Jared Parkinson
Assignment 2 – CS 162

Requirements: The requirements of this assignment I understand as the following items:

Overall: Create a shopping cart that uses a dynamic array to hold items. You must be able to add and delete items. The shopping cart will hold 4 items by default on creation. When the proposed number of items exceeds the number of array slots, increase the array size. This assignment does not indicate you must *shrink* the array if an item is deleted. Display a list of the Items on the Shopping List. Overload the == operator in the Item class to prevent duplicate items.

The user will input 4 things. The Item Name (spaces allowed), Item Unit Type (spaces allowed), Number of Units to Buy, and Price per Unit. The user will not be able to add duplicate items. I chose to omit the ability to update current items due to time constraints fixing bugs.

On the subject of the resizing: I chose to follow the PDF statement of increasing the array when the items exceeded the array size but did not decrease the array as the pdf did not state to decrease it.

On the subject of Overload: I chose to put the Overload in the Item class, and then access that overload on a per-item basis from within the List class. In addition, I chose to only pass one parameter to the operator and let it check against the item it is being called in versus the item name being passed to it.

Class Design:

Per the charts below, I wrote all of this out in Excel (more on this in reflection). Anything you see in red was changed at some point to accommodate design issues.

Some notable examples:

Item class didn't have too many changes other than setting a default constructor when I was working on getting a default Item to be created. I left this in the code just in case. I changed the initializations to 0 because Flip complained about doubles into NULLs. I placed function headers on all functions per TA instructions last assignment. Initially I had them 1 for getters and 1 for setters. extPrice was initially derived from a function. It now just happens during Construction. This just made it cleaner.

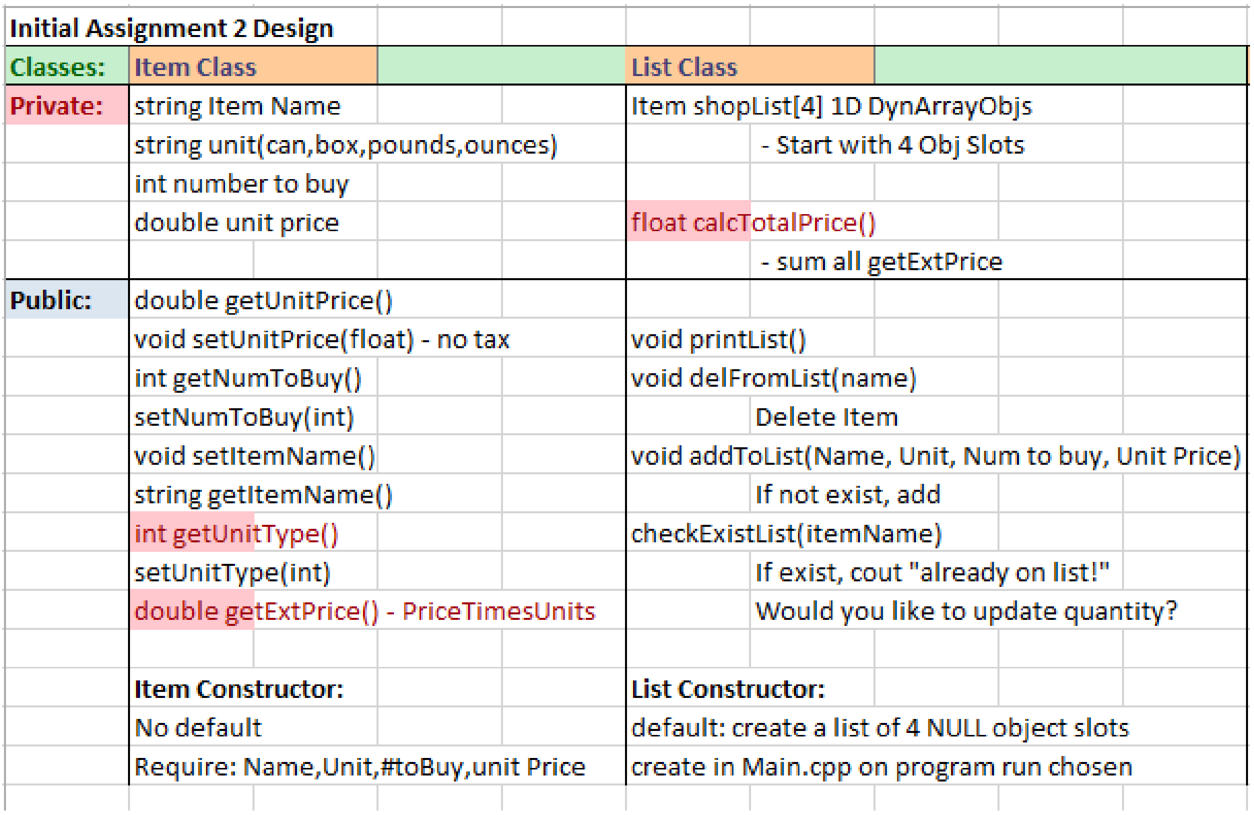
Item::setUnitType: Was going to be an int 1-4 since the pdf said we only needed 4 unit types. It was not specifically stated we needed the user to type this information in. After reading the forums, this method was changed from a number(int) system to a String system.

