CSE 1142 - COMPUTER PROGRAMMING II
Programming Assignment # 4

DUE DATE: 26/05/2018 23:59 (No extension)

In this homework, you will simulate a market. The market buys food from a set of firms. It keeps track of firms and bought food in txt files which makes it difficult to store and manage records systematically. Therefore, the owner of the market needs a program which can read txt files and store/manage information as he/she wants.

Your program should read two files: **firms.txt** and **products.txt**. The details of both files are given below.

1. Each firm with a unique id and name is recorded in the **firms.txt** file and format of the file is given below:
   - [id of the firm] [name of the firm]
     - For example;
       - 101 eti
       - 102 ulker etc.

a) You will read *id* and *name* of the firms from the **firms.txt** file and store in a **linked list**. To create the linked list, you will use the **Firm** struct with the following data fields:
   - *firmID* (int) : id of the firm
   - *firmName* (char array of size 100) : name of the firm
   - *struct Firm *nextfirm* : pointer to next firm.
   - Typedef struct Firm to firm.

b) To read the file you will write a function with name **readFirms.**

firm *readFirms(char filename[])
   - *filename[] :* name of the file to be read
   - This function should be called by your main function with the **firms.txt** file.
   - In this function, you will read the file whose name is given as argument *filename.*
   - You will create a **linked list** of the firms by using *firm* struct. Each time you read a firm from the file you will create a firm struct by using **malloc** and store read values in it. Then, you should add it as a new member to the linked list.
   - While adding a new member to the linked list, **id** fields of the firms should be arranged in descending order. For example,
     - Firstly, you read “101 eti” and this firm becomes the first element of the linked list.
     - Secondly, you read “300 ulker”. At this point, *ulker* becomes the first element of the linked list and *eti* becomes the second element of the linked list because of having smaller id than *ulker*.
     - Then, if you read “273 pinar”, *pinar* should be placed between *ulker* and *eti*. Consequently, *ulker* becomes the first, *pinar* becomes the second and *eti* becomes the third element of the linked list.
   - This function returns a pointer that points to the beginning of the linked list.

2. After constructing the linked list of the firms, you will print all firms in the linked list and their all data fields by using **printFirms** function.

void printFirms(firm *firmptr)
   - This function takes a pointer of type firm (*firmptr*) and prints all firms and their all data fields in the linked list pointed by *firmptr.*
   - For example, if the content of the firms.txt file is as given below

```
101 eti
300 superfresh
502 pinar
91 banvit
```

then, the output is

```
502 pinar
300 superfresh
101 eti
91 banvit
```

3. Information about bought food is stored in **products.txt** file and format of the file is given below:
- [name of the firm] [name of the food] [type of the food] [amount of food]
- For example, if the market buys *5* packages of a food of type *biscuit* with name *burcak* from firm *eti*, this purchase is written in the file as
  - eti burcak biscuit 5
- You will read the products from the **products.txt** file and store read values in a **linked list.**

a) To create a linked list, you will use the **FoodStock** struct. struct FoodStock contains another struct inside it which is *struct **Food. Food*** will be used to keep name and expire date of each food since they are kept by the market also. Expire date will be stored in day, month and year format. Create a Food struct with following data fields :
  - *prod_name* (char array of size 400) : name of the food
  - *exp_day* (int) : day of expire date
  - *exp_month* (int) : month of expire date
  - *exp_year* (int) : year of the expire date
  - Typedef struct Food to food.

b) Content of the **FoodStock** struct is given below
  - *struct Food ffood* : struct to store information about a specific food
  - *struct FoodStock *nextfood* : pointer to the next food
  - Typedef struct FoodStock to foodstock.

c) To read the **products.txt** file you will use the **readFoods** function. This function should be called by your main function.

foodstock *readFoods(char filename[])
  - *filename[] :* name of the file to be read
  - In this function you will create a linked list of read products. For example,
    - Firstly, you read "*eti burcak biscuit 5*" from the file. You will create a linked list with *5 foodstock* elements (you will use **malloc** to create structs) and *prod_name* of these elements will be "*eti burcak biscuit*". For expire date information of each *ffood* field of *foodstock* struct, you will assign 2018 or 2019 to *exp_year* randomly. You will assign 1,2,3 or 4 randomly for *exp_month* and a number created randomly between 1 and 14 will be assigned to *exp_day*.
    - Then, if you read "*ulker golf ice_cream 4*", you will add *4 foodstock* elements to the end of the linked list (immediately after the eti burcak biscuits). Each of them will have *prod_name* of "*ulker golf ice_cream*" and you will assign random day, month and year values for the expire date as explained above.
    - After adding above items, you should have a linked list with 9 **foodstock** structs.

