A

PROJECT REPORT

ON

LIBRARY MANAGEMENT SYSTEM

SUBMITTED BY

[DURGESH KUMAR KAUSHIK]

[Roll No. : 2120140080058]

A.N. COLLEGE, PATNA - 13

(A Constituent Unit of Patliputra University)

NAAC Accreditation – 'A' Grade

TABLE OF CONTENTS

DECLARATION

We hereby declare that this project on "*LIBRARY MANAGEMENT SYSTEM*" have been designed and done by us.

In completing this project we had to undergo a lot of research and a lot of thought was put into it before completing this work. We had searched various books, web-sites in order to complete this assignment. We had asked many persons who were related in this field for their generous help.
We have tried our level best to make it free of any possible errors. If, any there is any error then it is just a human error and we are solely responsible for it and are sorry for it .

Thanking you,

DURGESH KUMAR KAUSHIK

INTRODUCTION

As we know that today is the world of computers and it has entered in the each and every phase of everyday life . Computer plays an important role in day to day work . Hence, today is the world of computers .

The use of computer in the field of management of information is well known to us . The use of computer in the management of library can be very helpful to the management of library . As all of us know that the management of library is a tedious job . Traditionally it was done manually. But now as all the systems is going to be computerised , it is found necessary to computerize the management of library processes.

Library management system has been developed to automate the whole library management , so that the efficiency of the staff of the institute can be enhanced . The work of acquisition, issuing , maintaining the record of every student is very tough when it done manually , but the computer does these works and it becomes very much easier to maintain these things .library is the store-house of knowledge. It preserves the wisdom of the past age.

It is a place of study where we get books on various subjects and derive great benefit. There is a good library in Kautilya Computer Cources (KCC). It is housed in a big room. There are almirahs in the library which contains books on different subjects. Student and faculty of the KCC borrow books from the library. One of our centre staff looks after the library and he is called librarian. A student can borrow a book for a week. Next week he returns the books to the library and he gets other books. If he cannot finish the book in a week, he can get it re-issued for another week. If he fails to return the book on due date, he has to pay a fine for the delay. The library is meant for students and faculty only. Other person can not borrow books from it. There are about five thousand books in KCC. These books are in English only. The books have been classified into different sections. Every year a large number of new books are bought. The students increase their knowledge by reading different kinds of books. The library is very useful because, every student can not buy many books. He can borrow different kinds of books from the library. So the institute library is very useful to students. Any student can become a member of the library by filling a prescribed form and paying appropriate amount of fees. This membership is given for one-year duration, during which members can avail following facilities:

- They can get the books and journals for reading/reference
- They can get the books issued so that they can take them home and return them on or before due date.

For providing the reading services, library purchases from the market. When books arrive in the library, they are physically inspected and it they are found in order, they are taken into account, else they are rejected. Details of the books are noted in the acquisition register. After these formalities, book can be issued to the readers on their request.

When a member requests for a book, librarian searches it in the library, according to the acquisition number and accession number of the book. If it is available then it is issued to the member and details of the issue are noted down in a register called issue register.

After reading the book, members return the book to the librarian. At the time of returning of the book, librarian checks if the book is being returned within due date. If yes then he takes the book back else he calculates the amount of fine.

Fine amount is collected from the member and issue details are stroked off from the issue register.

Time to time librarian scans through the issue register and finds out list of books, which are overdue. There are following activities of librarian of library.

- Acquisition of books
- Membership maintenance
- Book issue
- Book return
- Renewal of membership

- Query

Acquisition of books

Purchase department of library purchases various books. After purchase, these books are inspected and if found in order a unique number called acquisition number is assigned to each book. Purchase department then sends these books to librarian. Based on the subject, author of the book etc. Librarian assigns a code to the book. This code is called as accession number .He also enters various details regarding a book in acquisition register. He changes entries of the acquisition register, if some change in book information takes place.

Membership maintenance

A person who wants to be a member of the library, has to fill up application form and give it to the librarian along with membership fee and caution money. Librarian assigns a unique membership number to him and enters the details of the member in membership register. After some time if any member wants to reduce and increase his caution money, librarian makes changes accordingly in the membership modification form.

If a member wants to discontinue in between, librarian strikes of his entry from the membership register and returns the caution money.

Book issue

Librarian issues only one book at a time to a member. A member fills a requisition slip and requests the librarian to issue the book to him. Format of the requisition slip is checked by the librarian and the librarian first checks in member register, whether is membership is still continuing. If not, he does not accept the request else he physically checks for the presence of the book in the library. If not present, he rejects the request of the member; otherwise he issues the book to member. Librarian notes down the details in issue register.

Book Return

When a member returns a book, librarian first checks whether the member is returning the book in time or not. If it is being returned late then he charges fine at the rate rs1.00 per day from the member. He updates the fine register by entering the date of actual return from the schedule return date and fine collected (if charged).

Renewal of membership

Librarian provides membership only for one-year duration. After this, if members want to continue their membership, they have to give an application to the librarian requesting him to extend their membership for next year duration. This application has to be supported by appropriate amount of renewal fee. On receiving the application librarian makes appropriate entries in the membership register and renews the

membership so that member can avail various facilities of the library, for next year also.

OBJECTIVES OF THE PROJECT

- To develop a software for library management system
- Effectiveness of present working system
- Reduces involvement of man power
- Effective query system for delayed /dead deals and immediate decision making on delayed /stopped operations.

Project category:

Category of this project will be “Object Oriented Programming System”. Object oriented technologies leads to reuse and reuse leads to faster software development and higher quality programs.

DRAWBACKS WITH EXISTING SYSTEM :

1) Time Consuming
2) Less Data Security
3) More Paper Works Are To Be Done.
4) No View Generation Facility
5) No Report Generation

ADVANTAGES OF THE PROPOSED SYSTEM :

1 . Less Time Consuming

2 . Less Paper Work

3. Report Generation Can Be Done Easily

4 . Query Can Be Done Easily

5. It Does Not Required Expert Hand

6. IT Is Fully User Friendly

7.Faster Access Of Information

TOOLS /PLATFORMS ,LANGUAGES USED

HARDWARE CONFIGURATION

PROCESSOR : PENTIUM III

Main Memory: : 64 MB

HDD (Free Space) : 10GB

FDD : 1.44MB

SVGA Colour monitor

Multimedia kit (CD drive, sound card, speaker, microphone)

System type: HCL infinity 2000 mtx system, with 32 bit file

Logitech mouse,105 keys Keyboard.

SOFTWARE:

OPERATING SYSTEM: windows 98

Programming language: c++

Documentation: MS-WORD XP

Why use C++ ?

C++ is a superset of c with features designed to support oops. OOP is a new way of organizing code and data that promises increased control over the complexity of the software development process.

Yet, C being a structured programming language offered the ease of software development but failed to support maintenance of large code.

With C++, it is much easier to build and maintain a really big code.

Object – Oriented Programming is cantered around new concepts such as Objects, Classes, Polymorphism, Inheritance, etc.

It is well-suited paradigm for the following: -

Modelling the real-world problem as close as possible to the user's perspective. Interacting easily with computational environment using

familiar metaphors. Constructing reusable software components and easily extendable libraries.

Easily modifying and extending implementations of components without having to recode every thing from scratch.

A language's quality is judged by twelve important criteria. They are a well defined syntactic and semantic structure, reliability, fast translation, efficient object code, orthogonality, machine independence, provability, generality, consistency with commonly used notations, subsets, uniformity, and extensibility. The various constructs of OOP languages (such as c++) are designed to achievethese with ease.

Unlike, the traditional methodology (Function-Oriented Programming), object-oriented programming emphasizes on the data rather than the algorithm. In oops, data is compartmentalized or encapsulated with the associated functions and this compartment or capsule is called an object.

In the OOP approach, the problem is divided into objects , whereas in function-oriented programming the problem is divided into functions. Although, both approaches adopt the same philosophy of divide and conquer, OOP conquers a bigger region, while function-oriented programming is content with conquering a smaller region. Object-Oriented Programming contains function-oriented programming and so OOP can be referred to as the superset of function-oriented programming and hence, it can be concluded that OOP has an edge over function-oriented programming.

Some of the most prominent features of C++ are classes, operator and function overloading, free store management, constant types, references, inline and friend function, inheritance, virtual functions, streams for console and file manipulations, templates and exception handling.

Advantage:-

Information hiding and data abstraction increase reliability and help decouple the procedural and representational specification from its implementation. Dynamic binding increases flexibility.

Inheritance coupled with dynamic binding enhances the reusability of code, thus increasing the productivity of a programmer. Code reuse is possible in conventional language as well, but object-oriented language greatly enhances the possibility of reuse.

Object–Orientation provides many other advantages in the production and maintenance of software; shorter development time, high degree of code sharing and malleability.

SOFTWARE PRINCIPLES USED:-

SYSTEM ANALYSIS

A. INTERACTION WITH USER: -

a. Get knowledge about existing system of the library

b. Assess the user's requirement and problems faced by

c. Re-defining the Problem.

d. To assess basic data, transactions of the book.

e. To review the various formats, forms, registers

f. To collect information about member/book.

g. Get the sense of various constraints

B. SOFTWARE DESIGNING:-

a. Preparing control flows.

b. Data base designing (preparing structures).

c. Setting up relationship among fields of various tables for effective updating and in ambiguity of data.

C. Program designing: -

a. Developing algorithm

b. Designing different input screens

c. Designing Reports

D. Programming: -

Writing source code for various modules

Main designing issues:- though it is the matter subjected to vary , yet some of the major designing issues to be covered which are:-

a. User friendly software through Event-driven techniques, Forms with control objects.

b. User of controls to provide easy navigation through different Database and operation

c. Back up.

d. Security

E. Documentation:-

a. User Manual :For smooth exploitation of software facilities by user

b. Technical Manual : For giving knowledge of technical shortcomings and problems likely to occur and methods to overcome that besides appraisal of technical features of this system.

F. Testing

G. Installation

H. Training to the users

TIME SCHEDULE FOR THE PROJECT PREPARATION

MODULE	DAYS

*	Interaction with user	5
*	System Analysis	5
*	Design	5
*	Program designing	5
*	Coding	10
*	Testing	5
*	Documentation	5
*	Installation	5
*	Implementation & training to user	3
Total Days Allotted	(8 weeks * 6 days each)=	48 days

SOFTWARE LIFE CYCLE:

Software systems pass through two principal phases during their life cycle.

1. The development phase.
2. The operation and maintenance phase.

The development phase begins when the need for the product is identified; it ends when the implemented product is tested and delivered for operation. Operation and maintenance include all activities during the operation of the software, such as fixing bugs discovered during operation, making performance enhancements, adapting the systems to its environment, adding minor features, etc.

Problem definition:-

The first stage in the development process understands the problem in question and its requirements. At this point, the managers and software specialists decide whether it is feasible to build the system.

Analysis: -

Analysis phase delivers requirements specification. Specification must be carefully checked for suitability, omission, inconsistencies and ambiguities. In this project the analysis is done with the help of the librarian and staffs after considering the needs of the library i.e. what is the basic need of the library and what it is requiring for the automation of the library. After going through the entire requirement we have come to the next phase of the Software Life Cycle.

Design: -

Design is the process of mapping systems requirements defined during analysis to an abstract representation of a specific-system implementation, meeting the cost and performance constraints. The purpose of design phase is to specify a particular software system that will meet the stated requirements. This project is divided into modules and their interactions. The modules may then be further decomposed into sub-modules and procedures until each module can be implemented easily.

Coding / Implementation: -

Once the specification and the design of the software is over, the choice of a programming language remains as one of the most critical aspect in producing reliable software. Implementation involves the actual production of code. Although it is one of the important phases, it takes only 20% of the total development time. In this project the generation of program is done with the use of appropriate programming language i.e. C++.

TESTING: -

Testing is the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies the specified requirements. Normally, most of the testing and debugging is done after the system has been implemented. A large percentage of errors

are discovered during testing originates in the requirement and design phase.

The techniques that have been propose for unit testing include the following: -

1. Path testing: - each possible path from input to output is traversed once.
2. Branch testing: - Each path must be traversed at least once.
3. Functional testing: - Each functional decomposition is tested at least once.
4. Special value testing: - testing for all values assumed to cause problems.

All the modules that have been developed before and tested individually are put together – integrated – in this phase and tested as a whole system. After doing this in the project the project will be almost ready to be delivered.

Maintenance: -

Once the system passes all the tests, it is delivered to the customer and enters the maintenance phase. Any modification made to the system after initial deliveries are usually attributed to this phase. Basically in this phase the software will be maintained and this is done prior to release of software

and I have reviewed the software and assured that all the documentation is available.

IMPORTANT FUNCTIONS AND THEIR PURPOSES

Important functions and purposes of the function are given below:-

1. **menu_console** :- function for adding modifying and deleting from LIB.DAT data file.
2. **new_library:**- function to enter new library books through the acquisition number. In an acquisition number there may be more than one book which contains several accession number.
3. **new_lib:**- function to append a record into LIB.DAT data file.
4. **lib_delete:**- function to delete a selected record from LIB .DAT data file.
5. **left_clear:**- function to perform a left clear on specified row from a specified column.
6. **right_clear:**- function to perform a right clear on specified row from a specified column.
7. **clear_area:**- function to clear a specified area in the screen.
8. **change_library**:- function for main menu to change the existing record in LIB.DAT data file. In the library the searching can only happen through the acquisition number and the accession number.
9. **replace**:- function to r ewrite the library contents at its original location.
10. **recordno**:- function to count the total number of records in LIB.DAT data file.

11. **delete_accno**:- function for main menu to delete the existing record in LIB.DAT data file.in the library the searching can only happen through the acquisition number and accession number

12. **record_delete**:- function to delete record from LIB.DAT data file by the matching acquisition number and accession number .

13. **book _issue**:- function to issue book to members.

14. **book _display**:- function to display the book information at the return time .

15. **report_menu**:- function to display various options related to report for answer.

16. **acqu_register**:- the function is used to print the report of acquisition register.

17**.memb_console**:- function for adding , modifying and deleting from MEMFILE.DAT data file.

18.**new_member**:- function for main menu to append new members in MEMFILE.DAT data file.

19**.memb_delete**:- function to delete a selected record from MEMFILE.DAT data file.

20**.memb_modify**:- library member to modification function .

21**.memb_check**:- function to membership check.

22**.expire_check**:- function to check membership date expired or not.

23**.change_member**:- function for main menu to change member

information in MEMFILE.DAT data file.

24.**display_member**:- function to display the existing member in screen to change the further information.

25.**replace**:- function to rewrite the member's contents at its original location.

26.**recordno**:- function to find the position in data base file.

27.**delete_member**:- function to delete member from MEMBAR.DAT data file. While executing the procedure it first display the member information and checks in the MASTER.DAT data file, Whether the member is issued any book or not. If the member is issued any book then the record can not be deleted, otherwise the record will be deleted from the MEMBER.DAT data file.

28.**member_renewal**:- function to renewal of membership.

29.**renewal_member**:- function to display the existing member and renewal the membership date.

30.**memb_register**:- function to display the membership register globally.

31.**expire_register**:- function to display the expired membership register.

32.**Issue_check**:-function to check the book is issued or not while deleting a library book.

33.**issue_memb_check**:- function to issue member check.

34.**issue_add**:- function to add issued book into MASTER.DAT data file.

35**.issue_delete**:- function to delete a selected record from MASTER.DAT data file.

36**.book _return**:- function to return books in library. The user has to enter the member number and the screen will display the details of the book which is returning.

37**.return_issue**:-function for displaying the returned book s information.

38**.issue_memb_delete**:- function to delete the issued book of member from MASTER.DAT data file.

39**.issue_register**:- function to display the issued book s in library.

40**.overdue_book :-** function to list overdue books.

41**.fine_add**:- function to add fines if return date exceeds.

42**.fine_register**:-function to display the fine collection register .

43.**lib_screen**:-function for displaying main menu for library management system.

44**.date_expire**:- function to find the expiry date of members.

45**.no_of_days**:- function to find the number of days in between two dates

46**.month_days**:- function to return the total days in a month.

