CS 302 Data Structures

Fall 2012

Programming Assignment 2

Due on Monday, Oct 1 at 1:00 pm

Write, run, and test a C++ program to simulate the operation of a grocery store checkout system. Your program should first build a data structure to contain information on all the products available in the store. Then the program should print the cash register receipt for each customer.

Input

The input for this program has two sources: 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.dat`", 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 12 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 ordered from smallest to largest product number.
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. (The store will soon be using a scanner to input the product number, but for now it is typed in by the cashier.)

Output

The program outputs should be written to a text file called "`Receipts.out`".

1. The inventory information is echo printed to the output file.
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. (The tax rate is 7.5% for taxable items.)

Processing

The program first reads in all of the inventory information from file "`Invent.in`", echo printing the information to the output file. 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. 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:
-------------------------------------------------------
SEP 10, 1998 6:00 pm (* date and time optional *)
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
```

Data Structures

The inventory information can be stored in an array of product records. Assume that the maximum number of products is 50, for purposes of writing this program. You should specify this data structure as an ADT, as described in the chapter; define and implement a set of operations to encapsulate it. The ADT should be tested using a test driver.

Deliverables

- Your design (either object-oriented or top-down)
- A listing of the ADT
- A listing of the test driver for the ADT
- A listing of the test plan as input for the driver
- A listing of the output from the test driver
- A listing of your program
- A listing of your test plan as input to the program
- A listing of the output file

PROGRAM SCHEDULE

Milestone	*Date Completed* *Planned*	*Actual*
Assignment received.		
Requirements understood; detailed specification recorded.		
Top level of design complete.		
All levels of top-down design complete; data structures determined.		
Coding complete (clean compile).		
Test plan complete.		
Testing complete		
Program ready to turn in; all external and internal documentation complete.		
Assignment turned in.		