```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>

#define MAX_BOOKS 200
#define MAX_BORROWERS 100
#define FILENAME "library_data.txt"
#define MAX_LENGTH 50
#define MAX_BORROWED 5

// Structure to store book information
typedef struct {
    char bookID[MAX_LENGTH];
    char title[MAX_LENGTH];
    char author[MAX_LENGTH];
    char publisher[MAX_LENGTH];
    int publicationYear;
    char genre[MAX_LENGTH];
    int availability;    // 1 = available, 0 = borrowed
    char borrowerID[MAX_LENGTH];  // ID of person who borrowed the book
    char dueDate[MAX_LENGTH];     // Due date for return
} Book;

// Structure to store borrower information
typedef struct {
    char borrowerID[MAX_LENGTH];
    char name[MAX_LENGTH];
    char contactInfo[MAX_LENGTH];
    int booksBorrowed;
} Borrower;

// Global variables
Book books[MAX_BOOKS];
Borrower borrowers[MAX_BORROWERS];
int numBooks = 0;
int numBorrowers = 0;

// Function prototypes
```

```
void displayMenu();
int getUserChoice();
void loadData();
void saveData();
void addBook();
void addBorrower();
void searchBooks();
void issueBook();
void returnBook();
void displayAllBooks();
void displayBorrowers();
void displayBorrowedBooks();
void generateReport();
void drawGraph();
char* getCurrentDate();
char* calculateDueDate();

int main() {
    int choice;

    do {
        displayMenu();
        choice = getUserChoice();

        switch (choice) {
            case 1:
                loadData();
                break;
            case 2:
                saveData();
                break;
            case 3:
                addBook();
                break;
            case 4:
                addBorrower();
                break;
            case 5:
                searchBooks();
                break;
```

```
            case 6:
                issueBook();
                break;
            case 7:
                returnBook();
                break;
            case 8:
                displayAllBooks();
                break;
            case 9:
                displayBorrowers();
                break;
            case 10:
                displayBorrowedBooks();
                break;
            case 11:
                generateReport();
                break;
            case 12:
                printf("Exiting program. Goodbye!\n");
                break;
            default:
                printf("Invalid menu choice!\n");
                break;
        }
    } while (choice != 12);

    return 0;
}

// Display the main menu
void displayMenu() {
    printf("\n");
    printf("XYZ LIBRARY - LIBRARY MANAGEMENT SYSTEM\n");
    printf("---------------------------------------\n");
    printf("1. Load Data\n");
    printf("2. Save Data\n");
    printf("3. Add Book\n");
    printf("4. Add Borrower\n");
    printf("5. Search Books\n");
```

```
    printf("6. Issue Book\n");
    printf("7. Return Book\n");
    printf("8. Display All Books\n");
    printf("9. Display All Borrowers\n");
    printf("10. Display Borrowed Books\n");
    printf("11. Generate Library Report\n");
    printf("12. Exit\n");
    printf("----------------------------------------\n");
    printf("Enter your choice: ");
}

// Get user choice from menu
int getUserChoice() {
    int choice;
    scanf("%d", &choice);
    getchar(); // Clear input buffer
    return choice;
}

// Load data from file
void loadData() {
    FILE *file = fopen(FILENAME, "r");

    if (file == NULL) {
        printf("Error opening file or file doesn't exist!\n");
        return;
    }

    // Reset the current data
    numBooks = 0;
    numBorrowers = 0;

    // Read the file line by line for books
    char line[256];
    int section = 0; // 0 for books, 1 for borrowers

    while (fgets(line, sizeof(line), file)) {
        // Check if we've reached the borrowers section
        if (strncmp(line, "BORROWERS", 9) == 0) {
            section = 1;
```

```
        continue;
    }

    // Remove newline character
    line[strcspn(line, "\n")] = 0;

    if (section == 0 && numBooks < MAX_BOOKS) {
        // Process book data
        char *token;

        // Extract book ID
        token = strtok(line, ",");
        if (token != NULL) {
            strcpy(books[numBooks].bookID, token);
        }

        // Extract title
        token = strtok(NULL, ",");
        if (token != NULL) {
            strcpy(books[numBooks].title, token);
        }

        // Extract author
        token = strtok(NULL, ",");
        if (token != NULL) {