DATA STRUCTURE(TABLES) OF MODULES:

LIB .DAT

Attribute	**Data Type**

Acquisition number	INT
Accession number	INT
Title	CHAR[30]
Author1	CHAR[30]
Author2	CHAR[30]
Publisher	CHAR[30]
Price	FLOAT
Dd/mm/yy	INT
Pages	INT
Status	CHAR
S_acquisition number	INT
S_accession number	INT
S_title	CHAR[30]
S_author1	CHAR[30]
S_author2	CHAR[30]
S_publisher	CHAR[30]
S_price	FLOAT
S_pages	INT
Smm/syy/sdd	INT
S_status	CHAR

MEMFILE.DAT

Attribute	**Data Type**

S_name	CHAR[30]
S_address	CHAR[30]
Memb_no	INT
Name	CHAR[30]
Address	CHAR[30]
Mdd/myy/mmm	INT
Edd/emm/eyy	INT
S_caut_fee	INT
S_memb_fee	INT
Memb_fee	FLOAT
Caut_fee	FLOAT
S_memb_no	INT
S_mdd/s_mmm/s_yyy	INT
S_edd/s_emm/s_eyy	INT

MASTER.DAT

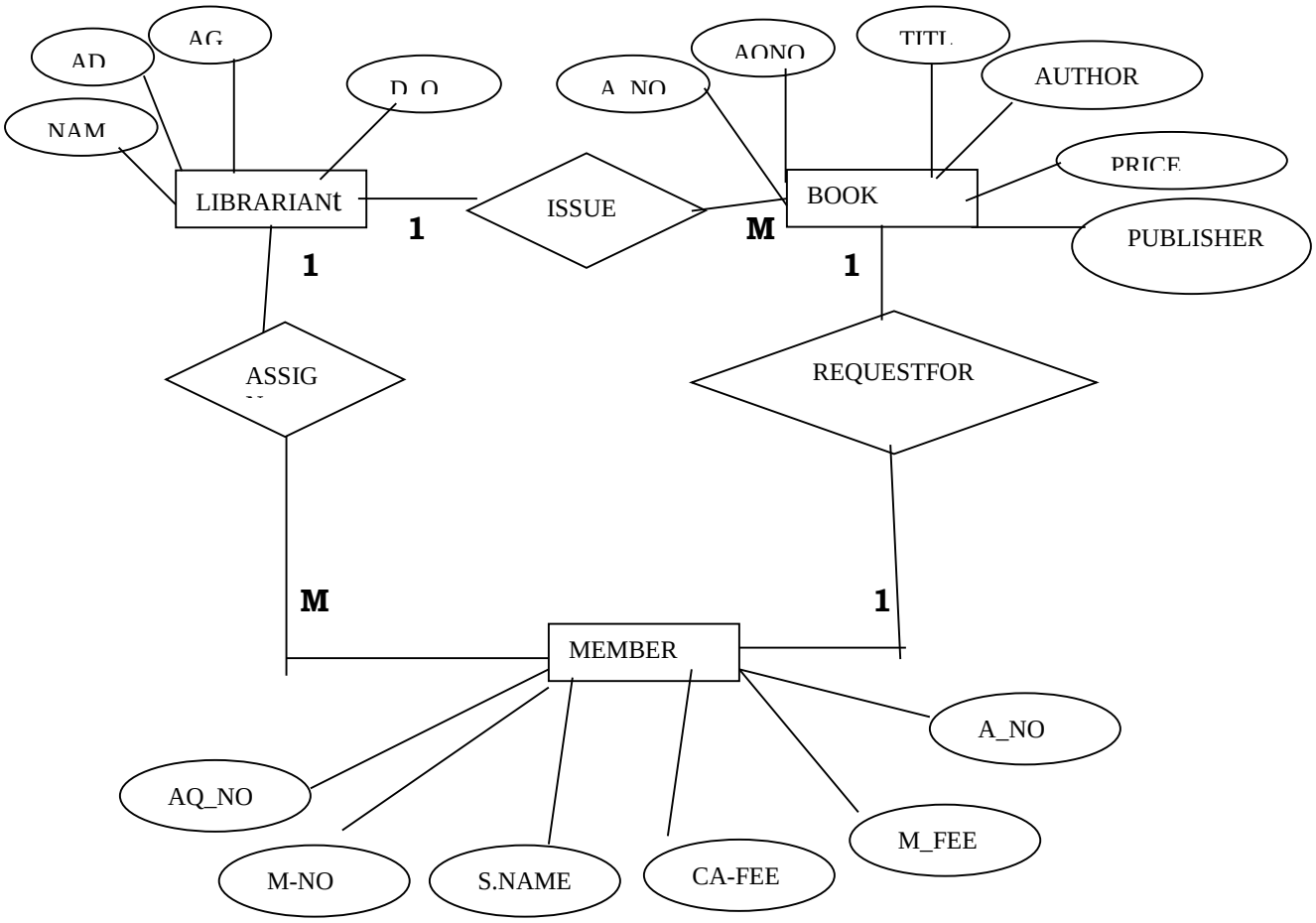
Attribute	**Data Type**
Acquisition number	INT
Accession number	INT
Memb_no	INT
S_acquisition number	INT
S_accession number	INT

S_memb_no	INT
S_idd/s_iyy/s_imm	INT
S_rdd/s_rmm/s_ryy	INT
S_cdd/s_cmm/s_cyy	INT
Idd /imm/iyy	INT
S_finfee	INT

FINE.DAT

Attribute	**Data Type**
Acquisition number	INT
Accession number	INT
Memb_no	INT
Rdd/rmm/ryy	INT
S_acquisition number	INT
S_accession number	INT
S_memb_no	INT
S_idd/s_iyy/s_imm	INT
S_rdd/s_rmm/s_ryy	INT
S_cdd/s_cmm/s_cyy	INT
Idd /imm/iyy	INT
S_finfee	INT
Fin_fee	INT

E-R DIAGRAM:-


AG
AD
D_O
NAM
LIBRARIANt
1
ISSUE
M
AONO
A_NO
TITL
AUTHOR
PRICE
BOOK
PUBLISHER
1
1
ASSIG
REQUESTFOR
M
1
MEMBER
A_NO
AQ_NO
M_FEE
M-NO
S.NAME
CA-FEE

SCREEN DESIGN:

This section displays the different screens, which have made in this project. The first is menu design.

MENU DESIGN:

The main menu has the following options:

1: Acquisition Section.

2: Membership Maintenance.

3: Book Issue

4:Book Return

5:Renewal Of Membership

6:Query

Acquisition Section:

1.Appending Record

2: Modify Record.

3: Delete Record.

0: Exit.

Membership Maintenance:

1:Recording New Member

2:Changing Member Data

3.Deleting Member Record

0.Exit

Book Issue :

Display the book issue form

Book Return :

Displays the book return form.

Renewal of membership :

Display the membership form

Query :

1.Acquisition Register

2.Membership Register

3.Issue Register

4.List Of Overdue Books

5.Fine Register

6.List Of Expired Members

0.Exit

What my project does:

This project starts with a menu, which tells the user to choose which area, he or she want to work in. The main menu has the following options:- acquisition section, membership maintenance, book issue, book return, renewal of membership, query, and Exit. The user has to choose any option from these options as per requirements.

If he choose the option of acquisition section then another menu will be get displayed on the screen ,in this screen first it displays the library console menu in which append, modify and delete record are defined.

If the user chooses the option 1 then he will have to fill the following options: the user fill up the following options:- acquisition number, accession number ,title, author name, publisher, price and pages of the book. After the entry record is appended to database file. If the user chooses the second option user can modify the record on the basis of acquisition number and accession number. If the user chooses the third option user can delete the record on the basis of acquisition number and accession number. If the user chooses the fourth option user can return from library console to main menu.

If he choose the option of membership maintenance then another menu will be get displayed on the screen, in this screen first it displays the membership console menu in which append, modify, delete record are defined.

If the user chooses the option 1 then he will have to fill up the membership form with the following options: - membership number, name, address, membership date, membership fees and caution money. After the entry of membership date, expiry date of membership automatically displayed. After the filling membership form, the record is appended to database file. If the user chooses the second option user can modify the record on the basis of membership number. If the user chooses the third option user can delete the record on the basis of membership number. If the user chooses the fourth option user can return from membership console to main menu.

If he choose the option of book issue then book issue form will be get displayed on the screen, in this screen he fill up the following options: accession number, acquisition number, title author, publisher price, page. Afterward record is appended to database file. If he choose the option of book return then book return form will be get displayed on the screen, in this screen he fill up the following options: membership number, name, accession number, address, membership date, membership fee, caution money issue date, title author, price, pages of the book. After the entry of issue date and membership date then return date and expiry date will be automatically displayed. All the entered records will be saved in the corresponding files, which can be further used for report generation.

If the user chooses the option 5 then he will have to fill up the new membership form with the following options: - membership number, renewal date. After the entry of membership number then name, address, membership date, expiry date, membership fees and caution money will be automatically displayed. After the entry of renewal date, the record will be saved in the corresponding files, which can be further used for report generation.

Another option from main menu is query option with the help of which one can get the reports. These options are acquisition register, membership register, issue register, list of overdue book, fine register, list of expired

members and exit. In the reporting options user can queries on the basis of accession number, acquisition number and membership number.

The last option of the main menu is exit from the gateway.

SOURCE CODE :

