P_1 sets `x` to 100 and then P_2 sets `x` to its cube—1000000 or, P_2 sets `x` to its cube 1000 and then P_1 sets `x` to the square of it, also 1000000.
1000:	P_2 accesses `x` (three times), then P_1 sets `x` to 100, then P_2 sets `x`.
100:	P_1 accesses `x` (twice), then P_2 sets `x` to 1000, then P_1 sets `x`.

If we instead execute serialized procedures:

```
(define x 10)

(define s (make-serializer))

(parallel-execute (s (lambda () (set! x (* x x))))
                  (s (lambda () (set! x (* x x x)))))
```

only the value 1000000 remain.