Programming Assignment #5

Small Electronics Store Inventory Redux

CS 2308.255 and 256, Spring 2013
Instructor: Jill Seaman

Due:
section 255: in class **Tuesday, 4/2/2013** (upload electronic copy by 4:00pm)
section 256: in class **Monday, 4/1/2013** (upload electronic copy by 1:00pm)

Write C++ **classes** that manage the inventory of a store that resells used electronics.

Item: For each item (electronic device), you should store the following info:

sku	(int)	Greater than 0
quantity	(int)	0 or more
price	(float)	0 or more
make+model	(a string)	Should not be empty

Note: You should NOT assume anything special about the sku. Items with different make+models may have the same sku. Items with different skus may have the same make+model.

ItemInventory:
You should be able to store 100 items. An attempt to add an item to the inventory when it already has 100 items should fail. (Note: skus may be larger than 100)

You should implement the following operations over the electronics store inventory:

addItem: takes an item and adds it to the inventory (unless it's full). Returns true if it succeeded.

removeItem: takes an Item and removes ALL matching entries for that item from the inventory. Returns the number of items that were removed.

showInventory: displays a listing of the store inventory to the screen, one item entry per line. Output sku, then quantity, then price, then make+model.

sortInventory: reorders the items in the list, using the < (or >) operator over the items (see instructions below) (does not display them).

getTotalQuantity: returns the total number of units of each item in the inventory.

getTotalPrice: returns the sum of prices of ALL of the actual units in the inventory.

Classes:

- Create classes for **Item** and **ItemInventory** with appropriate header files.
- Implement methods in the **ItemInventory** class to complete the operations described above.
  - You may add other private, helper, functions if you want.
- Implement functions in the **Item** class to:
  - set and get all instance variables (make instance variables private)
  - overload **==** (used to remove an item), **<**,and **>** operators (used in sorting)
    - for < and > use sku, and when the skus are the same, use the make+model.
    - for ==, two item are equal if all four instance variables are equal.
- You should implement **two constructors** for the Item class: one that takes no arguments (sku is -1, quantity and price are 0, make+model is empty), and one that takes a value for each of the member variables..

Input/Output:

The **main function** should be a driver program that tests the functionality of the Item and ItemInventory classes. See the website for a driver program that MUST work with your code (without changing the driver program). I recommend expanding the driver to do more complete testing of your code. Even if your program works correctly with the driver it may still have bugs not exposed by the driver.

Do not add extra I/O to the class functions. All the testing should happen in the driver.

NOTES:

- This program DOES need to be done in a Linux/Unix environment. Create and use a makefile to compile the executable program. There will be four goals in this makefile, because you will have three .cpp files. Use the following names for your files:

  Item.h
  Item.cpp
  ItemInventory.h
  ItemInventory.cpp
  ItemDriver.cpp

- Put a header comment at the top of each file.

- DO NOT change the names of the classes, functions or files.

- Follow the rest of the style guidelines from the class website.

Logistics:

Since there are multiple files for this assignment, you need to combine them into one file before submitting them. **Do NOT submit your ItemDriver.cpp file.** You can use the zip utility from the Linux/Unix command line:

```
[...]$zip assign5_xxxxxx.zip ItemInventory.cpp ItemInventory.h
Item.cpp Item.h makefile
```

This combines the 5 files into one zip file, assign5_xxxxx.zip (where xxxxx is your NetID). Then you should submit only assign5_xxxxx.zip.

There are two steps to the turn-in process:

1. Submit an electronic copy using the Assignments tool in TRACS no later than one hour before class the day the assignment is due (see top of page 1).

2. Submit a printout of the file 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.

   If you are unable to turn a printout in during class, you have until 5pm on the day the assignment is due to turn it in to the computer science department office (Nueces 247). They will stamp it and put it in my mailbox. DO NOT slide it under my office door.