IPC144 Exam – Part C -- Programming Question [50% of Final Assessment]

Seneca Groceries

This question is designed to be done by a group of 1-3 students. Submissions from groups of more than 3 students will not be accepted and all group members will receive zero for this portion of the assessment. Each group member must write and submit their own reflections which will be different from the reflections from other group members. Only 1 member of the group has to submit the code and every file must have a comment at the top identifying all members of the group.

You have been asked to write a program for a grocery store that will:

- Start by allowing the entry of the current stock in the store, which it will save and update during the day.
- Switch to sales mode where it will act like a cash register and allow customers to purchase things. It will update the inventory as items are sold and will produce an itemized receipt [CG1] for the customer.
- At the end of the day, print a summary of the sales, the inventory remaining and the top sellers in each category.

Products for sale are identified by name, and are placed in a category and the category is entered by its number:

1. produce
2. bakery
3. meat
4. dairy
5. baking
6. house wares
7. miscellaneous

Each item has a price and an indication of whether it is sold per unit or by weight which is entered as 0 if sold by the number purchased or 1 if sold by weight. It also indicates how much we have of this item either in units or by weight rounded down to the nearest kilogram.

When we sell items, they are sold by weight in fractional Kg or by integral unit counts. When the final amounts in inventory are calculated, they are rounded down to the nearest Kg for items sold by weight. If a customer tries to purchase more of something than we have in stock, we sell only the amount we have in stock. Most groceries are tax free except items in the house wares and miscellaneous categories which are taxed at 13%.

The list of items being purchased is terminated by an item with the ID of 0. A sale with no items added to it marks the end of sales which should be followed by the summary of the day's sales.

Input data should be checked for correct category range (1-7), sold by weight range (0-1). As the stock is entered, each item is assigned a numeric identifier from 1 upwards in the order the items are entered. For stock entry, you can assume that each item is only entered once.

When purchasing, the customer will identify the product to purchase by the identifier assigned to it when it was entered in the stock and listed in the initial stock report in the ID column. These product ID number should be checked to ensure that they are valid.

Below this section you will find:

- main.c which is the main for the program. You must use this main.c for your program without changes. This means that you will need to implement the data structures it uses and the functions it uses in the files stock.h and stock.c. You SHOULD create additional functions, other than the ones called from main, to create a highly modular, well-designed program. Your code will be tested against this main.c and your code must work and produce the expected output.
- A sample execution of the program to which your program's output should look very similar.
- Sample data which can be cut and pasted into the program for final testing. This is the data which generated the sample output.

You are required to:

- In a group of 1, 2 or 3 people create the missing code and have 1 group member submit the files stock.h and stock.c as a zip file. (Groups attempting the bonus question must also include main.c if they modified it) Submission from groups of more than 3 people will not be accepted and all members of these groups will receive 0 for this portion of the final assessment.
- Place a comment at the top of both stock.h and stock.c listing the names of every member of the group.
- Individually write a reflection based on the questions below and submit it on Blackboard. This means that a group of 3 students will need to submit 3 different reflections.
- BONUS: (5%) if you read the existing stock from a file containing the test data with the errors fixed. You will need to correct the errors in the data and place it in a file. Then, you can read it from the file without the error checking that is done with the manually typed version. You can change the value of the constant `STOCK_FROM_STDIN` declared in main to allow you to read from a file. Note that the sales will continue to be read from stdin, it is only the initial stock which will be read from the file. This will change the output to show the initial prompt for stock followed immediately the summary of the opening stock. The typing of the stock will vanish as it is being read from a file.

Rubric

CODE (30% of takehome mark, 15% of Final Assessment mark):

Correct Solution to problem	30%
Appropriate data structures	20%
Modular programming used	30%
Efficiency of coding	10%
Quality of code	10%
Bonus File Reading	5%

REFLECTIONS (70% of takehome mark, 35% of Final Assessment Mark):

Correctness	40%
Sufficient detail	50%
Grammar	10%

Reflections

In your reflections you should present your answers in depth and suitable answers should take at least 100 words for each of the reflections. That implies a minimum of 300 words for all of the reflections. Each member of the group must write their own reflections and submit them as an individual.

1. This is a highly modular program consisting of multiple functions. For each function you **personally** wrote, describe what it does and why it deserves to be a function. Be sure to include the functions you added which were not called directly from main.
2. The input for the program is different from what you are accustomed to seeing. Rather than prompting for each individual value, it prompts for several values at once. What benefit does this provide? How does it complicate the reading of the input? How were you able to handle the commas separating the input values?
3. Finding the top sellers in each category requires the development of an algorithm. Describe how you went about developing this algorithm. Then, describe how your algorithm works.

