Lab 5 CPSC 2310

DUE: Monday, February 27, 2024 midnight

Learning Objectives:

This assignment is designed to provide practice with the following:

- **Linked List**
- **Reading comma delimited files using scanset**
- **Structs**

Overview:

Read the entire document to ensure you are aware of all requirements.

This is an individual lab. You are not allowed to receive help from anyone other than the 2311 lab TA's.

There are many ways to complete this lab, but to force you to use several new functions/concepts in "C", I am going to be very specific on much of the instructions. I strongly suggest you read the entire document to make sure you understand completely what you are required to do. **Points will be deducted for not following directions.**

The program you are going to write will read data from a comma delimited file. Store the data in a linked list and write the formatted data to a file.

Requirements:

Scanset conversion:

Most of you know how to read and write information using scanf/printf or fscanf/fprintf. For this assignment, you are going to read data from a comma delimited file using the concept of scanset conversion. In class, we discussed and reviewed examples of scanset conversionand you are required to use scanset conversion when reading **ALL** of the data. Below are several links that may help you understand scanset:

https://www.geeksforgeeks.org/scansets-in-c/
https://www.knowprogram.com/c-programming/read-and-display-the-string-in-c-programming/
http://www.cplusplus.com/reference/cstdio/fscanf/
https://www.tutorialspoint.com/scansets-in-c

So, what is the data that you will read? In an email, I asked you to complete a google form with questions about yourself. This is the data you will read from a file. I will give you the input file to test your code.

Structs:
Each set of data will be stored in a linked list. You **must** use two structs; one for the birthday and one that will be used as a node for the linked list.

Lab 5 CPSC 2310

DUE: Monday, February 27, 2024 midnight

Linked List:

You should have seen linked lists in previous classes. If you need a refresher, here's a youtube tutorial: https://youtu.be/VOpjAHCee7c

You will define a struct for the linked list that will have the following data members:

- ❖ Character arrays for the first name(size 50), last name(size 50), major (size 10), and class standing(size 15).
- ❖ An instance of the struct that represents the birthday
- ❖ A pointer keeping track of the next node in the list

Files:
You will create functions.h, functions.c, and driver.c files.

driver.c
Main will be included in driver.c. Driver.c should have minimal amount of code in it. Things you may have in the driver:

- ❖ Create the input and output files pointers. These file names will be given on the command line. Use assert to check for the correct number of command line arguments.
- ❖ Open your input and output files and check that opened correctly. (use assert)
- ❖ Call the function createList to create and fill your linked list with input data.
- ❖ Call printList to print out the linked list
- ❖ Free memory from the linked list by calling deleteList.

Your driver should not have more code than necessary. DO NOT include functions.c in your driver.c file. As a rule, you should in include ".c" files. Points will be deducted if you have excessive code in driver.c

functions.h

This file contains the declaration of the structs, all #includes, and the function prototypes. You must also include header guards. Before each function prototype**,** you must have a detailed description of what the overall function does. What the parameters were and what is being returned. Here is an example of an overall function description.

```
/* Parameters: img - image_t pointer array holding the image data for
 *                    each of the input files
 * Return:      output - image_t struct containing output image data
 * This function averages every pixels rbg values from each of the
 * input images and puts those averages into a single output image
 */
```

You are required to have this type of comment block before each function in the **.h** file.

functions.c
You will implement all functions in this file.

Functions:

Below I will provide a short description of each function:

node_t* createList(FILE*, node_t)** – This function is called in main (driver.c) and starts the process of creating the list. The first argument will be a file pointer to your input file and the second will be a double pointer to the head of your list. Use a loop to read from your input file, calling **readNodeInfo** for each node and then calling **add** to add that node to the list. After all of the information from the input file has been added to the list, return a pointer to the list.

void add(node_t node, node_t** head)** – This is the function used to add the node to the linked list. You will take in two parameters, a double pointer to the node you want to add and a double pointer to the head of the list. You should check if the list is empty and add the node to the list. You will have to print out the data in order so add the node to the end of your linked list.

node_t* readNodeInfo(FILE* input) – (called by createList) This function will read the data from the input file, returning a pointer to the populated node. Use malloc to allocate the memory for the node that will eventually be added to the linked list. Using scanset conversion, read the data and store it in the node allocated. (You must use scanset conversion to read **ALL** the data, not just part of the data.)

void printList(FILE*, node_t*) – This function prints, to the output file, the data from the list. If the list is empty you are required to print a message, to **stderr,** indicating the list is empty and exit the program. If the list is not empty you are to print **LIST INFO:** then print the information for each node in the list. See example below for the required format. Described below is a function called printBorder which prints a line of 80 asterisk "*". You will call this function before printing the list and after printing the list.

An example of the print format:
Example:
```
********************************************************************************
LIST INFO:
Name: Jane Doe
Date of Birth:  January 1, 2000
Major:  CIS-BS
Year:  Senior
```

Then the next info

Then the next info
****
AKA:
printBorder()
\n
List Info:\n
Name:\t<firstname> <lastname>\n
Date of Birth:\t<month> <day>, <year>
Major:\t<major>\n
Year:\t<class standing>\n
\n
printBorder()

There is exactly one tab character between each of the labels and the data and a single space in between data such as first and last name.
The data should be printed out in the order it was entered.

void printBorder(FILE*) – This function prints, to the output file, 80 asterisk “*”.

void deleteList(node_t)** – After you are finished with the nodes in the list you need to give the memory back to the system. That is what this function does.

Submission Information:

You should submit the following files to gradescope:

- driver.c
- functions.c
- functions.h
- You should have a header in each of your files that contains the following information. If you neglect to place a header in a file, you will receive a 5-point deduction for each missing header.

```
/*************************/
 *Your name               *
 *CPSC 2310 Fall 23       *
 *UserName:               *
 *Instructor:  Dr. Yvon Feaster  *
/************************/
```

- Your code should be well documented.
- There should be no line of code longer than 80 characters.
- You must use proper and consistent indention.

Lab 5 CPSC 2310

DUE: Monday, February 27, 2024 midnight

- ❖ Your code should not contain “magic numbers”. For Example: for(int I = 0; i<89; i++){} 89 is a magic number; we do not know what it represents. Use a global variable, local variable, or a #define for any magic number in your program.

Failure to do any of the above items may result in a deduction of points for each offense.