a. (4 pts) Sammy's Music wants to reorganize its inventory database. Rather than maintain a list of stock items and assign each album to a single category, the company wants to maintain a hierarchy of categories. Each category can contain several subcategories. Each category also contains the albums that fall into that category. Each album should reside in the most specific subcategory to which it belongs.

b. (6 pts) In class, we discussed two data definitions for family trees (reproduced below). Some programs are easier to write using one data organization rather than the other. To explore this, write the following program in each of the two definitions

Write a program find-siblings which consumes a list of family trees and a symbol and produces a list of symbols. The symbols in the output list are the names of the siblings of the person named in the input symbol. You may assume that a name appears at most once in a family.

Be sure to write out the templates for both versions of the family tree, including the arrows that show the recursion. (You will get 2 points for the templates, 2 points for the program on child-centric trees and 2 points for the program on family-centric trees.)

Notes: This problem uses a list of family trees so that we may have multiple entry points into each family, as discussed in class (for example, in the descendant tree on page 205 of the text, the list would contain the parent structures for Carl, Bettina, and Fred). As a reminder, here are the two data definitions for family trees (the text provides sample pictures of both types of trees, on pages 185 and 205, respectively).
Child-centric family trees

;; A ftn (for family tree node) is either
;; - empty, or
;; - (make-child name father mother year eyes)
;; where name and eyes are symbols,
;;    father and mother are ftn, and
;;    year is a number
(define-struct child (name father mother year eyes))

Parent-centric trees

;; A parent is a structure
;; (make-parent name year eyes loc)
;; where name and eyes are symbols, year is a num,
;;    and loc is a list-of-children.
(define-struct parent (name year eye-color children))

;; A list-of-children is either
;; - empty, or
;; - (cons f r)
;; where f is a parent and r is a list-of-children