Programming Assignment #6

Small Store Inventory using a Linked List

CS 2308.251, 252, and 257 Spring 2016
Instructor: Jill Seaman

Due: Wednesday, 4/13/2016: upload electronic copy by 9:00am

Problem:

You will implement the ProductInventory class from Programming Assignment #5 using a linked list to represent the product inventory. You will be provided with the following files on the Resources tab on the TRACS site:

- **Product.h** and **Product.cpp** from the Assignment 5 solution
- **ProductDriver.cpp** nearly identical to the one from Assignment 5
- **ProductInventory.h** containing a modified ProductInventory class declaration, set up for the linked list representation.

You will be responsible for providing the ProductInventory.cpp file, including the implementation of the ProductInventory member functions (described below):

ProductInventory and **~ProductInventory**: creates an empty list, and deallocates all the nodes in the list, respectively.

addProduct(Product) ensures the product is unique, and price and quantity are valid. If so, adds a new node containing the product to either the beginning OR the end of the list. Returns true if successful, otherwise false.

removeProduct(string,string) removes a node with the given product name and locator from the linked list. Returns true if successful, otherwise false.

showInventory: displays a listing of the product inventory to the screen, nicely formatted, one product entry per line. Output locator, then quantity, then price, then product name.

getTotalQuantity: returns the total number of units of all of the products in the inventory.

findMinimum: returns the minimum product in the list, using the greaterThan() function over the products.

NOTE: if the list is empty, it should return a product made using the default constructor.

sortInventory: reorders the products in the list, using findMinimum.

Here is the algorithm you must use for implementing the sort function. It is a form of the selection sort. It uses a temporary head pointer variable to point to a new list. It repeatedly selects the next (minimum) element from the original list, appends a copy of it to the new list, and removes it from the original list. When it has removed all the nodes from the original list, it makes the old head pointer point to the new list.

Hints: call findMinimum() and removeProduct(). Incorporate the appendNode code to add the new node to the end of the new list (do not call addProduct).

I will go over this algorithm in more detail in class.

DO NOT create another ProductInventory local to the sortInventory function. I want you to use a ProductNode pointer that is local to the sortInventory function!

Input/Output:

Use the ProductDriver.cpp file on Tracs. Your solution should give the same results as running it with the Assignment 5 solution. I recommend adding some calls to the findMinimum() function to test it.

NOTES:

- This program should be done in a Linux/Unix environment. Use the PA5 makefile.
- DO NOT change the names of the classes, functions or files.
- Your ProductInventory.cpp file **must compile** with the files provided on TRACS (unchanged), otherwise you will receive a score of 0.
- Your program must pass **Test Case 0** or you will receive a score of 30 or less with no credit for the other grading categories (correctness/constraints/style). We will use basically the same Test Case 0 as we did for Programming Assignment 5. I have modified it to work with the ProductInventory.h file for this Assignment.

Logistics:

For this assignment you need to submit only the **ProductInventory.cpp** file. You do not need a zip file, you do not need a makefile, you do not need to provide your driver.

There are two steps to the turn-in process:

1. Submit an electronic copy using the Assignments tool on the TRACS website for this class.

2. Submit a printout of the source files at the beginning of class, the day the assignment is due. Please print your name on the front page, staple if there is more than one page.

See the assignment turn-in policy on the course website (cs.txstate.edu/~js236/cs2308) for more details.