



جامعة مصر للمعلوماتية  
EGYPT UNIVERSITY  
OF INFORMATICS

## Library Management System

Code Documentation

Submitted by:

- Sameer Ahmed 24-101318
- Omar Yasser 24-101451
- Kareem Ahmed 24-101379
- Mohamed Ayman 24-101127
- Kirollos Ihab 24-101383
- Youssef Khaled 24-101359
- Abdullah Abdulrhman 24-101329

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Code Overview</b>	<b>2</b>
2.1	Global Variables . . . . .	2
2.2	Input Procedure . . . . .	2
<b>3</b>	<b>Query Commands</b>	<b>3</b>
3.1	Number_Books . . . . .	3
3.2	Number_Members . . . . .	3
3.3	Book_ID_Min . . . . .	3
3.4	Books_Available . . . . .	3
3.5	Most_Borrowed . . . . .	4
3.6	List_Book_Borrowers . . . . .	4
3.7	List_Member_Books . . . . .	5
3.8	Members_Less . . . . .	5
3.9	Books_Unborrowed . . . . .	5
3.10	Books_Borrowed_Days . . . . .	6
3.11	Books_Per_Member . . . . .	6
3.12	Overlapping_Borrowers . . . . .	6

# 1 Introduction

This document provides a comprehensive explanation of the Library Management System code. It includes the structure, functionality of each section, and a user guide for available commands. The system is implemented in C and is designed to manage books, members, and borrowing transactions efficiently.

## 2 Code Overview

The program serves as a Library Management System that tracks books, members, and borrowing records. The core functionalities include adding books and members, recording borrow transactions, and providing various data queries for management purposes.

### 2.1 Global Variables

The program utilizes several global arrays to store data for books, members, and borrowing transactions:

- `book_ids`, `book_copies`, `book_borrowed_counts`: Store book details like IDs, number of copies, and borrow counts.
- `member_ids`, `member_borrowed_counts`: Store member details like IDs and the count of borrowed books.
- `borrow_book_ids`, `borrow_member_ids`, `borrow_dates`: Store borrowing transaction details including book IDs, member IDs, and dates.

### 2.2 Input Procedure

Input data is provided via the terminal in a structured format. Below is an example illustrating the expected input format:

Books:

```
101 5
102 3
103 7
104 2
105 0
```

Members:

```
201000
202000
203000
```

Borrowed\_Books:

```
101 201000 01/12/2024
101 202000 01/12/2024
102 202000 03/12/2024
103 201000 03/12/2024
104 202000 01/12/2024
^Z
```

The input begins with book data, followed by member data, and finally borrowing transaction data. Each section is marked with a specific header.

## 3 Query Commands

### 3.1 Number\_Books

Displays the total number of books in the library by simply printing the value stored in the variable `book_count`, which is updated whenever a book is added or removed from the system.

```
1 void number_books() {  
2     printf("%d\n", book_count);  
3 }
```

Listing 1: Number Books Function

### 3.2 Number\_Members

Displays the total number of members in the library by printing the value of `member_count`, which keeps track of all registered library members.

```
1 void number_members() {  
2     printf("%d\n", member_count);  
3 }
```

Listing 2: Number Members Function

### 3.3 Book\_ID\_Min

Finds the book with the smallest ID by iterating through the array `book_ids` and comparing each ID to the current minimum. The smallest ID is then printed. If there are no books, it outputs `none`.

```
1 void book_id_min() {  
2     if (book_count == 0) {  
3         printf("none\n");  
4         return;  
5     }  
6     int min_id = book_ids[0];  
7     for (int i = 1; i < book_count; i++) {  
8         if (book_ids[i] < min_id) {  
9             min_id = book_ids[i];  
10        }  
11    }  
12    printf("%d\n", min_id);  
13 }
```

Listing 3: Book ID Min Function

### 3.4 Books\_Available

Lists all books with available copies for borrowing by checking if the number of borrowed copies is less than the total copies for each book. If no such books exist, it outputs `none`.

```
1 void books_available() {  
2     int found = 0;  
3     for (int i = 0; i < book_count; i++) {  
4         if (book_copies[i] > book_borrowed_counts[i]) {  
5             printf("%d\n", book_ids[i]);  
6         }  
7     }  
8     if (found == 0) {  
9         printf("none\n");  
10    }  
11 }
```

```
6         found = 1;
7     }
8 }
9 if (found == 0) printf("none\n");
10 }
```