            strcpy(books[numBooks].author, token);
        }

        // Extract publisher
        token = strtok(NULL, ",");
        if (token != NULL) {
            strcpy(books[numBooks].publisher, token);
        }

        // Extract publication year
        token = strtok(NULL, ",");
        if (token != NULL) {
            books[numBooks].publicationYear = atoi(token);
        }
```

```
        // Extract genre
        token = strtok(NULL, ",");
        if (token != NULL) {
            strcpy(books[numBooks].genre, token);
        }

        // Extract availability
        token = strtok(NULL, ",");
        if (token != NULL) {
            books[numBooks].availability = atoi(token);
        }

        // Extract borrower ID
        token = strtok(NULL, ",");
        if (token != NULL) {
            strcpy(books[numBooks].borrowerID, token);
        }

        // Extract due date
        token = strtok(NULL, ",");
        if (token != NULL) {
            strcpy(books[numBooks].dueDate, token);
        }

        numBooks++;
    }
    else if (section == 1 && numBorrowers < MAX_BORROWERS) {
        // Process borrower data
        char *token;

        // Extract borrower ID
        token = strtok(line, ",");
        if (token != NULL) {
            strcpy(borrowers[numBorrowers].borrowerID, token);
        }

        // Extract name
        token = strtok(NULL, ",");
        if (token != NULL) {
            strcpy(borrowers[numBorrowers].name, token);
```

```
            }

            // Extract contact info
            token = strtok(NULL, ",");
            if (token != NULL) {
                strcpy(borrowers[numBorrowers].contactInfo, token);
            }

            // Extract books borrowed
            token = strtok(NULL, ",");
            if (token != NULL) {
                borrowers[numBorrowers].booksBorrowed = atoi(token);
            }

            numBorrowers++;
        }
    }

    fclose(file);
    printf("Data loaded successfully! %d books and %d borrowers
loaded.\n", numBooks, numBorrowers);
}

// Save data to file
void saveData() {
    FILE *file = fopen(FILENAME, "w");

    if (file == NULL) {
        printf("Error opening file for writing!\n");
        return;
    }

    // Write book data
    for (int i = 0; i < numBooks; i++) {
        fprintf(file, "%s,%s,%s,%s,%d,%s,%d,%s,%s\n",
                books[i].bookID,
                books[i].title,
                books[i].author,
                books[i].publisher,
                books[i].publicationYear,
```

```
                books[i].genre,
                books[i].availability,
                books[i].borrowerID,
                books[i].dueDate);
    }

    // Write separator for borrowers section
    fprintf(file, "BORROWERS\n");

    // Write borrower data
    for (int i = 0; i < numBorrowers; i++) {
        fprintf(file, "%s,%s,%s,%d\n",
                borrowers[i].borrowerID,
                borrowers[i].name,
                borrowers[i].contactInfo,
                borrowers[i].booksBorrowed);
    }

    fclose(file);
    printf("Data saved successfully!\n");
}

// Add a new book record
void addBook() {
    char addAnother;

    do {
        if (numBooks >= MAX_BOOKS) {
            printf("Error: Maximum number of books reached!\n");
            return;
        }

        printf("\nEnter details for new book:\n");

        printf("Book ID: ");
        scanf("%s", books[numBooks].bookID);
        getchar(); // Clear input buffer

        printf("Title: ");
        fgets(books[numBooks].title, MAX_LENGTH, stdin);
```

```
        books[numBooks].title[strcspn(books[numBooks].title, "\n")] = 0;
// Remove newline

        printf("Author: ");
        fgets(books[numBooks].author, MAX_LENGTH, stdin);
        books[numBooks].author[strcspn(books[numBooks].author, "\n")] = 0;
// Remove newline

        printf("Publisher: ");
        fgets(books[numBooks].publisher, MAX_LENGTH, stdin);
        books[numBooks].publisher[strcspn(books[numBooks].publisher,
"\n")] = 0; // Remove newline

        printf("Publication Year: ");
        scanf("%d", &books[numBooks].publicationYear);
        getchar(); // Clear input buffer

        printf("Genre: ");
        fgets(books[numBooks].genre, MAX_LENGTH, stdin);
        books[numBooks].genre[strcspn(books[numBooks].genre, "\n")] = 0;
// Remove newline

        // Set default values
        books[numBooks].availability = 1; // Available
        strcpy(books[numBooks].borrowerID, "None");
