CSCI 4963 —- Computer Architecture
Homework 01 —- Due Friday, September 19, 2025

Overview

- This homework is due by 11:59 pm EST on the above date via a Submitty gradeable named *"Homework 01"*.
- This homework is to be completed individually. Do not share your solutions with anyone else.
- Homework assignments are available approximately seven calendar days before they are due.
- Plan to start each homework early. You can ask questions during office hours or in the Submitty forum.
- You **must** use C for this homework assignment, and your code must successfully execute on Submitty to obtain full credit.

Objective

Implement a simple library management system in C to practice data structures, dynamic memory allocation, input output library functions, and modular programming.

Homework Requirements

1. **Features**
   You must implement the following functionalities:
   - Add a new book (with ID, title, author, year) with option number 1 and function `void addBook(char*)`. This function accepts a string that contains the option number (1), book id, book title in double quotations, book author in double quotations and the book's publication year. Then it adds the book to an array of books pointed by the pointer `library`. The array's length is dynamically increased by one when a new book is added to the end of it. You can use the `realloc()` function. Use `man realloc` in Linux terminal to know more about it. Always remember to put the result of `realloc()` function in a temporary pointer as the result may be `NULL` which destroys the current content of your `library` variable if you assign

it directly. Therefore, you need to check for that `NULL` value before using the result of `realloc`.

- To read a book title or author that has more than one word, you need to put them in double quotations (“”) and use a specific set of characters in `sscanf` to read them for `stdin`. In order to read them, you need to have a maximum number of characters to be read for each `title` or `author` in this case 99 characters since the total number of characters for these fields in type `Book` is 100 (100 - `NULL` byte). Instead of the regular character that specifies a string (`%s`) in `sscanf`, you need to use `\"%99[^\"]\"`. The character `\` is an escaping character and is used when we want to use a special character, in this case double quotation. `%99` specifies that it needs to read at most 99 characters. Everything that comes in brackets, shows the scanset of `sscanf`, meaning it should read all of these characters. Character `^` is a negating operator, so `^\”` means scan everything except the double quotation character. The last `\”` means match the ending double quotation but do not record it. For reading both title and author fields you need to use `\"%99[^\"]\"` in `sscanf`. Before `sscanf`, it is needed to use `fgets()` to read one line from the input file and then extract the required data. I have done this step in the code in function `main`. So you do not need to do it.

○ Search for a book by title (partial match allowed) with option 2 and function `void searchBook(char*)`. This function accepts a string that contains the option number (2), and the search phrase. Then, it searches the array that is pointed by the pointer `library` to find the book that its title matches the search phrase or all the books that their titles contain the search phrase. The book title needs to be in double quotations.

- You can use function `char* strstr(const char *haystack, const char *needle);`
- It finds the first occurrence of `needle` in `haystack`. You can Google it or use command `man strstr` in Linux terminal or WSL command prompt.

- Display all books by function `void displayBooks()` with option 3. It goes through the array library and prints the books like the following:

  ```
  1. 103 | The Great Gatsby | F. Scott Fitzgerald | 1925
  2. 2 | Dune | Frank Herbert | 1965
  3. 34 | Neuromancer | William Gibson | 1984
  ```

  - There is a pipe '|' between each field of the book and spaces surrounding the pipe character. You also need to have the same number of spaces and the index of each book exactly like shown. It is important for autograding purposes as an even one character difference means losing the points for that test case.
- Delete a book by ID by function `void deleteBook(char*)` with option 4. It receives a string that contains the option number 4 and an ID number of the book to be deleted. You need to use `sscanf` to ignore the open number and read the second integer which is the ID number. When the book is found, the books on the right side of it (books with higher indices) should be moved one index to the left (towards lower indices). Then, you may `realloc` the variable `library` with one less book.

2. **Data Structure**
   - Define a struct Book with fields:

     ```
     typedef struct {
         int id;
         char title[100];
         char author[100];
         int year;
     } Book;
     ```
   - Store books in a dynamically allocated array.

3. **Input File Structure**
   - The input file is composed of several lines. Each line represents one action in your code. As you have seen, there are four functions in your code and each function is called by a specific option value from the `main()` function. Each line of the input file, invokes one of the functions in your code. Here are the lines in the input file that call each of the four functions in your code:
     - addBook: `1 102 "Dune" "Frank Herbert" 1965`

        - searchBook: `2 "Dune"` (searches for a book titled Dune)
        - displayBooks: `3`
        - deleteBook: `4 102` (deletes the book with id=102)
    - One test input file along with its expected output is provided. The input file format is `.dat`, but you can open it with a text editor. In Linux the file extension does not dictate the application that opens the file unlike Windows. You can create any input file with any extension like `.txt` and feed it to your program.

4. **Submission Instructions**
    - Before you submit your code, be sure that you have added enough and clear comments to your code (this should not be an after-thought). Further, your code should have a clear and logical organization. You need to follow the following coding conventions:
        - **Naming Conventions:**
            1. Use meaningful names for variables, functions, classes, and other identifiers.
            2. Follow a consistent naming style, such as camelCase or snake_case.
        - **Indentation and Formatting:**
            1. Use consistent indentation for code blocks, typically using spaces or tabs.
            2. Format code for readability, including proper spacing around operators and keywords.
        - **Comments:**
            1. Add comments to explain complex sections of code or provide context.
            2. Avoid unnecessary or redundant comments; let the code speak for itself whenever possible.
    - To submit your assignment (and also perform final testing of your code), please use Submitty. Also as a reminder, your code must successfully execute on Submitty to obtain credit for this assignment. In order to comply with Submitty requirements, you are recommended to develop your code

with `gcc` in the Ubuntu environment on your local machine. You can use the following command to compile your code in terminal:

```
gcc -Wall -Werror <sourceName.c> -o <programName>
```

The option `-Werror` is used to turn warnings into errors. Therefore, you should write your code without any warnings. This is the command that is used on Submitty to compile your code for all C homework assignments in this class.

You may use the following command to run your program:

```
./hw01 < test.dat
```

There is no need to check for memory leaks.

- You can use input and output redirection in Linux to get the input for your program from an input file or redirect its output which is a bunch of `printf()`s to another output file if you need.
- The file `frame.c` is provided to you. You can use it to develop your code and use the comments in it to have a consistent output as it is important in autograding on Submitty.