# COMPUTER SCIENCE PROJECT

# LIBRARY MANAGEMENT

# INDEX

# 1.0 PROJECT STATEMENT

The project is about the management of books in a library. The basic idea behind our source code is to issue books from the library and also manage the return of the books to the library. The program contains two classes: Book and Member. This project implements the concepts of OOPs and data file handling.

This program helps the librarian to
1) **add** books to the book database file, **modify** book details, and **delete** book records from the book database file;

2) **add** member records to the members database file, **modify** member details, and **delete** member records from the member database file;

3) **modify** and **add** issue details of books to both the book and member files; and

4) **record** the return of books in the book and member files.

The program has been designed to be operated by the librarian of the school by giving a unique user login.

The project is all encompassing for all situations that can happen in real life because of the checks and error prompts that have been diligently thought about to suit a logical error when typing or a miss in observation. It is suitable for a large scale number of books and members also since it checks the validity and uniqueness of the each book or member.

# 2.0 CLASSES, FUNCTIONS AND FILE DETAILS

## 2.1 Class

Two classes are used in this program:

**Class identifier: Book**

- Purpose: Store details of books available in library
- Data members:
  - book name
  - ISBN
  - author name
  - Issue status of the book.
- It is used as a record storage class for the binary file "BOOK_DATA.DAT".

**Class identifier: Member**

- Purpose: Store details of members of library
- Data members:
  - unique identification number for each member
  - name of the member
  - name of book issued to the member.
- It is used for storing records in "MEMBER_DATA.DAT" binary file.

## 2.2 Functions in Program

There are 12 functions and 5 subfunctions in this program.

**Functions**:

1) LIBRARIAN_LOGIN()
2) menu()
3) Book_database()
4) member_database()
5) rewrite_book()
6) rewrite_mem()
7) Book_Issue()
8) return_books()
9) disbooks()
10) dismembers()
11) delete_book()
12) delete_member()

**Subfunctions**:

1) Book_valid()
2) Mem_valid()
3) memID_authentic()
4) mem_modify()
5) Book_modify()

## 2.3 Files in Program

Two binary files are used in the program:

1. **BOOK_DATA.DAT**
    - File type: Binary
    - Purpose: stores all records of books added to the library
    - acts as a database for the books
    - Class stored: Book

2. **MEMBER_DATA.DAT**
    - File type: Binary
    - Purpose: stores all records of members added to the library
    - acts as a database for the members
    - Class stored: Member

## 2.4 Data structures

Two data structures are used in this program:

1. Arrays
2. Classes.

# 3.0 SOURCE CODE

```
//Header files included for the program
#include<iostream.h>
#include<fstream.h>
#include<stdio.h>
#include<string.h>
#include<conio.h>
#include<process.h>
#define pas "glivew5194"
#define user "PSSSS"
 /*************************************************************
                              BOOK CLASS
 *************************************************************/
class Book
{
char auth[100];
protected:
        char ISBN[100];
        char bkname[100];
public:
char issue;
        void getB()  /***Function to get details of a new Book***/
        {
                cout << "Enter the name of the book: ";
                cin.ignore();
                gets(bkname);
```

```
        cout << "Enter the ISBN given: ";
        gets(ISBN);
        cout << "Enter author's name: ";
        gets(auth);
        cout<<"Enter whether book is issued or not: ";
        cin>>issue;
}
char* ret_ISBN()   /***Function to return ISBN***/
{
        return ISBN;
}
char* ret_bkname()          /***Function to return book name***/
{
        return bkname;
}
char* ret_auth()      /***Function to return author name***/
{
        return auth;
}
void putB()  /***Function to print details of a book***/
{
if(strlen(bkname)>6)
        cout<<"\t  "<<bkname<<"\t  "<<ISBN<<"\t  ";
else
        cout<<"\t  "<<bkname<<"\t\t  "<<ISBN<<"\t  ";
```

```
        if(strlen(auth)>5)
                cout<<auth<<"\t   "<<issue<<endl;
        else
                cout<<auth<<"\t\t   "<<issue<<endl;
        }
        /***Function to copy a new name to data member 'bkname'***/
        void set_name(char new_name[])
        {
        strcpy(bkname,new_name);
        }
        /***Function to copy a new ISBN to data member 'ISBN'***/
        void set_ISBN(char new_isbn[])
        {
        strcpy(ISBN,new_isbn);
        }
        /***Function to copy a new author name to data member 'auth'***/
        void set_auth(char* author)
        {
        strcpy(auth,author);
        }
};
```

```
/*****************************************************************
                        MEMBER CLASS
*****************************************************************/
class Member
{
/***Data members to store name of member, issued book***/
char mName[100],iss_bkname[100];
/***Data members to store member ID, issue status***/
int memID;int token;
public:
        void getM() /***Function to get details of a new member***/
        {
                cout << "Enter member ID: ";
                cin >> memID;
                cout << "Enter name of member: ";
                cin.ignore();
                gets(mName);
                token=0;
                strcpy(iss_bkname,'\0');
        }
        void putM();
        int ret_memID()     /***Function to return member ID***/
        {
                return memID;
        }
```

```
char* ret_name()    /***Function to return name***/
{
        return mName;
}
/***Function to copy new member ID to data member ‘memID’***/
void set_memID(int mem)
{
memID=mem;
}
/***Function to copy new member name to data member ‘mName’***/
void set_name(char new_name[])
{
strcpy(mName,new_name);
}
/***Function to change token data member***/
void addtoken(char bk[])
{
token=1;
strcpy(iss_bkname,bk);
}
void retoken()
{
token=0;
strcpy(iss_bkname,'\0');
}
```

```
        /***Function to reset token after book is returned***/
        int ret_token()
        {
        return token;
        }
        char* book_iss()
        {
        return iss_bkname;
        }
};
void Member::putM()       /***Function to print member details***/
        {
                cout <<"\t  "<<memID<<"\t     "<<mName;
        if(token= =1 && strlen(mName)<=5)
                cout<<"\t     "<<iss_bkname;
        else if(strlen(mName)>5 && strlen(mName)<=7 && token= =1)
                cout<<"\t  "<<iss_bkname;
        else if(strlen(mName)>7)
                {
                 if(token= =1)
                   cout<<"\t "<<iss_bkname;
                 else
                   cout<<"\t  No issue";
                }
         else
                cout<<"\t  No issue";
                cout<<endl;
        }
```

```
/*****************************************************************
                        LIBRARIAN LOGIN
*****************************************************************/

int main()

{

clrscr();

LIBRARIAN_LOGIN();

return 0;

}


***Function to authenticate the librarian and open menu***/

void LIBRARIAN_LOGIN()

{

char passw[10],o,userID[10];

int f_user=0,f_pass=0;

        cout<<"\t\t\t   LIBRARY INFORMATION SYSTEM\n\t\t\t  P.S.

SENIOR SECONDARY SCHOOL\n\t\t\t    ~~~~~~USER

LOGIN~~~~~~\n"<<endl;

cout<<"User Name: "<<endl;

        gets(userID);

cout<<"Password: "<<endl;

        gets(passw);
```

```
        do
        {
        if(strcmp(userID,user)= =0)
        {
          if(strcmp(passw,pas)= =0)
           {
        clrscr();
        menu();
           }
          else
        f_pass=1;
        }
        else
        f_user=1;
                if(f_user= =1)
                   {cout<<"User name is wrong"<<endl; getch();break;}
                else if(f_pass= =1)
                   { cout<<"Password is wrong"<<endl; getch();break;}
cout<<"\nDo you want to open main menu again? Y for yes, N for no:  ";
        cin>>o;
        }
        while(o= ='y'||o= ='Y');
}
```

```
/*****************************************************************
                        MAIN MENU
*****************************************************************/

/***Function to print the main menu***/

void menu()

{

int option;

        cout<<"\n~~~~~~~~~~~~~MENU~~~~~~~~~~~~~" << endl << "  1. Add books to database " << endl << "  2. Add members to database " << endl << "  3. Edit a book's details " << endl << "  4. Edit a member's details " << endl << "  5. Issue a book " << endl << "  6. Return a book " << endl << "  7. Show all books in database \n  8. Show all members in database \n  9. Remove a book from library database \n 10. Remove a member from database \n\n\n Enter your choice: ";

        cin >> option;

        clrscr();

        if (option = = 1)

                Book_database();

        else if (option = = 2)

                member_database();

        else if (option = = 3)

                rewrite_book();
```

```
        else if (option = = 4)
                rewrite_mem();


        else if (option = = 5)
                Book_Issue();
        else if (option = = 6)
                return_books();
        else if (option = = 7)
        {
                cout<<"\t BOOK NAME\t\tISBN\t\tAUTHOR\t ISSUE\n ";
                disbooks();
        }
        else if (option = = 8)
        {
                cout<<"\t MEMBER ID\tNAME\tBOOKS ISSUED\n ";
                dismembers();
        }
        else if (option = = 9)
                delete_book();
        else if (option = = 10)
                delete_member();
}
```

```
/***Function to write into BOOK_DATA.DAT file to store new Books***/
void Book_database()
{
        Book b;
        ofstream f("Book_data.dat", ios::binary | ios::app);
        char o;
do
{
        cout<<endl;
        b.getB();
if(Book_valid(b,b.ret_ISBN())= =0 && Book_valid(b,b.ret_bkname())= =0)
          f.write((char*)&b, sizeof(b));
else
          cout<<"This book already exists in file! "<<endl;
        cout << "Do you want to store another book's details? Y for yes: ";
                cin >> o;
        }
        while (o = = 'y' || o = = 'Y');
        f.close();
}
```

```
/***Function to write into MEMBER_DATA.DAT to store new records***/
void member_database()
{
        Member m;
        char o;
        ofstream fmem("member_data.dat", ios::binary | ios::app);
        do
        {
        m.getM();
                if (memID_authentic(m.ret_memID()) = = 1)
                  fmem.write((char*)&m, sizeof(m));
                else
                  cout << "This ID already exists! Please try again \n";
cout << "Do you want to store another member's details? Y for yes, N for no: ";
        cin >> o;
        }
        while (o = = 'y' || o = = 'Y');
        fmem.close();
}
```

