Exercise 3.68.

Louis Reasoner thinks that building a stream of pairs from three parts is unnecessarily complicated. Instead of separating the pair (S_0, T_0) from the rest of the pairs in the first row, he proposes to work with the whole first row, as follows:

```
(define (pairs s t)
  (interleave
   (stream-map (lambda (x) (list (stream-car s) x))
               t)
   (pairs (stream-cdr s) (stream-cdr t))))
```

Does this work? Consider what happens if we evaluate `(pairs integers integers)` using Louis's definition of pairs.

Answer.

No, it doesn't work. Using Louis's `pairs` procedure, the interpreter will immediately be overwhelmed by the sheer volume of process generated by the expression:

```
(pairs integers integers)
;Aborting!: maximum recursion depth exceeded
```

To find out the undoing, we'd better trace the behaviors of Louis's `pairs` procedure for a few steps. Since no assignment to local state variable is introduce into our stream paradigm, we can adopt the substitution model as a tool for inspection.

We start by applying the compound procedure `pairs` to the two same arguments `integers`:

```
(pairs integers integers)
```

According to the substitution model, we then evaluate the body of the procedure `pairs` with both `s` and `t` replaced by `integers`:

```
(interleave
 (stream-map (lambda (x) (list (stream-car integers) x))
             integers)
 (pairs (stream-cdr integers) (stream-cdr integers)))
```

It takes almost no effort to evaluate the first two subexpressions in this combination. However, it is the third one that traps the `pairs` procedure into an infinite recursion and overwhelms the interpreter:

```
(interleave
 (stream-map (lambda (x) (list (stream-car integers) x))
             integers)
 (interleave
  (stream-map (lambda (x) (list (stream-car integers) x))
              (stream-cdr integers))
  (pairs (stream-cdr (stream-cdr integers))
         (stream-cdr (stream-cdr integers)))))

(interleave
 (stream-map (lambda (x) (list (stream-car integers) x))
             integers)
 (interleave
  (stream-map (lambda (x) (list (stream-car integers) x))
              (stream-cdr integers))
  (interleave
   (stream-map (lambda (x) (list (stream-car integers) x))
               (stream-cdr (stream-cdr integers)))
```

*. Creative Commons  2013, Lawrence X. Amlord (颜世敏, aka 颜序).
Email address: informlarry@gmail.com

```
   (pairs (stream-cdr (stream-cdr (stream-cdr integers)))
          (stream-cdr (stream-cdr (stream-cdr integers)))))))
```

...