Exercise 3.44

Problem statement

Consider the problem of transferring an amount from one account to another. Ben Bitdiddle claims that this can be accomplished with the following procedure, even if there are multiple people concurrently transferring money among multiple accounts, using any account mechanism that serializes deposit and withdrawal transactions, for example, the version of make-account in the text above.

```
(define
  (transfer from-account to-account amount)
  ((from-account 'withdraw) amount)
  ((to-account 'deposit) amount))
```

Louis Reasoner claims that there is a problem here, and that we need to use a more sophisticated method, such as the one required for dealing with the exchange problem. Is Louis right? If not, what is the essential difference between the transfer problem and the exchange problem? (You should assume that the balance in `from-account` is at least amount.)

Solution

First let's recall the exchange procedure

```
(define (exchange account1 account2)
  (let ((difference (- (account1 'balance)
                       (account2 'balance))))
    ((account1 'withdraw) difference)
    ((account2 'deposit) difference)))
```

Notice that the key difference between `exchange` and `transfer` is that the former depends on the difference in balance *between the two accounts*. This means that if *any* of the two balances changes before the exchange operation has completed, the end result might have some issues — see Excercise 3.43. Using the `transfer` function we only care about the amount that is going to be transfered — which is a value that doesn't depend on any of the account balances. The withdraw and deposit operations can even be executed in reverse order, as long as both procedures get executed once and exactly once the final effect will be correct.

As always, Louis is wrong.