**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 conversion. Using scanset conversion when reading **ALL** of the data is a requirement. 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 (F23Data.csv)

**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.

**Linked List:**

As mentioned earlier, you will create a struct for the linked list that will have the following data members:

Character arrays for the first name(size 20), last name(size 20), Degree(size 7), Class Standing (size 12), and favorite type of book(size 30). (These can be individual data types or create another struct to represent a student.)

An instance of the struct that represents the birthday.

A self-referential pointer of the struct you created to represent each 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. Using assert, you are required to check that the appropriate number of command line arguments are passed to the command line. Also, using assert check that the files opened correctly.
* You will call the function createList.
* Call a function to print the data.
* Call a function to delete the list (deleteList)

Each of these functions are described below.

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 and starts the process of creating the list. You will use a loop to read each record in the file then add the record to the list. After all information has been added, this function returns a pointer to the head of the list.

**void add(node\_t\*\* node, node\_t\*\* head)** – This is the function used to add the node to the linked list.

**node\_t\* readNodeInfo(FILE\* input)** – (called by createList) This function will be used to read the data from the input file. Allocate 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

Degree: CISBS

Class Standing: Senior

Preferred Reading: Historical-Fiction

Then the next info

Then the next info

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**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 will submit your files to Handin

You should submit the following files.

* 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.

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