Assignment 2 –Arrays and Classes

You will submit a reflections document with every assignment. This is your chance to explain why you did what you did. Or why something you tried didn’t work. You can explain what you did to test your program to make sure it works. You can also document features or bugs that inhibit the program in some way.

You need to design, implement, and test a grocery shopping list program. The program should maintain and display a list of items.

You will use a class for the Items. **The class should have data elements for the following information:** item name, unit (i.e. can, box, pounds, or ounces), number to buy, and unit price. Do you need any functions other than the constructor(s)? How do you calculate the extended price for the item (number to by times unit price)? How do you print it to the screen?

You will also need a List class. **The List class will use a dynamic array to store Item objects.** As each item is entered an Item object must created and added to the array. What do you do if the array is full? One List object will have many Item objects. Do you need a print function in this class?

**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 of sale, the number needed, and the unit price. **The display should show:** for each item in the list, the number of items, the unit of sale, the unit price, the extended price for each item. Then 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 in correctly. If the item is already in the list give the user the option to add the new quantity to the existing Item object.

You must create a design document. It should include the design of the classes and how you will use the classes. Remember to free memory if no longer used. Figure that out, **before** you start coding. If you do the design properly the coding should be easier.

Your reflections document should include the design document, the testing results, and describe any changes you made to your original design.

You will provide a simple test plan. 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 or varying lengths. How do you handle spaces in names? Are the extended prices calculated correctly? Is the total amount correct?

**NOTE:** For testing please keep your array small, maybe only 4-5 elements. There is no ordering requirement so if you can add 2-3 Items and remove one there is not much more to test. To test resizing you (and the grader) will need to insert only a few items into the list. It saves time for everyone! 😊
NOTE: Please use incremental development! Start with the List class and maybe a simplified Item class (only the item name?). If you’re not comfortable with dynamic arrays start with a static array. Just remember to replace it later. Plan this out as part of your design. You will probably need to print the list early on so you can see what’s going on inside the list and the items. When you have a (partial) program that compiles and runs properly save a copy!

You must **include a makefile and put all files for your assignment in a zip file**. If you do not do this assignment will NOT be graded.

**Grading:**

- programming style and documentation (10%)
- create the list class and object (15%)
- create the item class and objects (10%)
- add and remove items to the list (10%)
- you properly manage memory, i.e. NO leaks (5%)
- when adding a duplicate item you offer to combine the quantities(5%)
- your array resizes properly (5%)
- Display the list to include the following: item name, unit for purchase, price per unit, number to buy, extended price for that item. Then at the bottom of the display indicate the total cost for that trip to the store. (15%)
- overload the equality operator (==) to prevent including duplicate items in the shopping list (10%)
- reflections document **to include the design description**, test plan, test results, and comments about how you resolved problems while designing and implementing your program (15%)