        strcpy(books[numBooks].dueDate, "None");

        numBooks++;

        printf("\nBook added successfully!\n");
        printf("Do you want to add another book? (Y/N): ");
        scanf(" %c", &addAnother);
        getchar(); // Clear input buffer

    } while (toupper(addAnother) == 'Y');
}

// Add a new borrower record
void addBorrower() {
    char addAnother;
```

```
    do {
        if (numBorrowers >= MAX_BORROWERS) {
            printf("Error: Maximum number of borrowers reached!\n");
            return;
        }

        printf("\nEnter details for new borrower:\n");

        printf("Borrower ID: ");
        scanf("%s", borrowers[numBorrowers].borrowerID);
        getchar(); // Clear input buffer

        printf("Name: ");
        fgets(borrowers[numBorrowers].name, MAX_LENGTH, stdin);
        borrowers[numBorrowers].name[strcspn(borrowers[numBorrowers].name,
"\n")] = 0; // Remove newline

        printf("Contact Info: ");
        fgets(borrowers[numBorrowers].contactInfo, MAX_LENGTH, stdin);

borrowers[numBorrowers].contactInfo[strcspn(borrowers[numBorrowers].contac
tInfo, "\n")] = 0; // Remove newline

        // Set default value
        borrowers[numBorrowers].booksBorrowed = 0;

        numBorrowers++;

        printf("\nBorrower added successfully!\n");
        printf("Do you want to add another borrower? (Y/N): ");
        scanf(" %c", &addAnother);
        getchar(); // Clear input buffer

    } while (toupper(addAnother) == 'Y');
}

// Search for books
void searchBooks() {
    int choice;
```

```
    char searchTerm[MAX_LENGTH];
    int found = 0;

    printf("\nSearch by:\n");
    printf("1. Book ID\n");
    printf("2. Title\n");
    printf("3. Author\n");
    printf("4. Genre\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    getchar(); // Clear input buffer

    printf("Enter search term: ");
    fgets(searchTerm, MAX_LENGTH, stdin);
    searchTerm[strcspn(searchTerm, "\n")] = 0; // Remove newline

    printf("\nSearch Results:\n");

printf("------------------------------------------------------------------------
-------------------------\n");
    printf("%-10s %-25s %-20s %-15s %-5s %-10s %-10s\n",
           "Book ID", "Title", "Author", "Genre", "Year", "Status", "Due
Date");

printf("------------------------------------------------------------------------
-------------------------\n");

    for (int i = 0; i < numBooks; i++) {
        int match = 0;

        switch (choice) {
            case 1: // Book ID
                if (strstr(books[i].bookID, searchTerm) != NULL) {
                    match = 1;
                }
                break;
            case 2: // Title
                if (strstr(books[i].title, searchTerm) != NULL) {
                    match = 1;
                }
```

```
                break;
            case 3: // Author
                if (strstr(books[i].author, searchTerm) != NULL) {
                    match = 1;
                }
                break;
            case 4: // Genre
                if (strstr(books[i].genre, searchTerm) != NULL) {
                    match = 1;
                }
                break;
        }

        if (match) {
            printf("%-10s %-25s %-20s %-15s %-5d %-10s %-10s\n",
                   books[i].bookID,
                   books[i].title,
                   books[i].author,
                   books[i].genre,
                   books[i].publicationYear,
                   books[i].availability ? "Available" : "Borrowed",
                   books[i].availability ? "N/A" : books[i].dueDate);
            found++;
        }
    }


printf("------------------------------------------------------------------
-------------------------\n");
    printf("Found %d books matching your search.\n", found);
}

// Issue a book to a borrower
void issueBook() {
    char bookID[MAX_LENGTH];
    char borrowerID[MAX_LENGTH];
    int bookIndex = -1;
    int borrowerIndex = -1;

    printf("\nEnter Book ID: ");
```

```c
scanf("%s", bookID);
getchar(); // Clear input buffer

printf("Enter Borrower ID: ");
scanf("%s", borrowerID);
getchar(); // Clear input buffer

// Find the book with the given ID
for (int i = 0; i < numBooks; i++) {
    if (strcmp(books[i].bookID, bookID) == 0) {
        bookIndex = i;
        break;
    }
}

// Find the borrower with the given ID
for (int i = 0; i < numBorrowers; i++) {
    if (strcmp(borrowers[i].borrowerID, borrowerID) == 0) {
        borrowerIndex = i;
        break;
    }
