```c
/*V.1
  LIBRARY MANAGEMENT SYSTEM
AUTHOR'S
DHARANEESH R S     CSE053        CSE-A
DEEPAN G           CSE043        CSE-A
GANESH PRABU BO    CSE063        CSE-A
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_STUDENTS 100
#define MAX_BOOKS 100

//STRUCTURE DECLARATION

struct Student {                                        //structure used to store student information like (name , id)
    int id;
    char name[50];
};

struct Book {                                           //structure used to store book information like ( book id , book title, due date)
    int id;
    char title[50];
    int due_date;
};

struct BorrowRecord {                                   //structure used to store Borrow information like ( borrow id , book id, borrow date, return date)
    int student_id;
    int book_id;
    int borrow_date;
    int return_date;
};

```

```c
//SETTING THE MAX LIMT OF ARRAY USING STRUCTS

struct Student students[MAX_STUDENTS];                          //It just add maximum limit OR upper limit that array can store
struct Book books[MAX_BOOKS];
struct BorrowRecord records[MAX_BOOKS];

//INTIALISATION OF VARIABLES (GLOBAL)

int student_count = 0;                                          //IT TRACKS NUMBER OF STUDENTS CURRENTLY ADDED TO THE SYSTEM (When a new student is added using addStudent, this counter is incremented.)
int book_count = 0;                                             //IT TRACKS NUMBER OF BOOKS CURRENTLY ADDED TO THE SYSTEM (It is incremented each time a new book is added using addBook.)
int record_count = 0;                                           //IT TRACKS NUMBER OF BORROW RECORDS CURRENTLY STORED IN THE SYSTEM (It is incremented each time a book is borrowed using borrowBook.)

//FUNCTION DECLARATION

int addStudent(int id, char name[]);                            //IT ADDS NEW STUDENT TO THE SYSTEM
int addBook(int id, char title[], int due_date);                //IT ADDS NEW BOOKS TO THE SYSTEM
int borrowBook(int student_id, int book_id, int borrow_date);   //IT RECORDS THAT A STUDENT BORROWED
int returnBook(int student_id, int book_id, int return_date);   //Handles the return of a borrowed book and calculates late fees if applicable.
void displayLateFees(int student_id);                           //Displays the total late fees for a specific student.
void displayNewBooks();                                         //Displays the list of books in the system, including their IDs, titles, and due dates.
int modifyBookName(int id, char new_title[]);                   //Updates the title of a book based on its ID.
int removeBook(int id);                                         //Removes a book from the system based on its ID.
void displayTeamMembers();                                      //Displays the names and IDs of the team members who created the program.
void displayNewStudents();                                      //Displays the list of students in the system, including their IDs and names.

//MAIN FUNCTION

int main() {
    int choice;
```

```c
    do {
        //INTIALISATION OF VARIABLES (LOCAL) #TEMPORILY
        int student_id;
        int book_id;
        int borrow_date;
        int return_date;
        int id;
        char name[50];
        char title[50];
        int due_date;

        //WELCOME MESSAGE

        printf("<-----------------------------------Welcome to Library Manager Application
  ----------------------------------------> \n");

        //PROMPTING FOR ACCOUNT TYPE

        printf("PRESS 1 : IF YOU ARE CREATING NEW ACCOUNT  OR  PRESS 2 : IF YOU ARE AN EXISTING STUDENT \n");
        int qa;
        printf("Enter Your Answer\n");
        scanf("%d", &qa);
        getchar();

        //HANDLES NEW ACCOUNT

        if (qa == 1) {
            printf("Enter Student ID : \n");
            scanf("%d", &id);
            getchar();
            printf("Enter Student Name (caps initial follows name example: Dharaneesh RS) : \n");
            fgets(name, 50, stdin);
            name[strcspn(name, "\n")] = '\0';
            addStudent(id, name);

        //HANDLES EXISTING STUDENT ACCOUNT

```

```
100         } else if (qa == 2) {
101             int qa1;
102             printf("<=======================================EXISTING STUDENT WINDOW=================================>\n
    ");
103             printf("PRESS 1 : TO ADD BOOK\nPRESS 2 : TO BORROW BOOK\nPRESS 3 : TO RETURN A BOOK \nPRESS 4 : TO DISPLAY NEW
     BOOKS\nPRESS 5 : TO MODIFY BOOK NAME \nPRESS 6 : TO REMOVE A BOOK \nPRESS 7 : TO DISPLAY NEW STUDENTS\nPRESS 8 : TO VIEW
    TEAM MEMBERS NAME \n");
104             scanf("%d", &qa1);
105             getchar();
106 
107             //ADD NEW BOOK OPTION
108 
109             if (qa1 == 1) {
110                 printf("<_________________________ADD NEW BOOK WINDOW IS OPENED NOW_____________________________>\n\n");
111                 printf("Enter New Book ID\n");
112                 scanf("%d", &book_id);
113                 getchar();
114                 printf("Enter New Book Name\n");
115                 fgets(title, 50, stdin);
116                 title[strcspn(title, "\n")] = '\0';
117                 printf("Enter the Permitted due date (YYYYMMDD) : \n");
118                 scanf("%d", &due_date);