Item::getExtPrice: Initially, this was supposed to do the calculation and return it. Then I thought that it would be just as easy to do this on the construction of the Item. Swapped it to there. Now this just returns the data member.

List class had some pretty significant changes over the life of the coding. I decided to use numItemsOnList and numArraySlots data members to increment and decrement and avoid additional validation with long code behind them. This meant I could just use those as checkers to see rather than cycle the array or check the size just to figure out what is going on.

List::resizeList: This was the big one. This single function took me hours of hitting my head to finally get correct. This is not so much because I have never done dynamic, but because I was attempting to get it to work with a const array into a dynamic array. I swapped between many different versions of the array finally to land on the correct one after finding a gem in stack overflow on the proper syntax.

Below is a portion of my excel sheet:



Initial Assignment 2 Design			
Classes:	**Item Class**	**List Class**	
Private:	string Item Name	Item shopList[4] 1D DynArrayObjs	
	string unit(can,box,pounds,ounces)		- Start with 4 Obj Slots
	int number to buy		
	double unit price	float calcTotalPrice()	
			- sum all getExtPrice
Public:	double getUnitPrice()		
	void setUnitPrice(float) - no tax	void printList()	
	int getNumToBuy()	void delFromList(name)	
	setNumToBuy(int)		Delete Item
	void setItemName()	void addToList(Name, Unit, Num to buy, Unit Price)	
	string getItemName()		If not exist, add
	int getUnitType()	checkExistList(itemName)	
	setUnitType(int)		If exist, cout "already on list!"
	double getExtPrice() - PriceTimesUnits		Would you like to update quantity?
	Item Constructor:	**List Constructor:**	
	No default	default: create a list of 4 NULL object slots	
	Require: Name,Unit,#toBuy,unit Price	create in Main.cpp on program run chosen	

Menu Class			
Main Menu			
	1 Play Game		
	2 Quit		
Game Menu			
	1 Display Current List		
		Display Shop List	
	2 Add Item		
		Name Item(s) Added	
	3 Remove Item		
		Cout has been removed	
	4 Main Menu		
		Back Menu	
Menu Constructor:			
default: create Menu, set choice 0			

What happens when user adds object?				
	Store cin (Name, Unit, Num to buy, Unit Price)			
	Initiate createItem(1, 2, 3, 4)		addToList(item)	
		Check to make sure array slot is available (NULL)		
		Check if item exists in List		
		If No NULL, then dynamically increase Array		
			Make New Array+1, Copy current Data, Add new item, delete old Array	
		If NULL exists in Array List		
			Add Object to Array in First NULL slot	
	cout Item has been added to List			

DESIGN CHANGE NOTES

Item	List
getUnitType = string changed to int	calcTotalPrice changed from float to double
getExtPrice = sent to List Class	data members for array and items on list
extPrice setup as data member and initialized	
	addItem() Scope of Pointer/object destroyed - not using new Item(), was making destroyable item
resize array only needed when EXCEEDs capacity according to PDF	
	resizeArray disaster redesign using dynamic on both arrays
	printArray not showing past 4 slots - trying to print extra slot non-existent
	remove not removing slot 1 - turns out not removing anything other than last slot

Test Plan: Here is my test plan in table form:

