Birla Institute of Technology & Science, Pilani
Second Semester 2021-22 (New Academic Calendar)
CS F111 – Computer Programming
Online Programming Test

SET PINK

==========

24/07/2022 **Max. Marks: 60M** **Duration: 120 mins**

==========

General Instructions

- This paper consists of only one question, which has various subparts whose details are described in the problem statement.
- Read all the instructions and the problem statement very carefully before attempting.
- Carefully follow the submission instructions mentioned at the end before uploading your solution on the Dom Judge portal.
- If you submit multiple submissions, only the latest one will be considered for evaluation. Whatever you submit on the Dom Judge portal will be considered as final. **It is your responsibility to make sure that you are submitting the right file(s)**. **Also, it is your responsibility to make sure that your solution is properly submitted on the Dom Judge portal.** Later on, if some student claims that he/she has submitted the solution and it doesn't appear on the Dom Judge portal, we won't be entertaining any such request.

Instructions to attempt the test

Create a directory with the name **labtest_<yourIDNumber> (Example: labtest_2021A1PS1234P)** in your home directory. The zip file that you have downloaded (**Q1_SET_PINK.zip**) and unzipped contains the question paper (which you are reading currently), and a few C Program files: "**grocerystore.h**", "**grocerystore.c**" and "**main.c**". Copy these three files into the directory which you just created (**labtest_<yourIDNumber>**). These are the three files you will be working upon. You should not make any changes to "**grocerystore.h**" and "**main.c**". You will have to complete the implementation of the functions given in "**grocerystore.c**". This is the only file you should work upon and make changes to. Also, you should not make any changes to the function parameters and the return types for any of the functions in "**grocerystore.c**". If you do any of these don'ts you will definitely incur a penalty.

Carefully observe how each of the functions that you will have to implement, is invoked and used in the **main()** function (main.c), before attempting to write their code. Also, carefully observe the sample execution shown at the end, while implementing each of the functions. Your functions should be implemented in such a way that the final compiled code when executed must give output similar to what is shown in the sample execution.

The Problem Statement

You have to maintain a set of grocery items. Each grocery item has the following attributes: **ID** (Integer), **Name** (Char array), **Price** (float), **Quantity** (Integer). You can follow the structure definition (**struct item**) in **grocerystore.h**. Complete the implementations of the functions in **grocerystore.c** whose details are given below:

(a) Implement the function **readGroceryList()** that receives a **count** of grocery items as a function parameter, and dynamically creates an array of size "**count**" that can store "**count**" number of grocery items. Then, in a loop you will have to read the details (**name**, **price** and **quantity**) of those "**count**" number of grocery items from the user. You should not be reading a value for **ID** of each item from the user. Rather, your program should assign a unique value to the **ID** starting from **1** for the first item, **2** for the second item, and so on. Finally, this function should return the array that you have just created. Assume that the name of every grocery item is only a single word (Eg: "Hamam" and not "Rin Supreme"). Also while giving the input, restrict the price of an item to have upto two decimal places only (Eg: 23.75 is valid price but 52.6987 is not a valid price). **10M**

(b) Implement the function **printGroceryList()** that receives an array of grocery items (**gItems**), and the number of grocery items in that array (**count**), as parameters. It then prints the details of all the grocery items present in that array. Follow the output shown in the sample execution (next page) carefully while implementing this function. **5M**

(c) Implement the function **findItem()** that receives an integer value (**qVal**), an array of grocery items (**gItems**), and the number of grocery items in that array (**count**), as parameters. It then searches for the first grocery item in the above array whose quantity is equal to **qVal** and returns that item. If such an item is not found, it simply returns an empty item with its **ID=-1** and other fields uninitialized. Please follow how the function is called in the main function as well as the sample execution before implementing this. **10M**

(d) Implement the function **findMaxPriceItem()** that receives an array of grocery items (**gItems**), and the number of grocery items in that array (**count**), as parameters. It then searches for the grocery item in the above array that has maximum price and returns that item. If there are multiple items with that maximum price, any one of them can be returned. Please follow how the function is called in the main function as well as the sample execution before implementing this. **10M**

(e) Implement the function **createGLL()** that that receives an array of grocery items (**gItems**), and the number of grocery items in that array (**count**), as parameters. It then dynamically creates a linked list using **stuct gLinkedList** and **struct gNode** for defining/storing a linked list and linked list nodes respectively. Please follow the structure definitions in **grocerystore.h**. The *linked list must store all the items of the **gItems** array in the reverse order*, i.e., the first item of the **gItems** array should become the last item in the linked list, the last item in the **gItems** array should become the first item in the linked list, etc. Basically the order of all the items should be reversed. Then this function should return the linked list just created. **15M**

(f) Implement the function **printGLinkedList()** that receives a linked list (**newList**) and then prints all the items present in it, while traversing the linked list from the first node to the last node. Follow the sample execution carefully before implementing this. **5M**

(g) Create a script file "**myScript.sh**" in the **labtest_<yourIDNumber>** directory. Put all the compilation commands that are required for compilation and linking to generate an executable with the name "**exe**". Make sure that name of the executable is "**exe**" only. **5M**

Sample Execution of the Code

[Note: Things in **bold red ink** are user entered values.**]**

```
jagat@XYZ:~/labTest$ sh myScript.sh
Enter number of unique grocery items in the store:3

Enter details for item 1:
Name:Hamam
Price:20.5
Quantity:15

Enter details for item 2:
Name:Sugar
Price:90.25
Quantity:20

Enter details for item 3:
Name:Milkmaid
Price:210.65
Quantity:6

Item ID: 1, Name: Hamam, Price: 20.500000, Quantity: 15
Item ID: 2, Name: Sugar, Price: 90.250000, Quantity: 20
Item ID: 3, Name: Milkmaid, Price: 210.649994, Quantity: 6

Enter the quantity of the item you wish to find: 20

The first item found with quantity 20 is:
ID: 2, Name: Sugar, Price: 90.250000, Quantity: 20

The item with maximum price is:
ID: 3, Name: Milkmaid, Price: 210.649994, Quantity: 6

Printing the Linked List created:
Item ID: 3, Name: Milkmaid, Price: 210.649994, Quantity: 6
Item ID: 2, Name: Sugar, Price: 90.250000, Quantity: 20
Item ID: 1, Name: Hamam, Price: 20.500000, Quantity: 15
```

Submission Instructions

The directory **labtest_<yourIDNumber>** now contains "**main.c**", "**grocerystore.c**", "**grocerystore.h**", and "**myScript.sh**" files. Create a zip file for this directory. The zip file should have the name **labtest_<yourIDNumber>.zip (Example: labtest_2021A1PS1234P.zip).** Upload this zip file as the solution to the Problem on the Dom Judge Portal.