Nassau Community College **Fall 2020**
CSC 217 D1: The C Programming Language **Prof. Christopher R. Merlo**

Project #2: Bookstore Part One
Out: Monday, 9 November 2020
Due: Wednesday, 25 November 2020 11:59:00 pm

Please note that you are meant to work on this assignment yourself.

1 Objectives

- Create a linked data structure in C
- Analyze and process text input
- Process command-line arguments

2 Overview

This project has been designed to give you an initial opportunity to design and implement a linked structure in C. Specifically, you will write a program that reads in formatted input, detects formatting errors, stores the data from well-formed input, and performs some brief analysis on the data.

3 Specifications

You have been hired by a local bookstore to help with the process of tracking inventory. In this project, you will be dealing with books as they arrive at the bookstore; later this semester, we will deal with sales. Your goal in this project is to read formatted input that describes a shipment of books, and store the information in a linked data structure so that it can be searched, sorted, and analyzed.

3.1 The Input

Each time the bookstore receives a shipment of books, the driver will hand the store manager a thumbdrive containing a file that describes the shipment, wherein each line of text represents one of the physical books on the truck – so that if two copies of the same title arrive, there will be two lines in the file that contain the same information. However, as this file is created by humans, there's no guarantee that the data will be in any particular order, or even that it will all be formatted correctly.

Your program must read from standard input a series of lines of data, until there is no more data to process. Each line will be in this format:

Sample Program Input

```
160078688X "Put It In the Book!: A Half-Century of Mets Mania" Rose, Howie
```

Every line of input *should* be in this format: a ten-digit ISBN number, followed by a title surrounded by double quotes, followed by an author in last-comma-first format. Your program ***must*** reject any lines of input that do not conform to this specification. One of the last things your program must do is report the amount of input lines that were accepted, and the amount that were rejected.

Note: We do not know how many books are arriving. Some trucks are big. Plan ahead.

3.1.1 ISBN Numbers

There are two kinds of ISBN[1] numbers. The newer standard is the 13-digit format, but the ten-digit format came first and is easier to analyze, and so every valid line of input to your program will start with a ten-digit ISBN number.

There are details regarding ISBN numbers that are not pertinent to this project. Here's what you need to know to complete this project:

- A valid ten-digit ISBN number may contain spaces and hyphens, which must be ignored. These strings should be considered by your program as representing the same ISBN number:
  - "`160078688X`"
  - "`1-6-00786-8-X`"
  - "`1 6007 8688-X`"
- The last digit of a ten-digit ISBN number is called the *check digit*, and is calculated as follows. Let's call the first digit n_{10}, the second digit n_9, and so on. Calculate a weighted sum s of the first nine digits by multiplying each by its subscript:

  ISBN Check Digit Calculation Algorithm

  $$s = \sum_{i=2}^{10} n_i \cdot i = (n_{10} \cdot 10) + (n_9 \cdot 9) + \cdots + (n_2 \cdot 2)$$

  In the example of Howie Rose's book, this calculation is:

  $$s = (1 \cdot 10) + (6 \cdot 9) + (0 \cdot 8) + (0 \cdot 7) + (7 \cdot 6) + (8 \cdot 5) + (6 \cdot 4) + (8 \cdot 3) + (8 \cdot 2) = 210$$

[1] ISBN stands for "International Standard Book Number." Yes, that makes the phrase "ISBN number" redundant, like "ATM machine," or "LCD display." These are examples of the aptly-named RAS Syndrome (https://en.wikipedia.org/wiki/RAS_syndrome).

The value of the check digit d is equal to $d = 11 - (s \bmod 11)$. For Rose's book, $d = 11 - (210 \bmod 11) = 11 - 1 = 10$. In instances such as this, where d is 10, publishers print the capital letter X as the check digit.

If the calculated check digit is not equal to the given check digit, we know that an error took place somewhere – perhaps a human typist transposed two digits, for instance. In such a case, we consider the ISBN number to be invalid.

3.1.2 Titles and Authors

Titles should always appear within double quotes, immediately after the ISBN number. Any line on which this doesn't happen must be rejected.

Similarly, an author's name must appear last, in the indicated format. Reject any line where this isn't the case.

Note: For this project, we will pretend that every book in the world has precisely one author, and that none of these authors have middle names or even middle initials. In production code, this wouldn't be the case.

3.2 Processing

Your program must dynamically create a linked structure (an unsorted list, or a sorted list, or perhaps a tree of some sort) in which each node represents one book, and stores how many times information about that book has been read in. During the input phase, this structure will start as an empty collection, and during the output phase, its contents will be displayed in some order to be determined by the user.

