Guidelines for using

set! and set-structure!

When should you use set! to solve a problem, and when should you use set-structure!? These guidelines should help you choose:

1. **If you're working with a list of structs, and the change is confined to one struct, use set-structure!**
   We did this for the `deposit` function. Although you could solve `deposit` by using set! and writing a helper that rebuilds the entire list, why go to all that work? set-structure! is a better choice here.

2. **If you're working with a list of structs, and the change will affect the list as a whole, you must use set!**
   We did this for the `remove-account` function. We wanted to remove an entire account, which would modify the list of accounts overall. set-structure! works only on structs, not on variables (such as a ListOfStruct) in general.

3. **If you're working with a single struct which is shared, and you want changes to the struct to be visible everywhere the struct is shared, use set-structure!**
   We did this when we used set-account-balance! on an account that was shared by two customers, Mcust and Pcust. Changes made to the account structure using set-account-balance! were visible everywhere.

4. **If you're working with a single struct which is shared, and you want changes to the struct to be visible in only one place, use set!**
   Say that our two customers, Maria and Phil, break up and no longer want to share the account defined by MPacct. We could change the definition of one of the customers by creating a new customer with set!:
   ```
   (set! Mcust (make-customer "Maria" 5554321 (make-account 123 200)))
   ```

   This replaces the definition for Mcust with a whole new customer structure (and a whole new account structure). The old definition for Mcust is no longer available.