    - Each time you read a new food from **products.txt** file you will add them to the end of the linked list.
    - You will return a pointer that points to the beginning of the linked list.

4. To print information about food, you will use the **printFood** function.

   void printFood(foodstock *foodptr, char filename[])

   - *foodptr* : pointer to a linked list
   - *filename[]* : name of the file to which the output will be written.
   - This function reads all elements and their data fields in a linked list pointed by *foodptr* and writes this information to a txt file whose name is taken as argument *filename*.

5. The market periodically controls expired products and expired products should be removed from the list of foods. To remove expired products, you will use the **stockOut** function.

   foodstock *stockOut(foodstock *foodptr, int day, int month, int year)

   - *foodptr* : pointer to linked list of foods
   - *day, month, year* : day, month and year of the expire date according to which the expired foods are detected. These values will be given by the user.
   - For example, if day = 3, month = 1 and year = 2019, the products that have expire date before 3/1/2019 will be deleted from the linked list.
   - Pointer to the new state of the linked list is returned.

6. In the main function of your program, implement the following scenario.
   - Create a ***firms*** pointer of type **firm**.
   - Read the firms by using the **readFirms** function from **firms.txt** file and update the value of ***firms*** pointer as the return value of **readFirms** function.
   - Print firms by using the **printFirms** function.
   - Create a ***stocks*** pointer of type **foodstock**.
   - Read the products from the **products.txt** file by using the **readFoods** function and update the value of ***stocks*** pointer as the return value of **readFoods** function.
   - Print the initial situation of the linked list pointed by ***stocks*** *by* calling ***printFoods*** function and name of the output file will be ***"initial_stock.txt"***
   - Print the following message on console display:
     "Please enter the day, month and year to be checked"
     Then, take the *day, month* and *year* from the user that will be used as the expire date for the **stockOut** function.
   - Call the **stockOut** function with ***stocks*** pointer and the expire date entered by the user. The updated version of the linked list will be pointed by ***stocks*** again.
   - Print the updated version of the linked list pointed by ***stocks*** pointer by using **printFoods** function and the name of the output file will be ***"final_stock.txt"*** for the second time.

7. Sample output:

- Assume that content of the **firms.txt** file is as given below

```
101 eti
300 superfresh
502 pinar
91 banvit
```

- Assume that content of the **products.txt** file is as given below

```
eti burcak biscuit 4
ulker golf ice_cream 2
sutas kaymakli yoghurt 3
```

- Your program should print the firms to the screen and requests an expire date from the user as shown below:

```
Firms :
firm no 502        firm name pinar
firm no 300        firm name superfresh
firm no 101        firm name eti
firm no 91         firm name banvit

Please enter the day, month and year to be checked:
1 4 2019
```

- Content of the **initial_stock.txt** file will become as shown below

```
eti burcak biscuit EXP : 2/3/2019
eti burcak biscuit EXP : 6/2/2019
eti burcak biscuit EXP : 11/1/2019
eti burcak biscuit EXP : 4/3/2019
ulker golf ice_cream EXP : 7/4/2019
ulker golf ice_cream EXP : 1/1/2018
sutas kaymakli yoghurt EXP : 3/1/2019
sutas kaymakli yoghurt EXP : 1/4/2019
sutas kaymakli yoghurt EXP : 3/3/2018
```

- Content of the **final_stock.txt** file will become as shown below

```
ulker golf ice_cream EXP : 7/4/2019
sutas kaymakli yoghurt EXP : 1/4/2019
```

This is a simple scenario to test your implementation. There might be other test cases too. Therefore, please pay attention to use the same function and variable names in your implementations. You can not decrease the number of functions.

Please do not forget that the number of entries in firms.txt and products.txt files can be greater or smaller than this scenario in other test cases.

Submission Instructions

Please zip and submit your files using filename YourNumberHW4.zip (ex: 150713852HW4.zip) to Canvas system (under Assignments tab).
Your zip file should contain the following files:

1. C source file: market_yourStudentNumber.c (ex: market_150713852.c)

Notes:

- Write a comment at the beginning of each program to explain the purpose of the program.
- Write your name and student ID as a comment.
- Include necessary comments to explain your actions.
- Select meaningful names for your variables.
- You are allowed to use the materials that you have learned in lectures & labs.
- Do not use things that you did not learn in the course.
- In case of any form of **copying and cheating** on solutions, all parts will get FF grade. You should submit your own work. In case of any forms of cheating or copying, both giver and receiver are equally culpable and suffer equal penalties.
  All types of plagiarism will result in FF grade from the course.
- No late submission will be accepted.