Source Code

```
// Declaration of header files

#include <iostream.h>
#include <fstream.h>
#include <process.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <iomanip.h>

// Function to find the expire dates of members
int date_expire(int, int, int);

// Function for printing Date and Time
void date_time()
{
        int xdd, xmm, xyy;
        struct time t;
        struct date d;                  // For extracting system date
        getdate(&d);
        xdd = d.da_day;
        xmm = d.da_mon;
        xyy = d.da_year;
        gotoxy(10, 3);
        textcolor(15);
        cout<<"Date:";
        cprintf("%d-%d-%d", xdd, xmm ,xyy);
        gotoxy(35,3);
        textcolor(7);
        cprintf("Time:");
        textcolor(15);
        gotoxy(65, 3);
        cout << "<0>=";
        textcolor(15);
        cprintf("Exit");
        while(!kbhit())
        {
                delay(100);
                gettime(&t);
                gotoxy(40,3);
                cprintf("%02d:%02d:%02d",t.ti_hour, t.ti_min, t.ti_sec);
                gotoxy(47,17);
```

```
        }
        textcolor(7);
}

// Function to find the no of days in between two dates
int no_of_days(int, int, int, int, int, int);

// Function to return the total days in a month
int month_days(int);


// class lib_class
class BOX_CLASS
{
        public:
                void DRAW_LINE(int, int, int, char);
                void DRAW_VER(int, int, int, char);
                void DRAW_BOX(int, int, int, int, char);
};

// Class for main control
class control
{
        public:
                void load(void);
                void lib_SCREEN(void);
                void help(void);
        private:
                void MENU_EDIT(void);
};


// Information regarding to library acquisition
class Library
{
        int acq_no;  // Acquisition number
        int acc_no;  // Accession number
        char title[30];
        char author1[30], author2[30];          // Authors
        char publisher[30];         // Name of the publisher
        float price;   // Price of the book
        int dd, mm, yy;     // Date of purchase
        int pages;    // Total pages in book
        char status;// Present status of book
        public:
```

```
        void Menu_console(void); // Menu for add, modify and delete
of library books
        void new_library(void);  // Function to enter new library
books through the acquisition number
        void NEW_lib(int, int, char s_title[30], char s_author1[30],
char s_author2[30], char s_publisher[30], float s_price, int sdd, int smm,
int syy, int s_pages, char s_status);
        void Lib_delete(int, int); // Deleting library information from
data file
        void Left_clear(int, int); // Function to perform a left clear on
specified row from a specified column
        void Clear_area(int, int, int, int); // Function to clear a
specified area in the screen
        void Right_clear(int, int); // Function to perform a right clear
on specified row from a specified column
        void Change_library(void);      // Function to change the
library information
        // Function to rewrite the library contents at its original
location
        void Replace(int, int, char s_title[30], char s_author1[30],
char s_author2[30], char s_publisher[30], float s_price, int sdd, int smm,
int syy, int s_pages, char s_status);
        int recordno(int, int);  // Function to find the position in file
        void Delete_Accno(void); // Function to delete records from
library
        void Record_delete(int, int);     // Function to delete the
record by copy method
        void Book_Issue(void);    // Function to issue books to
members
        void Book_display(int, int); // Function to display the book
information at return time


        // Report functions on Library of purchase department
        void Report_Menu(void);         // Function to display various
option on related to report for answer to management queires
        void Acqu_Register(void);       // Acquisition register for
report
};

// Class contains information containing the member details
class Member
{

    public:
```

```
        void Memb_console(void); // Menu for add, modify and delete
of Membership
        void new_member(void); // Function to select addition of
members
        // Deletes the same record after creating its position
        void Memb_delete(int);
        void Memb_modify(int); // Library member modification
function
        int Memb_Check(int);    // Membership check
        int Expire_Check(int); // Function to check membership date
expired or not
        void Change_member(void);    // Function to change the
members information
        void Display_member(int);      // Function to display the
existing information for modification
        // Function to rewrite the Member contents at its original
location
        void Replace(int, char s_name[30], char s_address[30], int,
int, int, int, int, int, float, float);
        int recordno(int);  // Function to find the position in file
        void Delete_member(void); // Function to delete member
from MEMBER.dat data file
        void Member_Renewal(void);    // Function to renewal of
membership
        void Renewal_member(int);     // Function where the
membership date is modified
        // Reporting function for member
        void Memb_Register(void); // Function to display the
membership register globally
        void Expire_Register(void); //Function to display the expired
membership register

    private:
        // Function to simply add records
        void NEW_memb(int, char s_name[30], char s_address[30],
int, int, int, int, int, int, float, float);

        int memb_no;      // Membership number

        char name[30];    // Name of the member
        char address[30]; // Address of the employee
        int mdd, mmm, myy;  // Membership date
        int edd, emm, eyy; // Expire date
        float memb_fee; // Membership fee
        float caut_fee;      // Caution money
```

```
};

// Information related to issue of books and issue register
class Issue
{
        public:
                int Issue_Check(int, int); // Function to check the book is
issued or not while deleting a library book
                void Issue_Add(int, int, int, int, int, int, int, int, int);
                void Issue_Delete(int s_acq_no, int s_acc_no, int s_memb_no,
int s_idd, int s_imm, int s_iyy, int s_rdd, int s_rmm, int s_ryy);
                int Issue_memb_check(int); // Function to issue member
check
                void Book_Return(void); // Function to return books
                void Return_Issue(int); // Function for displaying the
returned books information
                void Issue_member_delete(int); // Function to delete the
issued book of member from MASTER.dat data file
                void Issue_Register(void);              // Function to display
the issued books in library
                void OverDue_Book(void);        // Function for overdue books

                void Read_issue(void);
        private:
                int acq_no;  // Aquisition number
                int acc_no;  // Accession no
                int memb_no;        // Member number
                int idd, imm, iyy;  // Issue date
                int rdd, rmm, ryy; // Return date

};

// Information related to fine register
class Fine
{
        int memb_no; // Member number
        int     acq_no;       // fine acquisition no
        int acc_no;  // fine accession no

        int idd, imm, iyy;  // Book issue date

        int rdd, rmm, ryy; // Book purposed return date

        int cdd, cmm, cyy;          // Returned date
```

```
        int finfee;
        public:
                void Fine_add(int, int, int, int, int, int, int, int, int, int, int,
int, int);
                void Fine_Register(void);          // Function to display the fine
collection register
};

// Function for horizontal line draw
void BOX_CLASS::DRAW_LINE(int column1, int column2, int row, char c)
{
        for (column1; column1 <= column2; column1++)
        {
                gotoxy(column1, row);
                cout << c;
        }
}

// Function for vertical line draw
void BOX_CLASS::DRAW_VER(int row1, int row2, int column, char c)
{
        for (row1; row1 <= row2; row1++)
        {
                gotoxy(column, row1);
                cout << c;
        }
}

void BOX_CLASS::DRAW_BOX(int column1, int row1, int column2, int
row2, char c)
{
        char ch = 187;
        char c1, c2, c3, c4;
        char l1 = 196, l2 = 215;
        if (c == ch)
        {
                c1 = 218;
                c2 = 191;
                c3 = 217;
                c4 = 217;
                l1 = 196;
                l2 = 179;
        }
        else
        {
```

```
                c1 = c;
                c2 = c;
                c3 = c;
                c4 = c;
                l1 = c;
                c2 = c;
        }
        gotoxy(column1, row1);
        cout << c1;
        gotoxy(column2, row1);
        cout << c2;
        gotoxy(column1, row2);
        cout << c3;
        gotoxy(column2, row2);
        cout << c4;
        column1++;
        column2--;
        DRAW_LINE(column1, column2, row1, l1); //Horizontal line
        DRAW_LINE(column1, column2, row2, l1);
        column1--;
        column2++;
        row1++;
        row2--;
        DRAW_VER(row1, row2, column1, l2); // Vertical line
        DRAW_VER(row1, row2, column2, l2);
}

// Function to clear a row from left of the screen
void Library::Left_clear(int col, int row)
{
        for (int j = col; j <= 39; j++)
        {
                gotoxy(j, row);
                cout << " ";
        }
}

// Function to clear a row from right of the screen
void Library::Right_clear(int col, int row)
{
        for (int j = col; j <= 79; j++)
        {
                gotoxy(j, row);
                cout << " ";
        }
```

```
}

// Function to clear a specified area in the screen
void Library::Clear_area(int row, int till, int col, int tcol)
{
        for (int p = row; p <= till; p++)
        {
                int xcol = col;
                for ( xcol = xcol; xcol <= tcol; xcol++)
                {
                        gotoxy(xcol, p);
                        cout << " ";
                }
        }
}


//Function to display load about this project
void control::load(void)
        {
        textbackground(BLACK);
        clrscr();
                char uname[10];
                char pass[7];
                int i,t=100;

        _setcursortype(_NOCURSOR);
        gotoxy(37,24);
        textcolor(GREEN);
        cprintf("Loading...");
        for(i=1;i<80;i++)
        {
                gotoxy(i,25);
                textcolor(RED);
                printf("%c",221);
        }
        for(i=1;i<80;i++)
        {
                gotoxy(i,25);
                textcolor(GREEN);
                printf("%c",219);
                delay(t);
                if(i==25)t=80;
                if(i==50)t=20;
                if(i==75)t=10;
```

```
        }

        _setcursortype(_NORMALCURSOR);
        clrscr();
          textcolor(RED);
          gotoxy(20,10);
          cprintf("Enter User name :");
          gotoxy(20,12);
          cprintf("Enter Password  :");
          gotoxy(38,10);
          cin>>uname;

          gotoxy(38,12);


          int p=0;
          fflush(stdin);

          while((pass[p]=getch())!='\r')     { cout<<"*";
                p++;
          }
           pass[p]='\0';
           if(strcmpi(pass,"Patna")!=0||strcmpi(uname,"Kcc")!=0)
          { gotoxy(20,22);
          cout<<"Incorrect Password!!! coming out";
          getch();
          clrscr();
          system("cls");
          exit(1);
           }
        }



// Function that provides help about operations and options
void control::help(void)
{
        clrscr();
        BOX_CLASS s;
        s.DRAW_BOX(2, 1, 79, 25, 218);
        s.DRAW_BOX(25, 2, 54, 4, 219);
        gotoxy(27, 3);

        // cprintf  sends formatted output to the text window on the screen
        textcolor(GREEN);
```

```
        cprintf("Welcome to Library Project");

        // textcolor selects a new character color in text mode
        textcolor(LIGHTGRAY);
        delay(2);

        gotoxy(5, 6);
        cout << "This Software can be used to handle the Library
Managment System.It can ";
        gotoxy(5, 7);
        cout << "be used for keeping the detailed information about
Books/Member/Issue/";
        gotoxy(5,8);
        cout<<"Return/fine etc.in a automated and efficent mannerIt helps
in capturing ";
        gotoxy(5, 9);
        cout << "and manipulating data in a very easy and computerised
manner. It reduces ";
        gotoxy(5, 10);
        cout << "not only paperwork but response time also.";
        gotoxy(5, 13);
        cout << "It has the following options:- ";
        gotoxy(6, 15);
        cout << "-In second option you can add/modify or delete any
member.";
        gotoxy(6, 16);
        cout << "-In third option you can record the Issue of Book
operation.";
        gotoxy(6, 17);
        cout << "-In fourth option you can record the Recieving of book
operation.";
        gotoxy(6, 18);
        cout << "-The fifth option helps you in renewing a membership of a
member";
        gotoxy(6, 19);
        cout << "-The sixth option helps in answering the queries done by
management.";
        gotoxy(6, 22);
        cout << "Note-: To return from any menu press 0 (Zero) at any point
then you will";
        gotoxy(13, 23);
        cout << "you will be sent back to the previous menu or to the
shell.";
        textcolor(BLACK+BLINK);
        textbackground(WHITE);
```

```
        gotoxy(26, 25);
        cprintf("Press any key to continue...");
        textcolor(LIGHTGRAY);
        textbackground(BLACK);
        gotoxy(25, 2);
        getch();

}


// Function for main menu for library management system
void control::lib_SCREEN(void)
{
        char ch;
        struct time t;
        while (ch <= '6')
        {
                clrscr();
                BOX_CLASS s;
                s.DRAW_BOX(10, 5, 71, 21, 219);
                s.DRAW_BOX(9, 4, 72, 22, 218);
                textcolor(BLACK);
                textcolor(BLACK);
                textbackground(LIGHTGRAY);
                textcolor(RED);
                gotoxy(15,7);
                cprintf(" L I B R A R Y  M A N A G E M E N T  S Y S T E M ");
                gotoxy(28,9);
                textcolor(GREEN);
                cout << "1. Acquisition Section";
                gotoxy(28, 10);
                cout << "2. Membership Maintenance";
                gotoxy(28, 11);
                cout << "3. Book Issue";
                gotoxy(28, 12);
                cout << "4. Book Return";
                gotoxy(28, 13);
                cout << "5. Renewal of Membership";
                gotoxy(28, 14);
                cout << "6. Answer Management Queries";
                textbackground(0);
                gotoxy(10,24);
                cout<<"Created by :- ";
                textcolor(RED);
```

```
                cprintf("Manoj Kumar");
                gotoxy(10,25);
                cout<<"Reg. Number:- ";
                cprintf("######");

                gotoxy(28, 17);
                textcolor(RED);
                textbackground(0);
                cprintf("Enter Your Choice :");
                textbackground(BLACK);
                textcolor(7);

                date_time();
                ch = getche();
                textcolor(7);
                switch (ch)
                {
                        case '0':
                                exit(0); // Option to exit from main menu
                                break;
                        case '1':
                        {
                                Library lib;
                                textcolor(LIGHTGRAY);
                                textbackground(BLACK);
                                lib.Menu_console(); // Library books into LIB.dat
                                break;
                        }
                        case '2':
                        {
                                Member Memb;
                                textcolor(LIGHTGRAY);
                                textbackground(BLACK);
                                Memb.Memb_console(); // Members
management in MEMFILE.dat
                                break;
                        }
                        case '3':
                        {
                                Library lib;
                                textcolor(LIGHTGRAY);
                                textbackground(BLACK);
                                lib.Book_Issue();   // Function to issue books
                                break;
                        }
```

```
                        case '4':
                        {
                                Issue Iss;
                                textcolor(LIGHTGRAY);
                                textbackground(BLACK);
                                Iss.Book_Return(); // Function to return books
                                break;
                        }
                        case '5':
                        {
                                Member Memb;
                                textcolor(LIGHTGRAY);
                                textbackground(BLACK);
                                Memb.Member_Renewal();
                                break;
                        }
                        case '6':
                        {
                                Library lib;
                                textcolor(LIGHTGRAY);
                                textbackground(BLACK);
                                lib.Report_Menu(); // Function for displaying
report menu
                                break;
                        }
                }
        }

}

// Function for Adding, modifying and deleting from LIB.dat data file
void Library::Menu_console(void)
{
        char ch;

        while (1)
        {
                clrscr();
                clrscr();
                gotoxy(65,3);
                cout << "<0>=Exit";

                BOX_CLASS s;
                s.DRAW_BOX(10, 5, 71, 21, 219);
                s.DRAW_BOX(9, 4, 72, 22, 218);
```

```
                textcolor(BLACK);
                textbackground(WHITE);
                gotoxy(22, 8);
                cprintf("L I B R A R Y  C O N S O L E");
                textcolor(RED);
                textbackground(BLACK);
                gotoxy(25, 12);
                cout << "1: Keeping New Record";
                gotoxy(25, 13);
                cout << "2. Changing Existing Record";
                gotoxy(25, 14);
                cout << "3. Deleting Library Record";

                gotoxy(25, 16);
                cout << "0. Return";
                gotoxy(25, 18);
                cout << "Enter your choice : ";
                date_time();
                ch = getche();
                textcolor(7);
                if (ch == 27)
                        break;
                else
                        if (ch == '1')
                        {
                                new_library();
                        }
                        else
                                if (ch == '2')
                                {
                                        Change_library();
                                        break;
                                }
                                else
                                        if (ch == '3')
                                        {
                                                Delete_Accno();
                                                break;
                                        }
                                        else
                                                if (ch == '0')
                                                        break;
        }
}
```

```
// Function to append a record into LIB.dat data file
void Library::NEW_lib(int s_acq_no, int s_acc_no, char s_title[30], char