Test Runs	**Action**	**Expected**	**Pass/Fail**
Menu System	Press 1 - Begin	Go to Game Menu, display game menu	Initial: Fail 2nd: Pass, needed to mess with order of operations menu system
	Press 2 - Quit	Quit the program	Initial: Pass!
	Press 1-4 in Game menu	Do said action	Initial: All of these passed
	cin Enter character instead of digit	Fail with message	Initial: Pass! I used !cin to validate
Add Items	Add 1 through 4 items	Allow 4 adds, no crash	Initial: Used declared at compile array size, worked Pass!
	Add 1 – 4 items	Allow 4 adds, after updated dynamic starting array	Initial: Fail! 2nd: Pass! I had to re0think how to use the array system. I was accidentally using an array of Objects rather than pointer to objects on redesign.
	Add 5th item	Allow 5th item and resize array	Initial: Fail! 2nd: through 10th: Fail! FAIL! 11th: Pass! I had to rework the resize array function about 10 times before I finally got it to work properly. This included changing the starting array from at compile const 4 to dynamic creation.
	cin User input spaces in name	Allow spaces in Names AND in Unit Types	Initial: Fail! 2nd through 4th: Fail! I had to try many different cin methods to grab the line and not the characters entered on

			some menu beforehand. cin.ignore mostly resolved the issues I have seen. There could still be bugs somewhere
	Cin User input non-number in Menu system	Fail with message	Initial: Fails with message that you need to enter a digit. This system did not carry over to cin on Item adding due to time constraints
Remove Items	Rem first item 1	Remove the 1st item in list	Initial: Fail! 2nd: Pass. Needed to take into account that my if loop in remove was actually tripping the bool more than once and not returning when it found a match, it moved on to slot 2 and failed.
	Rem 2nd item 2	Remove 2nd item	Initial: Fail per above explanation unless last item on list. 2nd: Pass after rework
	Rem last item	Remove last item	Initial: Fail Per above 2nd: Pass after rework
	Rem item with name that has spaces	Remove item even with spaces in name	Initial: Fail 2nd: Pass. Needed to rework cin system because it would take into account previous digits in menu system as input into the name when checking
Operator == Overload	Run itemExists function to test overload in Item.cpp	Properly check name against current item	Initial: Fail due to itemList pointer not properly being called from Item Class 2nd: was able to confirm working when itemList slot

			called operator== and passed the Item name to test.

Reflections:

Besides the typical print the pdf out and mark it up, the first thing I did on Assignment 2 was different than Assignment 1. On Assignment 1, I wrote out everything on Paper. When I made changes, it looked like chicken scratch. I decided to use MS Excel to organize all of the classes and data that I needed to get done for this program. This method was quite a bit more helpful in the sense that I could see everything and manipulate, mark, change it without a lot of paper hassle. Not only that, but I could fit a lot more data and materials into one spot rather than having pages and pages of writing.

Dynamic Issues: Honestly, I spent WAY too much time on this assignment. I got stuck very hard on the Dynamic resize of the array. This is because I began this assignment using an Item *[4] and then tried to manipulate it and copy using the Dynamic new array of tempArray. This was a mistake in my mind because I spent a couple hours trying to get the syntax right and it just wasn't working. I wasted a bunch of time on it before I went and tried to redesign it. Sadly, I ended up spending more time on it until I was able to get it to work by swapping my initial array of 4 to a dynamically created array of 4 on construction of List.

Print Issues: I got stuck printing and had memory leaks because I was trying to print an array slot +1 greater than the actual size of the array. This was a simple fix and oversight. Another issue I had was that it was instantly failing and crashing. This was because I was trying to get the name of an item when it was a NULL slot. This is why I added the NULL checker in that section.

User Input issues: One main issue I had was properly getting cin to contain spaces. I ended up scouring the net to find cin examples that could include spaces. I started using getline, but found that a lot of the time the names would spill over into another portion of the program. I was able to (I think) fix it by the end of the program. I couldn't see obvious bugs with it like I could before. It would fill out 3 lines on the next cin input and not let the user work on that line. Per the assignment instructions, I assumed that all input would be correct.

Ultimately: I wish that I had a much firmer grasp on the Dynamic array vs const array initialization and how they could both be combined. I also hope that I get better at the coding process and methods so that these assignments do not take as long. I feel as if I am much slower than all the other people in the class and it takes me twice as long to write a program in comparison to them. It generally works well and looks great when finished, but takes longer.