When a valid line of input is detected, if this ISBN has never been read in before, create a new entry. However, if this ISBN *has* been seen before, determine whether the information on the input matches the information that has been stored already. If it does, update the count. If it doesn't, reject this line of input because it contains contradictory information.

3.3 The Output

Once all the input has been read and processed, your program must report how many lines of input were read in, list all of the books that were received along with a number of copies for each, and then report how many lines of input were accepted and rejected, either for bad formatting or for contradictory information. See the Sample Run in Section 4.

The list of books that your program displays must be sorted by title. In the case that two books by two different authors have the same title, any sub-ordering is fine - but you have to describe in your analysis file the choice you make here.

3.4 Files

You ***must*** design a `struct` to store all the details about one book, including how many copies were received in this shipment. As you've seen in class, the `struct` definition, and any associated macros and function prototypes, ***must*** appear in a header file with a filename that reminds the reader of the struct tag you chose; and the code for these functions ***must*** appear in a similarly-named source code module.

If you write any executable code in a header file, you will earn a grade of zero for this assignment.

You ***must*** also submit one or more other source code modules containing your `main` function and whatever other functions you find it useful to create. How many source code modules you create here is up to you, but you ***will*** be graded in part on how you justify your decision to use more or less modularity. It is also recommended that each source code module have an associated header file containing symbolic constants, function prototypes, and whatever else makes it easier for you to manage these source code modules.

You ***must*** also submit a Makefile that will compile only what needs to be compiled, based upon recent filesystem changes. This Makefile ***must*** create an executable called `project2`. I have uploaded a document I created about Makefiles to Blackboard under the "Files" section, and there's more information on the web.

Please see Section 7.2 on Page 7 for more information on what I will expect your Makefile to accomplish.

3.5 Other People's Code

You *are allowed* to call functions defined in `stdio.h`, `stdlib.h`, `string.h`, and `ctype.h`.

You *might be allowed* to use code that exists in other libraries, but you ***must ask for permission first*** and, in each source code module where such code is allowed, write a comment detailing the function(s), library/ies, date, time, and method (email, Slack, etc.) by which I gave such permission. When such code is used without prior permission or sufficient commenting, *no code in the module* will be considered when grading your project.

You *are allowed* to use any code that you find in the textbook – provided that this code exists in a separate module from any other code, and that it is very clearly commented to indicate whence it came. (*Please note* that this includes the `getline()` function we've been using.)

Similarly, you *are allowed* to use any code that you find on a *college or university website* (something whose domain ends in `.edu` or `.ac.uk` or something similar), with the same provisions as textbook code.

Other than these exceptions, ***all source code you submit must be written by you and by you alone.*** Failure to adhere to this rule will result in a grade of zero on this assignment, and might result in an automatic grade of F for the course.

4 Sample Run

Consider this input:

Sample Input

```
160078688X "Put It In the Book!: A Half-Century of Mets Mania" Rose, Howie
1550225480 "Ghost Rider: Travels on the Healing Road" Peart, Neil
160078688X "Put It In the Book!: A Half-Century of Mets Mania" Rose, Howie
160078688X "Put It In the Book?: A Half-Century of Mets Mania" Rose, Howie

160078688X Rose, Howie "Put It In the Book!: A Half-Century of Mets Mania"
  1-550-22548-0      "Ghost Rider: Travels on the Healing Road"      Peart,          Neil
0131130628 "The C Programming Language: 2nd Edition" Kernighan, Brian
160078688X "If I Did It" Simpson, OJ
1550225480 "Ghost Rider: Travels on the Healing Road" Peart, Neil
```

Lines 1, 2, and 3 are all valid, and lines 1 and 3 describe the same book.

On line 4, the ISBN number is the same as previous lines, but the title is different (someone typed a question mark instead of an exclamation point), and so this line must be rejected.

Line 5 is blank, and so your program should just ignore it and move on.

Line 6 is malformed – the author comes before the title – and so this line must also be rejected.

Line 7 contains the same contents as Line 2, and so that's a second copy of this book. Someone went ham on the space bar, and added dashes to the ISBN, but that's ok – the ISBN, title, last name, and first name are all the same.

Line 8 might look fine to you and me, but someone transposed two digits in the ISBN number, and so the check digit doesn't match the rest of the ISBN. This line must be rejected.

On Line 9, someone re-used Howie Rose's ISBN number on OJ's book, and so this line must be rejected as well.

Finally, Line 10 is the same as Line 2, and so it's accepted.