}

// Check if book and borrower exist
if (bookIndex == -1) {
    printf("Error: Book with ID %s not found!\n", bookID);
    return;
}

if (borrowerIndex == -1) {
    printf("Error: Borrower with ID %s not found!\n", borrowerID);
    return;
}

// Check if book is available
if (books[bookIndex].availability == 0) {
    printf("Error: Book is already borrowed!\n");
    return;
}
```

```
    // Check if borrower has reached the maximum number of books
    if (borrowers[borrowerIndex].booksBorrowed >= MAX_BORROWED) {
        printf("Error: Borrower has already borrowed the maximum number of
books allowed!\n");
        return;
    }

    // Update book information
    books[bookIndex].availability = 0;
    strcpy(books[bookIndex].borrowerID, borrowerID);
    strcpy(books[bookIndex].dueDate, calculateDueDate());

    // Update borrower information
    borrowers[borrowerIndex].booksBorrowed++;

    printf("\nBook issued successfully!\n");
    printf("Due date: %s\n", books[bookIndex].dueDate);
}

// Return a borrowed book
void returnBook() {
    char bookID[MAX_LENGTH];
    int bookIndex = -1;
    int borrowerIndex = -1;

    printf("\nEnter Book ID: ");
    scanf("%s", bookID);
    getchar(); // Clear input buffer

    // Find the book with the given ID
    for (int i = 0; i < numBooks; i++) {
        if (strcmp(books[i].bookID, bookID) == 0) {
            bookIndex = i;
            break;
        }
    }

    // Check if book exists
    if (bookIndex == -1) {
        printf("Error: Book with ID %s not found!\n", bookID);
```

```
        return;
    }

    // Check if book is borrowed
    if (books[bookIndex].availability == 1) {
        printf("Error: Book is not currently borrowed!\n");
        return;
    }

    // Find the borrower
    for (int i = 0; i < numBorrowers; i++) {
        if (strcmp(borrowers[i].borrowerID, books[bookIndex].borrowerID)
== 0) {
            borrowerIndex = i;
            break;
        }
    }

    // Update book information
    books[bookIndex].availability = 1;
    strcpy(books[bookIndex].borrowerID, "None");
    strcpy(books[bookIndex].dueDate, "None");

    // Update borrower information if found
    if (borrowerIndex != -1) {
        borrowers[borrowerIndex].booksBorrowed--;
    }

    printf("\nBook returned successfully!\n");
}

// Display all books
void displayAllBooks() {
    if (numBooks == 0) {
        printf("No books available!\n");
        return;
    }

    printf("\nALL BOOKS\n");
```

```
printf("-------------------------------------------------------------------
-------------------------\n");
    printf("%-10s %-25s %-20s %-15s %-5s %-10s\n",
           "Book ID", "Title", "Author", "Genre", "Year", "Status");

printf("-------------------------------------------------------------------
-------------------------\n");

    for (int i = 0; i < numBooks; i++) {
        printf("%-10s %-25s %-20s %-15s %-5d %-10s\n",
               books[i].bookID,
               books[i].title,
               books[i].author,
               books[i].genre,
               books[i].publicationYear,
               books[i].availability ? "Available" : "Borrowed");
    }


printf("-------------------------------------------------------------------
-------------------------\n");
    printf("Total Books: %d\n", numBooks);
}

// Display all borrowers
void displayBorrowers() {
    if (numBorrowers == 0) {
        printf("No borrowers available!\n");
        return;
    }

    printf("\nALL BORROWERS\n");
    printf("-----------------------------------------------------\n");
    printf("%-10s %-25s %-20s %-5s\n",
           "ID", "Name", "Contact Info", "Books");
    printf("-----------------------------------------------------\n");

    for (int i = 0; i < numBorrowers; i++) {
        printf("%-10s %-25s %-20s %-5d\n",
```

```
                borrowers[i].borrowerID,
                borrowers[i].name,
                borrowers[i].contactInfo,
                borrowers[i].booksBorrowed);
    }

    printf("--------------------------------------------------\n");
    printf("Total Borrowers: %d\n", numBorrowers);
}

// Display borrowed books
void displayBorrowedBooks() {
    int count = 0;

    printf("\nBORROWED BOOKS\n");

printf("-------------------------------------------------------------------
-------------------------\n");
    printf("%-10s %-25s %-20s %-15s %-15s %-10s\n",
           "Book ID", "Title", "Borrower ID", "Borrower Name", "Issue
Date", "Due Date");