/***Function to rewrite a particular book's details***/

```
void rewrite_book()
{
 char ch1[25];Book b,b2; int posi,flag=0,fl;
 cout<<"Enter ISBN of book that has to be edited: ";
  cin.clear();
 gets(ch1);
 fstream f("Book_data.dat",ios::binary|ios::in|ios::out);
 b2.set_name('\0'); b2.set_ISBN('\0'); b2.set_auth('\0');
  while(f.read((char*)&b,sizeof(b)))
  {
   if(strcmp(b.ret_ISBN(),ch1)= =0)
    {
    b2.set_name('\0'); b2.set_ISBN('\0'); b2.set_auth('\0');
    b2=b;
    if(Book_valid(b,b.ret_ISBN())= =1)
      {


    cout<<"\nOld book details: "<<endl;
    b.putB();
```

```
    cout<<"\nFor record modification: "<<endl;
    if(book_modify(b)= =1)
    flag=1;
    else{
    posi=(f.tellg()/sizeof(b));
    f.seekg(((posi-1)*sizeof(b)),ios::beg);
    f.write((char*)&b,sizeof(b));
    b2=b;  }
      }
    }
  }
if(Book_valid(b2,b2.ret_ISBN())= =1)
{
if(flag!=1)
   cout<<"\nBook has been edited "<<endl;
}
else if(Book_valid(b2,b2.ret_ISBN())= =0)
   cout<<"\nBook doesn't exist! "<<endl;
  getch();
f.close();
}
```

```
/***Function to rewrite a particular member's details***/

void rewrite_mem()

{

int ID,opt;Member m,m2;char ch[30];int posi,flag;

cout<<"Enter ID that has to be edited: ";

cin>>ID;

fstream f("member_data.dat",ios::binary|ios::in|ios::out);

while(f.read((char*)&m,sizeof(m)))

        {

        if(m.ret_memID()= =ID)

        {

        m2=m;

                if(Mem_valid(m,m.ret_memID())= =1)

                {

        cout<<"\nOld Member details: "<<endl;

          m.putM();

        cout<<"\nFor record modification: "<<endl;
```

```
            if(mem_modify(m)= =0)
              flag=1;
            else{
              posi=(f.tellg()/sizeof(m));
              f.seekg((posi−1)*sizeof(m),ios::beg);
              f.write((char*)&m,sizeof(m));
              m2=m;  }
            }
        }
        }
if(flag!=1 && Mem_valid(m2,m2.ret_memID())= =1 )
        {cout<<"\nRecord has been edited\n";}
else if(Mem_valid(m2,m2.ret_memID())= =0 )
        cout<<"\nMember does not exist "<<endl;
getch();
f.close();
}
```

```
/***Function to issue 1 book to an existing member***/
void Book_Issue()
{
char mem;int id,flag=-1;
fstream fin("Book_data.dat",ios::binary|ios::in|ios::out);
fstream f("member_data.dat",ios::binary|ios::in|ios::out);
cout << "Is this issue for a member? Y for yes, N for no: ";
cin >> mem;
Member m,m1;
Book b;
char bk[20];     int posi;
if (mem = = 'y' || mem = = 'Y')
{
cout << "Enter member ID: ";
cin >> id;
if (Mem_valid(m,id) = = 1) // m is passed here so that it has a copy of the required member record thru' reference
{
        cout << "ISSUE FOR " << m.ret_name() << endl;
```

```
        if(m.ret_token()= =0)
        {
        cout << "Enter the name of the book taken: ";
          cin.ignore();
          gets(bk);
if(Book_valid(b,bk)= =1)
{
        m.addtoken(bk);
         while (fin.read((char*)&b, sizeof(b)))
          {
           if (strcmp(b.ret_bkname(), bk) = = 0)
              {
              if (b.issue = = 'N')
               {
                    b.issue='I';
                    posi=fin.tellg()/sizeof(b);
                      fin.seekg(((posi-1)*sizeof(b)),ios::beg);
                      fin.write((char*)&b,sizeof(b));
                    flag=0;
               }
```

```
            else
            flag=1;
            }
          }
         if(flag= =0)
         cout << "Books have been issued!  Please note the date of book\n
Must return in 10 days " << endl;
         else
         cout << "Sorry! Book is unavailable currently. Please try again later
\n";
   }
   else
   cout<<"Book does not exist!"<<endl;
         }
         else
         cout<<“Issued book ”<<m.book_iss()<<“ has not been returned!\n”;
}
else
{
    cout <<“Member ID does not exist! Please try again ” << endl;
}
}
```

```
else
cout << "This is a MEMBERS ONLY library! \n";
//TO WRITE INTO MEMBER FILE SO THAT TOKEN IS ADDED IN THE FILE
if(flag= =0)
{
if(m.ret_token()= =1)
{
while(f.read((char*)&m1,sizeof(m1)))
{
  if(m1.ret_memID()= =m.ret_memID())
  {
   posi=f.tellg()/sizeof(m1);
   f.seekg((posi-1)*sizeof(m1),ios::beg);
   f.write((char*)&m,sizeof(m));
  }
}
}
}
f.close();
fin.close();
}
```

```
/***Function to return a book***/
void return_books()
{
cout << "~~~~~~~RETURNING BOOKS~~~~~~~" << endl;
int id,flag=-1,posi;
char mem, bk[20];
int a;
cout << "Enter if a member is returning book; Y for yes, N for no: ";
  cin >> mem;
Book b;
Member m,m1;
fstream f("member_data.dat",ios::binary|ios::in|ios::out);
fstream fin("Book_data.dat", ios::binary|ios::in|ios::out);
if (mem = = 'y' || mem = = 'Y')
{
  cout << "\nEnter member ID: ";
    cin >> id;
  if (Mem_valid(m, id) = = 1)
  {
  if(m.ret_token()= =1)
```

```
{
 cout << "\nRETURNING OF BOOK BY " << m.ret_name() << endl;
cout << "\nEnter the name of the book to be returned: ";
      cin.ignore();
      gets(bk);
if(Book_valid(b,bk)= =1)
{
if(strcmp(bk,m.book_iss())= =0)
{
      m.retoken();
      while (fin.read((char*)&b, sizeof(b)))
       {

         if (strcmp(b.ret_bkname(), bk) = = 0)
          {
          if (b.issue = = 'I')
           {
                b.issue='N';
                posi=fin.tellg()/sizeof(b);
                fin.seekg(((posi-1)*sizeof(b)),ios::beg);
```

```
                fin.write((char*)&b,sizeof(b));
               flag=0;
             }
            else
            flag=1;
            }
          }
           if(flag= =0)
            cout << "Book has been returned!"<<endl;
   }
   else
    cout<<"This book is not issued to this member! \n"<<m.ret_name()<<"
has not returned "<<m.book_iss()<<endl;
   }
   else
   cout<<"Book does not exist! "<<endl;
   }
  else
  cout <<"No books in this member's name!"<<endl;
  }
  else
  cout<<" Member ID does not match! "<<endl;
}
```

```
else
cout<<"Not eligible for issue or return  "<<endl;
//TO WRITE INTO MEMBER FILE SO THAT TOKEN IS RESET IN THE FILE
if(flag= =0)
{
   if(m.ret_token()= =0)
    {
while(f.read((char*)&m1,sizeof(m1)))
{
  if(m1.ret_memID()= =m.ret_memID())
  {
   posi=f.tellg()/sizeof(m1);
   f.seekg((posi-1)*sizeof(m1),ios::beg);
   f.write((char*)&m,sizeof(m));
  }
}
    }
}
f.close();
fin.close();
}
```

