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 be immediately overwhelmed by the sheer volume of process generated by the expression:

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

For Louis simply implemented `pairs` as a procedure application and Scheme uses applicative-order evaluation, which indicates the evaluator should obtain the value of `integers` before `pairs` is applied. This would exhaust the evaluator for the value of `integers` is an infinite set. The original `pairs` procedures in the text works because it is defined by the special form `cons-stream`, in that we can generate part of the answer given only partial information about the argument and successively proceed the evaluation.

Email address: informlarry@gmail.com