That's five valid lines (1, 2, 3, 7, 10) and four rejected lines (4, 6, 8, 9). Therefore, if that data exists in a file called `books.txt`, then the command `./project2 < books.txt` should generate output similar to this:

UNIX Session: Output from Sample Input

```
$ ./project2 < books.txt
10 lines of input were processed.

Ghost Rider: Travels on the Healing Road (Peart): 3 copies
Put It In the Book!: A Half-Century of Mets Mania (Rose): 2 copies

5 lines of input were accepted.
4 lines of input were rejected.
```

Please follow this formatting as closely as possible. When I grade your assignment, I'm going to run the same data through everyone's program and compare the output with my own using the `diff` command (see Section 7.2 on page 7), which will detect *any* differences, including in things like letter case and spacing. This clearly won't be the only basis upon which I grade your code, but it will give me somewhere to start.

5 Analysis

In addition to your source code, you must submit a PDF called `analysis.pdf` that contains your answers to the following questions:

- How much time did you spend working on this project?
- Who, if anyone, helped you with this project?
- What was the most difficult part of this project?
- What decision did you make regarding sorting two books with the same title? Why did you make this decision?
- What decision did you make regarding how many source code modules you chose to create? Why did you make this decision?

Your responses to these questions will be graded based upon grammar and spelling as well as on content. Please use complete sentences, proper punctuation, etc.

6 Submitting Your Project

You must submit your project via Blackboard; no other submissions will be compiled, ran, analyzed, or graded. You may re-submit this project as often as you want; I will only ever see the most recent submission. Notice this means that if you submit before the deadline, and then re-submit after the deadline, your project will be late, and you will lose lateness points.

It is difficult for me to predict how many files you will be submitting or what their names will be, but a makefile called `Makefile` and an analysis file called `analysis.pdf` are required. Besides that, be sure to upload all of the `.c` and `.h` files you create that I will need to run your makefile successfully.

Please ***do not*** submit a .zip file, as this will break the script I describe in the next section.

7 Assessment

It is clearly important that your program runs successfully. That is why **60%** of your grade will be based on the **correctness** of your program.

However, it is also important to write code that conforms to the rules we programmers have imposed upon ourselves, to ensure readability. **Twenty percent** of your grade will be based, therefore, on your adherence to the **style guidelines** listed on my web site.

Additionally, **proper testing and proper analysis** are the tools we use to convince non-programmers that our code works. Therefore, **twenty percent** of your grade will be based on your analysis.

7.1 Environment

I will be compiling your code using gcc version 9.3.0 on Ubuntu 20.04, and running it in that environment. You may develop in whichever environment you like, but if your code doesn't compile and run in this gcc/Ubuntu environment, you will lose a substantial amount of points.

In the case where compilation fails in this environment, I will try once more on `stargate.ncc.edu`, which is a CentOS version 7.8.2003 Linux environment running gcc version 4.8.5. If your code fails to compile and run on both environments, you will not earn any correctness points.

A successful compilation and run in either one of these environments will be considered a success for the purposes of grading.

7.2 Compiling and Running Your Program

Please note that when I run your project, all I'm going to do is place all the files you've submitted in a new directory and then, from a directory containing your directory and your classmates' directores, I'm going to run this script:

UNIX Session: Bash Script to Compile and Run Everyone's Submission

```
#! /bin/sh
for i in */; do
    cd $i
    echo $i
    make && ./project2 < ../books.txt > output.txt
    diff ../output.txt ./output.txt > diff.txt
    cd ..
done
```

If this doesn't compile and run your code, I won't have much to grade. I recommend in the strongest possible terms that you perform the same test with the files you plan to submit before submitting.

8 Things You Should Know

- All the advice that you read in the Project 1 specification regarding asking for help, sharing code, and storing your data in multiple places still applies.
- This project is due at 11:59:00 pm on November 25th. Late projects lose 10 points per 24 hour time period or portion thereof, starting at 11:59:01 on November 25th, regardless of weekends, holidays, weather, computer malfunction, etc. Any submissions uploaded after 11:59:00 pm on Monday, November 30th will be ignored, and a grade of 0 will be recorded.
- The second exam has been moved to Monday, November 30th, so that the day this is due can be spent asking questions about this project and/or the exam, or traveling for Thanksgiving if that's what you need to do.

9 Extra Credit

For ten points of extra credit, support the `-r` command-line switch, to display all of the rejected lines of output after the line that says how many there were.

For ten *different* points of extra credit, support the `-a` switch to display books in order by author's last name, then first name, then title.

No extra credit will be awarded in any case unless a clear indication is made in the analysis file that you believe your submission worthy of extra credit. Make it clear which extra credit challenge(s) you completed.

No partial extra credit will be awarded.

Do not expect any discussions about extra credit to take place during class.