**A3 Library Management System**

Note: This material was created with assistance of ChatGPT and Claude

1. **Assignment Overview**

You will design and implement a complete Library Management System in C++ that demonstrates:

* **Inheritance** (abstract base classes and derived classes)
* **Polymorphism** (virtual functions and runtime binding)
* **Operator Overloading** (custom operators for library operations)
* **File I/O** (saving and loading data)
* **Standard Template Library (STL) Containers** (vectors, maps)
* **Exception Handling** (custom exceptions): Optional

1. **Learning Objectives**

By completing this assignment, you will:

1. Master inheritance hierarchies with abstract base classes
2. Implement polymorphic behavior using virtual functions
3. Overload operators for intuitive object manipulation
4. Design a complete object-oriented system
5. Handle data persistence with file operations
6. (Optional) Apply exception handling for robust error management
7. Master separate compilation using the make utility

3. **System Requirements**

**Phase 1: Core Classes (Days 1-2)**

**1.1 LibraryItem Base Class (Abstract)**

Create an abstract base class LibraryItem with:

**Protected Members:**

* string itemID - Unique identifier
* string title - Item title
* string author - Creator/author
* int publicationYear - Year published
* bool isAvailable - Availability status
* string borrowedBy - Member ID who borrowed it
* string dueDate - Return due date

**Public Methods:**

* Constructor with parameters
* Virtual destructor
* Pure virtual void displayInfo() = 0
* Pure virtual double calculateLateFee(int daysLate) = 0
* Pure virtual string getItemType() = 0
* Getters and setters for all members
* void borrowItem(string memberID, string dueDate)
* void returnItem()

**1.2 Derived Classes**

**A. Book Class** (inherits from LibraryItem)

* Additional members: string ISBN, string genre, int pageCount
* Late fee: $0.50 per day
* Override all pure virtual functions
* Implement operator<< for output
* Implement operator== for comparison by ISBN

**B. Magazine Class** (inherits from LibraryItem)

* Additional members: int issueNumber, string month
* Late fee: $0.25 per day
* Override all pure virtual functions
* Implement operator<< for output
* Implement operator== for comparison

**C. DVD Class** (inherits from LibraryItem)

* Additional members: int duration (in minutes), string genre, string director
* Late fee: $1.00 per day
* Override all pure virtual functions
* Implement operator<< for output
* Implement operator== for comparison

**1.3 Member Class**

Create a Member class with:

**Members:**

* string memberID
* string name
* string email
* string phoneNumber
* vector<string> borrowedItems - List of borrowed item IDs
* double outstandingFees
* string membershipType - "Student", "Faculty", "Public"

**Methods:**

* Constructor
* void borrowItem(string itemID)
* void returnItem(string itemID)
* void payFee(double amount)
* void displayMemberInfo()
* Implement operator+= to add fees
* Implement operator-= to subtract paid fees
* Implement operator<< for output

**Phase 2: Library Management (Days 3-4)**

**2.1 Library Class**

Create a Library class that manages all items and members:

**Members:**

* vector<LibraryItem\*> items - Polymorphic collection
* vector<Member\*> members
* string libraryName
* vector<Transaction\*> transactions - Transaction log
* map<string, int> itemCountByType – Statistics tracking

**Methods:**

* Constructor and destructor (handle dynamic memory)
* **Item Management:**
* void addItem(LibraryItem\* item)
* void removeItem( )
* **Member Management:**
* void addMember(Member\* member)
* void removeMember()
* **Search Operations:**
* LibraryItem\* searchByID(string itemID)
* LibraryItem\* searchByTitle(string title) : Option
* vector<LibraryItem\*> searchByAuthor(string author) : Option
* Member\* findMember(string memberID)
* **Borrow and Return**
* void borrowItem(string memberID, string itemID, string dueDate)
* void returnItem(string memberID, string itemID, int daysLate)
* **Display Functions:**
* void displayAllItems(): Option
* void displayAvailableItems()
* void displayBorrowedItems() : Option
* void displayMemberInfo(string memberID)
* **Reporting:**
* void generateReport() - Display statistics
* **Operator Overloading:**
* operator+ to merge two libraries
* operator[] to access item by index
* operator<< for library output

**File I/O: Option**

* saveToFile() - Persist library data
* loadFromFile() - Load saved data
* saveTransactions() - Export transaction log

**2.2 Transaction Class**

Create a Transaction class to log all operations:

**Members:**

* string transactionID
* string memberID
* string itemID
* string transactionType - "BORROW" or "RETURN"
* string date
* double feesPaid

**Methods:**

* Constructor
* void displayTransaction()
* Implement operator<< for output

**2.3 File Operations: Option**

Implement save/load functionality:

* void saveToFile(string filename) - Save all library data
* void loadFromFile(string filename) - Load library data
* void saveTransactions(string filename) - Save transaction log

**2.4 Exception Handling: Option**

Create custom exception classes:

* ItemNotFoundException
* MemberNotFoundException
* ItemNotAvailableException
* InvalidOperationException

Each should inherit from std::exception and provide meaningful messages.

**Phase 4: User Interface & Testing (Days 5)**

**main() with test cases, sample outputs and file outputs are provides.**