Sample Output

Sample output from the program is shown below. Note that the input values are separated by commas and that you should do the same in your program.

```
Enter current stock in format amount, category, price, byWeight, name (0
amount to end):
21,9,1.99,0,broccoli
Invalid Category - Enter a number between 1 and 7: 1,1.99,2,broccoli
Invalid soldByWeight - Enter a number between 0 and 1: 0,broccoli
21,1,1.5,1,potato
3, 1,1.5,0,carrots
10,3,12.99,1,steak
2,3,9.5,1,Italian sausage
2,4,4.99,0,2% milk
2,4,3.99,0,coffee cream
2,2, 3.99,0, white bread
5,2,4.99,0,chocolate cup cakes
10,5,4.99,0,all purpose flour
8,5,5.99,0,whole wheat flour
5,5,2.99,0,raisins
2,6,0.99,0,scrub pads
4,7,13.95,0,laundry detergent
0

***************** Seneca Groceries - Opening Stock *****************
====================================================================
   ID         Product     Category     Price  Quantity
    1         broccoli     produce      1.99     21
    2          potato     produce      1.50     21
    3         carrots     produce      1.50      3
    4          steak       meat     12.99     10
    5    Italian sausage       meat      9.50     2
```

* scanf 할 때 %.2lf로 받으면 값 없어짐
  근데 printf에만 %.2lf 하면 잘 됨 그냥 %lf로 받고

* 각각 에러체크 하는데 추가하면 어떨까?
  ㄴ 근데 지금 for 안에 while만 2개

* flag 갯수?? ㄴ for loop 이랑 while loop 2개만 사용

70

* array of character로 값을 매번 초기화 후 strcpy로 입력함 혹시 pointer 방법 있나 생각 필요

```
   6          2% milk      dairy     4.99     2
   7        coffee cream      dairy     3.99     2
   8        white bread      bakery     3.99     2
   9     chocolate cup cakes      bakery     4.99     5
  10      all purpose flour      baking     4.99    10
  11      whole wheat flour      baking     5.99     8
  12          raisins      baking     2.99     5
  13        scrub pads      house wares     0.99     2
  14      laundry detergent    miscellaneous    13.95     4

************************ Now in Sales Mode ************************
=================================================================
Enter a product ID to purchase, and the quantity (0 to stop): 1,2
Enter a product ID to purchase, and the quantity (0 to stop): 20,3
Invalid Product - Enter a number between 0 and 14: 2, 200
Invalid quantity - Enter a number between 0.10 and 100.00: 2,2
Enter a product ID to purchase, and the quantity (0 to stop): 3,2
Enter a product ID to purchase, and the quantity (0 to stop): 4,1
Enter a product ID to purchase, and the quantity (0 to stop): 6,3
Enter a product ID to purchase, and the quantity (0 to stop): 8,1
Enter a product ID to purchase, and the quantity (0 to stop): 0

********************** Seneca Groceries **********************
=================================================================
          broccoli    1.99     3.98
           potato    1.50     3.00
          carrots    1.50     3.00
            steak     12.99    12.99
             2% milk      4.99     9.98
         white bread      3.99     3.99
Purchase Total             36.94
Tax                   0.00
Total                 36.94
Thank you for shopping at Seneca!

Enter a product ID to purchase, and the quantity (0 to stop): 5,2
Enter a product ID to purchase, and the quantity (0 to stop): 7,1
Enter a product ID to purchase, and the quantity (0 to stop): 10,2
Enter a product ID to purchase, and the quantity (0 to stop): 14,1
Enter a product ID to purchase, and the quantity (0 to stop): 0

********************** Seneca Groceries **********************
=================================================================
       Italian sausage    9.50    19.00
         coffee cream      3.99     3.99
      all purpose flour      4.99     9.98
      laundry detergent     13.95    13.95
Purchase Total             46.92
```

```
Tax                 1.81
Total               48.73
Thank you for shopping at Seneca!

Enter a product ID to purchase, and the quantity (0 to stop): 0

************************ End of Day Summary ************************
======================================================================
    Cost of items sold before taxes   83.86
         Number of Sales    2
      Average items per sale    5.00


************************** Closing Stock ***************************
======================================================================
  ID           Product     Category    Price Quantity
   1           broccoli     produce     1.99    19
   2            potato     produce     1.50    19
   3           carrots     produce     1.50     1
   4            steak       meat    12.99     9
   5       Italian sausage      meat     9.50     0
   6           2% milk      dairy     4.99     0
   7         coffee cream      dairy     3.99     1
   8         white bread     bakery     3.99     1
   9     chocolate cup cakes     bakery     4.99     5
  10      all purpose flour     baking     4.99     8
  11      whole wheat flour     baking     5.99     8
  12           raisins     baking     2.99     5
  13         scrub pads     house wares     0.99     2
  14      laundry detergent   miscellaneous   13.95     3

------------ Top 3 sellers in produce ------------
Rank          Product   Sales
   1          broccoli    2.00
   2            potato   2.00
   3          carrots    2.00
------------ Top 3 sellers in bakery -------------
Rank          Product   Sales
   1        white bread    1.00
   2          <none>   0.00
   3          <none>   0.00
------------- Top 3 sellers in meat --------------
Rank          Product   Sales
   1       Italian sausage    2.00
   2           steak    1.00
   3          <none>   0.00
------------- Top 3 sellers in dairy -------------
Rank          Product   Sales
```

```
   1          2% milk    2.00
   2        coffee cream    1.00
   3           <none>  0.00
------------ Top 3 sellers in baking -------------
Rank          Product   Sales
   1      all purpose flour    2.00
   2           <none>  0.00
   3           <none>  0.00
---------- Top 2 sellers in house wares ----------
Rank          Product   Sales
   1           <none>  0.00
   2           <none>  0.00
--------- Top 2 sellers in miscellaneous ---------
Rank          Product   Sales
   1      laundry detergent    1.00
   2           <none>  0.00
```