Listing 4: Books Available Function

### 3.5 Most\_Borrowed

Identifies the most borrowed book(s) by finding the maximum value in `book_borrowed_counts`. It then prints the IDs of books that match this maximum value. If no books have been borrowed, it outputs `none`.

```
1 void most_borrowed() {
2     if (book_count == 0) {
3         printf("none\n");
4         return;
5     }
6     int max_borrows = 0;
7     for (int i = 0; i < book_count; i++) {
8         if (book_borrowed_counts[i] > max_borrows) {
9             max_borrows = book_borrowed_counts[i];
10        }
11    }
12    if (max_borrows == 0) {
13        printf("none\n");
14        return;
15    }
16    for (int i = 0; i < book_count; i++) {
17        if (book_borrowed_counts[i] == max_borrows) {
18            printf("%d\n", book_ids[i]);
19        }
20    }
21 }
```

Listing 5: Most Borrowed Function

### 3.6 List\_Book\_Borrowers

Lists all members who have borrowed a specific book by iterating through the `borrow_book_ids` array and checking for matches with the given book ID. It then prints the corresponding member IDs.

```
1 void list_book_borrowers(int book_id) {
2     int found = 0;
3     for (int i = 0; i < borrow_count; i++) {
4         if (borrow_book_ids[i] == book_id) {
5             printf("%d\n", borrow_member_ids[i]);
6             found = 1;
7         }
8     }
9     if (!found) printf("none\n");
10 }
```

Listing 6: List Book Borrowers Function

### 3.7 List\_Member\_Books

Lists all books borrowed by a specific member by iterating through `borrow_member_ids` and matching the given member ID. The corresponding book IDs are then printed.

```
1 void list_member_books(int member_id) {
2     int found = 0;
3     for (int i = 0; i < borrow_count; i++) {
4         if (borrow_member_ids[i] == member_id) {
5             printf("%d\n", borrow_book_ids[i]);
6             found = 1;
7         }
8     }
9     if (found == 0) printf("none\n");
10 }
```

Listing 7: List Member Books Function

### 3.8 Members\_Less

Lists members who have borrowed fewer than a specified number of books by iterating through `member_borrowed_counts` and printing the IDs of members who meet this criterion. If no such members exist, it outputs `none`.

```
1 void members_less(int n) {
2     int found = 0;
3     for (int i = 0; i < member_count; i++) {
4         if (member_borrowed_counts[i] < n) {
5             printf("%d\n", member_ids[i]);
6             found = 1;
7         }
8     }
9     if (found == 0) printf("none\n");
10 }
```

Listing 8: Members Less Function

### 3.9 Books\_Unborrowed

Lists all books that have never been borrowed by checking if the borrow count for each book in `book_borrowed_counts` is zero. If no such books exist, it outputs `none`.

```
1 void books_unborrowed() {
2     int found = 0;
3     for (int i = 0; i < book_count; i++) {
4         if (book_borrowed_counts[i] == 0) {
5             printf("%d\n", book_ids[i]);
6             found = 1;
7         }
8     }
9     if (found == 0) printf("none\n");
10 }
```

Listing 9: Books Unborrowed Function

### 3.10 Books\_Borrowed\_Days

Counts the number of unique days on which books were borrowed by comparing the dates in `borrow_dates`. It checks for duplicates before incrementing the count of unique days.

```
1 void books_borrowed_days() {
2     int unique_days = 0;
3     for (int i = 0; i < borrow_count; i++) {
4         int is_unique = 1;
5         for (int j = 0; j < i; j++) {
6             if (strcmp(borrow_dates[i], borrow_dates[j]) == 0) {
7                 is_unique = 0;
8                 break;
9             }
10        }
11        if (is_unique) unique_days++;
12    }
13    printf("%d\n", unique_days);
14 }
```

Listing 10: Books Borrowed Days Function

### 3.11 Books\_Per\_Member

Displays the number of books borrowed by each member by iterating through `member_borrowed_counts` and printing each member ID along with their corresponding count.

```
1 void books_per_member() {
2     for (int i = 0; i < member_count; i++) {
3         printf("%d %d\n", member_ids[i], member_borrowed_counts[i]);
4     }
5 }
```

Listing 11: Books Per Member Function

### 3.12 Overlapping\_Borrowers

Identifies members who borrowed the same book on the same day by checking for matching `borrow_book_ids` and `borrow_dates` in the borrow records. It prints the IDs of overlapping borrowers.

```
1 void overlapping_borrowers(int book_id) {
2     int found = 0;
3     for (int i = 0; i < borrow_count; i++) {
4         if (borrow_book_ids[i] == book_id) {
5             for (int j = i + 1; j < borrow_count; j++) {
6                 if (borrow_book_ids[j] == book_id && strcmp(borrow_dates[i],
7                     borrow_dates[j]) == 0) {
8                     printf("%d\n%d\n", borrow_member_ids[i], borrow_member_ids[j]);
9                     found = 1;
10                }
11            }
12        }
13    }
14    if (!found) printf("none\n");
15 }
```

Listing 12: Overlapping Borrowers Function