1 Accumulator-Style Program Design

Practice Problems

Practice problems help you get started, if some of the lab and lecture material is not clear. You are not required to do these problems, but make sure you understand how you would solve them. Solving them on paper is a great preparation for the exams.

For each of the following problems work out the solution in four different ways:

- 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 about localities in the USA gives us the zip code, the name of the locality, the state in which it is located, and the latitude and longitude for this locality. Design the data to represent a city in the USA that contains this information.

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

Note 2: Define the `location` of each locality as a separate data item.

B. We would like to draw the cities on a map of the USA. Suppose our Canvas is 100 pixels wide and 100 pixels tall. We want to convert the latitude and longitude representation of the location into a `posn` on this map. Define the function `to-posn` that produces a `posn` that represents the given location on our Canvas.

   Assume that the latitude and longitude lines are parallel to the Canvas boundaries (*parallel projection*). Assume that

   - the left edge of the Canvas is the 125 degrees of longitude (in the Western hemisphere)
   - the right edge is 65 degrees of longitude (in the Western hemisphere)
   - the top of the Canvas is 50 degrees of latitude (in the Northern hemisphere)
   - the bottom is 20 degrees of latitude (in the Northern hemisphere)

   We focus only on the contiguous continental states (omitting the beautiful states of Alaska and Hawaii).

C. Define the function `distance` that computes the distance (in miles) between two cities. Estimates tell us that one degree of longitude (at our 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, include a comment explaining how you arrived at your estimates.)

D. Define the function `total-distance` that computes the length of a trip on which we visit all cities in a list of cities in the order in which they appear in the list.

   Note: Design this function in several stages: the first version should be done by following the *Design Recipe*. You should then modify the function to use the accumulator style. (Hand in only the accumulator style function.)

E. Define the function `all-states` that produces a list of all states we visit when our trip takes us to all cities in a given list of cities. Make sure each state appears in the list only once.

   Note: Design this function in several stages: the first version should be done by following the *Design Recipe* with the necessary auxiliary functions. You should then optimize the function so it traverses the list only once.

Note2: You will get a substantial *partial* credit if you hand in only the functions that follow the *Design Recipe*.

Note: Use the `check-expect` format of the `testing.ss` teachpack to define all your test cases.

1.2 Problem

Creative Project: An Interactive Game

During the next four weeks you will design and program an interactive game in the style you have done in the first course. A game consist of several different game pieces. The game pieces move either on each tick of the clock, or in response to the keys (typically the arrow keys). There may be other changes in the game pieces over the time or in response to the key events (x key launches a shot, an animal gets hungrier as the time goes on, ...). The game pieces interact in some predefined manner. Finally, something (the state of a game piece, the interaction between game pieces) triggers the end of the game.

A. Brainstorm with your partner about a game you would like to design. It should have at least two interacting game pieces, one of them should respond to the key events, one or both of them should respond to the time ticks. There should be an opportunity to expand the game to multiple game pieces (of at least one kind).

   Our drawing will be limited to an image composed of an overlay of basic shapes: an outline of a circle, a solid disk, a solid rectangle, a line, or a text.

B. Write a Design Proposal for your game that includes the following:

   (a) Game name and a sketch of one or more game scenes, including what the scene will look like at the start of the game.

   (b) A list of (Scheme) data definitions for the game pieces.

   (c) For each game piece describe (in English) its behavior in response to either the `tick` event or to the `key` event, or to both.

   (d) Describe (in English) what are the properties and the positions of the various game pieces at the start of the game.

   (e) Describe (in English) what conditions will lead to the end of the game.

C. Write a user's guide. This is a document you will give to the person that wants to play your game. Explain the goals of the game, how the user can interact with the game, what is the game objective, and how does the game end.

Note 1: The complete description should not exceed two pages. It must be typed, though the *screenshots* can be drawn by hand.

Note 2: You will get comments on your design describing what features are to be implemented, which should be postponed or modified.

Hand in the Design Proposal and the User's Guide to the instructor in class on **Thursday, May 13th.**