CS162 Project 2: Design and Testing with Classes

Ant Implementing Grocery Shopping List

Goals

- Develop a class from given requirements.
- Implement that class using object-oriented programming techniques.
- Overload an operator for a class.

In this project, you need to design, implement, and test a grocery shopping list program. The program **should maintain and display a list of items**.

You will design an **Item class**. The class should have data elements for the following information: **item name**, **unit** (i.e. can, box, pound, or ounce), **quantity to buy**, and **unit price**. Consider following things: do you need any functions other than the constructor? How do you calculate the extended price for the item (quantity to buy times unit price)? How do you print it to the screen?

You will also need a **List class**. **You will store Item objects in your List object.** When you enter a new item, an Item object must be created and added to the List object. Use a **dynamic array** to hold the Item objects. The dynamic array should start with a **capacity of 4 Item objects**. Do you need a print function in this class? Think about it.

Your program must perform the following activities: **create a list, add items, remove items, and display the shopping list**. To add an item, you should prompt the user to enter the name, unit, quantity to buy, and the unit price. The display should show: each item name in the list, item unit, the quantity to buy, the unit price, the extended price for each item, and the total price for all items. Oregon doesn't have a sales tax so you can ignore that. Debug and test your program.

Once you have the List and Item classes working correctly, **test if an item is already in your List before adding it**. **Overload the == operator to perform the test.** There is a simple example to overload this operator in the book. Keep it simple. How will you compare items? You can assume that the user will type the information correctly and compare them by the item names.

You should also test your program to make sure it works properly. Since this is a program with input and output, you should not need any driver functions. You do NOT need to test for every possible item in a grocery store, just a reasonable number. How do you handle spaces in names? Are the extended prices calculated correctly? Is the total amount correct?

Meanwhile, you need to create a **reflection document** and submit it with your program. It should include the design of your classes and how you will use the classes. Figure that out even **before** you start coding. If you do the design properly, the coding should be easier. It also needs to describe your test plan, test results, and comments about how you resolved problems while designing and implementing your program.

What to submit:

- Your program files.
- The reflections document (pdf).
- A makefile. If you do not provide a makefile it will not be graded.
- All the files must be submitted in a zip archive.

GRADING

- Programming style and documentation (10%)
- Create the list class and object (15%)
- Create the item class and objects (10%)
- Add and remove an item object to the list (15%)
- Resize the dynamic array correctly when the number of item objects exceeds the capacity (5%)
- Display the list to include the following: item name, unit, quantity to buy, unit price, extended price for that item. Then at the bottom of the display, indicate the total cost for that trip to the store (20%)
- Overload the equality operator (= =) to prevent including duplicate item objects in the shopping list (10%)
- Reflection document to include the design description, test plan, test results, and comments about how you resolved problems while designing and implementing your program (15%)