Sample Data

```
21,9,1.99,0,broccoli
1,1.99,2,broccoli
0,broccoli
21,1,1.5,1,potato
3, 1,1.5,0,carrots
10,3,12.99,1,steak
2,3,9.5,1,Italian sausage
2,4,4.99,0,2% milk
2,4,3.99,0,coffee cream
2,2, 3.99,0, white bread
5,2,4.99,0,chocolate cup cakes
10,5,4.99,0,all purpose flour
8,5,5.99,0,whole wheat flour
5,5,2.99,0,raisins
2,6,0.99,0,scrub pads
4,7,13.95,0,laundry detergent
0
1,2
20,3
2, 200
2,2
3,2
4,1
6,3
8,1
0
5,2
7,1
10,2
14,1
```

0

* numStockItems : the total number of products
(line 42) from readStockItems function
(kind of large)

* numSaleItems : the number of total Product ID
(line 70) from readSale function

* saleItem : How many a user input valid data in
(line 70) readSale function (kind of large)
- Declared as struct in line 30

* sale : total prices per a sale
(line 76) from printSalesReport

* totalSales : total prices sumed all of sales' prices
(line 76)

* totalSaleItems : the total number of all of the number of ID
(line 80) in every sale

* numSales : 2! cuz total 2 times displayed
(line 82) top 2 rank

* topSellers : maybe store the top 3 Items in there
(line 32) from getTopSellers function
- struct of SalesRecord

* cat : used to find a Category
(line 121) from getTopSellers function
- total 5 times displayed top 3 ranks
that's why cat is less than 5

* 3 or 2 : used to display as Top3 or Top2
(line 123) from printTopSellers function

* findValid fuction 구현

↳ 내 pc 구동을 원해서면 필요

* Negative value error Message

<0> broccoli ⇒ 1
<1> Potato ⇒ 2
<2> Carrot ⇒ 3
} "3"

① for (firstIndex = 3-1; firstIndex > 0; firstIndex--)

② for(secondIndex = 0; secondIndex < firstIndex; secondH)
2

③ if (secondIndex번째 숫자 < secondIndex+1 번째 숫자

④ emptyDouble = secondIndex번째 숫자

⑤ secondIndex번째 숫자 = secondIndex+1 번째 숫자

⑥ secondIndex의 변화과정 = emptyDouble

<0> broccoli ⇒ 1

<1> Potato ⇒ 2

<2> carrot ⇒ 3

(1)

① firstIndex=2, 2>0 True

② second=0, 0<2 True

③ 1 < 2 True

④ empty ⇐ 1 <0>=2

1 <= 2 <1>=0

2 <= 0 <2>=1

(2)

② second=1, 1<2 True

③ 0<1 True

④ empty<=0 <0>=2

0<=1 <1>=1

1<=0 <2>=0

(3)

② second=2, 2<2 false

(4)

① first=1, 1>0 True

② second=0, 0<1 True

③ 2<1 false

(5)

② second=1, 1<1 false

(6)

① first = 0 , 0 > 0 false

End of loop

③ firstIndex=2
secondIndex=0 | 1 > 2 : false

③ firstIndex=2
secondIndex=1 | 2 > 3 : false

① firstIndex=1 ~~low to high~~

② firstIndex=1
secondIndex=0 |

③ firstIndex=2
secondIndex=0 | 1 < 2 : True

3-1 emptydouble = 1

1)

3-2 1 = 2

3-3 2 = 0

③ firstIndex=2 secondIndex=1 | 2 < 3 : True

3-1 emptyDouble = 2

2 = 3

3 = 0

③ firstIndex=2 secondIndex=2 false

③ firstIndex=1 secondIndex=0 | 2 < 3 True

3-1 copy = 2

2 = 3

3 = 0

③ firstIndex = 1
secondIndex = 1 / 3 < 0 false

③ firstIndex = 0 : false

11