CIS 241 System-Level Programming and Utilities
Project 2 Grocery Store Management System

Due: Monday, 3/12/2018 by 10:00 pm

Educational Objectives:

This project is designed to help students gain hands-on experience with:

1) Data structure
2) Use of pointer
3) Implementation of linked list
4) Dynamic memory allocation
5) File I/O
6) makefile

Programming Environment

The project should be implemented with C programming language in a Linux environment. If you do not demonstrate your code with your instructor, your code will be tested on the EOS system with gcc compiler. A different C compiler and/or a different system can be used if you will demo your code working with your instructor.

Section 1 Description

In this project, you will write a program to facilitate the management of a grocery store. Your program must meet the requirements specified in the following subsection "Requirements" to receive full credit.

1.1 Requirements:

1) A linked list must be implemented and maintained for the underlying database. Product information is stored in a node on the list.
2) On the list, each node has the following data fields describing a product:
    a) Product name – a string of fewer than 20 characters without white spaces.
    b) Quantity value – a float point variable, representing the numerical amount of a product.
    c) Quantity unit (e.g. pounds, bottles, pieces, etc.) – a string of fewer than 20 characters without white spaces.
    d) Price value – a float point variable, representing the numerical value of the price.
    e) Price unit (e.g. dollars-per-pound, dollars-per-bottle, dollars-each, etc.) – a string of fewer than 20 characters without white spaces.
3) When the program starts, it should try to load available products from a file (say the file **data**) first. However, when the program starts for the first time, the file should be empty. And then provide user a menu of your choice similar to that shown in the screenshot below (You are more than welcome to design a better user interface):

```
Welcome to Xinli Wang grocery store!
Please let me know what you want to do by typing one of the numbers
=================================================================
1: Add product to store                 2: Purchase product from store
3: Check price of a product             4: Show products in store
5: Clean up a product from store        6: Find product
7: Inventory                            8: Done for today
What do you want to do?
```

4) Your program must provide the following functions:
    a) **Add product to store:** add a product to the database (linked list). If the same product (with the same name) is already on the list, update the quantity value accordingly (do not add additional node). If the product is not on the list yet, add a corresponding node (containing the information described in bullet 2)) to the list.
    b) **Purchase product from store:** the store will sell some product with certain amount of quantity. First, provide user with an interface to input the product name and quantity needed. Then, if the store has such product, update the quantity accordingly and record the amount of dollars made from this sell. If the store does not have such product, display a corresponding message.
    c) **Check price of a product:** First, provide user with an interface to input product name. Then, display the product (identified with product name) with its price (including value and unit) if this product is on the list. Otherwise, display a corresponding message.
    d) **Show products in store:** Display products on the list, including all of the information described in bullet 2) for each product.
    e) **Clean up a product from store:** First, provide user with an interface to input product name. Then, remove corresponding node from the list if the product if found. Otherwise, do nothing.
    f) **Find product:** First, provide user with an interface for the user to input product name. Then, display this product with the information describe in bullet 2) if it is found on the list. If not found, display such a message.
    g) **Inventory:** Display the amount of money that has been made for selling so far; and display all of the available products (nodes) with the information described in bullet 2).
    h) **Done for today:** Save the available products on the list with the information described in bullet 2) into a file and exit the program. When the program starts again, the saved data will be loaded.
5) You must implement this project in separate files. Examples will be:
    a) One **main.c** file: contains the main function.
    b) One **operations.h** file: contains all of the function headers.
    c) One **operations.c** file: contains the implementation of the functions.
6) You must write a makefile with the following sections, in addition to sections you think are necessary:
    a) **`all`**: compile the project and generate an executable.
    b) **`run`**: run the program.
    c) **`clean`**: delete all non-necessary files, such as .o files.

Section 2 Design Guidelines

You are encouraged to design your project in a way you think is good. This section provides a general guideline for your project design.

As a general guideline, you should divide your project into a number of different parts. Each part has a manageable and clearly defined task. Then, implement each task as a function.

For example, the following structure and type definition may be useful:

```
#define N 20
struct product
{
     char name[N];
     float quantity_value;
     char quantity_unit[N];
     float price_value;
     char price_unit[N];
     struct product *next;
};

typedef struct product product;
```

The following functions may be useful:

```
// insert a node to the list
int insert(product **l, product node);

// remove a node from list
void rmItem(product *l, product *node);

// show list
void showList(product * l);

// load data from file inf
int loadData(char inf[], product **l);

// save data to file outf
int saveData(char outf[], product *l);

// sell product product of quantity q
float purchase(product *l, char product[], float q);

// check out price of product p from list l
void check_price(product *l, char p[]);

// find a product p from list l
void findItem(product *l, char p[]);
```

```
// the job starts here, start with loading data from
// the file data, and perform the functions by calling
// related functions. Ends at saving data to the file data.
int doIt(char data[]);
```

Section 3 What Do You Submit?

Your project can get graded in two ways:

- First, try to demonstrate your code working with your instructor before the due time. You will receive full credit if your code meets the requirements. In this case, you don't need to turn in your code.
- If you cannot make the demo. Please submit all of your source codes, including your program files and makefile by due time to Blackboard. In this case, your project will be graded with the following grading policy.

Grading Policy

Your instructor will compile your program with gcc and run it on an EOS computer to test the functions specified in the requirements. You will receive full credit if your program meets the requirements. Otherwise, partial credits will be given as follows:

1) Compilation: If your program does not compile, you will receive 0 for this project. If your program compiles, you will receive 10 points.
2) 10 points for each of the functions described in bullet 2) (80 points in total).
3) 10 points for makefile.