/***Function to display all books stored in file***/

```
void disbooks()

{

ifstream f("Book_data.dat", ios::binary);

Book b;

while (f.read((char*)&b, sizeof(b)))

        {

        b.putB();

        getch();

        }

f.close();

}
```

```
/***Function to display all members stored in file***/

void dismembers()

{

ifstream f("member_data.dat", ios::binary);

Member m1;

while (f.read((char*)&m1, sizeof(m1)))

        {

        m1.putM();

        getch();

        }

f.close();

}
```

```
/***Function to delete a particular book from file***/
void delete_book()
{
        ifstream f;
        ofstream f1;
        f.open("Book_data.dat", ios::binary);
        f1.open("temp.dat", ios::binary|ios::app);
        Book b,b2;
        char a[25];int fl=0;
        cout << "Enter the ISBN of the book: ";
        cin.ignore();
        gets(a);
        if(Book_valid(b2,a)= =1)
        {
                if(b2.issue= ='N')
                        {
        while (f.read((char*)&b, sizeof(b)))
        {
        if (strcmp(a, b.ret_ISBN()) != 0)
        f1.write((char*)&b, sizeof(b));
        else
```

```
cout<<"Details of book to be deleted: \n";b.putB();
}
f.close();
f1.close();
remove("Book_data.dat");
rename("temp.dat", "Book_data.dat");
                }
        else
cout<<"This book is currently in issue! After return, you may delete it from database \n";
}
else
fl=1;
if(fl= =1)
cout<<"Book does not exist! "<<endl;
else if(fl= =0 && b2.issue= ='N')
cout<<"Book has been deleted"<<endl;
                                        }
```

```
/***Function to delete a particular member from file***/

void delete_member()

{

        ifstream f;

        ofstream f1;

        f1.open("temp.dat", ios::binary);

        Member m,m2;

        int id,fl;

        f.open("member_data.dat", ios::binary);

        cout << "Enter the member id : ";

        cin >> id;

        if(Mem_valid(m2,id)= =1)

        {

           if(strcmp(m2.book_iss(),'\0')= =0)

            {

                while (f.read((char*)&m, sizeof(m)))

                {

                   if (id != m.ret_memID())

                         f1.write((char*)&m, sizeof(m));
```

```
                else
                    {cout<<"Details of member to be deleted: \n";m.putM();}
                }
        f.close();
        f1.close();
        remove("member_data.dat");
        rename("temp.dat", "member_data.dat");
        }
           else
        cout<<"Books are issued to member! \n After returning, you may
delete record \n";
        }
        else
        fl=1;
if(fl= =1)
cout<<"Member does not exist! "<<endl;
else if(strcmp(m2.book_iss(),'\0')= =0 && fl!=1)
cout<<"Record has been deleted "<<endl;
}
```

```
/***Function to check if a Book exists in file***/
int Book_valid(Book &B, char bk[])
{
        ifstream f("Book_data.dat",ios::binary);
        Book b;
        int fl=0;
        while(f.read((char*)&b,sizeof(b)))
        {
          if((strcmp(b.ret_bkname(),bk)= =0)||(strcmp(b.ret_ISBN(),bk)= =0))
                {
                B=b;
                fl=1;
                }
        }
        f.close();
        return fl;
}
```

```
/***Function to check authenticity of member ID***/
int memID_authentic(int m_check)
{
	ifstream f("member_data.dat", ios::binary);
	Member mfile;
	int unique = 1;
	while (f.read((char*)&mfile, sizeof(mfile)))
	{
		if (mfile.ret_memID() = = m_check)
			unique = 0;
	}
	f.close();
	return unique;
}
```

```
/***Function to check if a member exists in file***/
int Mem_valid(Member &m1, int ID)
{
        ifstream fin("member_data.dat", ios::binary);
        Member m;
        int f = 0;
        while (!fin.eof())
        {
        fin.read((char*)&m, sizeof(m));
                if (m.ret_memID() = = ID)
                  {
                m1 = m;
                f = 1;
                  }
        }
        fin.close();
        return f;
}
```

```
/***Function to modify a specific field for a pass-by-reference Member***/
int mem_modify(Member &m)
{
        int opt,id; char name[25];
      cout<<"Enter 1 to change member ID;\t 2 to change name of member: ";
        cin>>opt;
        if(opt= =1)
        {
        cout << "\nEnter member ID: ";
                cin >> id;
        if(memID_authentic(id)= =0)
                cout<<"\nThe member ID already exists in file! \n";
                m.set_memID(id);
        }
        else if(opt= =2)
        {
        cout << "\nEnter name of member: ";
        cin.ignore();
        gets(name);
        m.set_name(name);
        }
        return memID_authentic(id);
}
```

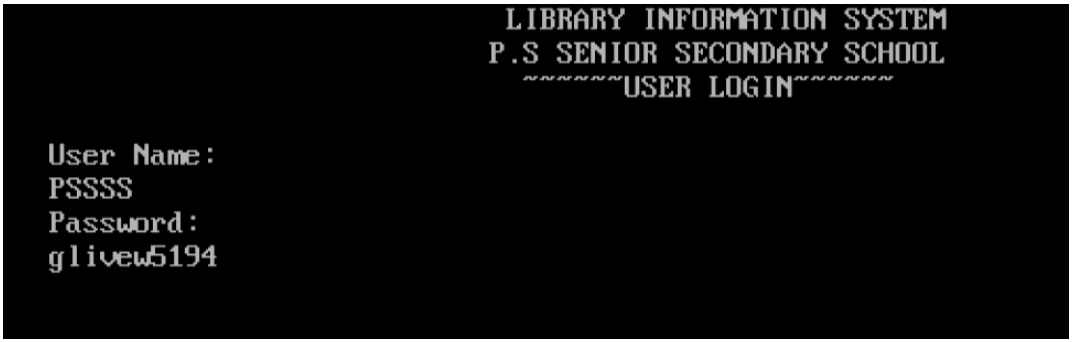
```
/***Function to modify specific field for a pass-by-reference Book***/
int book_modify(Book &b)
{
int opt;
char name[25],isbn[25],auth[25];
cout<<"Enter 1 to change name of book;    2 to change author's name; 3 to change ISBN: ";
cin>>opt;
        if(opt= =1)
          {
cout<<"\nEnter the name of the book: ";
cin.clear();
gets(name);
b.set_name(name);
          }
        else if(opt= =2)
          {
cout << "\nEnter author's name: ";
cin.clear();
gets(auth);
b.set_auth(auth);
          }
        else if(opt= =3)
          {
cout<<"\nEnter new ISBN: ";
```

```
cin.clear();
gets(isbn);
if(Book_valid(b,isbn)= =1)
cout<<"\nISBN already exists in file! "<<endl;
b.set_ISBN(isbn);
        }
return Book_valid(b,isbn);
}
```

