15CSE402 Structure and Interpretation of Computer Programs
Problem 4

Assume that a local bookstore (that sells story books) has contacted you to provide an inventory system (which is a collection of information related to all the books) for their book website. Since you wanted to try your hand in the functional programming paradigm, you chose to implement the inventory system in Racket.

You are fully aware that the first step towards realizing inventory system is to first abstract a book. The bookstore feels that the key information in the inventory system is the name of the book and its price. You have decided to implement the inventory system as a simple collection of these two pieces of information.

Once you have the inventory, finding a book in the inventory is an obvious facility you would like to provide. Having developed this far, the bookstore realized that the number in stock and number of copies sold for each book are other key information to be incorporated into the inventory. So is the option to find the number of books in stock given a book name. You may have to re-design/re-write your inventory and provide a procedural abstraction for the same. Also, the bookstore wanted to have provisions to find out how many books are not in stock and how many books have not been sold at all (even a single copy).

Since every business sector has started to embrace analytics, the bookstore wanted you to add a genre for each book so that it can find out which genre is fast selling and worst selling. Similarly adding authors and finding which author is fast selling and worst selling are also straightforward tasks.

The tasks you must do in this case study problem have all been provided in bold text. The quality of the solutions is assessed by the abstractions (both data and procedure) you have designed.

Data abstraction

book(title, (author, genre, price, noInStock, noSold))
inventory(list of book nodes)
genre-count(genre,count)
author-count(author,count)

Procedure abstraction

1. find-book(inventory,title){return pointer to book}
2. get-stock-count(inventory,title){return no of books in stock for the given book, return "no such book" if the pointer reaches the end and does not find a match}
3. get-sold-count(inventory,bookTitle){return no of books sold for the given book, return "no such book" if the pointer reaches the end and does not find a match}
4. find-not-in-stock-books-count(inventory){iterate and return count of books which have noInStock 0}
5. find-not-sold-books-count(inventory){iterate over and return the count of books which have noSold 0}
6. sell-book(inventory,title,count){
    a. iterate, find the book
        i. return "no such book" if the pointer reaches the end and does not find a match
    b. check if noInStock>count,
        i. if yes, noInStock-=count and noSold+=count
            1. Get genre
            2. Go to genre-count, iterate, find the genre
            3. Increase it by count
            4. Get author
            5. Go to author-count, iterate, find the author
            6. Increase it by count
        ii. If no, return saying count is more than what the store has
    c. }
7. increase-stock(inventory, title, count){
    a. iterate, find the book
        i. return "no such book" if the pointer reaches the end and does not find a match
    b. increase noInStock by count
    c. }
8. add-book(inventory, title, price, noInStock, author, genre){
    a. noSold=0 (by default)
    b. Create a new book with given details
    c. Append the book to inventory
    d. Check if author already in author-count else append with this author's name
    e. Check if genre already in genre-count else append with this genre
    f. }

9. best-genre(genre-count){iterate, find the highest count and it's related genre, return the genre}
10. worst-genre(genre-count){iterate, find the lowest count and it's related genre, return the genre}
11. best-author(author-count){iterate, find the highest count and it's related author and return the name}
12. worst-author(author-count){iterate, find the lowest count and it's related author and return the name}
13. get-field-from-book(book,field){in the book node, return the stock field}