Bryan Konshak Assignment 2 Reflection:

I was tasked to create an Item class that holds an item name, type, quantity, price, and then gets the total price based on quantity.

Initially I only needed the constructors, using the information needed from the user to create the Item objects as parameters. My default constructor created an "empty" item with fake names and no values. I added accessing functions as I needed them in my list class to print to my shopping list, as opposed to printing the Item from the Item class.

My list class stored pointers of Items in an array. It started with 4 Items but doubled when those 4 items were added. I used a print function in this class instead of the Item class.

My program was tasked to create a list, add items, remove items, and display the list. It prompts the user to enter in the items, and ignore duplicate names. It then displays the name, type, price, quantity, total price, and then the total list price.

The program uses the overloaded == to check for duplicate names.

Item Class:

The constructor takes as parameters the name, type, quantity, and price as parameters, then calculates total price quantity times price. It also has a default constructor with "Removed Item" as its name and type with 0 price. Also it contains accessor functions.

List Class:

The List Class has a constructor that initializes the array to 4, creates the ItemArray, sets values in array to NULL, established the endArray position to 0, and initializes the totalList price to 0.

It then runs the menu() function in main and the program begins. Then menu uses a utility menu function that displays three options: add item, remove item, or quit and print. Option one runs the addItem function. Option two runs removeItem. Option 3 prints the list and ends the program.

AddItem()

The addItem() function uses utility input validation functions and getline to account for space in strings. It gets the item string name, string type, double, price, double quantity (1.5 pounds for instance) and creates and Item* pointer object and assigns it to the endArray element position.

If endArray is greater than 0, it checks for duplicates using the == overloading function comparing itemName to the ItemArray and every element in a for loop. If there is a duplicate, endArray does not increase and the print out leaves it off the list (since endArray is used in the loops).

Otherwise it checks to double the size when endArray equals the size member -1 so there is enough space to keep adding items. It does this by calling the doubleList function, which creates a temp array to move the Items to and then back to the doubled ItemArray. Then it adds one to the endArray and prints the list via printList(), which shows all items, quantities and prices, and the final price using the accessing functions and keeping a running total.

removeItem()

I struggled figuring out what to do here but I created an Item with the string “Removed” and it replaces the item in the cart and changes its total to 0.

Test Plan:

Item class:

Test 1: unitPrice * quantity is computing correctly with cout statements.

Test 2: Assure the strings are printing to screen correctly

List class:

Test 1: Program accepts strings with spaces—FAILED—switched from cin to getLine

Test 2: Test input validation for price and quantity (numbers greater than 0)—PASSED—existing utility functions

Test 4: Enter in more than 4 Items and double the array—FAILED segmentation fault. Switched the doubleArray function to occur when endArray-1.

Test 5: Check the overload function—FAILED—causing fault when endArray was 0, if endArray>0 to fix

Test 5: Kick out duplicates with overload function—PASSED

Test 6: Assure program doesn’t crash with greater than 16 items: passed

Test 7: Assure list prints with accurate totals—PASSED

Test 8: Assure removeItem replaces Item as removed—PASSED

Test 9: Program quits with option 3—passed

Test 10: Various combinations of lists, removing, adding, of different sizes—2, 4, ,10 and 20 with accurate printouts and removed items

Test 11: check for memory leaks—FAILED—added delete to destructor and after moving items to temp array in doubleList()