Data Structure and Algorithms I

Programming Assignment #5 – option 2

Purposes:
Work with a class implementing a linked list, validating it with an interactive test program. Read in from a file and write to the file.

Description:
Implement a linked list ADT and an application program to simulate the operation of a grocery store checkout system using linked list ADT. You should first build a data structure to contain information on all the products available in the store. The program should print the cash register receipt for each customer.

Input for the program
There will be two sources of input for the program: the inventory information is input from a text file, and the customer transactions are input from the keyboard.

1. The information about each product carried by the store is listed on a single line in the inventory file "Invent.in", in the following format:
   <*product number*> <d*escription*> <*price*> <*tax*>
   where <product number> is a five-digit positive (nonzero) integer,
   <description> is a string of at most 20 characters with no embedded blanks,
   <price> is a real number, and
   <tax> is a character ('T' if the product is taxable; 'N' if it is not taxable).
   The data in the inventory file is NOT ordered.
2. The customer transactions are input from the keyboard, in the following format:
   <*product number*> <*times*>
   where <product number> is as described above, and <times> is an integer in the range 1 to 100, indicating the quantity of this product desired. A zero input for <product number> indicates the end of the customer's order. (Assume the store has no scanner to input the product number, but for now it is typed in by the cashier.)

Output of the program
The program outputs should be written to a text file called "Receipts.out".

1. The inventory information is echo printed to the output file, but *in ordered* from smallest to largest product number.
2. The cash register receipt for each customer is written to the output file. The receipts should be nicely formatted, with the product description (not the product number), number of items, item price, and total price for each product printed on a single line. At the end of each customer's order, the subtotal for all items, amount of tax on taxable items, and total bill should be printed, clearly labeled. (Determine the tax rate as you like, but reasonable.)

Processing
The program first reads in all of the inventory information from file "Invent.in", echo printing the information to the output file. The inventory information should be stored in

sorted linked list of product records. You may assume that the maximum number of products is 50. The program then prompts the cashier to begin inputting the order for the first customer. This customer's order is processed, and the receipt is printed to the output file. After each customer's order is processed, the program should ask the cashier if another customer is to be processed. For instance, if the answer is 'Y', the program sets up to process the next customer; if the answer is 'N', the program terminates with a friendly message.

Error Checking
The following input errors are possible and should be handled by an exception class:
1. Duplicate <product number> in inventory file. Write an error message to the output file and skip the second entry.
2. <Product number> not in inventory file. Write an error message on the receipt, ignore that product number, and continue with the next item.
3. <Times> not in the specified range. Write an error message to the output file, ignore that line, and continue with the next item.

Example
From File "Invent.in":
```
11012 gallon-milk 1.99 N
11014 butter 2.59 N
11110 pie-shells 0.99 N
20115 laundry-soap 3.60 T
30005 homestyle-br 0.99 N
```
From keyboard (one customer):
```
11110 2
40012 3
20115 1
0
```

To "Receipts.out" file:
```
------------------------------------------------------
Nov 10, 2008 6:00 pm
Customer 1
pie-shells 2 @ 0.99 1.98
*** item 40012 not in inventory ***
laundry-soap 1 @ 3.60 3.60 TX
Subtotal $5.58
Tax $0.27
Total $5.85
```

This is only an example. You are encouraged to add more features and make it look nicer and more realistic. Be sure to have the amount of money in the suitable output format.

Notes:

Your test driver should be interactive and clear to user. It should allow user to validate all the required functionalities thoroughly.
For the submission, make sure that your sample outputs demonstrate not only that your ADT has all the necessary implementations, but also that they work correctly.