Last updated 10 November, 2014

Assignment 8 (100 points):

Please see Submission Checklist (below) for submission requirements.

(10 points) Programming Styles and Convention.
http://classes.engr.oregonstate.edu/~jessjo/CS161/OSU_IntroCodeStandards_v1.1.pdf

(30) Remember to submit your report (just over the project, not the exercises)!
Which includes:

- Understanding: a description of what you understand is being asked of you for the assignment.
- Design: an outline in **pseudocode or a drawing** as to how your code will be implemented. The design section is to be done before you start coding.
- Testing: a description of the tests you will implement to ensure your code works along with the actual testing. The testing section is to be done before you start coding. **(Make a table with three columns named Input, Expected Output and Actual Output. Discuss any discrepancies.)**
- Reflection: a discussion of the problems you encountered and how you solved them.

(20) Exercise components:

1. (10) Define a struct called Item that consists of a string called name, a double called price, and an int called quantity. Create an array of Item called shoppingCart (you can assume a maximum of 100 Items). Display a menu that gives the following options:
   - Add item: this allows the user to enter the information for a new Item, which is then added to the cart.
   - List contents: this lists the name, price and quantity for all Items in the cart. All prices should be displayed with **two** decimal places. Use showpoint and setprecision for this (section 3.7).
   - Total price: this displays the total cost of all Items in the cart (with two decimal places) - no sales tax, since OSU is in Oregon. :)
   - Quit: exits the program.

   You should write a separate function to handle each of the first three options.

   Input validation: Verify that the user enters a valid double for price and a valid int for quantity. If an input isn't valid, prompt the user to try again.

   Discuss: Compare and contrast using the technique of parallel arrays (section 8.6) with using an array of structs. Do you see advantages to one or the other?

   File must be called: shopCart.cpp

2. (10) Write a function that takes as a parameter a vector of ints and returns a vector containing the mode(s) of that data. Remember that the mode is the value that appears most frequently. If there is just a single mode, then the vector you return will only contain one int, but if there is a tie, there will be more. The vector you return should not contain any duplicate values (each mode should only appear once). Your main method should prompt the user to keep entering another int until they want to stop. It should store these values into a vector and pass that to the function. Then it should print out the values in the returned mode vector.

   Input validation: Verify that the user enters valid ints. If an input isn't valid, prompt the user to try again.

   Discuss: Compare and contrast vectors with arrays. Are vectors always better?

   File must be called: findMode.cpp

(40) Project component

Write a struct called Date that consists of an int called day, an int called month and an int called year. Also, write a

struct called Car that consists of a string called make, a string called model, an int called year, a Date called datePurchased, a double called purchasePrice, a bool called isSold, a Date called dateSold, and a double called salePrice. Create a vector of Car. Display a menu that gives the following options:

- Add entry: this allows the user to enter the information for a car, which is then added to the vector. You should prompt the user for dateSold and salePrice if and only if isSold is true.
- List current inventory: this lists the data for all Cars that have been purchased, but not yet sold, and then prints out the count of how many cars that is.
- Profit for a month: this asks the user for a month and year and displays the total profit for sales in that month.
- Quit: exits the program.

You should write a separate function to handle each of the first three options. All prices and profits should be printed with **two** decimal places using showpoint and setprecision (as in exercise #1).

Input validation: Verify that values entered for day and month are valid. For example: 1/30 is a valid date, but 2/30 is not (don't worry about leap years). Of course 15/72 is also not valid. If an invalid date is entered, prompt the user to try again. No other input validation required.

File must be called: carLot.cpp

Remember to submit your **report** and **source files** to TEACH before the end of Sunday.

Remember to keep discussion going!

Submission Checklist:

makefile

Exercises (20 pts):
shopCart.cpp
findMode.cpp

Project (40 pts):
carLot.cpp

Report, ***in PDF format*** (30 pts):
Understanding
Design
Testing
Reflection

The implementation part of the project is the .cpp file you submit.