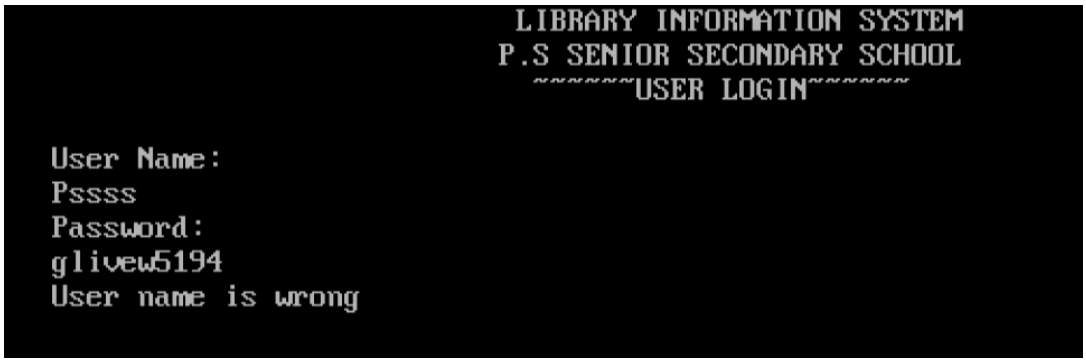
# 4.0 OUTPUT SCREENS OF PROGRAM

## LIBRARIAN_LOGIN()

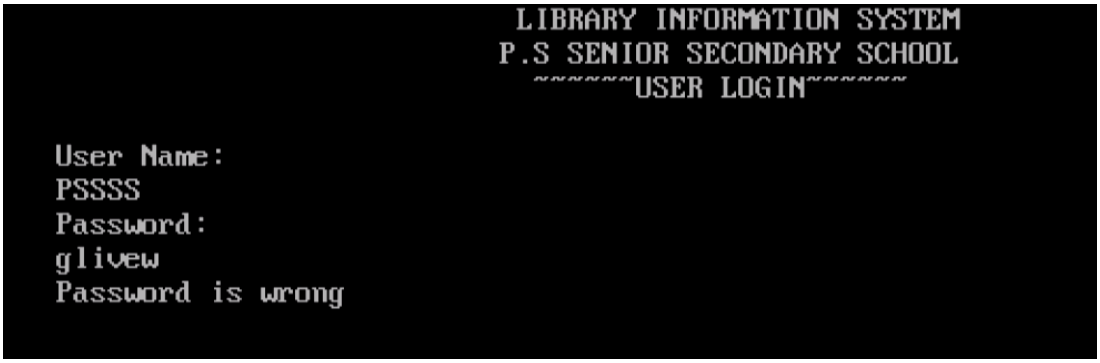
User login screen




Validation: user name authentication fails

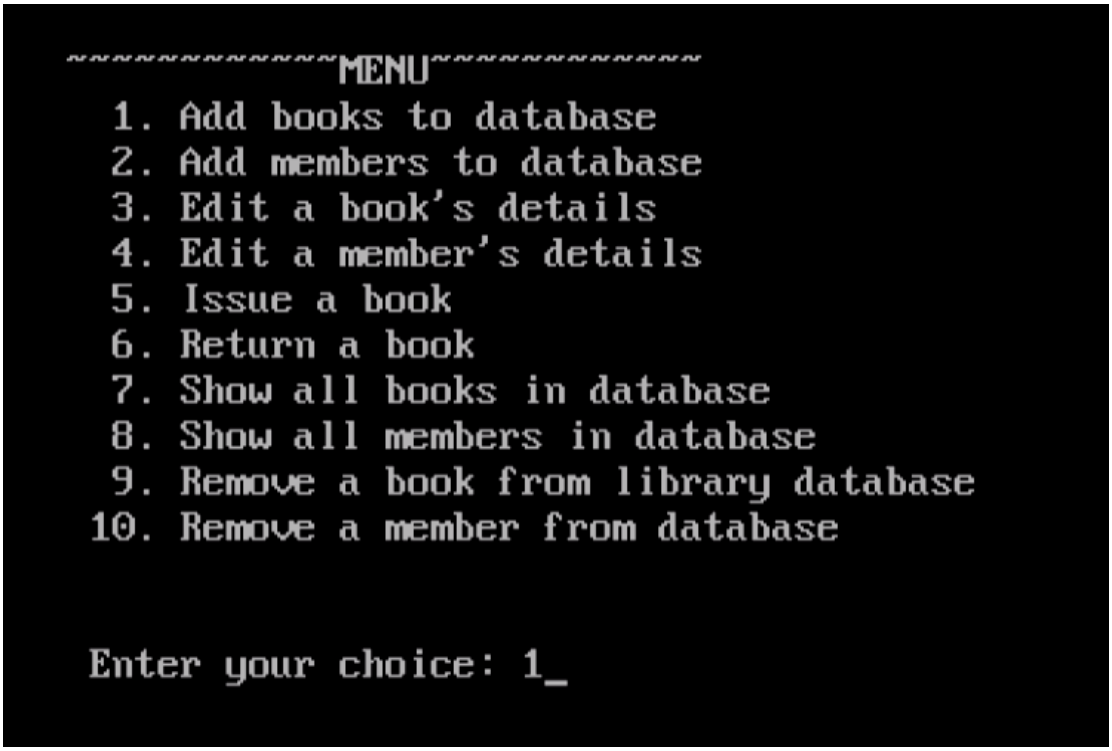



Validation: password authentication fails



# menu()

Main menu screen



# Book_database()

Input screen for book details showing entered correct details

```
Enter the name of the book: Stars
Enter the ISBN given: 9780539753647
Enter author's name: Robin Dexter
Enter whether book is issued or not: N
Do you want to store another book's details? Y for yes: n

Do you want to open main menu again? Y for yes, N for no:
```

Records created and stored in BOOK_DATA.DAT file

```
BOOK NAME              ISBN          AUTHOR        ISSUE
 Stars           9780439753647   Robin Dexter      N
 Cheerful        9781405274739   R. Hargreaves     N
 Becoming        9780241334140   M. Obama          N
 Illusions       9780743278904   Richard Bach      N
 Matilda         9780141346342   Raold Dahl        N
 Timekeeper      9780751541182   Mitch Albom       N
 Hotel           9798129108004   Arthur Hailey     N
 Paperweight     9780099457022   Stephen Fry       N
 Simple Korean   9781497445826   Go Billy          N
 Daydream        9781405235778   R. Hargreaves     N

Do you want to open main menu again? Y for yes, N for no: _
```

Validation: duplicate ISBN fails

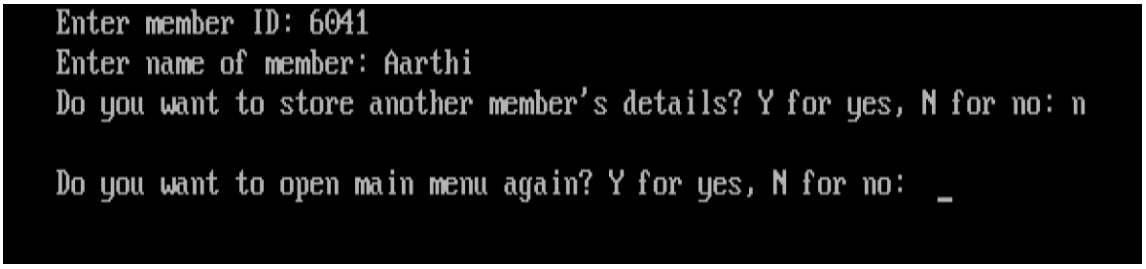
```
Enter the name of the book: No Safe Place
Enter the ISBN given: 9780439753647
Enter author's name: R. Patterson
Enter whether book is issued or not: N
This book already exists in file!
Do you want to store another book's details? Y for yes: n

Do you want to open main menu again? Y for yes, N for no:
```

