CPSC 131, Data Structures – Fall 2020
Book Cart: Container Adapters Homework

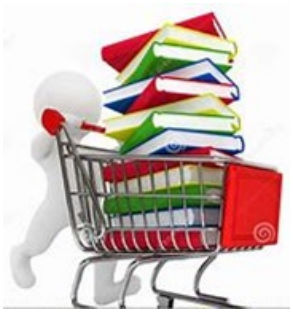




Learning Goals:

- Familiarization with stack and queue concepts
- Reinforce the concept of adapting the stack and queue abstract data type to an underlying implementation data structure
- Familiarization and practice using the STL's adapter container interface
- Increase recursion proficiency
- Reinforce modern C++ object-oriented programming techniques

Description:



Continuing with our Book and Book List themes, you are now at a bookstore shopping for the books on your list. As you walk up and down the aisles you place books onto your book cart, one book on top of the other. The last book you place onto your book cart will be on top and will be the first book you remove. In fact, if you want to get to something at the bottom of your cart you'll have to remove everything on top of it first. You're a very smart shopper so you know to start with heavy hardcover books first so they won't bend out of shape or break, and finish with the soft breakable popup baby books last.

As luck would have it, you've almost completed your shopping and have a pretty full cart when the wheel falls off rendering your cart unmovable. Determined to complete your book shopping you grab another cart and begin moving books from the broken cart to the new cart when you realize that your breakable books, like Ready for Life's "Just Like the Animals", will now be on the bottom. But that's an easy problem to solve, all you have to do is get a third cart and carefully move books between the two new carts so that the breakable books are always on top.

A recursive algorithm to carefully move books from the broken cart to a working cart is:

```
START
Procedure carefully_move_books (number_of_books_to_be_moved, broken_cart, working_cart, spare_cart)

   IF number_of_books_to_be_moved == 1, THEN
      move top book from broken_cart to working_cart
      trace the move

   ELSE
      carefully_move_books (number_of_books_to_be_moved-1, broken_cart, spare_cart, working_cart)
      move top book from broken_cart to working_cart
      trace the move
      carefully_move_books (number_of_books_to_be_moved-1, spare_cart, working_cart, broken_cart)

   END IF

END Procedure
STOP
```

See Data Structure & Algorithms - Tower of Hanoi, Tower of Hanoi video or Tower of Hanoi recursion game algorithm explained for more about the recursive algorithm.

A sample trace might look like:

```
After   0 moves:     Broken Cart            Working Cart           Spare Cart
                     ------------------------------------------------------------------
                     Like the Animals
                     131 Answer Key
                     Les Mis
                     Eat pray love
                     Hunger Games
                     ==================================================================



After   1 moves:     Broken Cart            Working Cart           Spare Cart
                     ------------------------------------------------------------------
                     131 Answer Key
                     Les Mis
                     Eat pray love
                     Hunger Games           Like the Animals
                     ==================================================================



After   2 moves:     Broken Cart            Working Cart           Spare Cart
                     ------------------------------------------------------------------
                     Les Mis
                     Eat pray love
                     Hunger Games           Like the Animals       131 Answer Key
                     ==================================================================



After   3 moves:     Broken Cart            Working Cart           Spare Cart
                     ------------------------------------------------------------------
                     Les Mis
                     Eat pray love                                 Like the Animals
                     Hunger Games                                  131 Answer Key
                     ==================================================================

And so on . . .

After  31 moves:     Broken Cart            Working Cart           Spare Cart
                     ------------------------------------------------------------------
                                            Like the Animals
                                            131 Answer Key
                                            Les Mis
                                            Eat pray love
                                            Hunger Games
                     ==================================================================
```

Once you fill your book cart you'll proceed to the checkout line. When it's your turn, you'll take all your books from your book cart and place them flat on the counter one after the other where they will be scanned in order and a total calculated. As a book is scanned, the ISBN is used to query the store's persistent database for the book's full title, author, and price. You take your receipt and your bags of books and leave the store.