s_author1[30], char s_author2[30], char s_publisher[30], float s_price, int
sdd, int smm, int syy, int s_pages, char s_status)
{
        acq_no = s_acq_no;
        acc_no = s_acc_no;
        strcpy(title, s_title);
        strcpy(author1, s_author1);
        strcpy(author2, s_author2);
        strcpy(publisher, s_publisher);
        price = s_price;
        dd = sdd;
        mm = smm;
        yy = syy;
        pages = s_pages;
        status = s_status;

        fstream sfile;

        // Append the new record into LIB.dat data file
        sfile.open("LIB.dat", ios::out|ios::app);
        sfile.write((char *)this, sizeof(Library));
        sfile.close();
}

// Function to delete a selected record from LIB.dat data file
void Library::Lib_delete(int s_acq_no, int s_acc_no)
{
        fstream file;
        file.open("LIB.dat", ios::in);
        fstream temp;
        temp.open("TEMP.dat", ios::out);
        file.seekg(0,ios::beg);

        // Delete the record using copy method into temporaty file
        while (!file.eof())
        {
                file.read((char *)this, sizeof(Library));
                if (file.eof())
                        break;
                if ((acq_no != s_acq_no) && (acc_no != s_acc_no))
                        temp.write((char *)this, sizeof(Library));
        }
        file.close();
```

```
        temp.close();
        file.open("LIB.dat", ios::out);
        temp.open("TEMP.dat", ios::in);
        temp.seekg(0, ios::beg);

        // Overwrites the temporay contents into original LIB.dat data file
        while (!temp.eof())
        {
                temp.read((char *)this, sizeof(Library));
                if (temp.eof())
                        break;
                if ((acq_no != s_acq_no) && (acc_no != s_acc_no))
                        file.write((char *)this, sizeof(Library));
        }
        file.close();
        temp.close();
}

// Function for main menu to append new library books LIB.dat
void Library::new_library(void)
{
        char ch;
        int i, valid;
        clrscr();

        gotoxy(3, 3);
        for (i = 1; i<= 76; i++)
                cprintf(" ");

        textbackground(BLACK);
        textcolor(BLACK+BLINK);
        textbackground(WHITE);
        gotoxy(30, 3);
        cprintf("Library Acquisition Entry");
        textcolor(LIGHTGRAY);
        textbackground(BLACK);
        gotoxy(65, 3);
        cout << "<0>=Exit";

        BOX_CLASS s;
        s.DRAW_BOX(10, 5, 71, 21, 219);
        s.DRAW_BOX(9, 4, 72, 22, 218);
        textcolor(BLACK);
        textbackground(LIGHTGRAY);
```

```
        int s_acq_no;       // Acquisition number

        do
        {
                // Function to clear a specified input box from the screen
                Clear_area(6, 18, 11, 70);

                int s_acc_no;       // Accession number
                char s_title[30];
                char s_author1[30], s_author2[30];   // Authors
                char s_publisher[30];     // Name of the publisher
                float s_price;      // Price of the book
                int sdd, smm, syy;          // Date of purchase
                int s_pages; // Total pages in book
                char s_status;      // Present status of book

                struct date d;              // For extracting system date
                getdate(&d);
                sdd = d.da_day;
                smm = d.da_mon;
                syy = d.da_year;

                gotoxy(50, 7);
                cout << "Date: " << sdd << '-' << smm << '-' << syy;

                gotoxy(10, 23);
                cout << "Enter the Acquisition number";
                gotoxy(13, 7);
                cout << "Acquisition No. " ;
                gotoxy(29, 7);
                cin >> s_acq_no;
                Left_clear(10, 23);
                Right_clear(39, 23);

                gotoxy(11, 8);
                for (i = 1; i <= 60; i++)
                cout << "=";

                do
                {
                        // Function to clear a specified input box from the
screen
                        Clear_area(9, 18, 11, 70);

                        gotoxy(10, 23);
```

```
                    cout << "Enter the Accession number";
                    gotoxy(13, 9);
                    cout << "Accession No. ";
                    gotoxy(28, 9);
                    cin >> s_acc_no;
                    Left_clear(10, 23);
                    Right_clear(39, 23);

                    if (s_acq_no == 1)
                    {
                            // Creates a temporaty space in LIB.dat data tile
                            NEW_lib(s_acq_no, s_acc_no, "aaa", "bbb", "ccc",
"ddd", 0, 0, 0, 0, 0, 'x');
                            // Deletes the same record after creating its
position
                            Lib_delete(s_acq_no, s_acc_no);
                    }
                    if ((s_acq_no == 0) || (s_acc_no == 0))
                            return;


                    gotoxy(13, 10);
                    cout << "Title : ";
                    gotoxy(13, 12);
                    cout << "Author(s) : ";
                    gotoxy(13, 14);
                    cout << "Publisher : ";
                    gotoxy(13, 15);
                    cout << "Price : ";
                    gotoxy(13, 16);
                    cout << "Pages : ";
                    gotoxy(13, 17);
                    cout << "Status : ";

                    do
                    {
                            Left_clear(10, 23);
                            gotoxy(10, 23);
                            cout << "Enter Title of the Book";
                            valid = 1;
                            gotoxy(22, 10);
                            gets(s_title);
                            strupr(s_title);
                            if (s_title[0] == '0')
                                    return;
```

```
                if (strlen(s_title) == 0 || strlen(s_title) > 30)
                {
                        valid = 0;
                        gotoxy(10, 23);
                        cout << "\7Title should not greater than
30";
                        getch();
                }
        } while (!valid);
        Left_clear(5, 23);
        Right_clear(39, 23);

        do
        {
                gotoxy(10, 23);
                cout << "Enter Author of the Book ";
                valid = 1;
                gotoxy(26, 12);
                gets(s_author1);
                strupr(s_author1);
                if (s_author1[0] == '0')
                        return;
                if (strlen(s_author1) == 0 || strlen(s_author1) >
30)
                {
                        valid = 0;
                        gotoxy(10, 23);
                        cout << "\7Author should not greater than
30";
                        getch();
                }
        } while (!valid);
        Left_clear(10, 23);
        Right_clear(39, 23);

        gotoxy(10, 23);
        cout << "Enter second author if there";
        // Second author
        gotoxy(26, 13);
        gets(s_author2);
        strupr(s_author2);
        Left_clear(5, 23);
        Right_clear(39, 23);

        do
```

```
                {
                        Left_clear(10, 23);
                        Right_clear(39, 23);

                        gotoxy(10, 23);
                        cout << "Enter name of the Publisher";
                        valid = 1;
                        gotoxy(26, 14);
                        gets(s_publisher);
                        strupr(s_publisher);
                        if (s_publisher[0] == '0')
                                return;
                        if (strlen(s_publisher) == 0 || strlen(s_publisher)
> 30)
                        {
                                valid = 0;
                                gotoxy(10, 23);
                                cout << "\7Publisher should not greater
than 30";
                                getch();
                        }
                } while (!valid);
                Left_clear(10, 23);
                Right_clear(39, 23);

                gotoxy(10, 23);
                cout << "Enter selling price of the book";
                gotoxy(22, 15);
                cin >> s_price;
                Left_clear(10, 23);
                Right_clear(39, 23);

                gotoxy(10, 23);
                cout << "Enter Total pages in the book";
                gotoxy(22, 16);
                cin >> s_pages;
                Left_clear(10, 23);
                Right_clear(39, 23);

                // The status is automatically fill as "Y", because the
accession is entering
                // fresh books

                gotoxy(22, 17);
                s_status = 'Y';
```

```
                        cout << s_status;

                        do
                        {
                                valid = 1;
                                gotoxy(10, 23);
                                cout << "Do you want to save the record <Y/N>:
";
                                ch = getche();
                                if (ch == '0')
                                        return;
                                ch = toupper(ch);
                        }while (ch != 'N' && ch != 'Y');
                        if (ch == 'N')
                                return;
                        NEW_lib(s_acq_no, s_acc_no, s_title, s_author1,
s_author2, s_publisher, s_price, sdd, smm, syy, s_pages, s_status);
                        Left_clear(10, 23);
                        Right_clear(39, 23);

                        gotoxy(12, 23);
                        cout << "Any Accession <Y/N>: ";
                        ch = getche();
                        if (ch == '0')
                                return;
                        Left_clear(12, 23);
                        Right_clear(12, 23);

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

                gotoxy(12, 23);
                cout << "Any Acquisition <Y/N>: ";
                ch = getche();
                if (ch == '0')
                        return;
                Left_clear(12, 23);
                Right_clear(12, 23);

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

}

// Function to count the total number of records in LIB.dat data file for
```

```
// counting the modification record till found
int Library::recordno(int s_acq_no, int s_acc_no)
{
        fstream file;
        file.open("LIB.dat", ios::in);
        file.seekg(0, ios::beg);
        int count = 0;

        // Counts the total records in DETAILS.dat data file
        while (file.read((char *)this, sizeof(Library)))
        {
                count++;
                if ((s_acq_no == acq_no) && (s_acc_no == acc_no))
                        break;
        }
        file.close();
        return count;
}

// Modification function for library record
void Library::Replace(int s_acq_no, int s_acc_no, char s_title[30], char
s_author1[30], char s_author2[30], char s_publisher[30], float s_price, int
sdd, int smm, int syy, int s_pages, char s_status)
{
        int recno;
        recno = recordno(s_acq_no, s_acc_no);
        fstream file;
        file.open("LIB.dat", ios::out|ios::ate);


        acq_no = s_acq_no;
        acc_no = s_acc_no;
        strcpy(title, s_title);
        strcpy(author1, s_author1);
        strcpy(author2, s_author2);
        strcpy(publisher, s_publisher);
        price = s_price;
        dd = sdd;
        mm = smm;
        yy = syy;
        pages = s_pages;
        status = s_status;

        // Finding the location in file
        int location;
```

```
        location = (recno-1) * sizeof(Library);

        // Search the position in file
        file.seekp(location);

        // Re-write the contents in DETAILS.dat data file
        file.write((char *)this, sizeof(Library));
        file.close();
        return;
}

// Function for main menu to change the existing recrod in LIB.dat
void Library::Change_library(void)
{
        char ch;
        int i, valid;
        clrscr();

        gotoxy(3, 3);
        for (i = 1; i<= 76; i++)
                cprintf(" ");

        textbackground(BLACK);
        textcolor(BLACK+BLINK);
        textbackground(WHITE);
        gotoxy(30, 3);
        cprintf("Library Modification");
        textcolor(LIGHTGRAY);
        textbackground(BLACK);
        gotoxy(65, 3);
        cout << "<0>=Exit";

        BOX_CLASS s;
        s.DRAW_BOX(10, 5, 71, 21, 219);
        s.DRAW_BOX(9, 4, 72, 22, 218);
        textcolor(BLACK);
        textbackground(LIGHTGRAY);

        int s_acq_no = 0;  // Acquisition number

        do
        {
                // Function to clear a specified input box from the screen
                Clear_area(6, 18, 11, 70);
```

```
int s_acc_no = 0;   // Accession number
char s_title[30];
char s_author1[30], s_author2[30];   // Authors
char s_publisher[30];     // Name of the publisher
float s_price;         // Price of the book
int sdd, smm, syy;         // Date of purchase
int s_pages; // Total pages in book
char s_status;       // Present status of book

struct date d;                // For extracting system date
getdate(&d);
sdd = d.da_day;
smm = d.da_mon;
syy = d.da_year;

gotoxy(50, 7);
cout << "Date: " << sdd << '-' << smm << '-' << syy;

gotoxy(10, 23);
cout << "Enter the Acquisition number";
gotoxy(13, 7);
cout << "Acquisition No. " ;
gotoxy(29, 7);
cin >> s_acq_no;
Left_clear(10, 23);
Right_clear(39, 23);

gotoxy(11, 8);
for (i = 1; i <= 60; i++)
cout << "=";

do

{
        // Function to clear a specified input box from the
screen
        Clear_area(9, 18, 11, 70);

        gotoxy(10, 23);
        cout << "Enter the Accession number";
        gotoxy(13, 9);
        cout << "Accession No. ";
        gotoxy(28, 9);
        cin >> s_acc_no;
        Left_clear(10, 23);
```

```
Right_clear(39, 23);
if ((s_acq_no == 0) || (s_acc_no == 0))
        return;

fstream file;
file.open("LIB.dat", ios::in);
file.seekg(0,ios::beg);

// Search the record for further modification
while (file.read((char *)this, sizeof(Library)))
{
        if ((s_acq_no == acq_no) && (s_acc_no == acc_no))
        {
                gotoxy(13, 10);
                cout << "Existing Information";
                gotoxy(13, 11);
                cout << "Title : ";
                gotoxy(13, 12);
                cout << "Author(s) : ";
                gotoxy(13, 14);
                cout << "Publisher : ";
                gotoxy(13, 15);
                cout << "Price : ";
                gotoxy(13, 16);
                cout << "Pages : ";
                gotoxy(13, 17);
                cout << "Status : ";
                gotoxy(22, 11);
                puts(title);

                gotoxy(26, 12);
                puts(author1);

                gotoxy(26, 13);
                puts(author2);
                gotoxy(26, 14);
                puts(publisher);
                gotoxy(22, 15);
                cout << price;

                gotoxy(22, 16);
                cout << pages;
                gotoxy(22, 17);
                cout << status;
                break;
```

```
                }
        }
        file.close();
        gotoxy(45, 10);
        cout << "New Information";

        do
        {
                Left_clear(10, 23);
                gotoxy(10, 23);
                cout << "Enter New Title of the Book";
                valid = 1;
                gotoxy(45, 11);
                gets(s_title);

                strupr(s_title);
                if (s_title[0] == '0')
                        return;
                if (strlen(s_title) == 0 || strlen(s_title) > 30)
                {
                        valid = 0;
                        gotoxy(10, 23);
                        cout << "\7Title should not greater than
30";

                        getch();
                }
        } while (!valid);
        Left_clear(5, 23);
        Right_clear(39, 23);

        do
        {
                gotoxy(12, 23);
                cout << "Enter New Author of the Book ";
                valid = 1;
                gotoxy(45, 12);
                gets(s_author1);
                strupr(s_author1);
                if (s_author1[0] == '0')
                        return;
                if (strlen(s_author1) == 0 || strlen(s_author1) >
30)
                {
                        valid = 0;
                        gotoxy(10, 23);
```

```
                        cout << "\7Author should not greater than
30";
                        getch();
                }
        } while (!valid);
        Left_clear(10, 23);
        Right_clear(39, 23);

        gotoxy(10, 23);
        cout << "Enter second author if there";
        // Second author
        gotoxy(45, 13);
        gets(s_author2);
        strupr(s_author2);
        Left_clear(5, 23);
        Right_clear(39, 23);

        do
        {
                Left_clear(10, 23);
                Right_clear(39, 23);

                gotoxy(10, 23);
                cout << "Enter name of the Publisher";
                valid = 1;
                gotoxy(45, 14);
                gets(s_publisher);
                strupr(s_publisher);
                if (s_publisher[0] == '0')
                        return;
        if (strlen(s_publisher) == 0 || strlen(s_publisher) > 30)
                {
                        valid = 0;
                        gotoxy(10, 23);
        cout << "\7Publisher should not greater than 30";
                        getch();
                }
        } while (!valid);
        Left_clear(10, 23);
        Right_clear(39, 23);

        gotoxy(10, 23);
        cout << "Enter selling price of the book";
        gotoxy(45, 15);
        cin >> s_price;
```

```
                    Left_clear(10, 23);
                    Right_clear(39, 23);

                    gotoxy(10, 23);
                    cout << "Enter Total pages in the book";
                    gotoxy(45, 16);
                    cin >> s_pages;
                    Left_clear(10, 23);
                    Right_clear(39, 23);

                    // The status is automatically fill as "Y", because the
accession is entering
                    // fresh books

                    gotoxy(45, 17);
                    s_status = 'Y';
                    cout << s_status;

                    do
                    {
                            valid = 1;
                            gotoxy(10, 23);
                            cout << "Do you want to Overwrite the record
<Y/N>: ";
                            ch = getche();
                            if (ch == '0')
                                    return;
                            ch = toupper(ch);
                    }while (ch != 'N' && ch != 'Y');
                    if (ch == 'N')
                            return;
                    Replace(s_acq_no, s_acc_no, s_title, s_author1,
s_author2, s_publisher, s_price, sdd, smm, syy, s_pages, s_status);
                    Left_clear(10, 23);
                    Right_clear(39, 23);

                    gotoxy(12, 23);
                    cout << "Any Accession <Y/N>: ";
                    ch = getche();
                    if (ch == '0')
                            return;
                    Left_clear(12, 23);
                    Right_clear(12, 23);

                    ch = toupper(ch);
```

```
                } while (ch == 'Y');

                gotoxy(12, 23);
                cout << "Any Acquisition <Y/N>: ";
                ch = getche();
                if (ch == '0')
                        return;
                Left_clear(12, 23);
                Right_clear(12, 23);

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

}

// Function to find the acquisition number and accession number is
exists or not
int Issue::Issue_Check(int s_acq_no, int s_acc_no)
{
        int tflag = 0;
        fstream file;
        file.open("MASTER.dat", ios::in);
        file.seekg(0,ios::beg);

        // Search the record for further modification
