Project 8, Program Design

A daycare center would like to maintain a list of book requests from the teachers for classroom libraries. Each book was stored with the title, author's first name, last name, price, and classroom. The program `book_requests.c` contains the `book struct` declaration, function prototypes, and the main function. Complete the function definitions so it uses a **dynamically allocated linked list** to store the book requests. Complete the following functions:

1. `add_to_end`:
    a. Ask the user to enter a book's title, author's first name, author's last name, and classroom (in the exact order).
    b. Check whether the book has already existed by title, author's name, and classroom. If the book has the same title, author's name, and classroom with an existing book in the list, the function should print a message about book already exists and exit the function.
    c. If the book does not exist, ask the user to enter price, allocate memory for it, store the data, and **add it to the end of the linked list**.
    d. If the list is empty, the function should return the pointer to the newly created linked list. Otherwise, add the book to the end of the linked list and return the pointer to the linked list.

2. `search`: search by a book's title. Ask the user to enter the book's title. Find all books with the title. Display title, author name, classroom, and price. If the book is not found, print a message.
3. `print_list`: print the title, author, price, and classroom of all books in the list.
4. `clear_list`: when the user exists the program, all the memory allocated for the linked list should be deallocated.

Note: use read_line function included in the program for reading in title, author first name, and last name.

Grading

Total points: 100

1. A program that does not compile will result in a zero.
2. Runtime error and compilation warning 5%
3. Commenting and style 15%
4. Functionality 80%:
    a. **Function implementation meets the requirement.**
    b. **Function processes the linked list by using the malloc and free functions properly.**

Before you submit

1. Compile with –Wall. Be sure it compiles on ***the student cluster*** with no errors and no warnings.

gcc –Wall book_requests.c

2. Be sure your Unix source file is read & write protected. Change Unix file permission on Unix:

chmod 600 book_requests.c

3. Test your program with Unix Shell script

chmod +x try_requests

./try_requests

4. Submit *requests.c* on Canvas.

Programming Style Guidelines

The major purpose of programming style guidelines is to make programs easy to read and understand. Good programming style helps make it possible for a person knowledgeable in the application area to quickly read a program and understand how it works.

1. Your program should begin with a comment that briefly summarizes what it does. This comment should also include your **name**.
2. In most cases, a function should have a brief comment above its definition describing what it does. Other than that, comments should be written only *needed* in order for a reader to understand what is happening.
3. Information to include in the comment for a function: name of the function, purpose of the function, meaning of each parameter, description of return value (if any), description of side effects (if any, such as modifying external variables)
4. Variable names and function names should be sufficiently descriptive that a knowledgeable reader can easily understand what the variable means and what the function does. If this is not possible, comments should be added to make the meaning clear.
5. Use consistent indentation to emphasize block structure.
6. Full line comments inside function bodies should conform to the indentation of the code where they appear.
7. Macro definitions (#define) should be used for defining symbolic names for numeric constants. For example: **`#define PI 3.141592`**
8. Use names of moderate length for variables. Most names should be between 2 and 12 letters long.
9. Use underscores to make compound names easier to read: **`tot_vol`** or **`total_volumn`** is clearer than totalvolumn.