ANGGERIK ARANDA COMMUNITY LIBRARY MANAGEMENT SYSTEM

The Kuala Lumpur City Hall (DBKL) has established a community library Sri Petaling named Anggerik Aranda Community Library (AACL). As a software development company, your company has been rewarded with the project of establishing the library management system (LMS) for AACL. The LMS main functionalities for the LMS is to manage the books and library patrons.

Books in AACL are groups into two categories of Fiction, Non Fiction. Those categories were then subdivided into genres as shown in Table 1. Each book title belong to one category and one genre. There are possibility of each book title to have more than one copies in the collection. For example, there are two copies of books entitled "One Thousands and One Nights". Therefore those two books must be uniquely identified. If one copy being borrowed, the other should be available to be borrowed by other patrons.

Fiction	**Non- Fiction**
Fantasy	Narrative
Science	Biography
Historical	Periodicals
Realistic	Self-help
Fan	Reference

Table 1. List of book genres

Besides book, the LMS should also be able to manage patrons. This include registering patrons, updating their information and most importantly books borrowing and returning functionalities. Every patron can may borrow at most of 3 books at any given times. By default, the patrons need to return the book within 15 days after the borrowing date. This duration however may be extended upon request, before the 15^{th} day.

Implementing appropriate data structure to store and manage the books and patrons information, you are required to develop a C++ application incorporating, at least, the following functionalities:

1. Books
   1.1 Book display – View all book titles available.
   1.2 Add book – Add individual book copy.
   1.3 Search book – Search books based on category, genre, title and availability.
   1.4 Update books' information.

2. Patrons
   2.1. Patron registration – registering a patron into the LMS.
   2.2. Search patron – Search for patron based on name, ID, etc.
   2.3. View all patrons with active book borrowed.
   2.4. Display the last 10 books borrowed by a patron.
   2.5. Update patrons' information.

Assignment Requirements

You are required to submit a **hardcopy** as well as a **softcopy** of assignment report and source code. The report should contain:

- Detailed explanation of the data structures and classes created, with proper justification on your decisions (include source code defining classes, data members, and method headers only).
- Brief explanation about the algorithms used to implement the functionalities stated above (include code snippets of important parts of implementation).
- Source code of the main function, with screenshots showing program's input and output interactions.

You need to submit your **proposal** on the LMS in **week 8**. The proposal should include the **proposed data structure** together with related **algorithms**. The algorithms should be presented in flowchart or pseudocode. This proposal will contribute **20%** for the final assignment marks (under Design criteria).

You have to **present your assignment solution and answers** to the lecturer during a Q&A session that will be conducted after the hand-in date.

If you use some code which has been taken or adapted from another source (book, magazine, internet, forum, etc.) then this must be **cited and referenced** using Harvard Referencing Style within your **source code**, and this must be mentioned explicitly in the **report**. Failure to reference code properly will be treated as plagiarism. **Automated tools for checking code similarities** among submissions will be used, and all detected cases will be treated as cheating. Assessment marks are divided as follows:

	Design	**Implementation**	**Documentation & Presentation**
Marks	20	60	20

What You Need to Hand In?

1. You are required to hand in the individual assignment report on or before the due date mentioned on the cover sheet of the assignment.

2. A softcopy of the report (in PDF format), in addition to the **C++ files** of the programs. The organization of files and folders **must adhere to the following instructions precisely**:
   - The report folder should be named using format <student ID>_<name>_<intake>.pdf. For example "**TP012345_Michael Jordan_ UC2F1910CS.pdf**"

   - All the source codes (.cpp and .h) should be zipped into one file and named following the above format. Make sure to **DELETE** all non-source-code files, including executables (***.exe**).

3. You should **present an executable solution** during Q&A session to demonstrate program execution, the working of the data structure, your understanding of the code, and ability to modify / fix it.

Marking Criteria:

The program submitted will be evaluated according to the following performance criteria:

Distinction (90% and above)

- Program compiles and executes perfectly
- At least 90% of the required functionalities are correctly implemented
- Efficient data structures and\or algorithms are used in the implementation
- Clear coding style and structure, and code is properly commented
- Functionalities are fully tested/validated in program execution

Credit (70% – 89%)

- Program compiles and executes
- Between 70% and 90% of the required functionalities are correctly implemented
- Implementation uses a data structure or algorithm that is not most efficient
- Clear coding style, and code is properly commented
- Functionalities are not fully tested/validated in program execution

Pass (50% - 69%)

- Program compiles perfectly and executes
- Between 50% and 70% of the required functionalities are correctly implemented
- Implementation uses inefficient data structures or algorithms
- Unclear coding style, or code is not properly commented
- Functionalities are not full tested/validated in program execution, or produce errors in some cases

Marginal Fail (30% - 49%)

- Program does not compile or run, but coding logic is almost correct
- Between 30% and 50% of the required functionalities are correctly implemented
- Implementation uses inefficient data structures or algorithms
- Unclear coding style, and no comments provided
- Functionalities are not tested/validated in program execution

Fail (below 30%)

- Program is not given
- Program does not compile or run
- Less than 30% of the required functionalities are implemented
- Implementation uses very inefficient data structures or algorithms
- No proper code structure and no comments provided