Page 3:

* Used for manipulating subexpressions:
  + get, set and modify a substructure of a given structure

Page 7:

* Data Store b a = Store { peek :: b -> a, pos :: b }
  + A value of type Store b a = collection of values of type a
  + Each element of the collection is indexed by a value of b (peek)
  + One special “selected” location (pos)

Page 10:

Note that the original collection ends up in the originally selected cell.

Page 12:

phoneLens is a getter/setter for the phone number

Page 42:

3 functional multireferences:

From A to its children of types A, B, C

From B to its children of types A, B, C

From C to its children of types A, B, C

Page 48:

We first provide the functional multireference that defines the reference to the children of each of these data types.

Page 51:

Note how we update purePlate and then pass it to mapFamily, which does all the recursion

Page 58:

We define appendPlate which combines plates over the applicative constant functor on a monoid.

Given a plate p that describes how to convert each type of data into a monoid type o, the preorderFold and postorderFold functions create a plate of functions that traverses all the descendants of the inputs and combine the results using the monoid operation.