#include <iostream>

#include <vector>

#include <algorithm>

#include <ctime>

#include <fstream>

#include <limits>

using namespace std;

class Date {

public:

    int day;

    int month;

    int year;

public:

    Date(int d, int m, int y) : day(d), month(m), year(y) {}

    void displayDate() const {

        cout << day << "/" << month << "/" << year;

    }

    bool isPastDate(const Date& other) const {

        if (year > other.year)

            return true;

        else if (year < other.year)

            return false;

        else {

            if (month > other.month)

                return true;

            else if (month < other.month)

                return false;

            else {

                if (day > other.day)

                    return true;

                else

                    return false;

            }

        }

    }

    static Date getCurrentDate() {

        time\_t now = time(0);

        tm\* ltm = localtime(&now);

        return Date(ltm->tm\_mday, ltm->tm\_mon + 1, ltm->tm\_year + 1900);

    }

};

class Student {

private:

    string name;

    int rollNumber;

public:

    Student(string n, int r) : name(n), rollNumber(r) {}

    void displayStudentInfo() const {

        cout << "Name: " << name << ", Roll Number: " << rollNumber << endl;

    }

    string getName() const {

        return name;

    }

    int getRollNumber() const {

        return rollNumber;

    }

};

class Book {

private:

    string title;

    bool available;

    Date returnDate;

    Student\* borrower;

public:

    Book(string t) : title(t), available(true), returnDate(0, 0, 0), borrower(nullptr) {}

    string getTitle() const {

        return title;

    }

    bool isAvailable() const {

        return available;

    }

    void borrowBook(Student\* student, int borrowDuration) {

        available = false;

        borrower = student;

        returnDate = Date::getCurrentDate();

        returnDate = addDays(returnDate, borrowDuration);

    }

    void returnBook() {

        available = true;

        borrower = nullptr;

        returnDate = Date(0, 0, 0);

    }

    bool isOverdue() const {

        Date currentDate = Date::getCurrentDate();

        return returnDate.isPastDate(currentDate);

    }

    void displayReturnDate() const {

        returnDate.displayDate();

    }

    void displayBorrowerInfo() const {

        if (borrower != nullptr) {

            borrower->displayStudentInfo();

        } else {

            cout << "Book not currently borrowed.\n";

        }

    }

private:

    Date addDays(Date date, int days) const {

        int daysInMonth[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

        date.day += days;

        if (date.day > daysInMonth[date.month - 1]) {

            date.day -= daysInMonth[date.month - 1];

            date.month++;

        }

        if (date.month > 12) {

            date.month -= 12;

            date.year++;

        }

        return date;

    }

};

class Library {

public:

    vector<Book> books;

    vector<Student> students;

public:

    void addBook(const Book& book) {

        books.push\_back(book);

        cout << "Book added successfully.\n";

    }

    void addStudent(const Student& student) {

        students.push\_back(student);

        cout << "Student added successfully.\n";

    }

    void displayAvailableBooks() const {

        if (books.empty()) {

            cout << "No books in the library.\n";

        } else {

            cout << "Books available in the library:\n";

            for (const auto& book : books) {

                if (book.isAvailable()) {

                    cout << "Title: " << book.getTitle() << endl;

                }

            }

        }

    }

    void displayStudents() const {

        if (students.empty()) {

            cout << "No students in the library.\n";

        } else {

            cout << "Students in the library:\n";

            for (const auto& student : students) {

                student.displayStudentInfo();

            }

        }

    }

    Student\* findStudent(int rollNumber) {

        auto it = find\_if(students.begin(), students.end(), [&rollNumber](const Student& s) {

            return s.getRollNumber() == rollNumber;

        });

        if (it != students.end()) {

            return &(\*it);

        } else {

            return nullptr;

        }

    }

    Book\* findBook(const string& title) {

        auto it = find\_if(books.begin(), books.end(), [&title](const Book& b) {

            return b.getTitle() == title;

        });

        if (it != books.end()) {

            return &(\*it);

        } else {

            return nullptr;