119                 addBook(book_id, title, due_date);
120             }
121 
122             //BORROW A BOOK OPTION
123 
124             else if (qa1 == 2) {
125                 printf("Enter Student ID : \n");
126                 scanf("%d", &student_id);
127                 printf("Enter Book ID\n");
128                 scanf("%d", &book_id);
129                 printf("Enter Borrow Date (YYYYMMDD) : \n");
130                 scanf("%d", &borrow_date);
131                 borrowBook(student_id, book_id, borrow_date);
132             }
133 
```

```
134             //RETURN A BOOK OPTION
135
136             else if (qa1 == 3) {
137                 printf("<----------------------------RETURN BOOK WINDOW IS OPENED NOW------------------------------>\
    n\n");
138                 printf("Enter Student ID : \n");
139                 scanf("%d", &student_id);
140                 printf("Enter Book ID\n");
141                 scanf("%d", &book_id);
142                 printf("Enter RETURN Date (YYYYMMDD) : \n");
143                 scanf("%d", &return_date);
144                 returnBook(student_id, book_id, return_date);
145             }
146             //DISPLAY ALL BOOKS OPTION
147
148             else if (qa1 == 4) {
149                 displayNewBooks();
150             }
151
152             //MODIFY BOOK NAME OPTION
153             else if (qa1 == 5) {
154                 printf("Enter Book ID to Modify: \n");
155                 scanf("%d", &book_id);
156                 getchar();
157                 printf("Enter New Book Title: \n");
158                 fgets(title, 50, stdin);
159                 title[strcspn(title, "\n")] = '\0';
160                 modifyBookName(book_id, title);
161             }
162             //REMOVE BOOK NAME OPTION
163             else if (qa1 == 6) {
164                 printf("Enter Book ID to Remove: \n");
165                 scanf("%d", &book_id);
166                 removeBook(book_id);
167             }
168             //DISPLAY NEW STUDENTS OPTION
169             else if (qa1 == 7) {
```

```c
                displayNewStudents();
            }
            //DISPLAY THE AUTHOR OF PROJECT AND TEAM MEMBERS NAME
            else if (qa1 == 8) {
                displayTeamMembers();
            }
            //ERROR HANDLING FOR INVALID INPUT
            else {
                printf("ENTER VALID INFORMATION \n");
            }
        }
        //ERROR HANDLING FOR INVALID INPUT
        else {
            printf("ENTER VALID INFORMATION \n");
        }
        //TO CONTINUE OR EXIT OPTION
        printf("Do you want to continue? \nPress : 1 for Yes(or)Press : 0 for No\nCHOOSE YOUR OPTION =>");
        scanf("%d", &choice);
    } while (choice != 0);//LOOP UNTILL EXIT

    return 0;
}

int addStudent(int id, char name[])                                     //FUNCTION SIGNATURE
{
    if (student_count < MAX_STUDENTS)                                   //CONDITION TO CHECK MAXIMUM STUDENTS
     {
        students[student_count].id = id;                                //ASSIGN THE VALUE OF STUDENT ID TO ID
    PRESENT IN STRUCT
        strcpy(students[student_count].name, name);                     //COPYS THE NAME STRING TO NAME FIELD IN
    STRUCT(WE USED STRCPY BECAUSE DIRECT ASSIGNMENT[=] OF STRING IS NOT POSSIBLE)
        student_count++;                                                //INCREMENTS THE COUNT OF THE STUDENT TO
    REFLECT THE ADDITION OF NEW STUDENT
        printf("Student added successfully!\n");
    }
    else                                                                //HANDLING THE OVERFLOW BECAUSE WHEN WE ADD
    MORE STUDENTS MORE THAN MAX_STUDENTS THE SYSTEM HAS NO SPACE TO STORE NEW STUDENT RECORDS
```

```
203     {
204         printf("Student limit reached!\n");
205     }
206 
207     return 0;
208 }
209 
210 int addBook(int id, char title[], int due_date)                    //Function Signature
211 {
212     if (book_count < MAX_BOOKS)                                    //CHECK FOR MAXIMUM BOOKS
213     {
214         books[book_count].id = id;                                 //Assigns the book's ID to the id field of
    the Book struct
215         strcpy(books[book_count].title, title);                    //Copies the title string to the title field
    of the Book struct.
216         books[book_count].due_date = due_date;                     //Stores the book's due date in the due_date
    field.
217         book_count++;
218         printf("Book added successfully!\n");
219     }
220     else                                                           //HANDLING OVERLOW
221     {
222         printf("Book limit reached!\n");
223     }
224 
225     return 0;
226 }
227 
228 int borrowBook(int student_id, int book_id, int borrow_date)       //FUNCTION SIGNATURE
229 {
230     if (record_count < MAX_BOOKS)                                  //CHECK FOR MAXIMUM BORROW RECORDS
231     {
232         records[record_count].student_id = student_id;             //Stores the student’s ID in the borrow
    record.
233         records[record_count].book_id = book_id;                   //Stores the book's ID in the borrow record.
234         records[record_count].borrow_date = borrow_date;           //Logs the date of borrowing.