Validation: duplicate book name fails

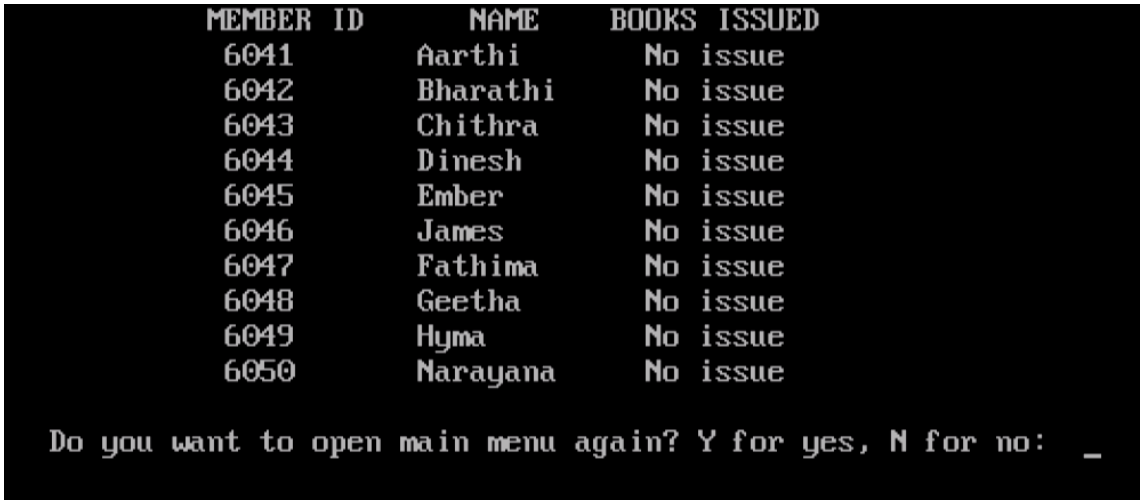
```
Enter the name of the book: Becoming
Enter the ISBN given: 9780099175322
Enter author's name: M. Obama
Enter whether book is issued or not: N
This book already exists in file!
Do you want to store another book's details? Y for yes: n

Do you want to open main menu again? Y for yes, N for no: _
```

# member_database()

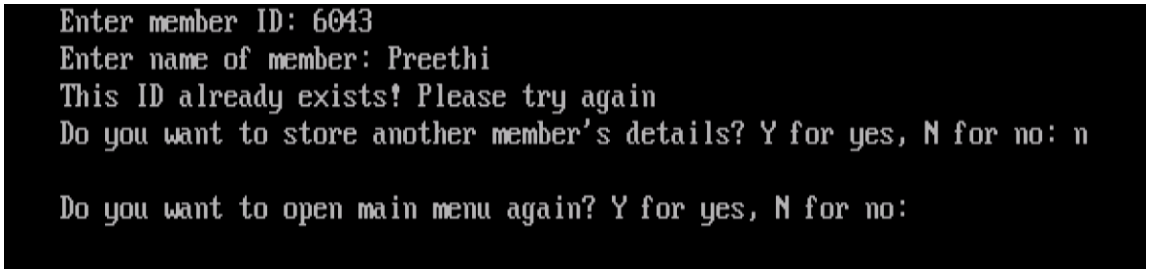
Input screen for member details showing entered correct details




Records created and stored in MEMBER_DATA.DAT file




Validation: duplicate member ID fails



# rewrite_book()

## Book database before modification

(*NOTE:* Book name: Becoming; ISBN 9780241**32**3**3**4140 wrong; correct ISBN: 9780241**33**4140)

```
          BOOK NAME              ISBN            AUTHOR    ISSUE
           Stars           9780439753647    Robin Dexter      N
           Cheerful        9781405274739    R. Hargreaves     N
           Becoming        97802413234140          M.Obama             N
           Illusions       9780743278904    Richard Bach      N
           Matilda         9780141346342    Raold Dahl        N
           Timekeeper      9780751541182    Mitch Albom       N
           Hotel           9798129108004    Arthur Hailey     N
           Paperweight     9780099457022    Stephen Fry       N
           Simple Korean   9781497445826    Go Billy          N
           Daydream        9781405235778    R. Hargreaves     N

 Do you want to open main menu again? Y for yes, N for no:
```

## Screen for modifying book details

(There are options for editing book name, author name or ISBN of book records)

```
 Enter ISBN of book that has to be edited: 97802413234140

 Old book details:
          Becoming        97802413234140          M. Obama          N

 For record modification:
 Enter 1-change name of book, 2-change author's name, 3-change ISBN:  3

 Enter new ISBN: 9780241334140

 Book has been edited

 Do you want to open main menu again? Y for yes, N for no:
```

## Book database after modification

```
          BOOK NAME              ISBN            AUTHOR    ISSUE
           Stars           9780439753647    Robin Dexter      N
           Cheerful        9781405274739    R. Hargreaves     N
           Becoming        9780241334140    M.Obama           N
           Illusions       9780743278904    Richard Bach      N
           Matilda         9780141346342    Raold Dahl        N
           Timekeeper      9780751541182    Mitch Albom       N
           Hotel           9798129108004    Arthur Hailey     N
           Paperweight     9780099457022    Stephen Fry       N
           Simple Korean   9781497445826    Go Billy          N
           Daydream        9781405235778    R. Hargreaves     N

 Do you want to open main menu again? Y for yes, N for no:
 _
```

Validation: Editing a book's ISBN to an existing one fails

```
Enter ISBN of book that has to be edited: 9780751541182

Old book details:
          Timekeeper      9780751541182   Mitch Albom      N

For record modification:
Enter 1-change name of book, 2-change author's name, 3-change ISBN:  3

Enter new ISBN: 9780439753647

ISBN already exists in file!

Do you want to open main menu again? Y for yes, N for no:
```

Validation: Editing a book's name to an existing one fails

```
Enter ISBN of book that has to be edited: 9780439753647

Old book details:
          Stars           9780439753647   Robin Dexter     N

For record modification:
Enter 1-change name of book, 2-change author's name, 3-change ISBN:  1

Enter the name of the book: Timekeeper

Book name already exists in file!

Do you want to open main menu again? Y for yes, N for no:  _
```

Validation: Editing a non-existent book fails

```
Enter ISBN of book that has to be edited: 9780741451183

Book doesn't exist!

Do you want to open main menu again? Y for yes, N for no:
```

# rewrite_mem()

Member database before modification

```
MEMBER ID      NAME     BOOKS ISSUED
 6041        Aarthi       No issue
 6042        Bharathi     No issue
 6043        Chithra      No issue
 6044        Dinesh       No issue
 6045        Ember        No issue
 6046        James        No issue
 6047        Fathima      No issue
 6048        Geetha       No issue
 6049        Hyma         No issue
 6050        Narayana     No issue

Do you want to open main menu again? Y for yes, N for no:  _
```

Screen for modifying member details

(There are options for editing name or member ID)

```
Enter ID that has to be edited: 6045

Old Member details:
          6045       Ember        No issue

For record modification:
Enter 1-change member ID, 2-change name of member:  1

Enter member ID: 6051

Record has been edited

Do you want to open main menu again? Y for yes, N for no:  _
```

Member database after modification

```
MEMBER ID      NAME     BOOKS ISSUED
 6041        Aarthi       No issue
 6042        Bharathi     No issue
 6043        Chithra      No issue
 6044        Dinesh       No issue
 6051        Ember        No issue
 6046        James        No issue
 6047        Fathima      No issue
 6048        Geetha       No issue
 6049        Hyma         No issue
 6050        Narayana     No issue

Do you want to open main menu again? Y for yes, N for no:
```

Validation: Editing a member's ID to an existing one fails

```
Enter ID that has to be edited: 6042

Old Member details:
          6042       Bharathi     No issue

For record modification:
Enter 1-change member ID, 2-change name of member:  1

Enter member ID: 6050

The member ID already exists in file!

Do you want to open main menu again? Y for yes, N for no:  _
```

Validation: Editing a non-existent member fails

```
Enter ID that has to be edited: 6055

Member does not exist

Do you want to open main menu again? Y for yes, N for no:
```

# Book_Issue()

Screen for issue of books

```
Is this issue for a member? Y for yes, N for no: y
Enter member ID: 6042
ISSUE FOR Bharathi
Enter the name of the book taken: Hotel
Books have been issued!  Please note the date of book
 Must return in 10 days

Do you want to open main menu again? Y for yes, N for no: _
```

