**Program Specifications**

You’re asked to implement an Order Processing program to keep track of orders for customers and process them if needed. The program will read a text file containing customer orders in arbitrary order and then process them

**Class Design**

* Re-use Item->InventoryItem->Product->eProduct and DairyProduct class hierarchy from lab 3
* Address: (street number, street name, zipcode, state): provide constructors/destructor/member functions as required in previous labs
* Order**: this is a template class** representing an Order of <something**>.**That <something> can be an eProduct object, a DairyProduct object, an OnlineProduct object, or a pointer to an Item object.**All constructors/destructor/member functions must be defined outside of class declaration**
  + Data: OrderId (string),  OrderDate (long): number of seconds after 1/1/1970,  item (<something>)
  + Constructors
    - default: initialize string to “”, long to 0
    - non-default constructor: take a string, a long, and <something>
  + Destructor: output order information
  + Functions: GetOrderId/SetOrderId/GetItem/SetItem/GetOrderDate/SetOrderDate  and other functions if needed
  + Operator overloading: overload == operator so that it returns true if 2 orders are the same. Two orders are the same if they have the same OrderId and the same <something>
* OrderProcessingSystem
  + Data: store name (string), store ID (string), pointer to Address, a map of customer orders (key: customer ID(string), value:       a deque containing Order<Item  \*> objects)
  + Constructors
    - default constructor: initialize strings to “”,  pointers to NULL
    - non-default constructor: take 2 string parameters for store name and store ID, an integer for street number, a string for street name, an integer for zipcode, and a string for state. It will dynamically allocate an Address object as well as initialize other members as appropriate
  + Destructor: de-allocate Address pointer and the Item \* in each Order in each deque container in the map. You will need to iterate thru the map. Each entry in the map contains customer orders (deque of Order <Item \*>) for a particular customer. Then iterate thru each deque container and delete all Item pointers in each deque.
  + Overload assignment operator operator= (you may ignore the map)
  + Member functions:
    - Init: read a text file to load customer orders
    - Menu: display a menu (invoked by start function below)

                            MENU

1. Show All Orders
2. Find Order by Customer ID
3. Process Order
4. Quit

      Please enter an option (4 to quit):  1

* Start: use a do while loop to display a menu and invoke ShowAllOrders, FindOrder, ProcessOrder per user option
* ShowAllOrders: display all orders by customer ID
* FindOrders: input a customer ID and display all orders by that customer
* ProcessOrder: input a customer ID and an order ID then process (remove it from the deque) the order for that customer

**Implementation Requirements**

* Each class must have default constructor, non-default constructor, destructor (to free memory or output information)
* Use STL containers (map and deque) and their iterators
* The text file must contain 16 customers (or more): 4 customer have one order, 4 customers have 2 orders, 4 customer have 3 orders, 4 customers have 4 orders
* In FindOrder and ProcessOrder functions (OrderProcessingSystem) throw an Exception whenever an invalid Customer ID or invalid Order ID is found. Whatever function that invokes those functions must do a try/catch to handle the exception. You may ignore and write an error message or simply terminate the program
* In text file reading throw an exception if there is a problem reading the file such as: can't open the file, invalid data. Do try/catch to handle the exception in the main program.

            int main  (  )  {

                 OrderProcessing \*  OrderSystem = ....................;

                 try {

                            OrderSystem->init ( ) ;

                 }   catch  (Exception e)  {

                               // handle exception here

                 }

           }

**Order Processing System Requirements**

* **Init:**read a text file with format shown below

**Sample of text file format:**

The text file format is almost exactly the same as lab 3’s text file format. You only need to add three more fields (Customer ID, Order ID, and Order Date at the end of each eProduct record or DairyProduct record to represent a full customer order.

<Product type> Name;Quantity;Price;<Product data>;**CustomerID;OrderId;OrderDate**

**What it mean is that your previous file reading code remains basically the same up to the point of creating an Item \*  item = new eProduct (….);**

**After that you may add code to read Customer ID, Order ID, and Order Date (by using getline function).**

**Now you will create an Order<Item \*> object and add it to the map (using Customer ID). Note that a Customer might already have an entry in the map. In that case you only need to add the order to the deque container for that customer.**

**////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////**

EP;Panasonic DVD Player;35;125.99;D;**BestBuy;BB123;1414936800**

DP;Dutch Cheese;234;4.95;1414836800;**Walmart;WM1234;1414846843**

EP;Sony Camcoder;125;395.99;U;**BestBuy;BB26781;1414857810**

EP;Apple iPhone 6;1200;399;N;**BestBuy;BB8918;1414832833**

DP;Horizon Milk;100;6.95;1414739810;**Walmart;WM56901;1414832655**

DP;Dutch Cheese;234;4.95;1414836800;**Lucky;LK1234;1414831234**

DP;Butter;235;3.25;1414936822;**Lucky;LK239;1414863512**

DP;Ice Cream;300;8.95;1414986807;**Lucky;LK345678;1414632100**

EP;Samsung TV;15;298.99;R;**Frys;FR6712456;1414837845**

**////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////**

**Main Program:**

* Declare a pointer to OrderProcessingSystem and initialize it with a dynamically allocated OrderProcessingSystem object (non-default constructor)
* Invoke Init function for the OrderProcessingSystem pointer
* Invoke Start function
* De-allocate the OrderProcessingSystem pointer