Here's your Library Management System code rewritten in a structured format:

System Requirements

- **Book Management**: The system should allow adding, viewing, and managing books in the library.
- **Member Management**: It should manage library members, allowing registration and tracking borrowed books.
- **Borrowing and Returning**: Members should be able to borrow books for a specific duration and return them.
- **Late Fee Calculation**: The system should calculate late fees for overdue books.

Solution Overview

The Library Management System consists of several classes:

1. **Book**: Represents a book with properties such as title, author, ISBN, type, availability, and due date.
2. **Member**: Represents a library member, allowing them to borrow and return books and calculate late fees.
3. **Library**: Manages the overall operations, including adding books, registering members, issuing books, returning them, and displaying available books and member information.

Classes and Functions

1. **Book Class**:
    - **Properties**: `title`, `author`, `isbn`, `type`, `isAvailable`, `dueDate`.
    - **Methods**: Constructor to initialize a book.
2. **Member Class**:
    - **Properties**: `memberId`, `name`, `borrowedBooks`.
    - **Methods**:
        - `borrowBook(Book&, int, int)`: Allows a member to borrow a book.
        - `returnBook(Book&)`: Allows a member to return a borrowed book.
        - `calculateLateFees(int)`: Calculates late fees based on overdue books.
3. **Library Class**:
    - **Properties**: `books`, `members`.
    - **Methods**:
        - `addBook(string, string, string, BookType)`: Adds a new book to the library.
        - `registerMember(int, string)`: Registers a new member.
        - `issueBook(int, string, int, int)`: Issues a book to a member.
        - `returnBook(int, string)`: Processes the return of a book.
        - `showAvailableBooks()`: Displays available books in the library.
        - `showMembers(int)`: Displays registered members and their late fees.

Complete Code

```cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <string>

using namespace std;

// Enum for Book Types
enum BookType { FICTION, NON_FICTION, REFERENCE };

// Book Class to represent a book
class Book {
public:
    string title;
    string author;
    string isbn;
    BookType type;
    bool isAvailable;
    int dueDate;  // Due date as an integer representing days from start

    // Constructor
    Book(string t, string a, string i, BookType bType)
        : title(t), author(a), isbn(i), type(bType), isAvailable(true), dueDate(0)
{}
};

// Member Class to represent a library member
class Member {
public:
    int memberId;
    string name;
    vector<Book*> borrowedBooks; // Pointers to borrowed books

    // Constructor
    Member(int id, string n) : memberId(id), name(n) {}

    bool borrowBook(Book& book, int currentDay, int borrowDuration) {
        if (borrowedBooks.size() < 3 && book.isAvailable) {
            borrowedBooks.push_back(&book);
            book.isAvailable = false;
            book.dueDate = currentDay + borrowDuration;  // Set due date
            return true;
        }
        return false; // Cannot borrow
    }

    void returnBook(Book& book) {
        for (size_t i = 0; i < borrowedBooks.size(); ++i) {
            if (borrowedBooks[i]->isbn == book.isbn) {
                borrowedBooks.erase(borrowedBooks.begin() + i);
                book.isAvailable = true;
                book.dueDate = 0;  // Reset due date
                break;
            }
        }
    }

    float calculateLateFees(int currentDay) {
        float lateFee = 0.0;
        for (auto& book : borrowedBooks) {
            if (book->dueDate < currentDay) {  // Book is overdue
                lateFee += 1.0 * (currentDay - book->dueDate);  // $1 per day
            }
        }
        return lateFee;
    }
};

// Library Class to manage books and members
class Library {
private:
    vector<Book> books;
    unordered_map<int, Member> members; // Maps member IDs to Member objects

public:
    void addBook(string title, string author, string isbn, BookType type) {
        books.emplace_back(title, author, isbn, type);
    }

    void registerMember(int memberId, string name) {
        members[memberId] = Member(memberId, name);
    }

    bool issueBook(int memberId, string isbn, int currentDay, int borrowDuration)
{
        if (members.find(memberId) == members.end()) return false; // Member not
found
        for (auto& book : books) {
            if (book.isbn == isbn && book.isAvailable) {
                return members[memberId].borrowBook(book, currentDay,
borrowDuration); // Try to borrow the book
            }
        }
        return false; // Book not found or not available
    }

    void returnBook(int memberId, string isbn) {
        if (members.find(memberId) != members.end()) {
            for (auto& book : books) {
                if (book.isbn == isbn) {
                    members[memberId].returnBook(book); // Return the book
                    break;
                }
            }
        }
    }
    void showAvailableBooks() {
        cout << "Available Books:\n";
        for (const auto& book : books) {
            if (book.isAvailable) {
                cout << "Title: " << book.title << ", Author: " << book.author <<
endl;
            }
        }
    }

    void showMembers(int currentDay) {
        cout << "Registered Members:\n";
        for (const auto& [id, member] : members) {
            cout << "Member ID: " << id << ", Name: " << member.name
                 << ", Late Fees: $" << member.calculateLateFees(currentDay) <<
endl;
        }
    }
};

// Main function demonstrating the Library Management System
int main() {
    Library library;
    int currentDay = 0; // Current day initialized to 0 for testing

    // Adding books to the library
    library.addBook("1984", "George Orwell", "1234567890", FICTION);
    library.addBook("Sapiens", "Yuval Noah Harari", "0987654321", NON_FICTION);

    // Registering members
    library.registerMember(1, "Alice");
    library.registerMember(2, "Bob");

    // Show available books
    library.showAvailableBooks();

    // Issuing a book to a member (14-day borrow period)
    library.issueBook(1, "1234567890", currentDay, 14); // Alice borrows "1984"

    // Show available books after issuing
    cout << "\nAfter issuing a book:\n";
    library.showAvailableBooks();

    // Simulate time passing (for demonstration purposes)
    currentDay += 15; // Simulating 15 days passing

    // Returning a book
    library.returnBook(1, "1234567890"); // Alice returns "1984"

    // Show available books after returning
    cout << "\nAfter returning a book:\n";
    library.showAvailableBooks();

    // Show members and their late fees
    library.showMembers(currentDay);

    return 0;
}
```

Explanation of the Code

- **Book Class**: Represents a book with attributes such as title, author, ISBN, type, availability, and due date. It includes a constructor to initialize a book.
- **Member Class**: Represents a library member. It manages borrowing and returning books, and it can calculate late fees based on the current day.
- **Library Class**: Central class managing books and members. It allows adding new books, registering members, issuing books, returning them, and displaying available books and member information.

This solution implements a comprehensive Library Management System, providing efficient management of books and members while ensuring smooth borrowing and returning processes.