        }

    }

    void issueBook(const string& title, int rollNumber, int borrowDuration) {

        Student\* student = findStudent(rollNumber);

        Book\* book = findBook(title);

        if (student != nullptr && book != nullptr && book->isAvailable()) {

            book->borrowBook(student, borrowDuration);

            cout << "You have issued the book: " << title << " to student: " << student->getName() << ". Please return by ";

            book->displayReturnDate();

            cout << endl;

        } else {

            if (student == nullptr) {

                cout << "Student with roll number " << rollNumber << " not found.\n";

            } else if (book == nullptr) {

                cout << "Book with title " << title << " not found or not available.\n";

            } else {

                cout << "Book with title " << title << " is not available for borrowing.\n";

            }

        }

    }

    void returnBook(const string& title) {

        Book\* book = findBook(title);

        if (book != nullptr && !book->isAvailable()) {

            book->returnBook();

            if (book->isOverdue()) {

                cout << "You have returned the book: " << title << ".\n";

                cout << "Book is overdue!\n";

            } else {

                cout << "You have returned the book: " << title << ".\n";

            }

        } else {

            cout << "Book not found or already available.\n";

        }

    }

};

template<typename T>

T getInput(const string& prompt) {

    T value;

    while (true) {

        cout << prompt;

        if (cin >> value) {

            break;

        } else {

            cout << "Invalid input. Please try again.\n";

            cin.clear();

            cin.ignore(numeric\_limits<streamsize>::max(), '\n');

        }

    }

    cin.ignore();

    return value;

}

void saveLibraryData(const Library& library) {

    ofstream bookFile("books.txt");

    ofstream studentFile("students.txt");

    for (const auto& book : library.books) {

        bookFile << book.getTitle() << endl;

    }

    bookFile.close();

    for (const auto& student : library.students) {

        studentFile << student.getName() << " " << student.getRollNumber() << endl;

    }

    studentFile.close();

}

void loadLibraryData(Library& library) {

    ifstream bookFile("books.txt");

    ifstream studentFile("students.txt");

    string title;

    while (getline(bookFile, title)) {

        library.addBook(Book(title));

    }

    bookFile.close();

    string name;

    int rollNumber;

    while (studentFile >> name >> rollNumber) {

        library.addStudent(Student(name, rollNumber));

    }

    studentFile.close();

}

int main() {

    Library library;

    int choice;

    string title, studentName;

    int rollNumber;

    const int BORROW\_DURATION = 15;

    loadLibraryData(library);

    do {

        cout << "\nLibrary Management System Menu:\n";

        cout << "1. Display available books\n";

        cout << "2. Display students\n";

        cout << "3. Add a book\n";

        cout << "4. Add a student\n";

        cout << "5. Issue a book\n";

        cout << "6. Return a book\n";

        cout << "7. Exit\n";

        choice = getInput<int>("Enter your choice: ");

        switch (choice) {

            case 1:

                library.displayAvailableBooks();

                break;

            case 2:

                library.displayStudents();

                break;

            case 3:

                cout << "Enter the title of the book: ";

                getline(cin, title);

                library.addBook(Book(title));

                break;

            case 4:

                cout << "Enter the name of the student: ";

                getline(cin, studentName);

                rollNumber = getInput<int>("Enter the roll number of the student: ");

                library.addStudent(Student(studentName, rollNumber));

                break;

            case 5:

                cout << "Enter the title of the book you want to issue: ";

                getline(cin, title);

                rollNumber = getInput<int>("Enter the roll number of the student: ");

                library.issueBook(title, rollNumber, BORROW\_DURATION);

                break;

            case 6:

                cout << "Enter the title of the book you want to return: ";

                getline(cin, title);

                library.returnBook(title);

                break;

            case 7:

                saveLibraryData(library);

                cout << "Exiting program. Goodbye!\n";

                break;

            default:

                cout << "Invalid choice. Please enter a number from 1 to 7.\n";

                break;

        }

    } while (choice != 7);

    return 0;

}