Functions

Shopping V2.0

Workshop 7 (worth 3% of your final grade)
URL:

In this workshop, you are to decompose your workshop 5 into seven functions.

LEARNING OUTCOMES

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

- use pointers to assign to variables from functions.
- decompose a problem into two or more modules.
- code a C function for each module.
- implement structured programming principles, including single-entry/single-exit logic.

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 7 from

Write this section of your code in shopping_2_lab.c

1.
[UNCHANGED from workshop 5]
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:

2.
[UNCHANGED from workshop 5]
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.
```

Display a welcome massage:

```
Welcome to the Shop
==================
```

3.
Write the following function to display a menu. This function takes no arguments and displays the following menu to the user.

```
void menu();
/*
Please select from the following options:
1) Display the inventory.
2) Add to the inventory.
3) Check price.
4) Clear Screen.
0) Exit.
*/
```

4.
Write the following function that prompts the user to input an integer. This function verifies that the integer is within a specified range (**`low`** and **`high`**), if not, the function displays a warning and prompts the user again. This function requires two parameters (low and high range) and returns the validated input from the user. You can assume the user will only enter numbers.

```
int validate(const int low, const int high);
```

5.
Write the following clear screen function. This simple function prints out 40 newlines to clear the screen. It requires no parameters and returns nothing.

```
void clear();
```

6.
Write the following function to display an inventory. This function takes the address of an array of objects of type Item (`items[]`), and an integer `size` representing the size of the array. The function displays the inventory in an informative output. See “program completion” section for the format of the output.

```
void displayInventory(const struct Item items[],const int size);
```

7.
Write the following function to perform a linear (naïve) search over the inventory array. This function receives the address of an array of type Item (`items[]`), an integer for the sku number of the desired item, and an integer `size` representing the size of the array. This function searches through the array for an item with the desired sku number and returns the index of the matching item if it is found, -1 if it is not found.

```
int searchInventory(const struct Item items[],const int sku_item,const int size);
```

8.
Write the following function to add an item to the inventory. This function takes the address of an array of objects of type Item (`items[]`), and the address of an integer `size` representing the size of the array.

```
void addItem(struct Item items[], int *size);
```

This function:

- Prompts the user to input the SKU number and the quantity of an item that one wants to add to the inventory.
- Use **`searchInventory`** function to 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 **item** array, and *size as its index, and assign sku, price and quantity. Increment size. Once the item is added to the inventory, display a message that the item is successfully added to the inventory.

9.

Write the following empty function. This function receives no arguments and displays a message “Not implemented”. You will write this function in at_home section.

```
void checkPrice();
```

10.

Write the main function that

- calls **clear** function to clear the screen,
- displays a welcome massage(Step 2),
- calls the function **menu** to display a menu to the user,
- calls the **validate** function to obtain the user’s input. Note that the user’s input must be within the range 0 and 3 (inclusive).
- Depending on the user’s input, call one of the following functions, and pass proper arguments:
    - If user’s input is 1: Call `displayInventory.`
    - If user’s input is 2: Call `addItem.`
    - If user’s input is 3: Call `checkPrice.`
    - If user’s input is 4: Call `clear.`
    - If user’s input is 0: The program exits and displays a goodbye message.
      (Note: The program only exits when the user selects Exit on the menu screen (looping required).

Program completion

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

```
Start OUPUT
><
><
><
```

><
><
><
><
><
><
><
><
><
><
><
><
><
><
><
>Welcome to the Shop
>==========
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: ***5***
>Invalid input, try again: ***2***
>Please input a SKU number: ***1234***
>Quantity: ***2***
>Price: ***45.63***
>The item is successfully added to the inventory.
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: ***2***
>Please input a SKU number: ***9010***
>Quantity: ***5***
>Price: ***23.50***
>The item is successfully added to the inventory.
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: ***2***
>Please input a SKU number: ***1234***
>Quantity: ***5***
>The item exists in the repository, quantity is updated.
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.

```
>Select: 1
><
><
>Inventory
>=======================================
>Sku         Price       Quantity
>1234        45.63       7
>9010        23.50       5
>=======================================
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: 0
>Goodbye!
><
```

End OUPUT

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_2_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_w7_in_lab <ENTER>`

and follow the instructions.

AT_HOME: TITLE (20%)

Copy shopping_2_lab.c to shopping_2_home.c and update the following function:

Update the following function to check the price for an item. This function receives the address of an array of Item (`items[]`), and an integer representing the size of the array. The function prompts the user to input the SKU number for the item they are

looking for. Then, this function searches through the inventory (use **`searchInventory`**), and displays the cost of the item. Note that if the item does not exist in the inventory, the program displays an informative message.

```
void checkPrice(const struct Item items[],const int size);
```

Program completion

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

Start OUPUT

```
><
><
><
><
><
><
><
><
><
><
><
><
><
><
><
><
><
><
>Welcome to the Shop
>==================
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: 2
>Please input a SKU number: 1234
>Quantity: 23.90
>Price: The item is successfully added to the inventory.
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: 2
>Please input a SKU number: 9010
>Quantity: 5
>Price: 89.20
>The item is successfully added to the inventory.
>Please select from the following options:
```

```
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: 2
>Please input a SKU number: 8721
>Quantity: 4
>Price: 19.99
>The item is successfully added to the inventory.
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: 1
><
><
>Inventory
>========================================
>Sku         Price       Quantity
>1234        0.90        23
>9010        89.20       5
>8721        19.99       4
>========================================
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: 3
>Please input the sku number of the item: 7777
>Item does not exist in the shop! Please try again.
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: 3
>Please input the sku number of the item: 9010
>Item 9010 costs $89.20
>Please select from the following options:
>1) Display the inventory.
>2) Add to the inventory.
>3) Check price.
>4) Clear Screen.
>0) Exit.
>Select: 0
>Goodbye!
><
```

End OUPUT

AT-HOME REFLECTION (10%)

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

1) What are the advantages of using functions?
2) C function syntax only allows for the return of a single value. What do you do if an application requires a function that returns more than one value?
3) Explain the differences between a local scope and a global scope?

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_2_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_w7_at_home <ENTER>
```

and follow the instructions.