Benjamin Tate
1/31/16
CS 162 – Section 400
Assignment 2 Reflection

Design:

My design process was done (quite messily) on paper, which makes it hard to replicate on a computer, but below is my best approximation, with the pseudocode cleaned up to represent the order in which I planned to implement my ideas, rather than the order in which I thought of them.

Item Class:

Required data members: string name, string unit, double price, int quantity

Required member funcitons:

- Item()
  - Initialize strings to “”
  - Initialize price to 0
  - Initialize quantity to 0
- Item(name, unit, price, quantity)
  - Set name, unit, price, and quantity to appropriate parameter values
- String getName()
- String getUnit()
- Double getPrice()
- Int getQuantity()
- Void changeQuantity(int)
  - Add parameter to quantity of this Item

Friend bool function to overload operator== (2 Items as parameters):

- Return true if names are the same & units are the same

List Class:

Required data members: int length, dynamic array list, int numItems

Required member functions:

- List()
  - Initialize length to 4
  - List = new Item[length]
  - Initialize numItems to 0
- Add(Item)
  - Checking for duplicate Item:
    - Iterate through Items in list:

                - If list[i] == parameter Item,
                    - Call change quantity at current Item, with quantity of parameter Item
                    - Stop function
        - Checking if list is at max Items:
            - Initialize int new_length = length + 4
            - Initialize temporary dynamic array of Items with new_length
            - Copy list to temp, item by item
            - Copy temp back into list to update list's length
            - Update length = new_length
        - List[numItems] = parameter Item
        - Increment numItems
- Remove(Item)
    - Iterate over list
        - If list[i] == parameter Item:
            - Iterate from current item to numItems-1
                - Shift contents over to replace parameter Item
- displayList()
    - initialize doubles totalPrice and exPrice, set totalPrice to 0
    - Print table:
        - Iterate over list
            - exPrice = Item.getPrice() * Item.getQuantity
            - Print relevant values in table format
            - Add exPrice to totalPrice
    - Print totalPrice
- deleteList()
    - Delete dynamic array

<u>Main function:</u>

- Create List object
- Initialize 2 strings, double, and int to hold user inputs for name, unit, price, and quantity.
- Prompt user for name, unit, price, and quantity, create Item with responses
    - Repeat this step at least 5 times total to make sure array will resize properly
- Call displayList()
- Prompt user to choose an item to remove
- Remove selected item
- Call displayList() again
- Call deleteList()

Test Plan:

1. Run program and follow prompts, creating 5 totally distinct items
    - Check that list has 5 items with appropriate extended and total prices
    - Choose any item to remove

- Check that list has the proper 4 items

2. Run program again, but this time make two items with the same name and different units, and two with the same name and units
    - List should show 4 items total
    - Quantity of repeated item should be sum of individual quantities
    - Same-name items should both display with different units
    - Try removing the repeated Item
        - The Item should go away entirely (not just reduce quantity)

Reflection and Test Results:

With this assignment, I spent a lot of time carefully thinking about my design, and my final program turned out to very closely resemble it as a result. Some variables changed names because they were easier to type or made more sense at the moment (like how temp became newList in my add() function), but other than that, the only major difference was in my remove() function. I realized while coding it out that the way I designed the remove() function would leave a "hanging" Item duplicate at the end of the list. I don't think that this would have necessarily been a problem, as long as I properly reduced numItems, but it seemed like bad form, and it bothered me, so got rid of it using much the same technique as I used in my add() function. That is to say that I ended up creating a temporary array from my list array, copying only the Items that I wanted, and then I copied that back into list.

When I first got my program compiling, the list wasn't being displayed, and totalPrice was always 0. Using some print-debugging in List.cpp and listMain.cpp, I tracked the problem to my add() function, where I found that I had made the stupid mistake of forgetting to do the simplest step of adding newItem to list[numItems]. I had gone the checks for duplicates and length, and by the time I finished, the main purpose of the function had slipped my mind. After adding that however, my testing went very smoothly. I followed my testing plan, also going through and repeating steps with the Items in different orders, and never encountered a problem.