Cody Hannan
CS162

PROJECT 2 - REFLECTIONS

DESIGN DESCRIPTION

Purpose:

The program we are tasked with making is simple grocery shopping list program. The basic function of it is to display and maintain a list of grocery items.

Requirements:

- It will need an item class that stores information of the item including name, units, quantity, price per unit, and total price.
- It will also require a list class that will store the item objects in a dynamic array. The array needs to start with a capacity of four.
- The program should create a list, add items, remove items, and display the shopping list.
- The display should show each item in the list including each item’s relevant information, such as name, units, quantity, price per unit, total price, and overall price of all items on the shopping list.
- The program should also test if the item being added is already in the list by overloading the == operator.

Design:

The requirements cover most of the design. There will be an item class, a list class, a menu function, an inputValidation function, and the main function.

<u>Item Class:</u>

The item class will store the item info and mostly contain getter methods. Its default constructor will initialize each item as “empty,” but it will also contain an overloaded constructor that will use user inputted parameters.

<u>List Class:</u>

The list class is where most of the primary functions of the program will be found. Its constructor will dynamically allocate four empty items. There will be an addItem function for adding item objects and a removeItem function for removing items. When removing an item it will shift all items up in the array and deallocate the last entry. It will also have a display function for displaying the current items in the list. The code for overloading the == operator will also be located in the list class and will be used in the addItem function to check if the item to be added is already in the list.

<u>Menu function:</u>

The menu function will display the menu options available to the user.

<u>Input Validation function:</u>

The inputValidation function will validate user inputs by checking that entered values are in the correct range and of the correct type.

<u>Main function:</u>

The main function will primarily be where user inputs are gathered for the item parameters and for navigating the menu options.

Pseudocode:

```
Item Class:
    Constructor takes parameters for name, units, quantity, and price.
        Initializes the item using the parameters

    Getter functions for:
    Name
    Unit
    Quantity
    Price
    And total price
        Which is price * quantity

List Class:
    Add Item function takes parameters for name, units, quantity, and price.
        Uses overloaded operator == to check if item is in list
            If true
                Notifies user

    Remove Item function take parameter for item location in list.
        Shifts all items in list up to overwrite item at given location
        Deallocates final item in list to lower overall list size

    Display function
        For loop for four cycles
            If list size is less than four
                Prints Item info for first four entries

        For loop for items outside of initial capacity
            If list size is more than 3
                Prints item info for entries 4 and on

        Calculates overall total price of shopping list and outputs to user

    Bool Overloaded operator (==) function takes reference parameter for item object
        For loop cycles through whole list
            If shopping list item name == object name
                Returns true
            Returns false

Menu function:
    Outputs menu options available to user:
      1. Add an item.
      2. Remove an item.
      3. Display current list.
      4. Quit program.
```

Input Validation function template:

```
Function call template is inputValid(int min, int max, string "variable")
    If variable == "some type of input that is needed"
        Cin >> User inputted value
        while(cin fails or input is outside of max and min range)
            If cin failed
                Notifies user to enter the proper value type.
                Cin >> user inputted value
            Else if value is outside of range
                Notifies user to enter a value within max and min range
                Cin >> user inputted value

    If variable == "some other type of input that is needed"
    .
    .
    .
```

Main function:

```
Initializes list object.
while(selected option isn't 4)
    Displays menu.
    User selects an option.
    If they chose 1
        Prompts user to enter item name, units, price, and quantity.
        Calls add item function from list class to add item to the list.
    If they chose 2
        Prompts user to enter the location of the item in the list that is to be removed.
        Calls removed item function from list class and removes the item from the list.
    If they chose 3
        Calls Display list function from list class to display the list to the user.
    If the user enters 4 then the while loop ends and the program ends.
```

TEST PLAN / RESULTS

Case	Input	Function	Expected Outcome	Actual Outcome
Display empty list.	Select option 3 from menu	list.Display()	Items 1 through 4 should show "empty" spaces.	Items 1 through 4 show "empty" spaces.
Adding to list.	Select option 1 from menu	list.addItem()	Adds item to shoppingList array	Successfully adds item
Adding past capacity to list.	Select option 1 from menu	list.addItem()	Adds item to shoppingList array	Successfully adds item

Removing from list (first, middle, and last item).	Select option 2 from menu	list.removeItem()	Removes item regardless of location and shifts items up in the list to fill the empty space	Removes item regardless of location and shifts items up in the list to fill the empty space
Removing item when list is past initial capacity.	Select option 2 from menu	list.removeItem()	Removes item and shifts list	Removes item and shifts list
Removing all items from list.	Select option 2 from menu	list.removeItem()	Removes all items leaving the initial four empty spaces	Removes all items leaving the initial four empty spaces
Adding item that is already in list.	Select option 1 from menu	list.addItem() Overloaded operator function (==)	Notifies user that that item is already on the list.	Notifies user that that item is already on the list.
Removing an item outside of list range	Select option 2 from menu	list.removeItem()	Tells user to enter a value within the range of listed items	Tells user to enter a value within the range of listed items
Removing an item from an empty list	Select option 2 from menu	list.removeItem()	Nothing happens. Still displays 4 empty items	Validation method keeps asking user for a value between 1 and 0
Display proper price values as a doubles	Price: $24.54 Quantity: 4		Total: $98.16	Total: $98.16

REFLECTIONS

Once again my main issues were dealing with dynamic memory allocation. I ended up running out of time to get the program to be memory leak free. Luckily it works and only a segmentation fault occurs when the user closes out the program. An error occurring at that point makes me think that there is an issue with my list destructor. I looked into it to try and see if I could fix the issue and I believe it is because I didn't resize the shoppingList array even though I thought I had been doing it correctly initially. It seems that I would create new item objects and add them to shoppingList but I didn't allocate a memory space within the array whereas I did for the item object. I noticed that these issues can be hard to pinpoint unless you use valgrind. I would compile my code in Xcode and it wouldn't indicate that there was a segmentation fault. However it would tell me if I deallocated space that was never allocated. My main method for checking these issues was a combination of running the program in Xcode and flip. I would have to enter some amount of items to the list and experiment with removing at different capacities. Once that wasn’t having any issues I would try to get the program to close without deallocating an object I had not previously allocated. Once Xcode wasn’t outputting errors, I tried running on flip, but would receive segmentation faults when closing the program. Overall, pointers and memory allocation continue to be my weakest area.