        while (file.read((char *)this, sizeof(Issue)))
        {
                if ((s_acq_no == acq_no) && (s_acc_no == acc_no))
                {
                        tflag = 1;
                        break;
                }
        }
        file.close();
        return (tflag);
}

// Function to check member for issue
int Issue::Issue_memb_check(int s_memb_no)
{
        int mflag = 0;
        fstream file;
        file.open("MASTER.dat", ios::in);
        file.seekg(0,ios::beg);
```

```
	// Search the record for further modification
	while (file.read((char *)this, sizeof(Issue)))
	{
		if (s_memb_no == memb_no)
		{
			mflag = 1;
			break;
		}
	}
	file.close();
	return (mflag);
}

// Function to find the MEMFILE.dat file that the member has EXISTS or
not
int Member::Memb_Check(int s_memb_no)
{
	int mflag = 0;
	fstream file;
	file.open("MEMFILE.dat", ios::in);
	file.seekg(0,ios::beg);

	// Search the record for further modification
	while (file.read((char *)this, sizeof(Member)))
	{
		if (s_memb_no == memb_no)
		{
			mflag = 1;
			break;
		}
	}
	file.close();
	return (mflag);
}

// Function to delete record from LIB.dat data file by the matching
acquition no. and accession no.
void Library::Record_delete(int s_acq_no, int s_acc_no)
{
	fstream file;
	file.open("LIB.dat", ios::in);
	fstream temp;
	temp.open("TEMP.dat", ios::out);
	file.seekg(0,ios::beg);
```

```
        // Delete the record using copy method into temporaty file
        while (!file.eof())
        {
                file.read((char *)this, sizeof(Library));
                if (file.eof())
                        break;
                if ((s_acq_no != acc_no) && (s_acc_no != acc_no))
                        temp.write((char *)this, sizeof(Library));
        }
        file.close();
        temp.close();
        file.open("LIB.dat", ios::out);
        temp.open("TEMP.dat", ios::in);
        temp.seekg(0, ios::beg);

        // Overwrites the temporay contents into original DETAILS.dat
data file
        while (!temp.eof())
        {
                temp.read((char *)this, sizeof(Library));
                if (temp.eof())
                        break;
                if ((s_acq_no != acc_no) && (s_acc_no != acc_no))
                        file.write((char *)this, sizeof(Library));
        }
        file.close();
        temp.close();
}


// Function for main menu to Delete the existing recrod in LIB.dat
void Library::Delete_Accno(void)
{
        char ch;
        int i;
        clrscr();

        int flag = 0;
        int tflag = 0;
        gotoxy(3, 3);
        for (i = 1; i<= 76; i++)
                cprintf(" ");

        textbackground(BLACK);
        textcolor(BLACK+BLINK);
```

```
textbackground(WHITE);
gotoxy(30, 3);
cprintf("Library Deletion");
textcolor(LIGHTGRAY);
textbackground(BLACK);
gotoxy(65, 3);
cout << "<0>=Exit";

BOX_CLASS s;
s.DRAW_BOX(10, 5, 71, 21, 219);
s.DRAW_BOX(9, 4, 72, 22, 218);
textcolor(BLACK);
textbackground(LIGHTGRAY);

int s_acq_no = 0;  // Acquisition number

do
{
        // Function to clear a specified input box from the screen
        Clear_area(6, 18, 11, 70);

        int s_acc_no = 0;  // Accession number
        char s_title[30];
        char s_author1[30], s_author2[30];   // Authors
        char s_publisher[30];     // Name of the publisher
        float s_price;        // Price of the book
        int sdd, smm, syy;        // Date of purchase
        int s_pages; // Total pages in book
        char s_status;      // Present status of book

        struct date d;              // For extracting system date
        getdate(&d);
        sdd = d.da_day;
        smm = d.da_mon;
        syy = d.da_year;

        gotoxy(50, 7);
        cout << "Date: " << sdd << '-' << smm << '-' << syy;

        gotoxy(10, 23);
        cout << "Enter the Acquisition number";
        gotoxy(13, 7);
        cout << "Acquisition No. " ;
        gotoxy(29, 7);
        cin >> s_acq_no;
```

```
                Left_clear(10, 23);
                Right_clear(39, 23);

                gotoxy(11, 8);
                for (i = 1; i <= 60; i++)
                cout << "=";

                do
                {
                        // Function to clear a specified input box from the
screen
                        Clear_area(9, 18, 11, 70);

                        gotoxy(10, 23);
                        cout << "Enter the Accession number";
                        gotoxy(13, 9);
                        cout << "Accession No. ";
                        gotoxy(28, 9);
                        cin >> s_acc_no;
                        Left_clear(10, 23);
                        Right_clear(39, 23);
                        if ((s_acq_no == 0) || (s_acc_no == 0))
                                return;

                        fstream file;
                        file.open("LIB.dat", ios::in);
                        file.seekg(0,ios::beg);

                        // Search the record for further modification
                        while (file.read((char *)this, sizeof(Library)))
                        {
                                if ((s_acq_no == acq_no) && (s_acc_no == acc_no))
                                {
                                        flag = 1;
                                        gotoxy(15, 11);
                                        cout << "Title : ";
                                        gotoxy(15, 12);
                                        cout << "Author(s) : ";
                                        gotoxy(15, 14);
                                        cout << "Publisher : ";
                                        gotoxy(15, 15);
                                        cout << "Price : ";
                                        gotoxy(15, 16);
                                        cout << "Pages : ";
                                        gotoxy(15, 17);
```

```
                        cout << "Status : ";
                        gotoxy(26, 11);
                        puts(title);

                        gotoxy(26, 12);
                        puts(author1);

                        gotoxy(26, 13);
                        puts(author2);
                        gotoxy(26, 14);
                        puts(publisher);
                        gotoxy(26, 15);
                        cout << price;

                        gotoxy(26, 16);
                        cout << pages;
                        gotoxy(26, 17);
                        cout << status;
                        break;
                }
        }
        file.close();

        Issue Iss;
        tflag = Iss.Issue_Check(s_acq_no, s_acc_no);
        if (tflag == 0)
        {
                do
                {
                gotoxy(10, 23);
cout << "Are you you sure to Delete this record <Y/N>: ";
                ch = getche();
                if (ch == '0')
                        return;
                        ch = toupper(ch);
                }while (ch != 'N' && ch != 'Y');
        }
        else
        {
                gotoxy(12, 23);
                cout << "The is already Issued ...";
        }
        Left_clear(10, 23);
        Right_clear(39, 23);
```

```
                        if ((ch == 'Y') && (flag == 1))
                        {
                                Record_delete(s_acq_no, s_acc_no);
                        }
                        if (flag == 1)
                        {
                                gotoxy(57, 23);
                                cout << "Record Deleted !";
                                getch();
                        }
                        Left_clear(10, 23);
                        Right_clear(39, 23);

                        gotoxy(12, 23);
                        cout << "Any Accession <Y/N>: ";
                        ch = getche();
                        if (ch == '0')
                                return;
                        Left_clear(12, 23);
                        Right_clear(12, 23);

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

                gotoxy(12, 23);
                cout << "Any Acquisition <Y/N>: ";
                ch = getche();
                if (ch == '0')
                        return;
                Left_clear(12, 23);
                Right_clear(12, 23);

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

}

// Function to append a record into MEMFILE.dat data file
void Member::NEW_memb(int s_memb_no, char s_name[30], char
s_address[30], int s_mdd, int s_mmm, int s_myy, int s_edd, int s_emm,
int s_eyy, float s_memb_fee, float s_caut_fee)
{
        memb_no = s_memb_no;
        strcpy(name, s_name);
        strcpy(address, s_address);
```

```
        mdd = s_mdd;
        mmm = s_mmm;
        myy = s_myy;
        edd = s_edd;
        emm = s_emm;
        eyy = s_eyy;
        memb_fee = s_memb_fee;
        caut_fee = s_caut_fee;

        fstream sfile;

        // Append the new record into MEMFILE.dat data file
        sfile.open("MEMFILE.dat", ios::out|ios::app);
        sfile.write((char *)this, sizeof(Member));
        sfile.close();
}

// Function to delete a selected record from MEMFILE.dat data file
void Member::Memb_delete(int s_memb_no)
{
        fstream file;
        file.open("MEMFILE.dat", ios::in);
        fstream temp;
        temp.open("TEMP.dat", ios::out);
        file.seekg(0,ios::beg);

        // Delete the record using copy method into temporaty file
        while (!file.eof())
        {
                file.read((char *)this, sizeof(Member));
                if (file.eof())
                        break;
                if (memb_no != s_memb_no)
                        temp.write((char *)this, sizeof(Member));
        }
        file.close();
        temp.close();
        file.open("MEMFILE.dat", ios::out);
        temp.open("TEMP.dat", ios::in);
        temp.seekg(0, ios::beg);

        // Overwrites the temporay contents into original MEMFILE.dat
data file
        while (!temp.eof())
        {
```

```
            temp.read((char *)this, sizeof(Member));
            if (temp.eof())
                  break;
            if (memb_no != s_memb_no)
                  file.write((char *)this, sizeof(Member));
      }
      file.close();
      temp.close();
}


// Function for main menu to append new members in MEMFILE.dat
void Member::new_member(void)
{
      char ch;
      int i, valid;

      Library lib;
      clrscr();
      gotoxy(65, 2);
      cout << "<0>=Exit";
      BOX_CLASS s;
      s.DRAW_BOX(10, 5, 71, 21, 219);
      s.DRAW_BOX(9, 4, 72, 22, 218);
      textcolor(BLACK);
      textbackground(LIGHTGRAY);

      gotoxy(3,3);
      for (i = 1; i<= 76; i++)
            cprintf(" ");

      textbackground(BLACK);
      textcolor(BLACK+BLINK);
      textbackground(WHITE);
      gotoxy(35, 4);
      cprintf("Membership Form");
      textcolor(LIGHTGRAY);
      textbackground(BLACK);

      do
      {
            lib.Clear_area(6, 18, 12, 70);
            int s_memb_no;    // Membership number
            char s_name[30];  // Name of the member
            char s_address[30]; // Address of the employee
```

```
        int s_mdd, s_mmm, s_myy;  // Membership date
        int s_edd, s_emm, s_eyy; // Expire date
        float s_memb_fee; // Membership fee
        float s_caut_fee;    // Caution money

        struct date d;              // For extracting system date
        getdate(&d);
        s_mdd = d.da_day;
        s_mmm = d.da_mon;
        s_myy = d.da_year;

        gotoxy(12, 7);
        cout << "Date: " << s_mdd << '-' << s_mmm << '-' << s_myy;
        gotoxy(12, 23);
        cout << "Enter new membership no.";
        gotoxy(12, 9);
        cout << "Membership No. " ;
        gotoxy(27, 9);
        cin >> s_memb_no;
        lib.Left_clear(12, 23);
        lib.Right_clear(12, 23);

        if (s_memb_no == 1)
        {
        // Creates a temporaty space in MEMFILE.dat data tile
NEW_memb(s_memb_no, "aaa", "bbb", 0, 0, 0, 0, 0, 0, 0.0, 0.0);
                // Deletes the same record after creating its position
                Memb_delete(s_memb_no);
        }
        if (s_memb_no == 0)
                return;

        gotoxy(12, 11);
        cout << "Name : ";
        gotoxy(12, 14);
        cout << "Address : ";
        gotoxy(12, 15);
        cout << "Memb. Dat : ";
        gotoxy(12, 16);
        cout << "Exp. Date : ";
        gotoxy(12, 17);
        cout << "Memb. Fees : ";
        gotoxy(12, 18);
        cout << "Caution Money : ";
```

```
do
{
        lib.Left_clear(12, 23);
        gotoxy(12, 23);
        cout << "Enter Name of the Member";
        valid = 1;
        gotoxy(19, 11);
        gets(s_name);
        strupr(s_name);
        if (s_name[0] == '0')
                return;
        if (strlen(s_name) == 0 || strlen(s_name) > 30)
        {
                valid = 0;
                gotoxy(12, 23);
                cprintf("\7Name should not greater than 30");
                getch();
        }
} while (!valid);
lib.Left_clear(12, 23);
lib.Right_clear(12, 23);

do
{
        lib.Left_clear(12, 23);
        gotoxy(15, 23);
        cout << "Enter Address of the Member ";
        valid = 1;
        gotoxy(23, 14);
        gets(s_address);
        strupr(s_address);
        if (s_address[0] == '0')
                return;
        if (strlen(s_address) == 0 || strlen(s_address) > 30)
        {
                valid = 0;
                gotoxy(12, 23);
                cprintf("\7Address should not greater than 30");
                getch();
        }
} while (!valid);
lib.Left_clear(12, 23);
lib.Right_clear(12, 23);

gotoxy(12, 23);
```

```
                cout << "Enter the membership date";
                gotoxy(25, 15);
                cin >> s_mdd;
                gotoxy(27, 15);
                cout << '-';
                gotoxy(28, 15);
                cin >> s_mmm;
                gotoxy(30, 15);
                cout << '-';
                gotoxy(31, 15);
                cin >> s_myy;
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

                // Calculation of expire date according to a validation of 1
year for
                // membership from membership date and will be calculated
in s_edd, s_emm, s_eyy

                s_edd = s_mdd;
                s_emm = s_mmm;
                s_eyy = s_myy + 1; // Added 1 with the membership year to
know the expire date

                gotoxy(25, 16);
                cout << s_edd;
                gotoxy(27, 16);
                cout << '-';
                gotoxy(28, 16);
                cout << s_emm;
                gotoxy(30, 16);
                cout << '-';
                gotoxy(31, 16);
                cout << s_eyy;

                gotoxy(12, 23);
                cout << "Enter membership fees for a member";
                gotoxy(25, 17);
                cin >> s_memb_fee;
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

                gotoxy(12, 23);
                cout << "Enter caution money ";
                gotoxy(28, 18);
```

```
                cin >> s_caut_fee;
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

                do
                {
                        valid = 1;
                        gotoxy(12, 23);
                        cout << "Do you want to save the record <Y/N>: ";
                        ch = getche();
                        if (ch == '0')
                                return;
                        ch = toupper(ch);
                } while (ch != 'Y');
                if (ch == 'N')
                        return;

                NEW_memb(s_memb_no, s_name, s_address, s_mdd,
s_mmm, s_myy, s_edd, s_emm, s_eyy, s_memb_fee, s_caut_fee);
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);
                gotoxy(12, 23);
                cout << "Do You Want New Record <Y/N>: ";
                ch = getche();
                if (ch == '0')
                        return;
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

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

// Function to count the total number of records in MEMBER.dat data
file for
// counting the modification record till found
int Member::recordno(int s_memb_no)
{
        fstream file;
        file.open("MEMFILE.dat", ios::in);
        file.seekg(0, ios::beg);
        int count = 0;

        // Counts the total records in MEMBER.dat data file
        while (file.read((char *)this, sizeof(Member)))
```

```
        {
                count++;
                if (memb_no == s_memb_no)
                        break;
        }
        file.close();
        return (count);
}

// Modification function for member information
void Member::Replace(int s_memb_no, char s_name[30], char
s_address[30], int s_mdd, int s_mmm, int s_myy, int s_edd, int s_emm,
int s_eyy, float s_memb_fee, float s_caut_fee)
{
        int recno = 0;
        recno = recordno(s_memb_no);

        memb_no = s_memb_no;
        strcpy(name, s_name);
        strcpy(address, s_address);
        mdd = s_mdd;
        mmm = s_mmm;
        myy = s_myy;
        edd = s_edd;
        emm = s_emm;
        eyy = s_eyy;
        memb_fee = s_memb_fee;
        caut_fee = s_caut_fee;

        fstream file;
        file.open("MEMFILE.dat", ios::out|ios::ate);

        // Finding the location in file
        int location;
        location = (recno-1) * sizeof(Member);

        // Search the position in file
        file.seekp(location);

        // Re-write the contents in DETAILS.dat data file
        file.write((char *)this, sizeof(Member));
        file.close();
        return;
}
```

```
// Function to display the existing member in screen to change the
further information
void Member::Display_member(int s_memb_no)
{
		fstream file;
		file.open("MEMFILE.dat", ios::in);
		file.seekg(0, ios::beg);
		while (file.read((char *)this, sizeof(Member)))
		{
			if (memb_no == s_memb_no)
			{
				gotoxy(12, 11);
				cout << "Name : ";
				gotoxy(19, 11);
				puts(name);

				gotoxy(12, 14);
				cout << "Address : ";
				gotoxy(23, 14);
				puts(address);

				gotoxy(12, 15);
				cout << "Memb. Dat : ";
				gotoxy(25, 15);
				cout << mdd;
				gotoxy(27, 15);
				cout << '-';
				gotoxy(28, 15);
				cout << mmm;
				gotoxy(30, 15);
				cout << '-';
				gotoxy(31, 15);
				cout << myy;

				gotoxy(12, 16);
				cout << "Exp. Date : ";
				gotoxy(25, 16);
				cout << edd;
				gotoxy(27, 16);
				cout << '-';
				gotoxy(28, 16);
				cout << emm;
				gotoxy(30, 16);
				cout << '-';
				gotoxy(31, 16);
```

```
                        cout << eyy;

                        gotoxy(12, 17);
                        cout << "Memb. Fees : ";
                        gotoxy(25, 17);
                        cout << memb_fee << setw(15)    // setwidth
                                << setprecision(2)    // set position
of decimal point
                                << setiosflags(ios::left)   // set left
justified output
                                << setiosflags(ios::showpoint)  //
always show decimal point
                                << setiosflags(ios::fixed);    // set
fixed notation for display

                        gotoxy(12, 18);
                        cout << "Caution Money : ";
                        gotoxy(28, 18);
                        cout << caut_fee << setw(15)
                                << setprecision(2)
                                << setiosflags(ios::left)
                                << setiosflags(ios::showpoint)
                                << setiosflags(ios::fixed)
;
                        break;
                }
        }
        file.close();
}

// Function for main menu to Change member informations in
MEMFILE.dat
void Member::Change_member(void)
{
        char ch;
        int i, valid;

        Library lib;
        clrscr();
        gotoxy(65, 2);
        cout << "<0>=Exit";
        BOX_CLASS s;
        s.DRAW_BOX(10, 5, 71, 21, 219);
        s.DRAW_BOX(9, 4, 72, 22, 218);
        textcolor(BLACK);
```

```
textbackground(LIGHTGRAY);

gotoxy(3,3);
for (i = 1; i<= 76; i++)
