# Linked List Of Books

BIT 143 – ASSIGNMENT 3

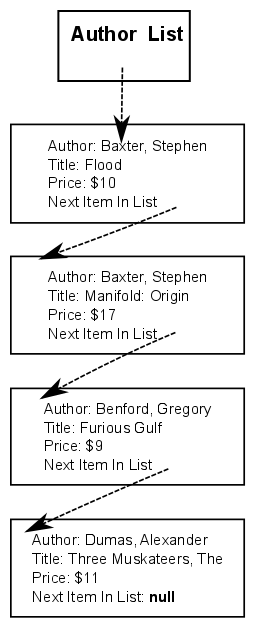
***Due: < Listed in the course schedule >***

|  |
| --- |
| You are not allowed to work in groups this assignment.  For this assignment, you should start, finish, and do all the work on your own.  If you have questions, please contact the instructor. |

#### What the program must accomplish

Imagine that you're going to keep track of some data. In this case, you're going to keep track of a collection of books. Each book has an author (a string), a title (another string), and a price.

For this assignment, you're going to create a program that will manage a linked list of books.

  
The list will be built out of linked list nodes, one per book. Each list node must contain information about a book (including the book’s author, title, and price) along with a link to the next item in the list.

Notice how the list is sorted alphabetically by the author’s name. Thus, if we were to print the books (by ***author***), we'd get:

Book Author: Baxter, Stephen

Book Title: Flood

Price: $10

Book Author: Baxter, Stephen

Book Title: Manifold: Origin

Price: $17

Book Author: Benford, Gregory

Book Title: Furious Gulf

Price: $9

Book Author: Dumas, Alexandre

Book Title: Three Musketeers, The

Price: $11

You need to implement a program that will allow someone to add a book, remove a book, and print the book. The Book objects must be stored in the ‘linked list' described above. If you've already got a book with a particular author AND title, and you try to add another book with exact same author & title, you can indicate that an error has occurred, and not add the (duplicate) book.

To be clear: you need to implement not just the data structures that store this information, but you also need to finish implementing the Console-based user interface that will allow the user to interactively create books, print the list, and remove books that is partially provided to you in the starter project. There are a couple of comments that start with // STUDENTS: that should identify areas that you need to complete (in addition to the multiply linked list work). Make sure that you fill in any error-handling code in the UserInterface class that needs to be finished, as well as any unfinished methods!

Just as a general FYI: DOUBLY LINKED lists are a standard term (in a doubly-linked list is one in which each node has both a next pointer, and a **previous** pointer – a link to the node prior to it in the list), but what you'll be doing here is different from a doubly linked list. **Do NOT use a doubly-linked list for this assignment.** (If you stick with what we covered in class you will do fine, regardless of whether you know what a ‘doubly linked list’ is or not)

## Commenting:

You should comment your code, paying particular attention to areas that are difficult to understand. If you found something to be tricky when you wrote it, make sure to comment it so that the next person (the instructor, who's grading you) understands what your code is doing. It is not necessary to comment every single line.

The purpose of this requirement is to both help you understand, and have you demonstrate, a thorough understanding of exactly how your program works.

*Every file that you turn in should have:*

* At the top of the file, you should put your name (first and last), the name of this class (“BIT 143”), and the year and quarter, and the assignment number, including the revision number, which starts at 0 (“A3.0”). If you’re handing this in again for a regrade, make sure to increase the minor version number by one (from “A3.0”, to “A3.1").

In general, you should make sure to do the following before handing in your project:

* All variables used should have meaningful names.
* The code should be formatted consistently, and in an easy to read format.

**What to turn in:**

o   The C# source code for the entire program.    
Please include all the .CS files, whether you've changed them or not, just on the off chance you forgot that you changed them.

**How to electronically submit your homework:**

This was covered in a separate document in this website..