printf("-------------------------------------------------------------------
-------------------------\n");

    for (int i = 0; i < numBooks; i++) {
        if (books[i].availability == 0) {
            // Find borrower name
            char borrowerName[MAX_LENGTH] = "Unknown";
            for (int j = 0; j < numBorrowers; j++) {
                if (strcmp(borrowers[j].borrowerID, books[i].borrowerID)
== 0) {
                    strcpy(borrowerName, borrowers[j].name);
                    break;
                }
            }

            printf("%-10s %-25s %-20s %-15s %-15s %-10s\n",
                   books[i].bookID,
                   books[i].title,
```

```
                    books[i].borrowerID,
                    borrowerName,
                    "N/A", // Issue date not tracked
                    books[i].dueDate);
            count++;
        }
    }

printf("------------------------------------------------------------------
-------------------------\n");
    printf("Total Borrowed Books: %d\n", count);
}

// Generate library report
void generateReport() {
    int availableBooks = 0;
    int borrowedBooks = 0;

    // Count available and borrowed books
    for (int i = 0; i < numBooks; i++) {
        if (books[i].availability == 1) {
            availableBooks++;
        } else {
            borrowedBooks++;
        }
    }

    printf("\nLIBRARY REPORT\n");

printf("------------------------------------------------------------------
-------------------------\n");
    printf("Total Books: %d\n", numBooks);
    printf("Available Books: %d\n", availableBooks);
    printf("Borrowed Books: %d\n", borrowedBooks);
    printf("Total Borrowers: %d\n", numBorrowers);

printf("------------------------------------------------------------------
-------------------------\n");
```

```
    // Count books by genre
    printf("\nBooks by Genre:\n");

printf("------------------------------------------------------------------
-------------------------\n");

    // Create a temporary array to store genre counts
    struct {
        char name[MAX_LENGTH];
        int count;
    } genres[MAX_BOOKS];
    int numGenres = 0;

    // Count books by genre
    for (int i = 0; i < numBooks; i++) {
        // Check if genre already exists in the array
        int genreIndex = -1;
        for (int j = 0; j < numGenres; j++) {
            if (strcmp(genres[j].name, books[i].genre) == 0) {
                genreIndex = j;
                break;
            }
        }

        if (genreIndex == -1) {
            // Add new genre
            strcpy(genres[numGenres].name, books[i].genre);
            genres[numGenres].count = 1;
            numGenres++;
        } else {
            // Increment count for existing genre
            genres[genreIndex].count++;
        }
    }

    // Print genre counts
    for (int i = 0; i < numGenres; i++) {
        printf("%-20s: %d\n", genres[i].name, genres[i].count);
    }
```

```
printf("------------------------------------------------------------------
-------------------------\n");

    // Draw a graph showing books by genre
    drawGraph(genres, numGenres);
}

// Draw a bar graph showing books by genre
void drawGraph(struct { char name[MAX_LENGTH]; int count; } genres[], int
numGenres) {
    printf("\nBOOKS BY GENRE\n");

printf("------------------------------------------------------------\n");

    for (int i = 0; i < numGenres; i++) {
        printf("%-12s |", genres[i].name);

        for (int j = 0; j < genres[i].count; j++) {
            printf("*");
        }

        printf(" %d\n", genres[i].count);
    }


printf("------------------------------------------------------------\n");
    printf("             0    5    10   15   20   25   30   35   40   45
50\n");

printf("------------------------------------------------------------\n");
}

// Get current date as a string
char* getCurrentDate() {
    static char date[11]; // Format: YYYY-MM-DD
    time_t now = time(NULL);
    struct tm *t = localtime(&now);

    sprintf(date, "%04d-%02d-%02d",
```

```
            t->tm_year + 1900, t->tm_mon + 1, t->tm_mday);

    return date;
}

// Calculate due date (14 days from current date)
char* calculateDueDate() {
    static char date[11]; // Format: YYYY-MM-DD
    time_t now = time(NULL);
    time_t due = now + (14 * 24 * 60 * 60); // 14 days in seconds
    struct tm *t = localtime(&due);

    sprintf(date, "%04d-%02d-%02d",
            t->tm_year + 1900, t->tm_mon + 1, t->tm_mday);

    return date;
}
```