Grocery List:

Problem Domain:
The grocery list program needs to allow a user to add an item, its unit (each, box, pound, etc), its price and cost to a list of items. If the item already exists on the list the item will not be added to the list. Once the user is done adding items, the program should print the total cost of each item plus the sum of all the items.

Thoughts on class

Item Class
member vars:
- Name
- Unit (do we make an enum???)
- Quantity
- Price Per

member funs:
- Constructor
- Set Func for each variable
- Get Func for each variable
- Print Func (do we have the print function here or in the list class?)

List Class
Member vars:
- pointer to pointer of item
- another pointer to pointer of item
- totalItems;
- arraySize;

member funs:
- addItem function
- constructor
- print item function (should we have it here or in list class?)

Ideas:

How to add new items *stuff = new Item(args)
after item is added, point pointer to NULL and create a new item

add item function within the list class will take in user input and create a new item. list class must include the item.hpp file.

we create a list class that includes an array of Item objects.

when the user wants to add an item to the grocery cart, he/she uses an addItem function that gets user input and dynamically stores it in the next available location in the items array.

the addItem function will use pointer notation to input all new items using dynamically allocated items.

the constructor of of the item class will initialize all elements in the beginning array to NULL

the addItem function will use a while loop with an exit function for customers to discontinue the addItem process.

the List class will have a print function that uses pointer notation to access the getFunction for each item. it will calculate total prices of list by adding each item's total price (price * qt)

Pseudocode:

```
create item;

start a while loop:
get user input to string variables for item name and unit;
get user input for price and quantity variables;

point pointer to new dynamically created Item object;

if (total Items is more than 0)
        check to see if new Item object is already in the array;

if item is in array
        start loop over;
else
        add item to array using the addItem function

point pointer to Null

if (totalItems is equal to arraySize)
        use the expand array function to expand the array to allow for more products

ask if user wants to add more products;
```

run loop or exit;

Testing

Test 1:
Input Type - enter random characters, numbers and symbols for the string entry

Expected Result - Because using getline, program should run as expected. Be sure to cin.clear and cin.ignore after entry

Actual Result - Program runs as expected. No issues with user input.

Test 2:
Input Type - enter strings or characters for integer and double variables.

Expected Result - Using cin.good user validation in a while loop we can eliminate errors in user input. No errors are expected.

Actual Result - when entering bad input into integer variables, we successfully requested new user input. however, the array population works improperly when user inputs characters into the array. Not sure how to keep the issue from occurring. Attempted to change to a getline function and (atoi) the string, but still unable to get the characters or strings to properly convert to integers.

Test 3:
Input Type - make duplicate item

Expected Result - the overloaded == operator should successfully check for equal variable names in the array of items. if the items have the same name, then the item will not be added to the array

Actual Result - the overload operator worked successfully and duplicate items were not added to the array.

Test 4:
Input Type - add unknown data types to the user question " would you like to add more items"

Expected Result - the input requested is a character. using an if statement and toupper function, the program will continue to ask for new items unless the user enters an 'n' or 'N' for "No."

Actual Result - the result was as expected and the user could add new items when necessary or print the list when finished.