Book record modified after book is issued

| BOOK NAME | ISBN | AUTHOR | ISSUE |
|---|---|---|---|
| Stars | 9780439753647 | Robin Dexter | N |
| Cheerful | 9781405274739 | R. Hargreaves | N |
| Becoming | 9780241334140 | M.Obama | N |
| Illusions | 9780743278904 | Richard Bach | N |
| Matilda | 9780141346342 | Raold Dahl | N |
| Timekeeper | 9780751541182 | Mitch Albom | N |
| Hotel | 9798129108004 | Arthur Hailey | I |
| Paperweight | 9780099457022 | Stephen Fry | N |
| Simple Korean | 9781497445826 | Go Billy | N |
| Daydream | 9781405235778 | R. Hargreaves | N |

```
Do you want to open main menu again? Y for yes, N for no: _
```

Member record modified after book is issued

| MEMBER ID | NAME | BOOKS ISSUED |
|---|---|---|
| 6041 | Aarthi | No issue |
| 6042 | Bharathi | Hotel |
| 6043 | Chithra | No issue |
| 6044 | Dinesh | No issue |
| 6051 | Ember | No issue |
| 6046 | James | No issue |
| 6047 | Fathima | No issue |
| 6048 | Geetha | No issue |
| 6049 | Hyma | No issue |
| 6050 | Narayana | No issue |

```
Do you want to open main menu again? Y for yes, N for no:
```

Validation: Issue of an already issued book fails

```
Is this issue for a member? Y for yes, N for no: y
Enter member ID: 6044
ISSUE FOR Dinesh
Enter the name of the book taken: Hotel
Sorry! Book is unavailable currently. please try again later

Do you want to open main menu again? Y for yes, N for no:  _
```

Validation: Issue of a non-existent book fails

```
Is this issue for a member? Y for yes, N for no: y
Enter member ID: 6044
ISSUE FOR Dinesh
Enter the name of the book taken: Examguru Physics
Book does not exist!

Do you want to open main menu again? Y for yes, N for no:
```

Validation: Issue of book to a member having book fails

```
Is this issue for a member? Y for yes, N for no: y
Enter member ID: 6042
ISSUE FOR Bharathi
Issued book Hotel has not been returned!

Do you want to open main menu again? Y for yes, N for no:  _
```

Validation: Issue to non-existent member fails

```
Is this issue for a member? Y for yes, N for no: y
Enter member ID: 6055
Member ID does not exist! Please try again

Do you want to open main menu again? Y for yes, N for no:
```

Validation: Issue allowed only for member

```
Is this issue for a member? Y for yes, N for no: n
This is a MEMBERS ONLY library!

Do you want to open main menu again? Y for yes, N for no:
```

# return_books()

Screen for return of books

```
~~~~~~~~RETURNING BOOKS~~~~~~~~
Enter if a member is returning book; Y for yes, N for no: y

Enter member ID: 6042

RETURNING OF BOOK BY Bharathi

Enter the name of the book to be returned: Hotel
Book has been returned!

Do you want to open main menu again? Y for yes, N for no: _
```

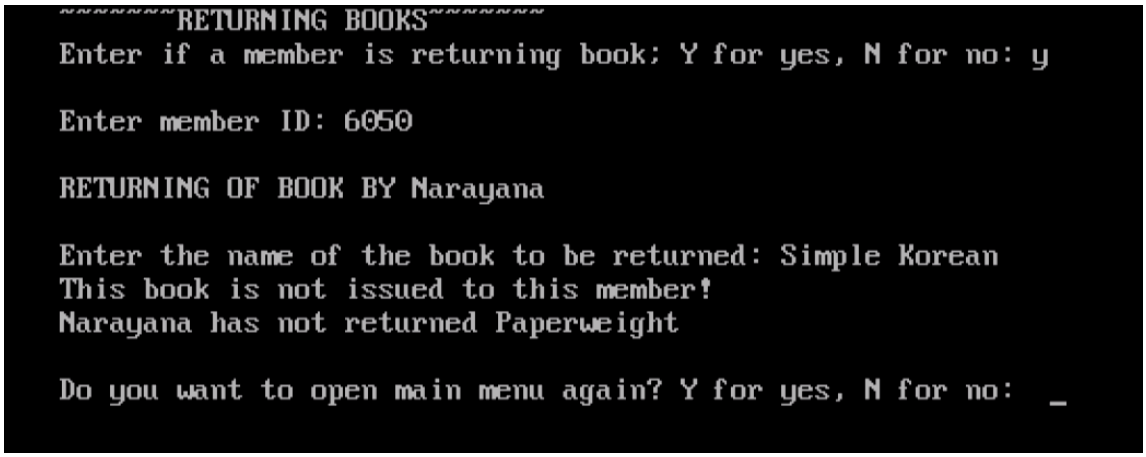
Book record modified after return

```
BOOK NAME              ISBN            AUTHOR      ISSUE
 Stars           9780439753647   Robin Dexter       N
 Cheerful        9781405274739   R. Hargreaves      N
 Becoming        9780241334140   M.Obama            N
 Illusions       9780743278904   Richard Bach       N
 Matilda         9780141346342   Raold Dahl         I
 Timekeeper      9780751541182   Mitch Albom        N
 Hotel           9798129108004   Arthur Hailey      N
 Paperweight     9780099457022   Stephen Fry        I
 Simple Korean   9781497445826   Go Billy           N
 Daydream        9781405235778   R. Hargreaves      N

Do you want to open main menu again? Y for yes, N for no:
```

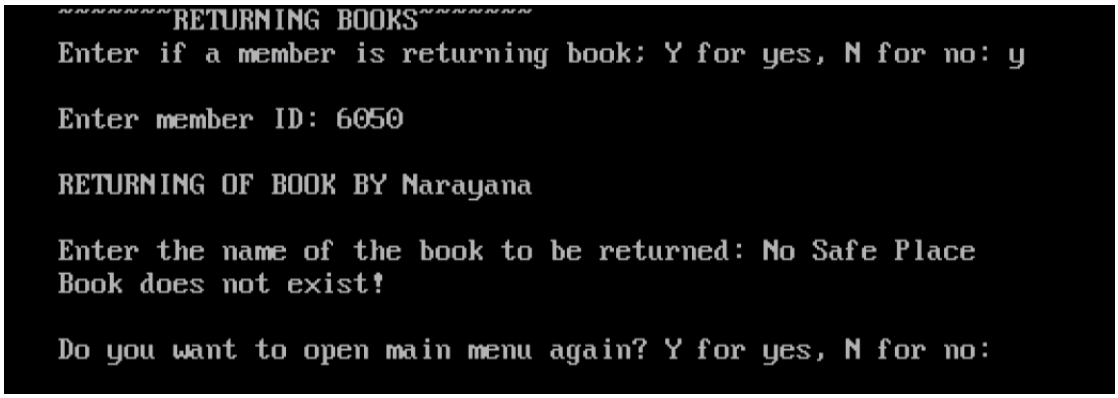
Member record modified after return

```
MEMBER ID      NAME     BOOKS ISSUED
 6041        Aarthi       No issue
 6042        Bharathi     No issue
 6043        Chithra      No issue
 6044        Dinesh       Matilda
 6051        Ember        No issue
 6046        James        No issue
 6047        Fathima      No issue
 6048        Geetha       No issue
 6049        Hyma         No issue
 6050        Narayana    Paperweight

