Structure Types

Shopping V1.0

Workshop 5 (worth 3% of your final grade)
URL: https://github.com/Seneca-144100/IPC-WS5

In this workshop, you are to write an application that manages an inventory.
Application Features:

- Customers are able to view an inventory of items.
- Customers are able to quickly search for an item and get the price.
- Your application interacts with the customer.
- Your application adds items to the inventory.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities to:

- design a structure type
- access the members of an object of structure type.

SUBMISSION POLICY

Your workshops are divided into two sections; in_lab and at_home.

The "in_lab" section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of the workshop. If you do not attend the workshop, you can submit the "in_lab" section along with your "at_home" section (a 20% late deduction will be assessed). The "at_home" portion of the lab is **due the day before your next scheduled workshop**

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible for regularly backing up your work.

IN-LAB: ITEM CLASS (70%)

Download or clone workshop 5 from https://github.com/Seneca-144100/IPC-WS5

Write this section of your code in shopping_lab.c

1.
In this workshop, you are going to use a C structure type to represent an **Item** of an inventory. A person is able to view the inventory, add an item to the inventory, and check prices for items in the inventory. Here is the C structure type that should be used in this workshop:

```
struct Item{
        int sku_;
        float price_;
        int quantity_;
};
```

sku_: Stock Keeping Unit number for an item.
price_: Price for an item.
quantity_: Quantity of an item.

Also, use **const** or **#define directives** to define the following number:
MAX_ITEMS: the maximum number of items that exists in an inventory. Assume **MAX_ITEMS** is 10**.**

In your main function, implement the following steps:

1. Define the following variables in your main program:
struct Item item[MAX_ITEMS]; //An array of Item representing the inventory
int size=0; //Number of items in the inventory. The inventory is initially empty.

2. Display a welcome massage:
Welcome to the Shop
==========

3. Display the following menu, and prompt the user to select an option from the menu.

Please select from the following options:
1) Display the inventory.
2) Add to the inventory.
0) Exit.

4. You must verify that the input integer is between zero and two inclusive. If the input number is not between zero and two, you must display a warning and prompt the user again. You can assume the user will only enter numbers. The program only exits when the user selects 0 (Exit) on the menu screen (looping required).

5. Depending on the user's input, one of the following scenarios happens.

Add to the inventory:

- Prompt the user to input the SKU number and the quantity of an item that one wants to add to the inventory.
- Search through the inventory (the **`item`** array) to find if the item exists in the array.
- If the item is found in the inventory, do the following:
    - Update the quantity of the item by adding the quantity read to the quantity of the item in array.
    - Display a message that the item quantity is successfully updated.
- If the item is not found in the inventory, do the following:
    - If the inventory is full (**`size== MAX_ITEMS`**), display the inventory is full.
    - If the inventory is not full, the function prompts the user to input item price. Then, update the inventory. Hint: Use the **`inventory`** array, and size as its index, and assign sku, price and quantity. Increment size.

Display the inventory:

- Display the inventory in an informative format. See sample output.

Exit:

- Display a goodbye message:
  Goodbye!

Program completion

Your program is complete if your output matches the following output. Red numbers show the user's input.

```
Welcome to the Shop
==================
Please select from the following options:
1) Display the inventory.
2) Add to shop.
0) Exit.
Select:8
Invalid input, try again: Please select from the following options:
1) Display the inventory.
2) Add to shop.
0) Exit.
Select:2
Please input a SKU number:2341
Quantity:3
Price:12.78
The item is successfully added to the inventory.
Please select from the following options:
1) Display the inventory.
2) Add to shop.
0) Exit.
Select:2
Please input a SKU number:4567
Quantity:9
Price:98.2
The item is successfully added to the inventory.
```

```
Please select from the following options:
1) Display the inventory.
2) Add to shop.
0) Exit.
Select:1


Inventory
=========================================
Sku         Price       Quanity
2341        12.78       3
4567        98.20       9
=========================================
Please select from the following options:
1) Display the inventory.
2) Add to shop.
0) Exit.
Select:2
Please input a SKU number:2341
Quantity:5
The item exists in the repository, quanity is updated.
Please select from the following options:
1) Display the inventory.
2) Add to shop.
0) Exit.
Select:1


Inventory
=========================================
Sku         Price       Quanity
2341        12.78       8
4567        98.20       9
=========================================
Please select from the following options:
1) Display the inventory.
2) Add to shop.
0) Exit.
Select:0
Good bye!
```

For submission instructions, see the SUBMISSION section below.

IN_LAB SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above or any information needed.

If not on matrix already, upload your **`shopping_lab.c`** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit ipc_w5_in_lab <ENTER>
```

and follow the instructions.

AT_HOME: TITLE (20%)

Copy shopping_lab.c to shopping_home.c and add the following option to the menu:

3) Price check.

Make sure that the entry validation (step 4) in menu entry is updated to include 3 as a valid entry.
Then add the following logic:

Price check

- Prompt the user to input the SKU number for the item they are looking for.
- Search through the inventory and display the cost of the item.
- Note that if the item does not exist in the inventory, the program displays an informative message.

Program completion

Your program is complete if your output matches the following output. Red numbers show the user's input.

```
Welcome to the Shop
==================
Please select from the following options:
1) Display the inventory.
2) Add to shop.
3) Price check.
0) Exit.
Select:2
Please input a SKU number:2341
Quantity:4
Price:12.78
The item is successfully added to the inventory.
Please select from the following options:
1) Display the inventory.
2) Add to shop.
3) Price check.
0) Exit.
Select:2
Please input a SKU number:4567
Quantity:9
Price:98.20
The item is successfully added to the inventory.
Please select from the following options:
1) Display the inventory.
```

```
2) Add to shop.
3) Price check.
0) Exit.
Select:1


Inventory
=======================================
Sku         Price       Quanity
2341        12.78       4
4567        98.20       9
=======================================
Please select from the following options:
1) Display the inventory.
2) Add to shop.
3) Price check.
0) Exit.
Select:3
Please input the sku number of the item:
2322
Item does not exist in the shop! Please try again.
Please select from the following options:
1) Display the inventory.
2) Add to shop.
3) Price check.
0) Exit.
Select:3
Please input the sku number of the item:
2341
Item 2341 costs $12.78
Please select from the following options:
1) Display the inventory.
2) Add to shop.
3) Price check.
0) Exit.
Select:0

Good bye!
```

AT-HOME REFLECTION (10%)

Please provide brief answers to the following questions in a text file named **`reflect.txt`**.

1) What is a structure type?
2) What is the difference between a structure type and an array?
3) How do you access a member of an object of a structure type?

AT_HOME SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload your **shopping_home.c and reflect.txt** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit ipc_w5_at_home <ENTER>
```

and follow the instructions.