**Program Specifications**

Rewrite lab 2 adding polymorphism and abstract classes. Also implement operator overloading.

**Class Design**

*class InventorySystem (*now contains an array of pointers to abstract class Item objects instead pointers to InventoryItem objects)

* BuildInventory: Need to be modified to read DairyProduct in addition to eProduct. The new file format is shown at the end of the lab specs
* ShowInventory: No longer need to do static\_cast. Polymorphism will allow the correct information display depending on the objects.
* ReportExpiredItems: **New function**. Traverse the entire array of pointers and downcast the pointers using dynamic\_cast. If it is a DairyProduct object then display it if the product is expired
* ReportDefectiveItems: **New function**. Traverse the entire array of pointers and downcast the pointers using dynamic\_cast. If it is an eProduct then display it if it is defective
* SearchItem: must be returning a pointer to Item now instead of pointer to InventoryItem

*class  Item*  (**New class**. Minimum implementation specified below)

* Private data members: None
* Constructors: only default constructor is needed which will output “Item constructor”.
* Destructor: output “Item destructor”. Make sure you define destructor properly in the context of polymorphic implementation
* Public member functions: **(all of them are pure virtual functions)**
  + IsNew: takes no parameter and returns a bool. Default implementation is to return false.
  + Clone: takes no parameter and returns a pointer to an Item object. Default implementation is to return NULL pointer.
  + MoreValuable: takes a reference to a const Item object and returns a bool. Default implementation is to return false.
  + DisplayItem: take no parameter and return nothing. Default implementation is to display an empty string (“”).

*class InventoryItem*: derived from class Item. Prefix DisplayItem with keyword virtual.

*class Product*: derived from InventoryItem. Prefix DisplayItem with keyword virtual.

* Provide implementation for MoreValuable function based on Price data member.

*class eProduct*: derived from Product class. Prefix DisplayItem with keyword virtual.

* Provide implementation to IsNew function based on Condition data member
* Provide implementation to Clone function which dynamically allocates an eProduct object (initializing the cloned object with data from the current object either by way of constructor or by way of get/set functions or by way of accessing protected data) and then returns the pointer to the newly allocated eProduct object
* Overload the following operators for this class: insertion operator << and equality operator ==

*class DairyProduct*: **New class** derived from Product class. Prefix DisplayItem with keyword virtual.

* Private data members: ExpirationTime (time\_t data type. ExpirationTime is the number of seconds has elapsed since 1/1/1970. For example: the number of seconds has elapsed since 1/1/1970 on 10/31/2014 is 1414738800).
* Constructors: default and non-default constructors
* Destructor
* Public member functions: GetExpirationTime, SetExpirationTime, DisplayItem
* Provide implementation for IsNew based on ExperationTime and current time. A diary product is new if the expiration time is at least two days (2 x 24 x 3600 = 172800 seconds) more than the current time. To get the current time:

time\_t   now = time (NULL);      // now is the current time which is the number of

                                                    // seconds that has elapsed since 1/1/1970

* Provide implementation for Clone function similar to eProduct

**Implementation Requirements**

* Here is how your main program should be coded
  + Declare a pointer to InventorySystem object
  + Dynamically allocate an InventorySystem object
  + Invoke BuildInventory
  + Invoke ShowInventory
  + Invoke the two reports on DiaryProduct and eProduct
  + Your own code to test if your overloading operator implementation is correct for << and ==
  + Use a loop to ask user either enter ‘S’ or ‘s’ for Search an item or ‘Q’ or ‘q’ to quit the program