

Accumulator-Style Program Design

Due: 1/18/2011

Portfolio Problems

For each of the following problems design four different solutions (functions):

- Using the design recipe
- Modifying the previous solution by using an accumulator
- Implementing the solution using the Scheme loop `foldl`
- Implementing the solution using the Scheme loop `foldr`

Problems:

1. Problem 31.3.4 in HtDP
2. Produce a list of all authors of the books in the given list. (Use the book definitions used in the lecture and the lab.)

Pair Programming Assignment

1.1 Problem

A. A database of information on localities (i.e., cities) in the USA gives us the zip code, name, state, and latitude and longitude for each locality. Design a data definition to represent a `city` in that contains this information for a single locality.

Note 1: We may have several different entries for a given `city`, for example Boston would include the zip code 02115 and 02116, though each has a slightly different latitude and longitude.

Note 2: Define the `location` of each locality as a separate data definition (*long/lat*).

B. We want to draw the cities on a map of the United States. Suppose our map is 100 pixels wide and 100 pixels tall. Design a function, named `to-posn`, that converts a representation of a `location` into a `posn` on the map.

   Assume that the *latitude* and *longitude* lines are parallel to the Scene boundaries (known as *parallel projection*). Also assume the following:

   - the left edge of the Scene is at 125 degrees longitude (in the Western Hemisphere)
   - the right edge is at 65 degrees longitude (also in the Western hemisphere)
   - the top of the Scene is at 50 degrees latitude (in the Northern hemisphere), and,
   - the bottom is at 20 degrees latitude (in the Northern hemisphere)

   Assume that we are focused only on the contiguous continental states (i.e., we omit the beautiful states of Alaska and Hawaii... please accept our appologies).

C. *Optional* Modify the function `to-posn` so that it consumes the width and the height of the Canvas (with the US map stretching over the entire Canvas).

D. Design a function `distance` that computes the distance (in miles) between two cities. Estimates tell us that one degree of longitude (at the USA's latitude) is approximately 55 miles, and one degree of latitude is approximately 70 miles. (Feel free to make more accurate estimates. If you do so, explain how you arrived at your numbers.)

E. Design a function `total-distance` that computes the total length of a trip where you visit all cities in a list of cities in the order given in the list (assume you are a bird... or a plane... or superman).

   Note: First design the function following the *Design Recipe*, then modify the function to be in accumulator style. (Hand in only the accumulator style function. *Make sure your purpose statement includes a line that explains the role of the accumulator*.)

F. Design a function `all-states` that produces a list of all the states visited on a given trip through a list of cities. *Make sure each state appears in the list only once*.

Note 1: Design the function in stages: first follow the *Design Recipe* with the necessary **`local`** function definitions, then optimize the function so it traverses the list of cities only once.

Note 2: You will get *partial* credit if you hand in only the functions that follow the *Design Recipe*.

Important: Use **`check-expect`**s to test your functions.