Programming Assignment #2

Manage a Small Store Inventory

CS 2308.256, Spring 2014
Instructor: Jill Seaman

Due: before class **Wednesday, 2/12/2014** (upload electronic copy by 1:30pm)

Problem:

Write a C++ program that will allow a user to manage the inventory of a small store.

The inventory for the small store will contain the following information for each product in the inventory:
product name (i.e. "Apple iPhone 3GS 8GB", may contain spaces in it, must be unique)
locator   (string with no spaces, used to physically locate product, **not** unique)
quantity   (how many of this product in stock, greater than or equal to 0)
price   (in dollars and cents, greater than 0)

Note: Your program should be able to store up to 100 different products.

The program should offer the user a menu with the following options:
1. Add a new product to the inventory (prompt user for input values).
2. Remove a product from the inventory (by product name).
3. Adjust the quantity of a product (given the product name and change amount).
4. Display the information for a product (given the product name).
5. Display the inventory sorted by product name.
6. Quit

The program should perform the selected operation and then re-display the menu.

Do not change the menu numbers associated with the operations.

Note for options 1 and 2, you are not changing the quantity of a product. You are adding (or removing) the information about a product from the inventory (you are adding or removing an entry from an array).

For the **Add** operation, a complete solution will ensure that the inventory is not full (100 products) before adding a new product, and it should perform input validation (see constraints above). If it fails to add a product, it will output a message indicating why.

For the **Remove** operation, the program should indicate whether the operation was successful or not (if the product was not found).

The implementation of option 3, Adjust the quantity, is optional. The option must be in the menu, but if chosen it may output "This feature is not yet implemented". It should ask for the desired <u>change</u> to the quantity (a positive value increases the quantity, a negative value decreases the quantity by that amount). If implemented, it should ensure the quantity does not become negative (but 0 is valid).

For **option 4**, if the product is not found, display an appropriate message.

For **option 5**, display the information for each product on a separate line. The values should line up in columns (headers for the table are optional). If the inventory is empty, you may output an empty table (no need to display an error message).

NOTES:

- This is a long program! Start asap! Read all the instructions carefully!

- This program should be developed using a Linux or Unix environment.

- Do not use global variables (global constants are encouraged, especially for the maximum capacity of the inventory).

- Use an array of structures to store the inventory in the program. The structure definition should be global, but the array of structures should NOT be global.

- You MUST use partially filled arrays, there should be no empty slots or gaps in the inventory. Keep all the products at the front of the array, in locations 0..count-1 where count is the number of products.

- You MUST use **binary search** for <u>all</u> searches! No linear searches anywhere!

- The program must be modular, with significant work done by functions. Each function should perform a single, well-defined task (Hint: each menu choice). Also note that some arguments will need to be passed by reference!

- To input the product name (which may contain spaces) use this each time:

    ```
    cin.ignore(100,'\n');  // skips whitespace after prev input
    getline(cin, productName); // where productName is a string
    ```

- OR you may require that the productName has no spaces and use cin >> productName; (for a small point deduction).

- You may use the following code from the book. See the Resources tool in TRACS:
  - Program 5-8: use this as a pattern for the menu portion of the program.
  - Program 8-2: this contains the binary search code.
  - Program 8-4: this contains the bubble sort code.

- I recommend implementing the features in this order (do one per day):
  - The menu (output a sentence for each menu option).
  - Add a product.
  - Display the inventory (first unsorted, then (when that works) sorted).
  - Display the information for ONE product (by product name, requires search).
  - Remove a product (requires search).
  - Adjust quantity (requires search) (implementing this feature is optional!).

- I will put sample output on the class website in a separate file.

Logistics:

Name your file **assign2_xxxxx.cpp** where xxxxx is your TX State NetID (your txstate.edu email id). The file name should look something like this: assign2_js236.cpp

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 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.