**Introduction to Computer Science – 150005**

**Homework Assignment #12**

**Records**

**Comments:**

1. At the end of each question, you need to provide a main program that shows the correctness of the functions that you wrote.
2. Use meaningful variable names
3. Comment each program, including a comment before the main program and each function explaining their purpose and how they work, and a comment at the end of the program containing test runs with input and output.
4. Be careful on code readability and appearance (indentation)
5. Make sure to compute exactly what is requested in each question.
6. In the examples below, output is marked in green, input in yellow.
7. Reminder: submit your own work!

**Important notice: At times the automatic checker does not accept the delete statement, and therefore won’t assign a grade of 80. In this case, comment out the delete statement before resubmitting. (Note, comment out but don’t delete the statement, since a missing delete statement will not get full credit on the manual grading.)**

A store has many different items for sale. For each item in the store, we need to record the following information:

* Product code (integer)
* Product name (string up to 20 characters)
* Current amount in stock (integer)
* Minimum amount needed in stock (integer)
* Price of product (real number)

The store needs to handle the following

* Add a new product to the store
* Cost of product given a product code
* Sale of a product (reduces amount in stock)
* Reorder of product whenever amount in stock falls below minimum
* Print stock

Write a program that contains the following data:

* Define a struct whose name is *Item* to represent the structure of a product.
* Define an array called store which represents the products in the store. At the start of the program, you should input the maximum number of products which the store can have and build the array accordingly.

The program should define the following functions:

* addItem – Adds a new product. The function receives the array *store,* the maximum number of products it can have, and the current number of products in *store*. The function inputs from the user the information about the new product as follows:  
    
    
  If there is room to add the new product to *store*, it is added to the end of *store* and the actual number of products is incremented. If a product with the same code already exists, then if the name is also the same, them the function just updates the amount in stock. If the name is different, or if there is no more room in the array, the function prints ERROR.
* findPrice – Finds the price of a product. The function receives the array *store* and the current number of products in *store*. The function inputs from the user the desired product code If the product exists in the system, the function prints  
  price: xxx.  
  If not, it prints ERROR.
* sold – Records a sale. The function receives the array *store* and the current number of products in *store*. The function inputs from the user the desired product code and the number of items to be sold.  
    
    
  The function updates the number of items in stock for the product such that the amount sold is deducted from the number of items in stock. If the product code does not exist, the function prints ERROR.
* order – Update the stock. The function receives the array *store* and the current number of products in *store*. The function iterates over all products in the database and for those products whose current number of items in stock is less than the minimum required for the product, the function prints out the name of the product and the amount to order (so that the number in stock for the product is 5 more than its minimum).  
  item name: xxx  
  code: yyy  
  amount to order: zzz  
    
  It then updates the number of items in stock for the product.
* print – Prints out current stock. The function receives the array *store* and the current number of products in *store*. The function prints out all the information for each product each on a separate line.  
  name: xxx  
  code: yyy  
  amount: zzz  
  minimum amount: aaa  
  price: bbb

Advice: Use top down design approach. You can use the main program provided below. Feel free to make changes. You must write the missing functions.

Note: It is recommended to hand in a partial solution if you did not complete the assignment. The solution will receive an appropriate grade from the manual grading.

enum cases{EXIT,ADD,FIND,SOLD,ORDER,PRINT};

int main()

{

Item\* store;

int maxItems;

int numItems = 0;

int choice;

cout << "enter max number of items: " << endl;

cin >> maxItems;

store = new Item[maxItems];

do {

cout << "enter 0-5:\n";

cin >> choice;

switch (choice) {

case EXIT: break;

case ADD: addItem(store, maxItems, numItems);

break;

case FIND: findPrice(store, numItems);

break;

case SOLD: sold(store, numItems);

break;

case ORDER: order(store, numItems);

break;

case PRINT: print(store, numItems);

break;

default: cout << "ERROR" << endl;

}

} while (choice != 0);

system("pause");

return 0;

}

enter max number of items:

2

enter 0–5:

1 // code 1: add item

enter code:

100 // new item

enter name:

paper

enter amount:

200

enter minimum amount:

50

enter price:

28.90

enter 0–5:

1 // code 1: add item

enter code:

101 // new item

enter name:

pens

enter amount:

75

enter minimum amount:

80

enter price:

6.90

enter 0–5:

1 // code 1: add item

enter code:

100 // update existing item

enter name:

paper

enter amount:

50

enter minimum amount:

50

enter price:

28.90

enter 0–5:

1 // code 1: add item

enter code:

102

enter name:

pencils

enter amount:

300

enter minimum amount:

250

enter price:

1.75

ERROR // No space

enter 0–5:

5 // code 5: print

name: paper

code: 100

amount:250

minimum amount:50

price: 28.90

name: pens

code: 101

amount: 75

minimum amount: 80

price: 6.90

enter 0–5:

2 // code 2: search price

enter code:

103

ERROR

enter 0–5:

2 // code 2: search price

enter code:

101

price: 6.90

enter 0–5:

4 // code 4: order

item name: pens

code: 101

amount to order: 10

enter 0–5:

5 // code 5: print

name: paper

code: 100

amount:250

minimum amount:50

price: 28.90

name: pens

code: 101

amount: 85

minimum amount: 80

price: 6.90

enter 0–5:

3 // code 3: sell item

enter code:

100

enter amount:

200

enter 0–5:

5 // code 5: print

name: paper

code: 100

amount:50

minimum amount:50

price: 28.90

name: pens

code: 101

amount: 85

minimum amount: 80

price: 6.90

enter code:

0