CSE102 – Computer Programming

Homework #6

Due Date: 26/04/2023

Hand in: A student with number 20220000001 should hand in a zip file named 20220000001.zip for this homework.

Homework Description: You have been tasked with creating an inventory management system capable of managing the stock of a supermarket with multiple branches. This assignment will provide you with practical experience working with functions, arrays, multi-dimensional arrays, pointers, file I/O, and data manipulation.

The program includes several data manipulation operations, including querying products or stocks, filtering query results, adding, deleting, or updating a product or stock entry, and more. Each requested operation is explicitly explained in the part descriptions, so it is important to read them carefully.

To store the product and stock data, you will need two comma-separated text files: one for products and one for stocks. In the "products.txt" file, the various features of each product will be stored. Meanwhile, the stock status of each product in each branch will be stored in the "stocks.txt" file. Both text files will contain only one entry (one product or one stock) per line.

The first line of the products file will consist of the names of the product features. Similarly, the first line of the stocks file will contain the names of the stock features. By default, each product will have five different features. **However, please note that the number of features may be increased during runtime by adding new features to the system.**

- pID (Integer): This is the unique identifier of the product, and it will be auto-incremented by one for each new product added to the system.
- Type (Character): This represents the type of product and can be one of the following categories: D (for drinks), F (for foods), C (for cleansers), and O (for other products).
- Name (Character[]): This feature represents the name of the product, and it can be a maximum of 8 characters in length.
- Brand (Character[]): This feature represents the brand of the product, and it can be a maximum of 5 characters in length.
- Price (Double): This feature represents the price of the product.

Here is an example of what the product file could look like:

pID,Type,Name,Brand,Price

1,D,Ayran,Sutas,20.5

2,F,Simit,Firin,8.0

…

The stocks file will contain four features:

- sID: This is the unique identifier of the stock entry, and it will be auto-incremented by one for each new entry added to the system.
- pID: This feature represents the unique identifier of the product and must match the id of the product in the products.txt file.
- Branch (Character[]): This represents the name of the city where the branch is located, and it can be a maximum of 15 characters in length.
- Current_stock (Integer): This feature represents the number of products currently stored in the branch.

An example product file is shown below:

sID,pID,branch,current_stock

1,3,Istanbul,100

2,121,Kocaeli,51

…

Part 1. [10 pts] Menu

When the program is executed, it will display the following menu to the user:

Welcome operator, please select an option to continue:

1- File Operations
2- Query products
3- Check stock status
4- Stock control by brand
5- Export report

The available operations are explained in more detail in the following sections. After the user has selected an operation, the menu will be displayed again until the user selects option 5 to terminate the program.

Part 2. [30 pts] File Operations

This option provides a submenu for file operations, allowing the user to perform various actions on the product and stock files. The following actions are available:

1. Add a new product: The user will be prompted to enter the values for a new product, including pID (which will be auto-incremented), Type, Name, Brand, and Price. The new product will be added to the products file.

2. Delete a product: The user will be prompted to enter the pID of the product to be deleted. The product will be removed from the products file, and the pID values of the products beneath the deleted product will be decreased by one, and their lines will be shifted up by one.

3. Update a product: The user will be prompted to enter the pID of the product to be updated, as well as the name of the feature to be updated and the new value for that feature. The pID cannot be updated.

4. Add feature to products: The user will be prompted to enter the name of the new feature to be added to the products file. The feature values will be initialized to None for all products.

5. Add a new stock entry: The user will be prompted to enter the values for a new stock entry, including sID (which will be auto-incremented), pID (matching the id of the product in products.txt file), Branch, and Current_stock. The new stock entry will be added to the stocks file.

6. Delete a stock entry: The user will be prompted to enter the sID of the stock entry to be deleted. The stock entry will be removed from the stocks file, and the sID values of the stock entries beneath the deleted entry will be decreased by one, and their lines will be shifted up by one.

7. Update a stock entry: The user will be prompted to enter the sID of the stock entry to be updated, as well as the name of the feature to be updated and the new value for that feature. The sID cannot be updated.

8. Back to main menu.

Part 3. [15 pts] Search Product

This option prompts the user to a menu for performing various product search operations. The menu options are as follows:

1. List all products

2. Filter products by brand, type, price, or a user-defined feature

3. Back to main menu.

If the user selects option 1, the program should list all products stored in the products file.

If the user selects option 2, the program should prompt the user to select the type of filter they want to apply (brand, type, price, or a user-defined feature) and ask for the corresponding filter value/s. The user can use more than one value for filtering, separated by commas. For example, if the user wants to filter by brand and type, they can input "brand,Samsung,type,O".

For brand, type, and user-defined filters, the program should return the products that match the filtering value/s exactly. For the price filter, the program should ask for the minimum and maximum price values and return the products that have a price within the given range.

Please note that, the program must only store the products that remain after filtering in the memory, otherwise you will get 0 for this part.

Part 4. [15 pts] Check Stock Status

This option prompts a menu to the user, enabling them to query the stock of products in various ways. In menu, Option 1 allows the user to query the stock of a given product in a specified branch by using the product ID and branch name. Option 2 allows the user to list the stock of all products in a specified branch. Lastly, option 3 allows the user to list the out-of-stock products in a specified branch. Option 4 returns to the main menu.

Please note that, the program must only store the stock entries that remain after filtering in the memory, otherwise you will get 0 for this part.

Part 5. [15 pts] Brand Stock Control

The company wants to list all products of a given brand with their current stock. In this part, you need to list the products of the given brand by storing their pID, price, and current stock values in a 2D array and print that array.

Part 6. [15 pts] Stock Report

You are asked to export the 2D array in part 5 to a text file as a report in the following format. In this report, there are three comma-separated features in each row: the pID of a product, its minimum stock value, its maximum stock value, and its median number of stock in all the branches.

Notes:

**** You can use static arrays but not global variables.**

****Attach the screenshots of the gameplay.**

****Do not forget to prepare a makefile (-50 points)**

General Rules:

1. We do not give you any function prototypes. We expect that you are experienced enough to understand when to use methods and name them. These will also be graded.
2. Make sure to include appropriate comments and variable names in your code to make it easy to understand.
3. The program must be developed on given version of OS and must be compiled with GCC compiler, any problem which rises due to using another OS or compiler won't be tolerated.

4. Note that if any part of your program is not working as expected, then you can get zero from the related part, even it is working partially.
5. Zip your homework files before uploading them to MS Teams. The zip file must contain the C file with your solution and screenshots of the valid outputs of the program.
6. You can ask any question about the homework by sending an email to b.koca@gtu.edu.tr or by using the homework channel on MS Teams page of the course.