        cprintf(" ");

textbackground(BLACK);
textcolor(BLACK+BLINK);
textbackground(WHITE);
gotoxy(35, 4);
cprintf("Membership Form");
textcolor(LIGHTGRAY);
textbackground(BLACK);

do
{
        lib.Clear_area(6, 18, 12, 70);
        int s_memb_no;    // Membership number
        char s_name[30];  // Name of the member
        char s_address[30]; // Address of the employee
        int s_mdd, s_mmm, s_myy;  // Membership date
        int s_edd, s_emm, s_eyy; // Expire date
        float s_memb_fee; // Membership fee
        float s_caut_fee;    // Caution money

        struct date d;              // For extracting system date
        getdate(&d);
        s_mdd = d.da_day;
        s_mmm = d.da_mon;
        s_myy = d.da_year;

        gotoxy(12, 7);
        cout << "Date: " << s_mdd << '-' << s_mmm << '-' << s_myy;
        gotoxy(12, 23);
        cout << "Enter new membership no.";
        gotoxy(12, 9);
        cout << "Membership No. " ;
        gotoxy(27, 9);
        cin >> s_memb_no;
        lib.Left_clear(12, 23);
        lib.Right_clear(12, 23);

        if (s_memb_no == 0)
                return;
```

```
Display_member(s_memb_no);
do
{
        lib.Left_clear(12, 23);
        gotoxy(12, 23);
        cout << "Enter New Name of the Member";
        valid = 1;
        gotoxy(45, 11);
        gets(s_name);
        strupr(s_name);
        if (s_name[0] == '0')
                return;
        if (strlen(s_name) == 0 || strlen(s_name) > 30)
        {
                valid = 0;
                gotoxy(12, 23);
                cprintf("\7Name should not greater than 30");
                getch();
        }
} while (!valid);
lib.Left_clear(12, 23);
lib.Right_clear(12, 23);
do
{
        lib.Left_clear(12, 23);
        gotoxy(15, 23);
        cout << "Enter New Address of the Member ";
        valid = 1;
        gotoxy(45, 14);
        gets(s_address);
        strupr(s_address);
        if (s_address[0] == '0')
                return;
        if (strlen(s_address) == 0 || strlen(s_address) > 30)
        {
                valid = 0;
                gotoxy(12, 23);
                cprintf("\7Address should not greater than 30");
                getch();
        }
} while (!valid);
lib.Left_clear(12, 23);
lib.Right_clear(12, 23);

gotoxy(12, 23);
```

```
                cout << "Enter New the membership date";
                gotoxy(45, 15);
                cin >> s_mdd;
                gotoxy(47, 15);
                cout << '-';
                gotoxy(48, 15);
                cin >> s_mmm;
                gotoxy(50, 15);
                cout << '-';
                gotoxy(51, 15);
                cin >> s_myy;
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

                // Calculation of expire date according to a validation of 1
year for
                // membership from membership date and will be calculated
in s_edd, s_emm, s_eyy
                s_edd = s_mdd;
                s_emm = s_mmm;
                s_eyy = s_myy + 1; // Added 1 with the membership year to
know the expire date

                gotoxy(45, 16);
                cout << s_edd;
                gotoxy(47, 16);
                cout << '-';
                gotoxy(48, 16);
                cout << s_emm;
                gotoxy(50, 16);
                cout << '-';
                gotoxy(51, 16);
                cout << s_eyy;

                gotoxy(12, 23);
                cout << "Enter membership fees for a member";
                gotoxy(45, 17);
                cin >> s_memb_fee;

                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

                gotoxy(12, 23);
                cout << "Enter caution money ";
                gotoxy(45, 18);
```

```
                cin >> s_caut_fee;
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

                do
                {
                        valid = 1;
                        gotoxy(12, 23);
                        cout << "Do you want to Modify the record <Y/N>: ";
                        ch = getche();
                        if (ch == '0')
                                return;
                        ch = toupper(ch);
                } while (ch != 'Y');
                if (ch == 'N')
                        return;

                Replace(s_memb_no, s_name, s_address, s_mdd, s_mmm,
s_myy, s_edd, s_emm, s_eyy, s_memb_fee, s_caut_fee);
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);
                gotoxy(12, 23);
                cout << "Do You Want New Record <Y/N>: ";
                ch = getche();
                if (ch == '0')
                        return;
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

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


// Function to delete a member from MEMBER.dat data file.
void Member::Delete_member()
{
        char ch;
        int i, flag = 0;

        Library lib;
        clrscr();
        gotoxy(65, 2);
        cout << "<0>=Exit";
        BOX_CLASS s;
```

```
s.DRAW_BOX(10, 5, 71, 21, 219);
s.DRAW_BOX(9, 4, 72, 22, 218);
textcolor(BLACK);
textbackground(LIGHTGRAY);

gotoxy(3,3);
for (i = 1; i<= 76; i++)
        cprintf(" ");

textbackground(BLACK);
textcolor(BLACK+BLINK);
textbackground(WHITE);
gotoxy(35, 4);
cprintf("Membership Deleted Form");
textcolor(LIGHTGRAY);
textbackground(BLACK);

do
{
        lib.Clear_area(6, 18, 12, 70);
        int s_memb_no;    // Membership number
        char s_name[30];  // Name of the member
        char s_address[30]; // Address of the employee
        int s_mdd, s_mmm, s_myy;  // Membership date
        int s_edd, s_emm, s_eyy; // Expire date
        float s_memb_fee; // Membership fee
        float s_caut_fee;    // Caution money

        struct date d;              // For extracting system date
        getdate(&d);
        s_mdd = d.da_day;
        s_mmm = d.da_mon;
        s_myy = d.da_year;

        gotoxy(12, 7);
        cout << "Date: " << s_mdd << '-' << s_mmm << '-' << s_myy;
        gotoxy(12, 23);
        cout << "Enter membership no. to delete";
        gotoxy(12, 9);
        cout << "Membership No. " ;
        gotoxy(27, 9);
        cin >> s_memb_no;
        lib.Left_clear(12, 23);
        lib.Right_clear(12, 23);
```

```
            if (s_memb_no == 0)
                  return;

            Display_member(s_memb_no);
            Issue Iss;
            flag = Iss.Issue_memb_check(s_memb_no);

            if (flag == 1)
            {
                  gotoxy(12, 23);
                  cout << "Member has already a book. Can't Delete ...!";
                  getch();
                  lib.Left_clear(12, 23);
                  lib.Right_clear(12, 23);
            }
            else
            {
                  gotoxy(12, 23);
                  cout << "Are You Sure to delete <Y/N>: ";
                  ch = getche();
                  ch = toupper(ch);
                  if (ch == 'Y')
                  {
                        Memb_delete(s_memb_no);
                        gotoxy(12, 23);
                        cout << "Member Deleted";
                        getch();
                        lib.Left_clear(12, 23);
                        lib.Right_clear(12, 23);
                  }
            }
            gotoxy(12, 23);
            cout << "Do you want to Delete any Record <Y/N>: ";
            ch = getche();
            if (ch == '0')
                  return;
            lib.Left_clear(12, 23);
            lib.Right_clear(12, 23);

            ch = toupper(ch);
      } while (ch == 'Y');
}
```

```
// Function to display the existing member and renewal the membership
date
void Member::Renewal_member(int s_memb_no)
{
		char s_name[30];  // Name of the member
		char s_address[30]; // Address of the employee
		int s_mdd, s_mmm, s_myy;  // Membership date
		int s_edd, s_emm, s_eyy; // Expire date
		float s_memb_fee; // Membership fee
		float s_caut_fee;    // Caution money

		fstream file;
		file.open("MEMFILE.dat", ios::in);
		file.seekg(0, ios::beg);
		while (file.read((char *)this, sizeof(Member)))
		{
			if (memb_no == s_memb_no)
			{
				strcpy(s_name, name);
				strcpy(s_address, address);
				s_mdd = mdd;
				s_mmm = mmm;
				s_myy = myy;

				s_edd = edd;
				s_emm = emm;
				s_eyy = eyy;

				s_memb_fee = memb_fee;
				s_caut_fee = caut_fee;

				gotoxy(12, 11);
				cout << "Name : ";
				gotoxy(19, 11);
				puts(s_name);

				gotoxy(12, 14);
				cout << "Address : ";
				gotoxy(23, 14);
				puts(s_address);

				gotoxy(12, 15);
				cout << "Memb. Dat : ";
				gotoxy(25, 15);
				cout << s_mdd;
```

```
gotoxy(27, 15);
cout << '-';
gotoxy(28, 15);

cout << s_mmm;
gotoxy(30, 15);
cout << '-';
gotoxy(31, 15);
cout << s_myy;

gotoxy(12, 16);
cout << "Exp. Date : ";
gotoxy(25, 16);
cout << s_edd;
gotoxy(27, 16);
cout << '-';
gotoxy(28, 16);
cout << s_emm;
gotoxy(30, 16);
cout << '-';
gotoxy(31, 16);
cout << s_eyy;

gotoxy(12, 17);
cout << "Memb. Fees : ";
gotoxy(25, 17);
cout << s_memb_fee << setw(15)
                   << setprecision(2)
                   << setiosflags(ios::left)
                   << setiosflags(ios::showpoint)
                   << setiosflags(ios::fixed);

gotoxy(12, 18);
cout << "Caution Money : ";
gotoxy(28, 18);
cout << s_caut_fee << setw(15)
                   << setprecision(2)
                   << setiosflags(ios::left)
                   << setiosflags(ios::showpoint)
                   << setiosflags(ios::fixed);

gotoxy(40, 15);
cout << "Renewal. Dat : ";
gotoxy(55, 15);
cin >> s_mdd;
```

```
                        gotoxy(58, 15);
                        cout << '-';
                        gotoxy(59, 15);
                        cin >> s_mmm;
                        gotoxy(61, 15);
                        cout << '-';
                        gotoxy(62, 15);
                        cin >> s_myy;

                // Membership contain for 1 year valid after renewal date
                        s_edd = s_mdd;
                        s_emm = s_mmm;
                        s_eyy = s_myy + 1;

                        gotoxy(40, 16);
                        cout << "Exp. Date : ";
                        gotoxy(55, 16);
                        cout << s_edd;
                        gotoxy(58, 16);
                        cout << '-';
                        gotoxy(59, 16);
                        cout << s_emm;
                        gotoxy(61, 16);
                        cout << '-';
                        gotoxy(62, 16);
                        cout << s_eyy;

                        break;
                    }
                }
                file.close();
                if (s_eyy > 0)
                    Replace(s_memb_no, s_name, s_address, s_mdd,
s_mmm, s_myy, s_edd, s_emm, s_eyy, s_memb_fee, s_caut_fee);
}

// Function for main menu to renewal of membership
void Member::Member_Renewal(void)
{
        char ch;
        int i, valid;

        Library lib;
        clrscr();
        gotoxy(65, 2);
```

```
cout << "<0>=Exit";
BOX_CLASS s;
s.DRAW_BOX(10, 5, 71, 21, 219);
s.DRAW_BOX(9, 4, 72, 22, 218);
textcolor(BLACK);
textbackground(LIGHTGRAY);

gotoxy(3,3);
for (i = 1; i<= 76; i++)
        cprintf(" ");

textbackground(BLACK);
textcolor(BLACK+BLINK);
textbackground(WHITE);
gotoxy(35, 4);
cprintf("Membership Renewal Form");
textcolor(LIGHTGRAY);
textbackground(BLACK);

do
{
        lib.Clear_area(6, 18, 12, 70);
        int s_memb_no;    // Membership number
        char s_name[30];  // Name of the member
        char s_address[30]; // Address of the employee
        int s_mdd, s_mmm, s_myy;  // Membership date
        int s_edd, s_emm, s_eyy; // Expire date
        float s_memb_fee; // Membership fee
        float s_caut_fee;    // Caution money

        struct date d;              // For extracting system date
        getdate(&d);
        s_mdd = d.da_day;
        s_mmm = d.da_mon;
        s_myy = d.da_year;

        gotoxy(12, 7);
        cout << "Date: " << s_mdd << '-' << s_mmm << '-' << s_myy;
        gotoxy(12, 23);
        cout << "Enter new membership no.";
        gotoxy(12, 9);
        cout << "Membership No. " ;
        gotoxy(27, 9);
        cin >> s_memb_no;
        lib.Left_clear(12, 23);
```

```
                lib.Right_clear(12, 23);

                if (s_memb_no == 0)
                        return;

                Renewal_member(s_memb_no);

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

// Function for Adding, modifying and deleting from MEMFILE.dat data
file
void Member::Memb_console(void)
{
        char ch;
        while (1)
        {
                clrscr();
                gotoxy(65, 3);
                cout << "<0>=Exit";

                BOX_CLASS s;
                s.DRAW_BOX(10, 5, 71, 21, 219);
                s.DRAW_BOX(9, 4, 72, 22, 218);
                textcolor(BLACK);
                textbackground(WHITE);
                gotoxy(22, 8);
                cprintf("M E M B E R S H I P  C O N S O L E");
                textcolor(GREEN);
                textbackground(BLACK);
                gotoxy(25, 12);
                cout << "1: Recording New Member";

                gotoxy(25, 13);
                cout << "2. Changing Member Data";
                gotoxy(25, 14);
                cout << "3. Deleting Member Record";

                gotoxy(25, 16);
                cout << "0. Return";
                gotoxy(25, 17);
                cout << "Enter your choice : ";
                date_time();
                ch = getche();
```

```
                if (ch == 27)
                        break;
                else
                        if (ch == '1')
                        {
                                new_member();
                        }
                        else
                                if (ch == '2')
                                {
                                        Change_member();
                                        break;
                                }
                                else
                                        if (ch == '3')
                                        {
                                                Delete_member();
                                                break;
                                        }
                                        else
                                                if (ch == '0')
                                                        break;
        }
}

// Function to add issued books into MASTER.dat data file
void Issue::Issue_Add(int s_acq_no, int s_acc_no, int s_memb_no, int
s_idd, int s_imm, int s_iyy, int s_rdd, int s_rmm, int s_ryy)
{
        acq_no = s_acq_no;
        acc_no = s_acc_no;
        memb_no = s_memb_no;
        idd = s_idd;
        imm = s_imm;
        iyy = s_iyy;

        rdd = s_rdd;
        rmm = s_rmm;
        ryy = s_ryy;
        fstream sfile;

        // Append the new record into MASTER.dat data file
        sfile.open("MASTER.dat", ios::out|ios::app);
        sfile.write((char *)this, sizeof(Issue));
        sfile.close();
```

```
}

// Function to delete a selected record from MASTER.dat data file
void Issue::Issue_Delete(int s_acq_no, int s_acc_no, int s_memb_no, int
s_idd, int s_imm, int s_iyy, int s_rdd, int s_rmm, int s_ryy)
{
        fstream file;
        file.open("MASTER.dat", ios::in);
        fstream temp;
        temp.open("TEMP.dat", ios::out);
        file.seekg(0,ios::beg);

        // Delete the record using copy method into temporaty file
        while (!file.eof())
        {
                file.read((char *)this, sizeof(Issue));
                if (file.eof())
                        break;
                if ((acq_no != memb_no) && (acc_no != s_acc_no))
                        temp.write((char *)this, sizeof(Issue));
        }
        file.close();
        temp.close();
        file.open("MASTER.dat", ios::out);
        temp.open("TEMP.dat", ios::in);

        temp.seekg(0, ios::beg);

        // Overwrites the temporay contents into original MASTER.dat
data file
        while (!temp.eof())
        {
                temp.read((char *)this, sizeof(Issue));
                if (temp.eof())
                        break;
                if (memb_no != s_memb_no)
                        file.write((char *)this, sizeof(Issue));
        }
        file.close();
        temp.close();
}

// Function to find the expire dates of members
int date_expire(int s_edd, int s_emm, int s_eyy)
{
```

```
        int s_cdd, s_cmm, s_cyy;        // Current date
        int pdays = 0;
        struct date td;
        getdate(&td);

        s_cdd = td.da_day;
        s_cmm = td.da_mon;
        s_cyy = td.da_year;
        // when expire day less than current day, expire month equal to
current month
        // and expire year equal to current year
        if ((s_edd < s_cdd) && (s_emm == s_cmm) && (s_eyy == s_cyy))
                pdays = 1;
        else

        // when expire day less than current day, expire month less than
current month
        // and expire year equal to current year

        if ((s_edd < s_cdd) && (s_emm < s_cmm) && (s_eyy == s_cyy))
                pdays = 1;
        else

        // when expire day less than current day, expire month less than
current month
        // and expire year less than current year

        if ((s_edd < s_cdd) && (s_emm < s_cmm) && (s_eyy < s_cyy))
                pdays = 1;
        else

        // when expire day greater than current day, expire month less
than current month
        // and expire year equal to current year

        if ((s_edd > s_cdd) && (s_emm < s_cmm) && (s_eyy == s_cyy))
                pdays = 1;
        else

        // when expire day greater than current day, expire month less
than current month
        // and expire year less current year

        if ((s_edd > s_cdd) && (s_emm < s_cmm) && (s_eyy < s_cyy))
                pdays = 1;
```

```
        else

        // when expire day greater than current day, expire month equal to
current month
        // and expire year less than current year

        if ((s_edd > s_cdd) && (s_emm == s_cmm) && (s_eyy < s_cyy))
                pdays = 1;
        else

        // when expire day greater than current day, expire month greater
than current month
        // and expire year less than current year

        if ((s_edd > s_cdd) && (s_emm > s_cmm) && (s_eyy < s_cyy))
                pdays = 1;
        else

        // when expire day equal to current day, expire month equal to
current month
        // and expire year less than current year

        if ((s_edd == s_cdd) && (s_emm == s_cmm) && (s_eyy < s_cyy))
                pdays = 1;
        else

        // when expire day less than current day, expire month equal to
current month
        // and expire year less than to current year

        if ((s_edd < s_cdd) && (s_emm == s_cmm) && (s_eyy < s_cyy))
                pdays = 1;
        else

        // when expire day equal to current day, expire month less than
current month
        // and expire year less than current year

        if ((s_edd == s_cdd) && (s_emm < s_cmm) && (s_eyy < s_cyy))
                pdays = 1;
        else

        // when expire day less than current day, expire month greater
than current month
        // and expire year less than to current year
```

```
        if ((s_edd < s_cdd) && (s_emm > s_cmm) && (s_eyy < s_cyy))
                pdays = 1;
        else

        // when expire equal to current day, expire month greater than
current month
        // and expire year less than current year

        if ((s_edd == s_cdd) && (s_emm > s_cmm) && (s_eyy < s_cyy))
                pdays = 1;
        return (pdays);
}

// Function to check membership date expired or not
int Member::Expire_Check(int s_memb_no)
{
        int xflag = 0;

        fstream file;
        file.open("MEMFILE.dat", ios::in);
        file.seekg(0, ios::beg);
        while (file.read((char *)this, sizeof(Member)))
        {
                if (s_memb_no == memb_no)
                {
                        xflag = date_expire(edd, emm, eyy);
                        break;
                }