The output of your program is an itemized receipt with the total amount due, perhaps something like this:

```
"9780895656926", "Just Like the Animals", "Ready for Life", 41.97
"54782169785" (131 Answer Key) not found, book is free!
"0140444300", "Les Misérables (1st edition)", "Victor Hugo", 28.96
"9780399576775", "Eat pray love made me do it (1st edition)", "Rebecca Asher", 36.99
"9780545310581", "The Hunger Games - Library Edition", "Suzanne Collins", 67.56
-------------------------
Total $175.48
```

How to Proceed:

The following sequence of steps are recommended to get started and eventually complete this assignment.

1. Review the solution to the last homework assignment. Use the posted solution to fix your solution and verify it now works. Your Book class needs to be working well before continuing with this assignment.
2. Implement the database functions first. Details are in BookDatabase.hpp and BookDatabase.cpp.
    a. The constructor should open a text file containing Books and populate a memory resident data store with the contents of the text file. You must implement the memory resident data store with a standard vector. You are given a sample database text file to test your work. The actual database file used to grade your work may be different.
    b. The find() function takes an ISBN, searches the memory resident data store, and returns a pointer to the book if found and a null pointer otherwise. You must implement this function recursively.
    c. The size() function takes no arguments and returns the number of entries in the database.
3. Implement the segments in main.cpp from top to bottom next. Details are embedded in main.cpp.
    a. Implement the carefully_move_books recursive algorithm first, then
    b. Snag an empty cart
    c. Shop for a while placing books onto a book cart
    d. A wheel on your cart falls off so carefully move books from the broken cart to a new cart that works
    e. Checkout and pay for all this stuff by choosing a checkout line and placing books on the counter. Once all the books have been moved from the cart to the counter the cashier will begin scanning the books in the order you placed them on the counter.
    f. Scan the books accumulating the amount due and creating a receipt with full product descriptions obtained from the database
        i. Don't assume the book's ISBN will be in the store's persistent database.
        ii. Assume the database text file will have formatting errors and that your Book's extraction operator will handle the errors.

Rules and Constraints:

1. You are to modify only designated TO-DO sections. Do not modify anything outside such designated areas. Designated TO-DO sections are identified with the following comments:

```
//////////////////////// TO-DO ////////////////////////////
...
//////////////////////// END-TO-DO ////////////////////////
```

   Keep these comments and insert your code between them. In this assignment, there are 12 such sections you are being asked to complete. 7 of them are in main.cpp, 2 are in BookDatabase.hpp, and 3 are in BookBookDatabase.cpp.

Reminders:

- The C++ using directive `using namespace std;` is never allowed in any header or source file in any deliverable products. Being new to C++, you may have used this is the past. If you haven't done so already, it's now time to shed this crutch and fully decorate your identifiers.
- Use Build.sh to compile and link your program on Tuffix – it employs the correct compile options.
- Filenames are case sensitive, both in source code and in your OS file system. Windows doesn't care about filename case, but Linux does.
- You may redirect standard input from a text file, and you must redirect standard output to a text file named output.txt. Failure to include output.txt in your delivery indicates you were not able to execute your program and will be scored accordingly. A screenshot of your terminal window is not acceptable. See How to build and execute your programs. Also see How to use command redirection under Linux if you are unfamiliar with command line redirection.

Deliverable Artifacts:

Provided files	Files to deliver	Comments
Book.hpp	1. Book.hpp	You should not modify this file. The grading process will overwrite whatever you deliver with the ones provided with this assignment. It is important that you deliver complete solutions, so don't omit these files in your delivery.
Book.cpp	2. Book.cpp	You should replace the provided file stub with your (potentially) updated files from the previous assignment.
main.cpp BookDatabase.hpp BookDatabase.cpp	3. main.cpp 4. BookDatabase.hpp 5. BookDatabase.cpp	Start with the files provided, make your changes in the designated TO-DO sections (only), and delivery your final solution. Changes made outside the designated TO-DO sections will be discarded during the grading process.
sample_output.txt sample_test_results.txt	6. output.txt	Capture your program's output to this text file and include it in your delivery. Failure to deliver this file indicates you could not get your program to execute.
Sample_Book_Database.dat		Text file with a book store's database. Do not modify this file. It's big and unchanged, so don't include it in your delivery. The grading process may use a different database.
BookDatabaseTests.cpp BookTests.cpp CheckResults.hpp		When you're far enough along and ready to have your class tested, then place these files in your working directory. These tests will be added to your delivery and executed during the grading process.