CMSC 106 Project #5 Fall 2011

Due date: Monday, December 12, 2011 - 11:00pm

1 Purpose

In this project you will write a program using structures and file input. You will be running a grocery store. It involves maintaining an inventory (an array of Items) and processing a shopping list (an array of Items) to see how much you can sell to the individual.

You will have a structure called an Item which is defined in the shopping.h file. The item contains a name, a quantity on hand and a price for a single instance of that item. When this item appears in the inventory list, all three fields are needed, but when it appears in a shopping list, the price will be 0 (since people shopping don't get to set the price for the item they want to buy).

2 Project description

As you read this section, you may also want to look at the `public` files and their corresponding `output` files. The input files are also included an noted in comments at the top of each test file. You will want to create additional input files and possibly additional test files, too. Do not modify the `shoppingHelpers.c` file included. No `shoppingMain.c` file is included - you should create one to test all of the parts together and to make it easier to change which input file is being used for inventory and which for the shopping list. All code you write should be in `shopping.c`. The output of each of the public source file must match the corresponding output file exactly.

You should not use input redirection in this project at all. All input in the required functions should be from the file named. In your main you may use standard input to get the names of the input files - as demonstrated in class.

You must implement all of the functions as described in the `shopping.c` file.

3 Project requirements

All your C programs in this course should be written in ANSI C, which means they must compile and run correctly with `gcc -Wall -Werror -ansi -pedantic-errors` on the grace system as was setup at the beginning of the course. You will lose credit if your program generates any warning messages when it is compiled. Even if you already know what they are, you may not use any C language features other than those introduced so far in lecture or lab. Note specifically that `structs` may not be used. In addition, the `goto` may not be used. Using C features not presented in the lectures before the date of the assignment, or using the `goto` statement or any dynamic structures **will result** in losing credit.

You may not modify the `shoppingHelpers.c` file at all. All of your code must appear in the `shopping.c` file. You may add other methods or constants to the `shopping.c` if you would like, but they will not have a prototype in the `shopping.h`.

All work for this project must be your own individual work. You may not work with any other student or compare code in any way. If you need help see either the instructor or TA. Obtaining help from others will be considered cheating. Getting code from a web page or other location will be considered plagerism.

4 Project requirements

1. All of your projects for this course must have a comment near their top which contains your name, your Glue ID, your section number, your TA's name, and your university ID number.
2. All your C programs in this course should be written in ANSI C, which means they must compile and run correctly when compiled with the compiler `gcc`, with the options `-Wall`, `-Werror`, `-ansi` and `-pedantic-errors`.
3. You must implement each of the functions defined in the shopping.c file without changing the name, return type, parameter(s) or what it does.
4. make sure it runs with the test input/output files given before submitting it, and make sure you test it on others that are different from those given.

5. You **may not** use any C language features except those introduced thus far in classs. Note that using any features of C other than these will result in **losing credit** when your project is graded. For this project you may not use `goto` or `dynamic structures`.

6. The Campus Senate has adopted a policy asking students to include the following statement on each assignment in every course: "I pledge on my honor that I have not given or received any unauthorized assistance on this assignment." Consequently you're asked to include this pledge in a comment near the top of your program. See below for important information regarding violations of academic integrity.

7. Your program should be written using good programming style and formatting. Any instances of bad style will result in losing points. The good style issuse we will be grading for in this project consists of:
   - having an initial block comment with information as stated above.
   - use of adequate descriptive comments throughout, to indicate what your program is doing and how, also make sure there are not too many comments so that readability is hindered (points lost for both too many or too few)
   - neat and readable indentation and formatting, including consistent alignment of braces and indentation of subsidiary statements, and avoiding writing any statements or comments which are wider than the standard terminal window width of 80 columns,
   - writing clear and readable code,
   - not using any global (or file scope) variables,
   - having descriptive identifiers (variable names, function names, names for symbolic constants, etc) [note: using single letter variable names or names that are not descriptive of their purpose will result in loss of credit],
   - making sure that you have no "dead code" (code that can never be reached) and making sure that you have no "code duplication" (code that already appears another place in the program - or very similar code that could have been done once by creating a helper funciton,
   - use of symbolic constants for any special values which don't change and which are used in several places in your code,
   - avoiding disallowed C language features as discussed above,
   - and, I also suggest fully blocking all loops and conditionals – use a curly brace to make it more obvious what lines are dependent on which headers.