        }
        file.close();
        return (xflag);
}

// Function to issue books to members
void Library::Book_Issue()
{
        char ch;
        int i, tflag = 0; //, valid;
        int mday = 0;
        clrscr();
        gotoxy(65, 3);
        cout << "<0>=Exit";
        BOX_CLASS s;
        s.DRAW_BOX(10, 5, 71, 21, 219);
```

```
s.DRAW_BOX(9, 4, 72, 22, 218);
textcolor(BLACK);
textbackground(LIGHTGRAY);

gotoxy(3,3);
for (i = 1; i<= 76; i++)
        cprintf(" ");

textbackground(BLACK);
textcolor(BLACK+BLINK);
textbackground(WHITE);
gotoxy(35, 4);
cprintf("Issue Form");
textcolor(LIGHTGRAY);
textbackground(BLACK);

int s_acq_no = 0;  // Acquisition number

int s_acc_no = 0;  // Accession number
char s_title[30];
char s_author1[30], s_author2[30];   // Authors
char s_publisher[30];     // Name of the publisher
float s_price;         // Price of the book
int s_idd, s_imm, s_iyy;  // Date of Issue
int s_rdd, s_rmm, s_ryy; // Date of return
int s_pages; // Total pages in book
char s_status;       // Present status of book

int s_memb_no;    // Member number for issuing books

struct date d;                // For extracting system date
getdate(&d);

s_idd = d.da_day;
s_imm = d.da_mon;
s_iyy = d.da_year;

// Steps to find the return date of issued book
mday = month_days(s_imm);

s_rdd = s_idd + 3;
if (s_rdd <= mday)
{
        s_rdd = s_rdd;
        s_rmm = s_imm;
```

```
        s_ryy = s_iyy;
}
else
{
        s_rdd = 3 - (mday - s_idd);
        if (s_imm == 12)
        {
                s_rmm = 1;
                s_ryy = s_iyy + 1;
        }
        else
        {
                s_rmm = s_imm + 1;
                s_ryy = s_iyy;
        }
}

gotoxy(48, 7);
cout << "Issue date: " << s_idd << '-' << s_imm << '-' << s_iyy;
gotoxy(48, 8);
cout << "Return date: " << s_rdd << '-' << s_rmm << '-' << s_ryy;

gotoxy(10, 23);
cout << "Enter the Acquisition number";
gotoxy(13, 7);
cout << "Acquisition No. " ;
gotoxy(29, 7);
cin >> s_acq_no;
Left_clear(10, 23);
Right_clear(39, 23);


gotoxy(11, 8);
for (i = 1; i <= 60; i++)
        cout << "=";

gotoxy(10, 23);
cout << "Enter the Accession number";
gotoxy(13, 9);
cout << "Accession No. ";
gotoxy(28, 9);
cin >> s_acc_no;
Left_clear(10, 23);
Right_clear(39, 23);
```

```
if ((s_acq_no == 0) || (s_acc_no == 0))
{
        Issue Iss;
        Iss.Issue_Add(0, 0, 0, 1, 1, 1, 1, 1, 1);
        Iss.Issue_Delete(0, 0, 0, 1, 1, 1, 1, 1, 1);
        return;
}

fstream file;
file.open("LIB.dat", ios::in);
file.seekg(0,ios::beg);

// Search the record to display before issue
while (file.read((char *)this, sizeof(Library)))
{
        if ((s_acq_no == acq_no) && (s_acc_no == acc_no))
        {
                gotoxy(15, 11);
                cout << "Title : ";
                gotoxy(15, 12);
                cout << "Author(s) : ";
                gotoxy(15, 14);
                cout << "Publisher : ";
                gotoxy(15, 15);
                cout << "Price : ";
                gotoxy(15, 16);
                cout << "Pages : ";
                gotoxy(15, 17);
                cout << "Status : ";
                gotoxy(26, 11);
                puts(title);

                gotoxy(26, 12);
                puts(author1);

                gotoxy(26, 13);
                puts(author2);
                gotoxy(26, 14);
                puts(publisher);
                gotoxy(26, 15);
                cout << price;

                gotoxy(26, 16);
                cout << pages;
                gotoxy(26, 17);
```

```
                cout << status;

                // Function check whether this book is already issued
or not to any member
                Issue Iss;
                tflag = Iss.Issue_Check(s_acq_no, s_acc_no);

                if (tflag == 1)
                {
                        gotoxy(12, 23);
                        cout << "This book is aleardy issued";
                        getch();
                        Left_clear(10, 23);
                        Right_clear(39, 23);
                        return;
                }
                else
                {
                        tflag = 0;
                        gotoxy(45, 9);
                        cout << "Membership No. ";
                        gotoxy(64, 9);
                        cin >> s_memb_no;

                        // Function to check such a member is their in
MEMBER.dat data file or not
                        Member Memb;
                        tflag = Memb.Memb_Check(s_memb_no);

                        if (tflag == 1)
                        {
                                tflag = 0;
                                // Function to check membership expired
or not
                                tflag = Memb.Expire_Check(s_memb_no);

                                if (tflag == 0)
                                {
                                        tflag = 0;
                                        // Function to check whether the
member already book issued or not by tflag
                                        tflag =
Iss.Issue_memb_check(s_memb_no);

                                        if (tflag == 0)
```

```
                                        {
                                                // Function to append the
issued book into MASTER.dat data file
                                                Iss.Issue_Add(s_acq_no,
s_acc_no, s_memb_no, s_idd, s_imm, s_iyy, s_rdd, s_rmm, s_ryy);
                                                gotoxy(12, 23);
                                                cout << "Book Issued";
                                                getch();
                                                Left_clear(10, 23);
                                                Right_clear(39, 23);
                                        }
                                        else
                                        {
                                                gotoxy(12, 23);
                                        cout << "The member has already a book";
                                                getch();
                                        }
                                }
                                else
                                {
                                        gotoxy(12, 23);
                                        cout << "The membership Expired";
                                        getch();
                                }
                        }
                        else
                        {
                                gotoxy(12, 23);
                                cout << "The member does not exists";
                                getch();
                        }
                }
                break;
        }
    }
    file.close();
}

// Function to add fines if return date is exceeds
void Fine::Fine_add(int s_memb_no, int s_acq_no, int s_acc_no, int s_idd,
int s_imm, int s_iyy, int s_rdd, int s_rmm, int s_ryy, int s_cdd, int s_cmm,
int s_cyy, int s_finfee)
{
    memb_no = s_memb_no;
    acq_no = s_acq_no;      // fine acquisition no
```

```
        acc_no = s_acc_no;        // fine accession no

        idd = s_idd;          // Book issue date
        imm = s_imm;
        iyy = s_iyy;

        rdd = s_rdd;          // Book purposed return date
        rmm = s_rmm;
        ryy = s_ryy;

        cdd = s_cdd;                // Returned date
        cmm = s_cmm;
        cyy = s_cyy;

        finfee = s_finfee;
        fstream sfile;

        // Append the fined record into FINE.dat data file if the member
returns the book after due date
        sfile.open("FINE.dat", ios::out|ios::app);
        sfile.write((char *)this, sizeof(Fine));
        sfile.close();
}

// Function to display the book information at return time
void Library::Book_display(int s_acq_no, int s_acc_no)
{
        fstream file;
        file.open("LIB.dat", ios::in);
        file.seekg(0,ios::beg);

        // Search the record for displaying the book information
        while (file.read((char *)this, sizeof(Library)))
        {
                if ((s_acq_no == acq_no) && (s_acc_no == acc_no))
                {
                        gotoxy(40, 8);
                        cout << "Title : ";
                        gotoxy(40, 9);
                        cout << "Author(s) : ";
                        gotoxy(40, 11);
                        cout << "Publisher : ";
                        gotoxy(40, 15);
                        cout << "Price : ";
                        gotoxy(40, 16);
```

```
                        cout << "Pages : ";
                        gotoxy(40, 17);
                        cout << "Status : ";
                        gotoxy(49, 8);
                        puts(title);

                        gotoxy(53, 9);
                        puts(author1);

                        gotoxy(53, 10);
                        puts(author2);
                        gotoxy(53, 11);
                        puts(publisher);
                        gotoxy(52, 15);
                        cout << price;

                        gotoxy(52, 16);
                        cout << pages;
                        gotoxy(52, 17);
                        cout << status;
                        break;
                }
        }
        file.close();
}

// Function to return the total days in a month
int month_days(int s_cmm)
{
        int dmon = 0;
        switch (s_cmm)
        {
                case 1: dmon = 31; break;
                case 2: dmon = 28; break;
                case 3: dmon = 31; break;
                case 4: dmon = 30; break;
                case 5: dmon = 31; break;
                case 6: dmon = 30; break;
                case 7: dmon = 31; break;
                case 8: dmon = 31; break;
                case 9: dmon = 30; break;
                case 10: dmon = 31; break;
                case 11: dmon = 30; break;
                case 12: dmon = 31; break;
        }
```

```
	return (dmon);
}

// Function to find the days in between two dates
int no_of_days(int s_cdd, int s_cmm, int s_cyy, int s_rdd, int s_rmm, int
s_ryy)
{
	int mdays = 0;
	int pdays = 0;
	mdays = month_days(s_rmm);
	if ((s_rdd < s_cdd) && (s_rmm == s_cmm) && (s_ryy == s_cyy))

		pdays = s_cdd - s_rdd;
	else
	if ((s_rdd < s_cdd) && (s_rmm < s_cmm) && (s_ryy == s_cyy))
		pdays = (mdays - s_rdd) + s_cdd + ((s_cmm-1) - s_rmm) * 30;
	else
	if ((s_rdd < s_cdd) && (s_rmm < s_cmm) && (s_ryy < s_cyy))
		pdays = (mdays - s_rdd) + s_cdd + ((s_cmm-1) - s_rmm) * 30 +
(s_cyy - s_ryy) * 12 * 30;
	else
	if ((s_rdd > s_cdd) && (s_rmm < s_cmm) && (s_ryy == s_cyy))
		pdays = (mdays - s_rdd) + s_cdd + ((s_cmm-1) - s_rmm) * 30;
	else
	if ((s_rdd > s_cdd) && (s_rmm < s_cmm) && (s_ryy < s_cyy))
		pdays = (mdays - s_rdd) + s_cdd + ((s_cmm-1) - s_rmm) * 30 +
(s_cyy - s_ryy) * 12 * 30;
	else
	if ((s_rdd > s_cdd) && (s_rmm == s_cmm) && (s_ryy < s_cyy))
		pdays = (mdays - s_rdd) + s_cdd + ((s_cmm-1) - s_rmm) * 30 +
(s_cyy - s_ryy) * 12 * 30;

	else
	if ((s_rdd > s_cdd) && (s_rmm > s_cmm) && (s_ryy < s_cyy))
		pdays = (mdays - s_rdd) + s_cdd + ((s_cmm-1) - s_rmm) * 30 +
(s_cyy - s_ryy) * 12 * 30;
	else
	if ((s_rdd == s_cdd) && (s_rmm == s_cmm) && (s_ryy < s_cyy))
		pdays = (s_cyy - s_ryy) * 12 * 30;
	else
	if ((s_rdd < s_cdd) && (s_rmm == s_cmm) && (s_ryy < s_cyy))
		pdays = (s_cdd - s_rdd) + (s_cyy - s_ryy) * 12 * 30;
	else
	if ((s_rdd == s_cdd) && (s_rmm < s_cmm) && (s_ryy < s_cyy))
		pdays = (s_cmm - s_rmm) * 30 + (s_cyy - s_ryy) * 12 * 30;
```

```
        else
        if ((s_rdd < s_cdd) && (s_rmm > s_cmm) && (s_ryy < s_cyy))
                pdays = (mdays - s_rdd) + s_cdd + (12 - s_rmm) * 30 +
(s_cmm - 1) * 30 + ((s_cyy - (s_ryy+1)) * 12 * 30) ;
        else
        if ((s_rdd == s_cdd) && (s_rmm > s_cmm) && (s_ryy < s_cyy))
                pdays = (12 - s_rmm) * 30 + (s_cmm) * 30 + ((s_cyy -
(s_ryy+1)) * 12 * 30) ;
        cout << pdays;
        return (pdays);
}

// Function display the books at the time of return
void Issue::Return_Issue(int s_memb_no)
{
        fstream file;
        file.open("MASTER.dat", ios::in);
        file.seekg(0,ios::beg);
        int s_acq_no, s_acc_no;
        int s_idd, s_imm, s_iyy;  // Issue date
        int s_rdd, s_rmm, s_ryy; // Return date
        int s_cdd, s_cmm, s_cyy;         // Current date

        int s_finfee; // Fine amount
        struct date d;                // For extracting system date
        getdate(&d);
        s_cdd = d.da_day;
        s_cmm = d.da_mon;
        s_cyy = d.da_year;
        int tdays = 0;        // Variable store the date difference
        // Search the record for further displaying the issue and return
dates
        while (file.read((char *)this, sizeof(Issue)))
        {
                if (s_memb_no == memb_no)
                {
                        s_acq_no = acq_no;
                        s_acc_no = acc_no;
                        s_idd = idd;
                        s_imm = imm;
                        s_iyy = iyy;

                        s_rdd = rdd;
                        s_rmm = rmm;
                        s_ryy = ryy;
```

```
                        // Function to display the book information at the time
of return
                        Library lib;
                        lib.Book_display(s_acq_no, s_acc_no);

                        gotoxy(12, 19);
                        cout << "Issue Date: " << s_idd << "-" << s_imm << "-"
<< s_iyy;
                        gotoxy(40, 19);
                        cout << "Return Date: " << s_rdd << "-" << s_rmm << "-"
<< s_ryy;

                        // The function no_of_days() calculate the difference
betwee
                        // return date and current date
                        tdays = no_of_days(s_cdd, s_cmm, s_cyy, s_rdd,
s_rmm, s_ryy);
                        if (tdays > 0)
                        {
                                s_finfee = tdays;
                                cout << s_finfee;
                                // Function add a fine record if the member
returned the book after due date
                                Fine fin;
                                fin.Fine_add(s_memb_no, s_acq_no, s_acc_no,
s_idd, s_imm, s_iyy, s_rdd, s_rmm, s_ryy, s_cdd, s_cmm, s_cyy, s_finfee);
                        }
                        break;
                }
        }
        file.close();
}

// Function to delete member data from MASTER.dat data file
void Issue::Issue_member_delete(int s_memb_no)
{
        fstream file;
        file.open("MASTER.dat", ios::in);
        fstream temp;
        temp.open("TEMP.dat", ios::out);
        file.seekg(0,ios::beg);

        // Delete the record using copy method into temporaty file
        while (!file.eof())
```

```
        {
                file.read((char *)this, sizeof(Issue));
                if (file.eof())
                        break;
                if (memb_no != s_memb_no)
                        temp.write((char *)this, sizeof(Issue));
        }
        file.close();
        temp.close();
        file.open("MASTER.dat", ios::out);
        temp.open("TEMP.dat", ios::in);
        temp.seekg(0, ios::beg);

        // Overwrites the temporay contents into original MASTER.dat
data file
        while (!temp.eof())
        {
                temp.read((char *)this, sizeof(Issue));
                if (temp.eof())
                        break;
                if (memb_no != s_memb_no)
                        file.write((char *)this, sizeof(Issue));
        }
        file.close();
        temp.close();
}

// Function to return books in library
// The user has to enter the member no and the screen will display the
details of the book which is returning
void Issue::Book_Return(void)
{
        char ch;
        int i, flag = 0;

        Library lib;
        clrscr();
        gotoxy(65, 3);
        cout << "<0>=Exit";
        BOX_CLASS s;
        s.DRAW_BOX(10, 5, 71, 21, 219);
        s.DRAW_BOX(9, 4, 72, 22, 218);
        textcolor(BLACK);
        textbackground(LIGHTGRAY);
```

```
        gotoxy(3,3);
        for (i = 1; i<= 76; i++)
                cprintf(" ");

        textbackground(BLACK);
        textcolor(BLACK+BLINK);
        textbackground(WHITE);
        gotoxy(35, 4);
        cprintf("Book Return Form");
        textcolor(LIGHTGRAY);
        textbackground(BLACK);

        do
        {
                lib.Clear_area(6, 18, 12, 70);
                int s_memb_no;    // Membership number
                char s_name[30];  // Name of the member
                char s_address[30]; // Address of the employee
                int s_mdd, s_mmm, s_myy;  // Membership date
                int s_edd, s_emm, s_eyy; // Expire date
                float s_memb_fee; // Membership fee
                float s_caut_fee;    // Caution money

                struct date d;                // For extracting system date
                getdate(&d);
                s_mdd = d.da_day;
                s_mmm = d.da_mon;
                s_myy = d.da_year;

                gotoxy(12, 7);
                cout << "Date: " << s_mdd << '-' << s_mmm << '-' << s_myy;
                gotoxy(12, 23);
                cout << "Enter membership no. to Return Book";
                gotoxy(12, 9);
                cout << "Membership No. " ;
                gotoxy(27, 9);
                cin >> s_memb_no;
                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

                if (s_memb_no == 0)
                        return;

                // Function to display members information at the time of
book return
```

```
                Member Memb;
                Memb.Display_member(s_memb_no);
                // Function to find the fine and display the issue and return
dates
                Return_Issue(s_memb_no);

                // Function to check whether the member takes book or not
                flag = Issue_memb_check(s_memb_no);

                if (flag == 1)
                {
                        // Member deleted from MASTER.dat data file i.e.,
book returned
                        Issue_member_delete(s_memb_no);
                        // Return_function displays the book information and
add fine if avail
                        lib.Left_clear(12, 23);
                        lib.Right_clear(12, 23);
                }
                else
                {
                        gotoxy(12, 23);
                        cout << "Member Does not have any book";
                        getch();

                        lib.Left_clear(12, 23);
                        lib.Right_clear(12, 23);
                }

                gotoxy(12, 23);
                cout << "Do you want to Return any Record <Y/N>: ";
                ch = getche();
                if (ch == 'n' || ch == 'N')
                        return;

                lib.Left_clear(12, 23);
                lib.Right_clear(12, 23);

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

// Function for displaying the report menu
void Library::Report_Menu(void)
{
```

```
char ch;
while (1)
{
        clrscr();

        BOX_CLASS s;
        s.DRAW_BOX(10, 5, 71, 21, 219);
        s.DRAW_BOX(9, 4, 72, 22, 218);
        textcolor(BLACK);
        textbackground(WHITE);
        gotoxy(21, 7);
        cprintf("R E P O R T I N G   C O N S O L E");
        textcolor(RED);
        textbackground(BLACK);
        gotoxy(25, 9);
        cout << "1: Acquisition Register";
        gotoxy(25, 10);
        cout << "2. Membership Register";
        gotoxy(25, 11);
        cout << "3. Issue Register";
        gotoxy(25, 12);
        cout << "4. List of OverDue Books";
        gotoxy(25, 13);
        cout << "5. Fine Register";
        gotoxy(25, 14);
        cout << "6. List of Expire Members";

        gotoxy(25, 15);
        cout << "0. Return";
        gotoxy(25, 17);
        cout << "Enter your choice : ";
        date_time();
        ch = getche();
        if (ch == 27)
                break;
        else
                if (ch == '1')
                {
                        Acqu_Register();