235         records[record_count].return_date = -1;                    //initializes the return date with -1 to
```

```
indicate the book has not yet been returned.
        record_count++;
        printf("Book borrowed successfully!\n");
    }
    else
    {
        printf("Borrow record limit reached!\n");                          //HANDLING OVERFLOW
    }

    return 0;
}

int returnBook(int student_id, int book_id, int return_date)               //FUNCTION SIGNATURE
{
    for (int i = 0; i < record_count; i++)                                 //Search for Matching Borrow Record using for
 loop
    {
        if (records[i].student_id == student_id && records[i].book_id == book_id && records[i].return_date == -1)
                                                                           /*Checks three conditions:
                                                                               # Matching student_id: Ensures the record
 belongs to the student returning the book.
                                                                               # Matching book_id: Ensures the record
corresponds to the correct book.
                                                                               # return_date == -1: Ensures the book has
 not already been returned.
                                                                           */
        {
            records[i].return_date = return_date;                           //Update the Return Date
            int late_days = return_date - books[i].due_date;                //Calculate Late Fees
            int late_fees = (late_days > 0) ? late_days * 10 : 0;
            printf("Book returned successfully!\n");                        //PRINT SUCCESS MESSAGE
            printf("Late fees: %d units(100 units = 1 rupees)\n", late_fees);
            printf("\nIn Rupees : %d\n",late_fees/100);
            return 0;
        }
    }
    printf("No matching borrow record found or book already returned!\n");    //ERROR HANDLING
```

```c
    return -1;
}

void displayLateFees(int student_id)                                    //FUNCTION SIGNATURE
{
    int total_fees = 0;                                                 //Initialize total_fees to keep track of the
 accumulated late fees.
    for (int i = 0; i < record_count; i++)                              //Iterate through Borrow Records
    {
        if (records[i].student_id == student_id && records[i].return_date != -1)
                                                                        /* Condition 1 (records[i].student_id ==
student_id):
                                                                              Ensures the record belongs to the
specific student whose fees are being calculated.
                                                                           Condition 2 (records[i].return_date != -1
):
                                                                              Ensures that the book has been
returned (i.e., return_date is not -1).*/
        {
            int late_days = records[i].return_date - books[i].due_date;     //Calculate Late Fees for Each Record
            total_fees += (late_days > 0) ? late_days * 5 : 0;              //Calculate late_days & Calculate Late Fees
        }
    }
    printf("Total late fees for Student ID %d: %d units\n", student_id, total_fees);


}

void displayNewBooks()                                                  //FUNCTION SIGNATURE
{
    printf("ID\tTitle\t\t\tDue Date\n");                                //Print Table Header
    printf("----------------------------------------\n");
    for (int i = 0; i < book_count; i++)                                //Iterate Over Books and Display Details
    {
        printf("%d\t%s\t\t%d\n", books[i].id, books[i].title, books[i].due_date);
    }
    printf("\n");
```

```c
}

int modifyBookName(int id, char new_title[])                    //FUNCTION SIGNATUTRE
{
    for (int i = 0; i < book_count; i++)                        //Iterate Over Books to Find Matching ID
    {
        if (books[i].id == id)
        {
            strcpy(books[i].title, new_title);                  /*Loop through the books array to search for a book with the matching id.
                                                                  books[i].id == id checks if the current book ID matches the provided id.
                                                                  If a match is found, the book title is updated with new_title using strcpy().*/
            printf("Book title modified successfully!\n");
            return 0;                                           //SUCESS
        }
    }
    printf("Book with ID %d not found!\n", id);                 //BOOK NOT FOUND MESSAGE
    return -1;                                                  //NOT FOUND
}

int removeBook(int id)                                          //FUNCTION SIGNATURE
{
    for (int i = 0; i < book_count; i++)                        //ITERATE OVERBOOKS COUNT TO FIND MATCHING ID
    {
        if (books[i].id == id)
        {
                                                                /*Loop through the books array to find the book with the matching id.
                                                                   If a matching id is found, shift all subsequent books one position to the
                                                                   left, effectively removing the book from the array.*/
            for (int j = i; j < book_count - 1; j++)
            {
```

```c
                books[j] = books[j + 1];
            }
            book_count--;
            printf("Book removed successfully!\n");
            return 0;                                                   //SUCESS
        }
    }
    printf("Book with ID %d not found!\n", id);
    return -1;                                                          //ERROR HANDLING
}
void displayNewStudents()                                               //FUNCTION SIGNATURE
{
    printf("ID\tName\n");                                               //Print Table Header
    printf("----------------------\n");
    for (int i = 0; i < student_count; i++)                             //Loop Through Students and Print
Information
    {
        printf("%d\t%s\n", students[i].id, students[i].name);
    }
    printf("\n");
}

void displayTeamMembers()                                               //DISPLAYING THE TEAM MEMBERS NAME
{
    printf("\n<----------------------------Our Team---------------------------------------->\n");
    printf("\nDharaneesh RS      CSE053\nDeepan G           CSE043\nGanesh Prabu BO    CSE063           \n\n");
    printf("\nThank You\n");
}

```