DESIGN AND REFLECTION

Cassidy Bullock. CS 162 Online Section

DESIGN

Classes:

Items

Functions: N/A

Data: Public

- String name: this will hold a string for the name of the grocery item
- String unit: this will hold a string with the type of unit
- Int amount: this will hold an integer for the number of this item to buy
- Double price: this will hold the price per unit of the item
- Double subtotal: this will hold the calculation of the amount times the price, so the total price for the item

List

Functions:

- Run(): public, runs the other private functions within the class
- addItem(): private, adds an item to the list and is called by run() if the user requests to add an item
- fillItem(): private, asks user to input the properties of the item they want to add to the grocery list, is called by addItem()
- removeItem(): private, called by run if the user requests to remove an item, asks the user what to remove then systematically checks the array for a matching name, then removes that from the list and shifts everything else up so there is no blank spot
- MoveUp(int index): private, called by removeItem() to shift all the items in the array up one from the spot where an item was removed
- printList(): private, called by run() if the user requests to see the list, prints each item from the list with its name, unit, price, amount and subtotal

- printMenu(): private, called by run() to print the menu to the user while the user has not selected to quit. Also displays the total price for the shopping trip at the bottom of the menu
- checkSize(): private, checks the size of the array to see if the counter is the same as the maximum size of the array and returns true if the array size needs to be increased
- doubleAll(): private, calls checkSize to see if the array size needs to be doubled, then doubles the size and creates a temporary array of the doubled size to store all of the items, deletes the original array and then gives back the new, larger array, is called by run to just immediately check if the array needs to be increased and to increase automatically before the user puts in a new item to prevent segmentation faults.

Data: Public

- Int ctr: stores how many items there are
- Int size: stores the maximum size of the array holding all the items
- Double total: holds the total price of the shopping trip
- Items all: pointer to the array holding the items

Main:

- Create an object "grocery" for the class list
- Calls the function run() within grocery

REFLECTIONS

Testing:

Function: addItem()

Test Case	**Input Values**	**Expected Outcome**	**Actual Outcome**	**Comments**
input first item	eggs, dozens, 2, 2.05	item is added to list and properties are assigned to it	item is added to list and properties are assigned to it	works as expected
input subsequent item	milk, gallons, 1, 3.99	item is added to list and properties are assigned to it	item is added to list and properties are assigned to it	works as expected
input item greater than original size	watermelon, pounds, 5, 1.97	array should have already doubled to accommodate more items	item is added to list and properties are assigned to it	this indicates that the function doubleSize() is working so the array is dynamic

Function: removeItem()

Test Case	**Input Values**	**Expected Outcome**	**Actual Outcome**	**Comments**

remove item on the end of the list	flour	item is removed from the array	item is removed from the array	works as expected
remove item in the middle of the array	milk	item is removed from the array and all following items are shifted up one	item is removed from the array and all following items are shifted up one	works as expected
ask to remove item that isn't there	apples	nothing will happen, program will continue	nothing will happen, program will continue	works as expected, while not ideal is also not detrimental to the overall functioning of the program, could add function to reask user what to remove if no matching item is found

Function: printList()

Test Case	**Input Values**	**Expected Outcome**	**Actual Outcome**	**Comments**
print list with 3 items on it	none	prints all the items and their properties: price, amount, etc.	prints all the items and their properties: price, amount, etc.	works as expected
print list with no items on it	none	prints nothing	prints nothing	works as expected

Function: run()

Test Case	**Input Values**	**Expected Outcome**	**Actual Outcome**	**Comments**
ask to add item	1	prints menu, calls function addItem()	prints menu, calls function addItem()	works as expected
ask to remove an item	2	prints menu, calls function removeItem()	prints menu, calls function removeItem()	works as expected
ask to print the list	3	prints menu, calls function printList()	prints menu, calls function printList()	works as expected
ask to quit program	4	prints menu, exits program	prints menu, exits program	works as expect

General Other Thoughts:

Most things work as expected. There was a lot of trouble trying to get the dynamic array to work, and I tried several different approaches and did a lot of research both out of the book and online but it turned out I had an off by one error in the function checkSize(). As soon as that was fixed, the program appears to be working as it should.

The only thing is that there is not much error handling, I intended to work on this more but I prioritized getting the array to be dynamic and ran out of time to write these functions. What I did do was make sure the user entered a number for the menu. I also made the program to use getline when getting the name of the item in case the enter an item that has spaces in the name. This program assumes a smart user who enters valid inputs for the most part and doesn't do things like add duplicate items.

To Use the Makefile:

Type **make -f make_grocery.txt** into the command line