                }
                else
                        if (ch == '2')
                        {
                                Member Memb;
                                Memb.Memb_Register();
```

```
                        }
                        else
                                if (ch == '3')
                                {
                                        Issue Iss;
                                        Iss.Issue_Register();
                                }
                                else
                                        if (ch == '4')
                                        {
                                                Issue Iss;
                                                Iss.OverDue_Book();
                                        }
                                        else
                                                if (ch == '5')
                                                {
                                                        Fine fin;
                                                        fin.Fine_Register();
                                                }
                                                else
                                                        if (ch == '6')
                                                        {
                                                                Member Memb;

        Memb.Expire_Register();
                                                        }
                                                        else
                                                                if (ch == '0')
                                                                        break;
        }
}


// The function is used to print the report of acquisition register.
void Library::Acqu_Register(void)
{
        fstream file;
        file.open("LIB.dat", ios::in);
        file.seekg(0,ios::beg);
        clrscr();

        int s_acq_no;       // Acquisition number
        int s_acc_no;       // Accession number
        char s_title[30];
        char s_author1[30], s_author2[30];   // Authors
```

```
char s_publisher[30];     // Name of the publisher
float s_price;         // Price of the book
int s_dd, s_mm, s_yy;     // Date of purchase
int s_pages; // Total pages in book
char s_status;       // Present status of book
gotoxy(30, 2);
cout << "ACQUISITION REGISTER";
gotoxy(30, 3);
cout << "--------------------";
int xdd, xmm, xyy;
struct date d;                // For extracting system date
getdate(&d);
xdd = d.da_day;
xmm = d.da_mon;
xyy = d.da_year;
gotoxy(63, 3);
cout << "Date: "<< xdd << "-" << xmm << "-" << xyy;

gotoxy(1, 4);
for (int i = 0; i <= 79; i++)
        cout << "=";
gotoxy(1, 5);
cout << "Sl.";
gotoxy(6, 5);
cout << "Acq.";
gotoxy(12, 5);
cout << "Acc.No.";
// For displaying only 12 characters of each
gotoxy(22, 5);
cout << "Title &";
gotoxy(40, 5);
cout << "Publisher";
gotoxy(54, 5);
cout << "Price &";
gotoxy(63, 5);
cout << "Date(p)";
gotoxy(75, 5);
cout << "Status";

gotoxy(1, 6);
cout << "No";
gotoxy(6, 6);
cout << "No.";

// For displaying only 12 characters of each
```

```
gotoxy(22, 6);
cout << "Author(s)";
gotoxy(54, 6);
cout << "Pages";

gotoxy(1, 7);
for (i = 0; i <= 79; i++)
        cout << "=";
int sln = 0;
int row = 8;
// Displaying the book information
while (file.read((char *)this, sizeof(Library)))
{
        ++sln;
        s_acq_no = acq_no;
        s_acc_no = acc_no;
        strcpy(s_title, title);
        strcpy(s_author1, author1);
        strcpy(s_author2, author2);
        strcpy(s_publisher, publisher);
        s_price = price;
        s_dd = dd;
        s_mm = mm;
        s_yy = yy;
        s_pages = pages;
        s_status = status;

        gotoxy(1, row);
        cout << sln;
        gotoxy(6, row);
        cout << s_acq_no;
        gotoxy(14, row);
        cout << s_acc_no;
        // For displaying only 12 characters of each
        gotoxy(21, row);
        cout << s_title;
        gotoxy(40, row);
        cout << s_publisher;
        gotoxy(55, row);
        cout << s_price;

        gotoxy(63, row);
        cout << s_dd << "-" << s_mm << "-" << s_yy;
        gotoxy(78, row);
        cout << s_status;
```

```
                row++;
                gotoxy(21, row);
                cout << s_author1;
                gotoxy(55, row);
                cout << s_pages;

                row++;
                gotoxy(21, row);
                cout << s_author2;


                gotoxy(1, row);
                for (i = 0; i <= 79; i++)
                        cout << "-";

                row++;
                if (row > 23)
                {
                        row = 8;
                        gotoxy(4, 24);
                        cout << "Press any key to continue.... ";
                        getch();
                        clrscr();
                }

        }
        gotoxy(1, row);
        for (i = 0; i <= 79; i++)
                cout << "=";

        getch();
        file.close();

}

// Function for Membership register
void Member::Memb_Register(void)
{
        clrscr();
        gotoxy(30, 2);
        cout << "MEMBERSHIP REGISTER";
        gotoxy(30, 3);
        cout << "--------------------";
        int xdd, xmm, xyy;
        struct date d;                // For extracting system date
```

```
getdate(&d);
xdd = d.da_day;
xmm = d.da_mon;
xyy = d.da_year;
gotoxy(63, 3);
cout << "Date: "<< xdd << "-" << xmm << "-" << xyy;

gotoxy(1, 4);
for (int i = 0; i <= 79; i++)
        cout << "=";
gotoxy(1, 5);
cout << "Sl.No";
gotoxy(7, 5);
cout << "Memb.No.";
gotoxy(17, 5);
cout << "Name &";
gotoxy(40, 5);
cout << "Memb.Date";
gotoxy(51, 5);
cout << "Expir.Date";
gotoxy(63, 5);
cout << "Membership";
gotoxy(74, 5);
cout << "Caution";

gotoxy(17, 6);
cout << "Address";
gotoxy(65, 6);
cout << "Fees";
gotoxy(75, 6);
cout << "Money";

int s_memb_no;    // Membership number
char s_name[30];  // Name of the member
char s_address[30]; // Address of the employee
int s_mdd, s_mmm, s_myy;  // Membership date
int s_edd, s_emm, s_eyy; // Expire date
float s_memb_fee; // Membership fee
float s_caut_fee;    // Caution money


gotoxy(1, 7);
for (i = 0; i <= 79; i++)
        cout << "=";
int sln = 0;
```

```
int row = 8;
// Displaying the Members information

fstream file;
file.open("MEMFILE.dat", ios::in);
file.seekg(0, ios::beg);
while (file.read((char *)this, sizeof(Member)))
{
        ++sln;
        s_memb_no = memb_no;
        strcpy(s_name, name);
        strcpy(s_address, address);
        s_mdd = mdd;
        s_mmm = mmm;
        s_myy = myy;
        s_edd = edd;
        s_emm = emm;
        s_eyy = eyy;
        s_memb_fee = memb_fee;
        s_caut_fee = caut_fee;

        gotoxy(2, row);
        cout << sln;
        gotoxy(7, row);
        cout << s_memb_no;
        gotoxy(16, row);
        cout << s_name;
        gotoxy(40, row);
        cout << s_mdd << "-" << s_mmm << "-" << s_myy;
        gotoxy(51, row);
        cout << s_edd << "-" << s_emm << "-" << s_eyy;

        gotoxy(65, row);
        cout << s_memb_fee << setw(15)
                                << setprecision(2)
                                << setiosflags(ios::left)
                                << setiosflags(ios::showpoint)
                                << setiosflags(ios::fixed);
        gotoxy(75, row);
        cout << s_caut_fee << setw(15)
                                << setprecision(2)
                                << setiosflags(ios::left)
                                << setiosflags(ios::showpoint)
                                << setiosflags(ios::fixed);
        row++;
```

```
                gotoxy(16, row);
                cout << s_address;
                row++;
                gotoxy(1, row);

                for (i = 0; i <= 79; i++)
                        cout << "-";

                row++;

                if (row > 23)
                {
                        row = 8;
                        gotoxy(4, 24);
                        cout << "Press any key to continue.... ";
                        getch();
                        clrscr();
                }
        }
        gotoxy(1, row);
        for (i = 0; i <= 79; i++)
                cout << "=";

        getch();
        file.close();
}

// Function for issue register
void Issue::Issue_Register()
{
        clrscr();
        gotoxy(35, 2);
        cout << "ISSUE REGISTER";
        gotoxy(35, 3);
        cout << "--------------";
        int xdd, xmm, xyy;
        struct date d;                // For extracting system date
        getdate(&d);
        xdd = d.da_day;
        xmm = d.da_mon;
        xyy = d.da_year;
        gotoxy(63, 3);
        cout << "Date: "<< xdd << "-" << xmm << "-" << xyy;

        gotoxy(1, 4);
```

```
for (int i = 0; i <= 79; i++)
        cout << "=";
gotoxy(1, 5);
cout << "Sl.No";
gotoxy(9, 5);
cout << "Memb.No.";
gotoxy(18, 5);
cout << "Acq.no";
gotoxy(28, 5);
cout << "Acc.No";
gotoxy(49, 5);
cout << "Books Issued";
gotoxy(63, 5);
cout << "Purposed Date";

gotoxy(55, 6);
cout << "On";
gotoxy(65, 6);
cout << "Of Return";

int s_acq_no;       // Aquisition number
int s_acc_no;       // Accession no
int s_memb_no;    // Member number
int s_idd, s_imm, s_iyy;  // Issue date
int s_rdd, s_rmm, s_ryy; // Return date


gotoxy(1, 7);
for (i = 0; i <= 79; i++)

        cout << "-";
int sln = 0;
int row = 8;
// Displaying the Issued books information

fstream file;
file.open("MASTER.dat", ios::in);
file.seekg(0,ios::beg);
// Search the record for further modification
while (file)
{
        file.read((char *)this, sizeof(Issue));
        if (file.eof())
                break;
        ++sln;
```

```
                s_acq_no = acq_no;
                s_acc_no = acc_no;
                s_memb_no = memb_no;
                s_idd = idd;
                s_imm = imm;
                s_iyy = iyy;
                s_rdd = rdd;
                s_rmm = rmm;
                s_ryy = ryy;

                gotoxy(1, row);
                cout << sln;
                gotoxy(10, row);
                cout << s_memb_no;
                gotoxy(20, row);
                cout << s_acq_no;
                gotoxy(30, row);
                cout << s_acc_no;
                gotoxy(50, row);
                cout << s_idd << "-" << s_imm << "-" << s_iyy;
                gotoxy(63, row);
                cout << s_rdd << "-" << s_rmm << "-" << s_ryy;
                row++;
                if (row > 23)
                {
                        row = 8;
                        gotoxy(4, 24);
                        cout << "Press any key to continue.... ";
                        getch();
                        clrscr();
                }
        }
        getch();
        file.close();
}

// Function for overdue books
void Issue::OverDue_Book()
{
        clrscr();
        gotoxy(35, 2);
        cout << "OVERDUE BOOK";
        gotoxy(35, 3);
        cout << "------------";
        int xdd, xmm, xyy;
```

```
struct date d;                  // For extracting system date
getdate(&d);
xdd = d.da_day;
xmm = d.da_mon;
xyy = d.da_year;
gotoxy(63, 3);

cout << "Date: "<< xdd << "-" << xmm << "-" << xyy;

gotoxy(1, 4);
for (int i = 0; i <= 79; i++)
        cout << "=";
gotoxy(1, 5);
cout << "Sl.No";
gotoxy(9, 5);
cout << "Memb.No.";
gotoxy(18, 5);
cout << "Acq.no";
gotoxy(28, 5);
cout << "Acc.No";
gotoxy(40, 5);
cout << "Issue Date";
gotoxy(53, 5);
cout << "Return Date";

gotoxy(66, 5);
cout << "Days Overdue";

gotoxy(66, 6);
cout << "As on Sys.Date";

int s_acq_no;       // Aquisition number
int s_acc_no;       // Accession no
int s_memb_no;    // Member number
int s_idd, s_imm, s_iyy;  // Issue date
int s_rdd, s_rmm, s_ryy; // Return date
int dmonth, ovrdue = 0;

gotoxy(1, 7);
for (i = 0; i <= 79; i++)
        cout << "-";
int sln = 0;
int row = 8;
// Displaying the Issued books information
```

```
fstream file;
file.open("MASTER.dat", ios::in);
file.seekg(0,ios::beg);
// Search the record for further modification
while (file.read((char *)this, sizeof(Issue)))
{
        s_acq_no = acq_no;
        s_acc_no = acc_no;
        s_memb_no = memb_no;
        s_idd = idd;
        s_imm = imm;
        s_iyy = iyy;
        s_rdd = rdd;
        s_rmm = rmm;
        s_ryy = ryy;
        ovrdue = no_of_days(xdd, xmm, xyy, s_rdd, s_rmm, s_ryy);
        if (ovrdue > 0)
        {
                ++sln;
                gotoxy(2, row);
                cout << sln;
                gotoxy(10, row);
                cout << s_memb_no;
                gotoxy(20, row);
                cout << s_acq_no;
                gotoxy(30, row);
                cout << s_acc_no;
                gotoxy(40, row);
                cout << s_idd << "-" << s_imm << "-" << s_iyy;
                gotoxy(53, row);
                cout << s_rdd << "-" << s_rmm << "-" << s_ryy;
                gotoxy(70, row);
                cout << ovrdue;
                row++;
                gotoxy(1, row);
                for (int i = 0; i <= 79; i++)
                        cout << "-";
                row++;
                if (row > 23)
                {
                        row = 7;
                        gotoxy(4, 24);
                        cout << "Press any key to continue.... ";
                        getch();
                        clrscr();
```

```
                        }
                }
        }
        getch();
        file.close();
}


// Function to display the fine register
void Fine::Fine_Register()
{
        clrscr();
        gotoxy(35, 2);
        cout << "FINE REGISTER";
        gotoxy(35, 3);
        cout << "-------------";
        int xdd, xmm, xyy;
        struct date d;                  // For extracting system date
        getdate(&d);
        xdd = d.da_day;
        xmm = d.da_mon;
        xyy = d.da_year;
        gotoxy(63, 3);
        cout << "Date: "<< xdd << "-" << xmm << "-" << xyy;

        gotoxy(1, 4);
        for (int i = 0; i <= 79; i++)
                cout << "=";
        gotoxy(1, 5);
        cout << "Sl.No";
        gotoxy(9, 5);
        cout << "Memb.No.";
        gotoxy(18, 5);
        cout << "Acq.no";
        gotoxy(28, 5);
        cout << "Acc.No";
        gotoxy(39, 5);
        cout << "Issue";
        gotoxy(49, 5);
        cout << "Returned";
        gotoxy(61, 5);
        cout << "Returned";
        gotoxy(74, 5);
        cout << "Fine";
```

```
gotoxy(39, 6);
cout << "Date";
gotoxy(51, 6);
cout << "Date";
gotoxy(63, 6);
cout << "On";

int s_memb_no;
int     s_acq_no;
int s_acc_no;
int s_idd, s_imm, s_iyy;  // Book issue date

int s_rdd, s_rmm, s_ryy; // Book purposed return date

int s_cdd, s_cmm, s_cyy;         // Returned date
int s_finfee;

gotoxy(1, 7);
for (i = 0; i <= 79; i++)
        cout << "-";
int sln = 0;
int row = 8;
// Displaying the Issued books information

fstream file;
file.open("Fine.dat", ios::in);
file.seekg(0,ios::beg);
// Search the record for further modification
while (file.read((char *)this, sizeof(Fine)))
{
        ++sln;
        s_memb_no = memb_no;
        s_acq_no = acq_no;
        s_acc_no = acc_no;
        s_idd = idd,
        s_imm = imm,
        s_iyy = iyy;

        s_rdd = rdd,
        s_rmm = rmm,
        s_ryy = ryy;

        s_cdd = cdd,
        s_cmm = cmm,
        s_cyy = cyy;
```

```
                s_finfee = finfee;

                gotoxy(1, row);
                cout << sln;
                gotoxy(9, row);
                cout << s_memb_no;
                gotoxy(18, row);
                cout << s_acq_no;
                gotoxy(28, row);
                cout << s_acc_no;
                gotoxy(37, row);
                cout << s_idd << "-" << s_imm << "-" << s_iyy;
                gotoxy(49, row);
                cout << s_rdd << "-" << s_rmm << "-" << s_ryy;
                gotoxy(61, row);
                cout << s_cdd << "-" << s_cmm << "-" << s_cyy;

                gotoxy(75, row);
                cout << s_finfee << setw(5)
                                        << setprecision(2)
                                        << setiosflags(ios::left)
                                        << setiosflags(ios::showpoint)
                                        << setiosflags(ios::fixed);
                row++;
                if (row > 23)
                {
                        row = 8;
                        gotoxy(4, 24);
                        cout << "Press any key to continue.... ";
                        getch();
                        clrscr();
                }
        }
        getch();
        file.close();
}

// Function to list out expiring membership persons
void Member::Expire_Register()
{
        clrscr();
        int tflag = 0;
        gotoxy(20, 2);
        cout << "List of Expiring Membership Person(s)";
        gotoxy(20, 3);
```

```
cout << "-------------------------------------";
int xdd, xmm, xyy;
struct date d;                  // For extracting system date
getdate(&d);
xdd = d.da_day;
xmm = d.da_mon;
xyy = d.da_year;
gotoxy(63, 3);
cout << "Date: "<< xdd << "-" << xmm << "-" << xyy;

gotoxy(1, 4);
for (int i = 0; i <= 79; i++)
        cout << "=";
gotoxy(1, 5);
cout << "Sl.No";
gotoxy(7, 5);
cout << "Membership ";
gotoxy(20, 5);
cout << "Name & ";
gotoxy(40, 5);
cout << "Membership";
gotoxy(58, 5);
cout << "Membership";

gotoxy(10, 6);
cout << "No.";
gotoxy(20, 6);
cout << "Address ";
gotoxy(40, 6);
cout << "Beginning Date";
gotoxy(58, 6);
cout << "Expire Date";

int s_memb_no;    // Membership number
char s_name[30];  // Name of the member
char s_address[30]; // Address of the employee
int s_mdd, s_mmm, s_myy;  // Membership date
int s_edd, s_emm, s_eyy; // Expire date
float s_memb_fee; // Membership fee
float s_caut_fee;    // Caution money


gotoxy(1, 7);
for (i = 0; i <= 79; i++)
        cout << "=";
```

```
        int sln = 0;
        int row = 8;
        // Displaying the Members information

        fstream file;
        file.open("MEMFILE.dat", ios::in);
        file.seekg(0, ios::beg);
        while (file.read((char *)this, sizeof(Member)))
        {
                tflag = 0;
                s_memb_no = memb_no;
                strcpy(s_name, name);
                strcpy(s_address, address);
                s_mdd = mdd;
                s_mmm = mmm;
                s_myy = myy;

                s_edd = edd; // Actual expire date
                s_emm = emm;
                s_eyy = eyy;

                // The function returns the value 1 or 0 from date_expire()
function
                tflag = date_expire(s_edd, s_emm, s_eyy);
                if (tflag == 1)
                {
                        ++sln;
                        gotoxy(2, row);
                        cout << sln;
                        gotoxy(9, row);
                        cout << s_memb_no;
                        gotoxy(19, row);
                        cout << s_name;
                        gotoxy(40, row);
                        cout << s_mdd << "-" << s_mmm << "-" << s_myy;
                        gotoxy(58, row);
                        cout << s_edd << "-" << s_emm << "-" << s_eyy;

                        row++;
                        gotoxy(19, row);
                        cout << s_address;
                        row++;
                        gotoxy(1, row);
                        for (i = 0; i <= 79; i++)
                                cout << "-";
```

```
                        row++;

                        if (row > 23)
                        {
                                row = 8;
                                gotoxy(4, 24);
                                cout << "Press any key to continue.... ";
                                getch();
                                clrscr();
                        }
                }
        }
        gotoxy(1, row);
        for (i = 0; i <= 79; i++)

                cout << "=";

        getch();
        file.close();
}

// Main Program starts at here
void main(void)
{
        control menu;
        menu.load();
        menu.help();
        menu.lib_SCREEN();
```

****