**Program Specifications**

Write a simple inventory management system for an electronics store. The inventory system should have the following functionalities:

* BuildInventory: read a text file containing electronics products information and store them in an array of pointers
* ShowInventory: display all inventory items
* ShowDefectInventory: display inventory items that are defective
* QueryProduct: given a product ID from user input display the product information

**Class Design**

You need at least three classes.

**class  InventorySystem** (minimum implementation specified below)

* Private data members
  + StoreName
  + StoreID
  + ItemList (array of pointers to InventoryItem objects, max of 512)
  + ItemCount (tracking how many items are stored in the array)
* Constructors (always initialize all pointers in the array to NULL)
  + Default constructor: set data members to 0, NULL, or “” as appropriate
  + Non-default constructor: taking a string for store name, an integer for store ID
* Destructor: de-allocate dynamic memory in the array of pointers (up to ItemCount elements)
* Public member functions:
  + BuildInventory: read a text file containing eProduct records, dynamically allocate eProduct objects and store the objects in the array ItemList
  + ShowInventory: display all items in the inventory. Output must be properly formatted (2 digit after decimal point)
  + ShowDefectInventory: display inventory items whose conditions are defective
  + SearchItem: ask user for a product ID, then search the ItemList array and display the product information if found or an error message if not found

**class InventoryItem**  (minimum implementation specified below)

* Protected data members: Name, Quantity
* Constructors
  + Default constructor
  + Non-default constructor
* Destructor
* Public member functions: SetName, GetName, GetQuantity, AddQuantity, DisplayItem

**class Product**: derived from InventoryItem class (minimum implementation specified below)

* Protected data members: ProductID (to be generated randomly. See code below), Price
* Constructors
  + Default constructor (set ProductID to 0, Price to 0.0)
  + Non-default constructor (randomly generate Product ID and set Price)
* Destructor
* Public member functions: SetID, GetID, SetPrice, GetPrice, DisplayItem (invoke DisplayItem from base class, then display its own data)
* Define one more member function that is specific to this class

**class eProduct:**derived from Product class (minimum implementation specified below)

* Private data members: Condition

Note: Condition is an enum variable describing the eProduct object’s current condition. Possible eProduct object’s conditions are: NEW, REFURBISHED, USED, DEFECTIVE

* Constructors
  + Default constructor
  + Non-default constructor
* Destructor
* Public member functions: GetCondition, SetCondition, DisplayItem (invoke DisplayItem from base class, then display its own data)
* Define one more member function that is specific to this class

**Implementation Requirements**

* Use member initializer syntax in constructors
* Destructor should output "<class name> object [data] is destroyed ..." For example: "eProduct object [NEW] is destroyed ..." "Product object [1234,85.95] is destroyed .." "InventoryItem object [DVD Player, 25] is destroyed ..."
* Member functions that do not modify data members must be defined as const
* The order of class declarations in your source file is InventorySystem, InventoryItem, Product, eProduct. Since InventorySystem contains InventoryItem array of pointers you must use forward declaration on InventoryItem
* Must use static\_cast to downcast the InventoryItem pointers to eProduct pointers before accessing eProduct objects' functions
* Your text file must contain at least 16 inventory items
* 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 ShowDefectInventory
  + Use a loop to ask user either enter ‘S’ or ‘s’ for Search an inventory item or ‘Q’ or ‘q’ to quit the program
  + De-allocate InventorySystem object