XJCO 1921- Programming Project - Coursework 1

**Deadline:** 12 PM GMT on Friday 1 April 2022

This work is the first coursework for this module. It corresponds to **30%** of the overall assessment for this module. Submissions should be made via Minerva.

# Course Specification

Create a software program in C for the management of a library, according to the specification below. Use a git repository host like GitHub or Gitlab for version control throughout the project.

* 1. **Functionalities [20 marks]:**

Your program should offer the following functionalities:

1. Allow new users to register with the system. To register, the user should supply their name, username and password to login into the system.
2. Allow a registered user to login into the system using their registered credential (username and password)
3. Allow authentic users to search for books by title, author or year of publication
4. Allow users to borrow and return books
5. The system should provide a special user account, the librarian, to add and remove books to the library;
6. Persistence: The state of the library (books, users, and loans) is saved to a text or binary file (no databases is required) and restored at a new execution. Your submission can contain a data file to use as a starting point for the demo. However, your code should not hard-code any paths. All file names should be specified by command-line arguments.
7. Use appropriate dynamic data structures to manage the data (e.g., Link lists). *If static arrays are used, this part will be capped at 10 (out of 20).*

**Notes:**

* Your program must have more than one module (C file). One of the modules must implement the provided interface book\_management.h.
* You can add new data fields to the existing data structures (i.e., Book and BookList) and create new data structures within book\_management.h. However, you may not change the function prototype provided by book\_management.h.
* Your program should make use of the functions defined in book\_management.h, but you are free to create additional head files, data structures and functions to use by your program.
* The program should provide adequate error handling, i.e., display meaningful messages if an error occurs and handle common errors. We will be running our own test cases in order to mark the assignments and doing our best to come up with creative ways to break your code! You should therefore develop your own test cases to beat us to it.
* You also do NOT need to implement a graphical user interface for this task. A neat text-based interface working in the CLI (command-line interface) is sufficient for this task.
* An FAQ for CW1 is also available on Minerva. A demo video for an example implementation is also available on Minerva. This demo video is for illustration only, and your solution does not need to be exactly the same as the demo video (but should follow the coursework spec).

**1.2 Good Programming Practice [8 marks]:**

You should follow good software development practices. For this exercise, you are specifically asked to:

* Follow modular development by making sure the code is modular and well structured; code with proper comments
* Use Makefile for code compilation;
* Use a git repository for version control

**Notes**:

1. **Makefile**: You should also submit a Makefile with at least two targets: *all* and *clean*. “*make all*” compiles your code to generate an executable binary, while “*make clean*” removes all object files (.o), all executables, and any *temporary* files created during your run.
2. **Version control**: We will check the commit logs of your git repository. We expect to see steady progress towards completion, as revealed in the pattern of git commits. One of the implications of this is that we will be penalising any student who develops their code without git then dumps it all into git at the last minute.

**1.3 Reflection [2 marks]:**

During marking, you will be asked to look back and reflect on your project development. Specifically, we will ask you to reflect on:

1. What went well with this project? Include specific areas of the work, programming, design, or testing.
2. What was the most challenging part of this work? Why, and what will you do to address this in the future?

*Please avoid generic statements about time management. Focus on your C coding, design, and testing processes.*

**[Total: 30 marks]**

# Submission

Submit your entire git repository (containing your code and a Makefile), along with a ReadMe file containing the URL of your git repository and a screenshot of your git commit history, **all in a single zip (.zip or .gz)** **file** through Minerva. The submitted zip file should NOT include your build directory or compiled object files, or the executable binary, because these are machine-dependent.

Your git commit and push must be made prior to the deadline.

*You should follow the instructions below on how to prepare your submission. Late submissions are accepted up to 7 days late. Each day, or part of a day, will incur a 5% penalty.*

Important notes on the submission:

* Write the program in standard C. If you write your code in any other language, it will not be assessed, and you will get a zero mark.
* This is an ***individual project***, and you are not supposed to work in groups or pairs with other students.
* Be aware that plagiarism in your code will earn you a zero mark and will have very serious consequences. If two (or more) students have large portions of their files nearly identical, they will be accused of plagiarism or collusion. If found guilty, all parties involved will incur the penalty, regardless of who was the author of the code. For this reason, never show or give access to your code to anyone. Do not help a colleague by sharing your code, or you will both be found guilty of collusion.
* It is your responsibility to make sure that nobody has access to your code. Lock the session if you leave your computer unattended.
* Make sure to download and check your submission. Corrupted files, binary files, wrong versions, copies of your project (over the years, we have seen it all), or anything other than what is requested in this document will be considered an invalid submission.
* We will not accept submissions other than through Minerva.

# Marking and Demonstration

* This exercise will be marked during a lab session after the submission deadline.
* You will be asked to demonstrate your work ***from your Minerva*** ***submission***.
* You need to demonstrate your work to a member of the course team; failing to do so will result in 0 marks.
* You need to come to the marking session with your exercise completed. We will not be able to provide support for this exercise during marking.
* You should be able to explain what you have done clearly, to show that you understand the concepts introduced.
* Checks for plagiarism and collusion will be carried out on all work.