Adam Kniffin
OSU ID#: 931492308
Email: kniffina@oregonstate.edu

Project 2 – Design Document and Testing

For this assignment there will be 2 classes

1) Item

- Will be similar to an item that you get at the grocery store
- 4 member variables
    1) Name
    2) Unit
    3) Number of Units to Buy
    4) Price per unit
- Functions
    1) Get and set functions for each of the member variable
    2) Calculate Price. Function will multiply number of units to buy times price per unit to calculate the total price of Item.
    3) Print. Will use the get functions to print all information about member variables. Also use calculate price to print the total.
    4) Overloaded equality operator to determine if the two Items being compare are equal to each other or not. (compares member variables)

Reason for Item class.

This class is going to be used to create different Item objects that are going to resemble an item that you would see in a grocery store. Most of the functions and member variables are extremely straight forward and easy to create.

2) List

- 2 Member variables
    1) An array of pointers to Item objects.
        - An array that will hold Item objects
    2) Integer value to contain the number of Items in the list.

- Functions

    1) Add Item function. Will take a reference to an Item object as its parameters. If the Item is equal to any of the Item's stored in the List's array, it will not be added.

    2) Remove Item function. Will take a reference to an Item object. If the Item is found in the array, it will be deleted, and the array will be shifted over.

Requirements of the Program

- When using the program with these two classes we want to be able to do multiple things.
    - Create Item objects based on what the user states (user is creating the object).
    - Remove an Item based on what the user has specified.
        - For this portion of the assignment, I decided make my remove function take the Item out of the array IF they user specified the name and unit of measure for the Item.
            - Could of made more strict, but to me, this seemed like enough.
- The program must create a list, add items, remove items, and display the shopping list.
    - Display should show:
        - Each item in the list
        - Number of items
        - Unit of sale, unit price
        - Extended price for each item
        - Total of all the items in the list

Plan to Test Program

- Starting with the Item class.
    - Once declaration and implementation files have been formatted:
        - Test to make sure an Item object can be created properly.
            - Use get functions to output all the member variables
            - Test print function doing the same thing
    - Test overloaded equality operator
        - Create two item objects that are the same, and 2 that are different

                - Use if else statements to make sure the right output is occurring (1 for true, 0 for false)
        - Calculate price function
            - Output a calculate price function for an Item object
                - If the total is equal to the number of units, times the units it is working properly
- List Class
    - Start with making sure I can create a List object with no errors.
        - Output the member variables to make sure they are initialized properly
    - Add Item function
        - Add an Item to the List and then use the display function to output what has been added
        - Also need to test when the number of Items reach the max amount of values allowed in the array
    - Remove Item function
        - Same as add Item but different syntax
    - Display list function
        - Add items, and use this function to output
            - If no output, was it the display's fault?

1st Testing Session

As I worked through the two classes, they were both extremely straight forward. I had no problem with any of the functions besides the addItem function and the removeItem function. The equality operator was also very straight forward for me and had no problem with that. However, I did have a lot of struggles throughout my testing phase that will be documented below.

addItem Function

1) When creating this function I started off with the idea that I wanted to be able to add items to the List and then I would take care of resizing the array, once it had reached its capacity.

2) I started with a for-loop that that was set to loop for how many items there were. If the Item being passed into the function could not be found, the Item would be added.

a) This ended up being a huge problem during my testing for one major reason, if there are no items in the array, then the number of items = 0, and there-for the loop won't even initiate.

- How I solved the problem. I added a new condition at the end of the addItem function that would determine if the number of items member variable was equal to 0. If it did, the Item would be added automatically, since there was nothing to compare it to.

3) My biggest hurdle in the addItem function came from the resizing of the array. In my original design document, my member variable is an array of Item pointers, not a pointer to an array of Items.

a) During my testing I started out by trying to just resize the array from 4 to 8, not worrying about any member variables but hardcoding the numbers into the size of the array.

- While trying to make my array larger it kept crashing, and the array could not be set to the new array copy that I had created. I did some research online and found that I was using an array of pointers not, pointing to an array of Items.

- **How I solved the problem**. I had to change the member variable to a pointer to an Item variable. Then in the constructor for the List class I initialized the variable to a new Item array of 4 elements.

b) The array was no resizing from 4 elements to 8 elements (numbers hardcoded in) but I wanted the array to double in size every time its capacity had been reached.

- I tried doubling the number of items member variable but quickly realized that by doing that, I was losing track of how many ACTUAL items I had in my array.

- **How I solved the Problem.** I created a new variable called maxSize that would be set to 4 in the constructor. Once the number of items in the array reached the same value as maxSize, it would be doubled. So, essentially every time the array was full and a new item was going to be added, the array would double in size.

<u>2nd Testing Session</u>

I was extremely excited because I had the functions working how I wanted them. The addItem function, and removeItem function worked great, and would add whatever I had sent in, as long as it was an Item object. I double checked the Item using the equality operator to make sure the Item could be found in the array. However, I ran into trouble once I had to incorporate the user experience into my program

1) Add Item Function

- The function that I thought was working had to be restructured. I needed to be able to call it as many times as I wanted without having to worry about creating a new Item object to send to the List.

- **How I solved the Problem.** Everytime the addItem function was called I had the user enter in the information for each member variable. I would check the array to

make sure an Item did not already exist with the same name, or unit of measurement. As long as those two requirements could be met, a new Item would be made by setting the end of the array to all of the values that the user had entered (using the set functions for each member variable).

2) Remove Item function

- This was an extremely similar problem I had for add item function. I had to allow the user to enter information and then find the Item in the array, and remove it. I had to take out the Item reference parameters and set it to void.

3) Main Function

- Once I was able to get my remove item and add item functions working properly, the main function went pretty smooth. I simply set up a while loop that would loop as long as the user did not enter quit.

Overall Thoughts on the Project

This was probably not the most difficult assignment, but I definitely ran into the most amount of changes that I have had to, compared to any other assignment. Looking back at my design document, if I would have had a better understanding of what needed to be accomplished from the start, I could of saved myself a lot of time and energy and not having to switch around my code so often.

Overall, it was a great learning experience and helped get me some more practice with dynamic memory, and pointers (which I needed help with).