Program #2
CS 202 Programming Systems

***** Make sure to read the Background Information first!**
It applies to all programming assignments this term***

Program #2 - Information

Have you ever thought about what happens behind the scenes at a restaurant? Even something as simple as a foot cart? There are ingredients to keep track of. How much does each recipe require? And, then how do you know when it is time to restock and buy more of a particular ingredient? I can imagine that management of this could be more time consuming than one would expect. Since this is a lot to keep track of, we will write a program to help out!

Program #2 – Building a Hierarchy using Dynamic Binding

For your second program, you will be creating a C++ OOP solution for helping manage a restaurant (in terms of purchasing ingredients) based on the items ordered from the menu. You are allowed to keep the menu relatively simple (2 or 3 entrees, 2 or 3 appetizers and then drinks). There will be two external data files. One to hold the menu and ingredients required for each menu item. The other to hold the ingredients in stock (with the quantity of the item and expiration dates for those items that require it).

The requirement for this application is to have at least **THREE DIFFERENT TYPES OF MENU ITEMS (entrees, appetizers, and drinks)**, derived from **a common abstract base class** and through DYNAMIC BINDING! With dynamic binding, we want each class to have a self-similar client interface. For each menu item, these could include:

a. Allow the client to order this item from the menu
    - Entrees come with extra items such as a starch (French fries), salad and/or soup
    - Appetizers do not have any extra items
b. Remove Item from menu,
c. Substitute ingredients (for gluten free situations or allergies),
d. Purchase ingredients (needed for that menu item);
    - with drinks there would be no expiration date but with entrees there would be for meat and dairy ingredients.

OOP and dynamic binding are THE PRIMARY GOALS of this assignment!

-

Program #2 – Building a Data Structure

Every program this term will experiment with combinations of data structures. In this assignment, the required data structure is a tree organized by the ingredient for a menu item. For each node in the tree, there must be a LLL (all the menu items that use this ingredient). Implementation of the data structures requires full support of insert, removal *(not required for a balanced tree),* display, retrieval, and remove-all.

Efficiency must be part of your data structure design. If the user enters the data by term, the result will be the worst case scenario. Make sure that doesn't happen. You may elect to use a balanced tree or an algorithm that ensures that the tree is reasonably structures (and not a linear sequence!).

EVERYONE this term will implement a balanced tree at some point during the term. You may ELECT to do that with this assignment and gain points for efficiency. Or wait until a future program.