Do you want to open main menu again? Y for yes, N for no:
_
```

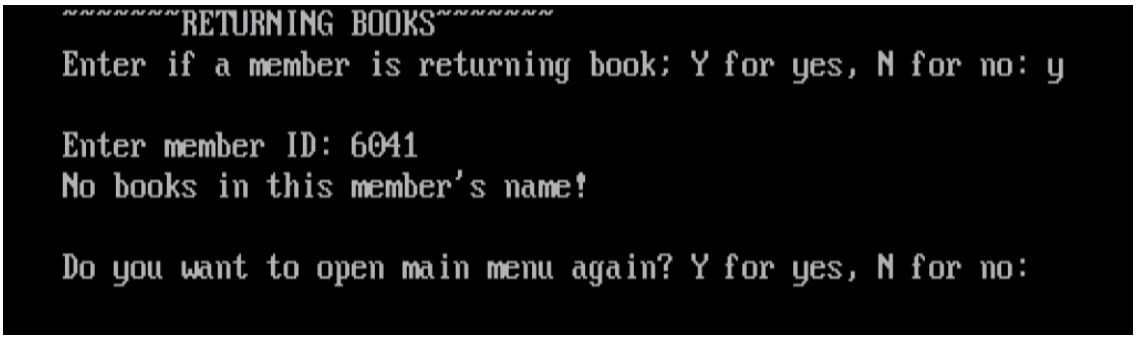
Validation: Return of wrong book fails



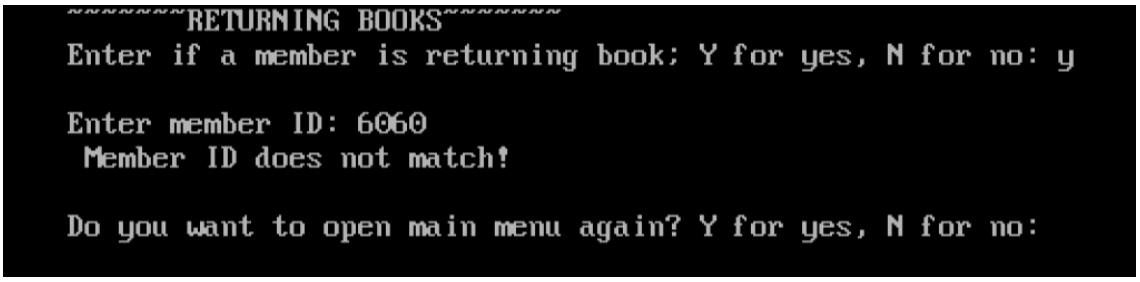

Validation: Return of non-existent book fails



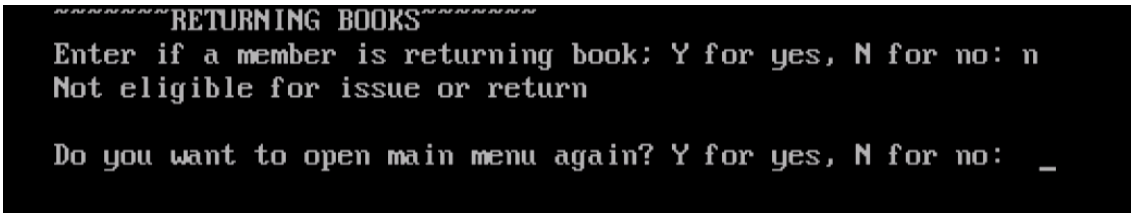

Validation: Return of book when no book is issued fails



Validation: Return of book by non-existent member




Validation: Return of books allowed only for members

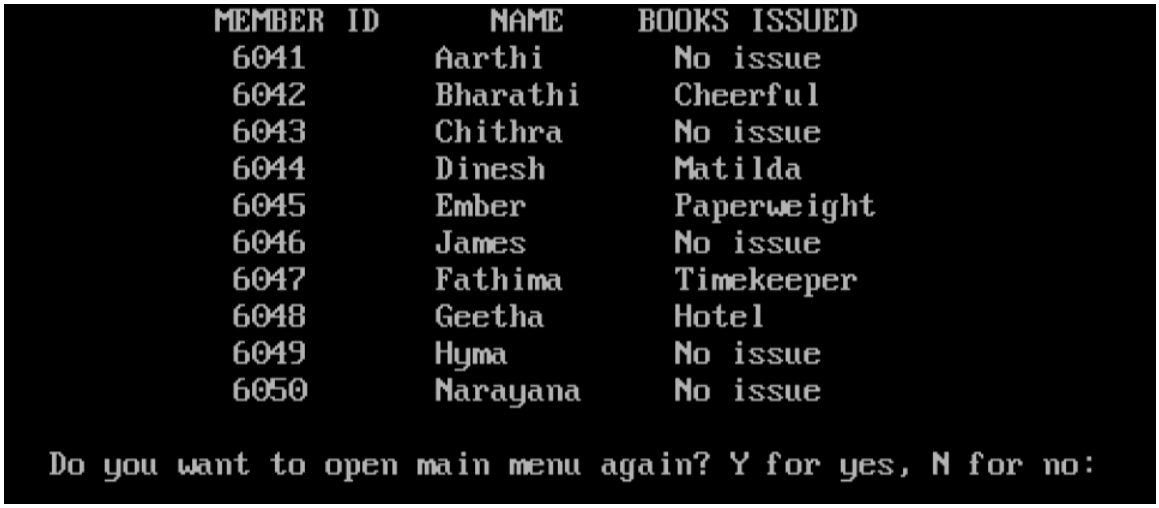


# disbooks()

Displayed books screen

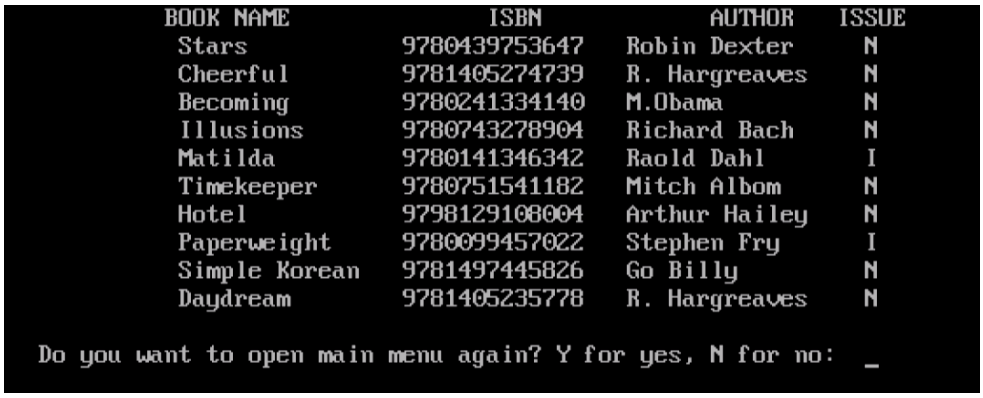
# dismembers()

Displayed members screen

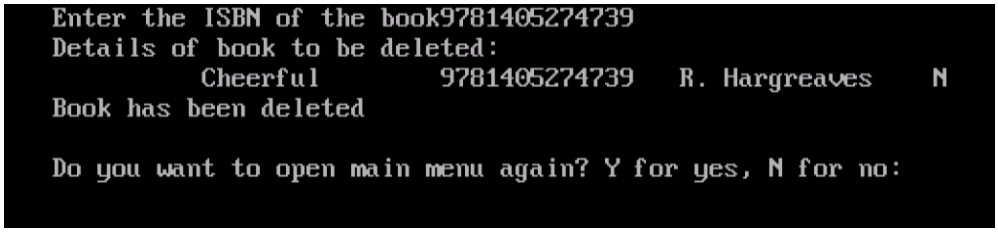


# delete_book()

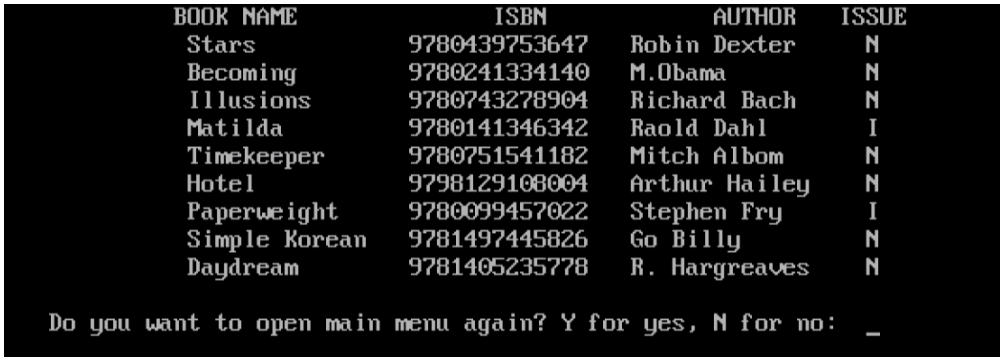
Book database before deletion of record



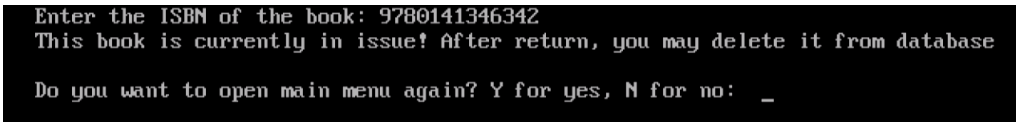

Deletion input screen



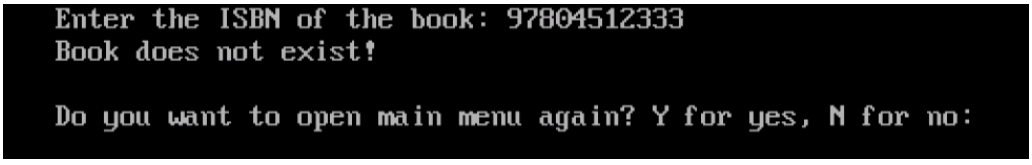

Book database after deletion of record




Validation: Deletion of issued book fails

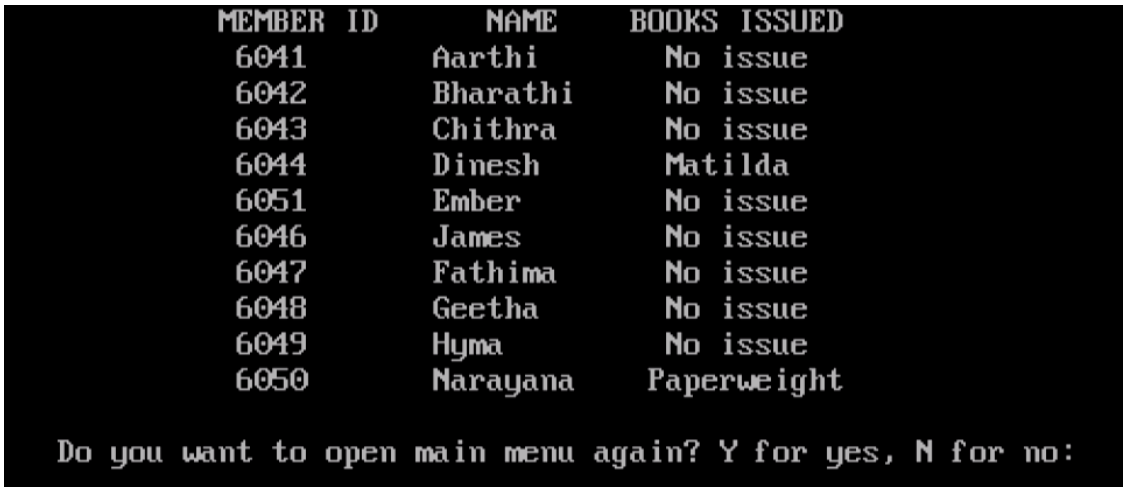



Validation: Deletion of non-existent book fails

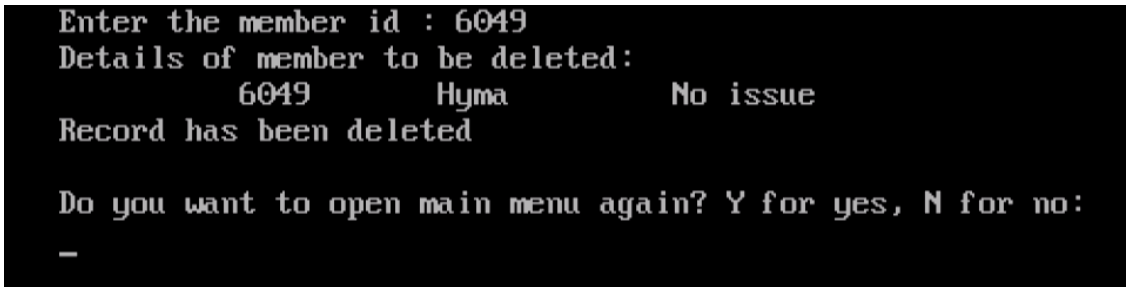


# delete_member()

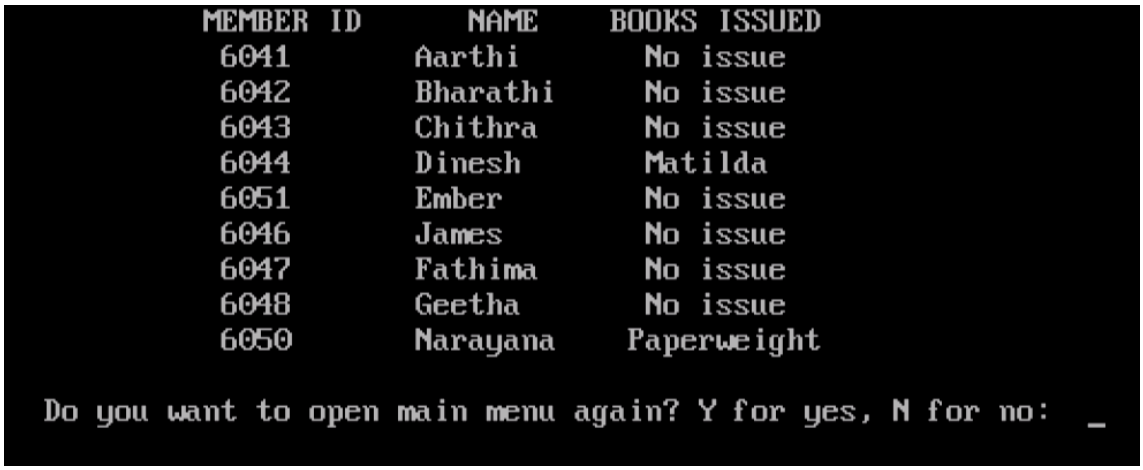
Member database before deletion



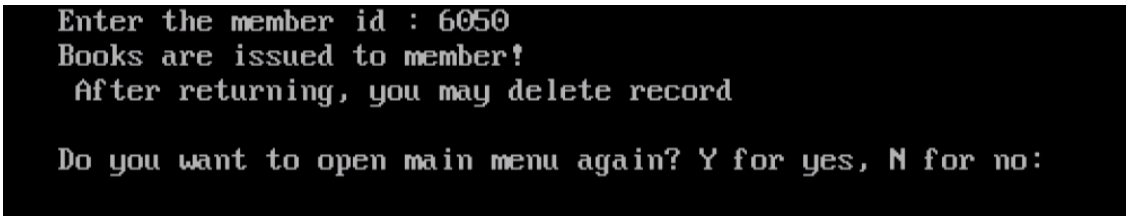

Deletion input screen



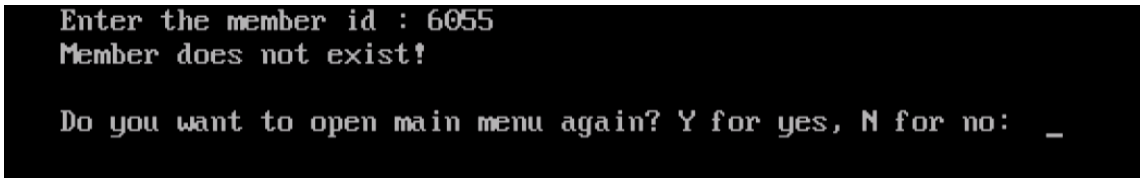

Member database after deletion



Validation: Deletion of member with issued book fails




Validation: Deletion of non-existent member fails



# 5.0 SCOPE OF IMPROVEMENT OF THE PROJECT

## Limitations

1. Sorting of records to be handled
2. Modification of more than a data member at a time to be handled
3. Unrestricted character length by using heap and pointers to be handled
4. Calculation & validation of due dates for issued books to be handled.

## Possible improvements to be incorporated

1. Providing member IDs for new members through automation
2. Searching and displaying books by the same author
3. Searching common words in book and author names
4. Handling screen display of a huge database of books and members
5. Inclusion of more details such as book price, member addresses and phone numbers etc.
6. Including friendly prompts at user interfaces.
7. Handling the issue and return of multiple books to the same member
8. Keeping track of the number of book copies that the library has

# 6.0 BIBLIOGRAPHY

Class notes

Arihant All in One Computer Science, Class 12

Websites:

stackoverflow.com
geeksforgeeks.org

# 7.0 CONCLUSION

The project titled Library Management done by

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

for the academic year 2019−2020 has been completed and compiled, tested and executed successfully.