LIBRARY MANAGEMENT SYSTEM

PROJECT ABSTRACT:

A library management program basically deals with all the operations performed in a library, ranging from the user's logging in to viewing the organisation of various books or other materials, which would have otherwise been very hectic for the users and librarians as well. So, in order to avoid the complexities and confusions, a Library Management System was devised.

In this project, we will be dealing with various operations, namely:

a) Welcome message.
b) Users' valid login.
c) Adding a book in the database.
d) Searching for a book.
e) Viewing a book.

f) Deleting a book from the database.
g) Updating the users' credentials.

It will be a menu driven program where one may perform the above operations after inputting a valid login username and password.

The programming language used here is 'C'. This program includes all the major concepts of 'C', i.e., structures, file handling, string, arrays, pointers, functions, looping (for, while) and conditional statements (if, if else, switch case).

MODULE WISE DESCRIPTION OF THE APPLICATION :

MODULE – 1 : WELCOME PAGE WITH MESSAGE

When entered into the application , A welcome screen is showed upon , saying – WELCOME TO LIBRARY MANAGEMENT SYSTEM. After that , it will ask to press any key from the keyboard to proceed.

MODULE – 2 : USER ACCOUNT AUTHENTICATION (LOGIN)

After pressing a key , the application will for ward us the Login page , asking for Username and Password. If You Entered the wrong credentials , thrice , it will say that you are an unknown user , else , failed to login , if once failed , else , move to the main menu.

MODULE – 3 : MAIN MENU

The Main Menu will offer a lot of options to operate the system. Like Adding the books name , searching books by name , deletion and

viewing , changing your credentials , and at last , exiting the application (all numbered from 1 to 5 and 0).

MODULE – 4 : ADDING BOOK(S)

When asked for an option to proceed , one might have to add books , thus pressed one (1). This will direct to another page , asking for the information of the book(s) to be added to the system. The fields required are as follows –

1. Book ID
2. Book Name
3. Author Name
4. Student Name
5. Date of adding.

After filling the necessary information , you are again directed to the main menu.

MODULE – 5 : SEARCH FOR THE BOOK

After going to the main menu , you decided to search for a book in the system , thus gave the option 2. This will direct you to the Search page.

Here the only thing asked will be – Name of the book. After entering the correct name of the book , you shall get all the information of that particular book. Then the application asked for pressing a key - that will direct you to the main menu.

MODULE – 6 : VIEW BOOK(S)

You also have the option to view all the books available in the database / system – providing full information about them. It also depicts a count value – showing how many books are available.

MODULE – 7 : BOOK DELETION

There can be a book or many books that wasn't used for a long time OR the book is no more available (physical presence) , thus , that has to be deleted from the system as well.

While deleting a book , the application will just ask for the Book ID , after entering that , the book information will get deleted and you will be directed to the main menu.

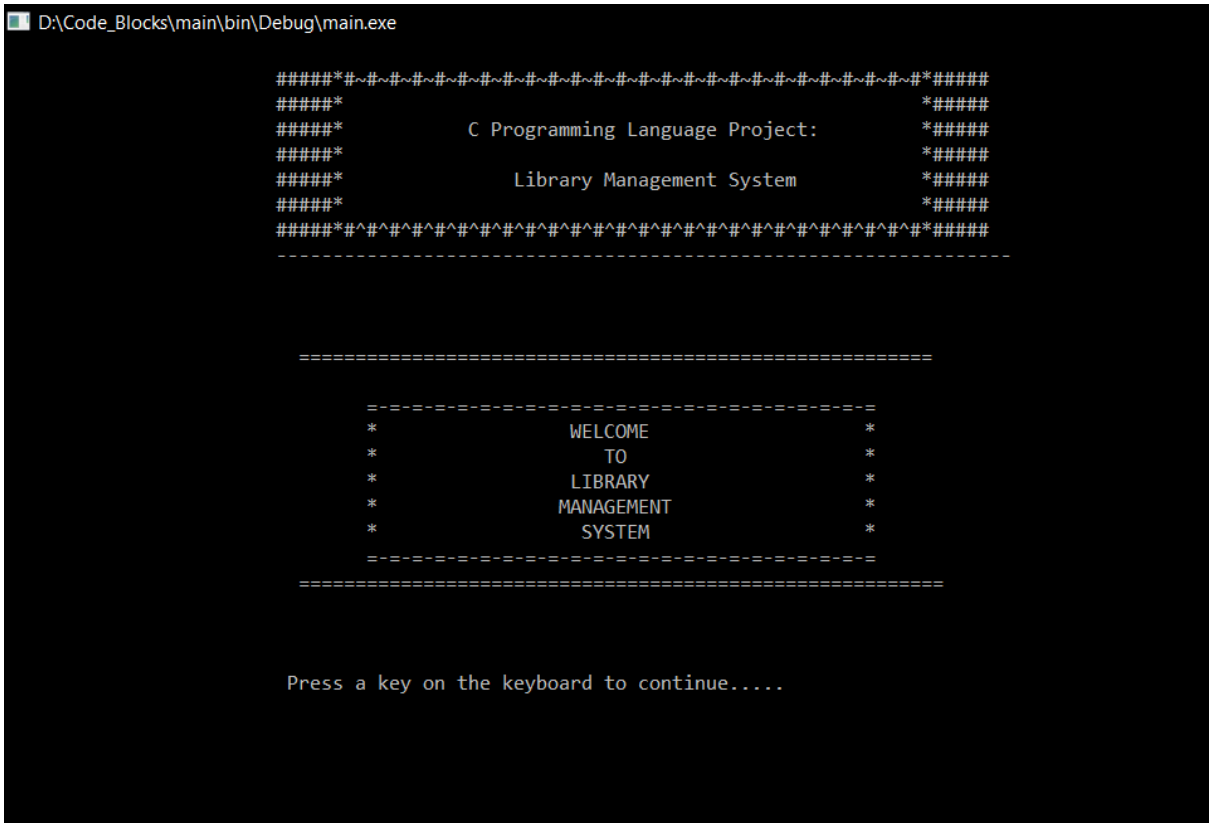
MODULE – 8 : PASSWORD UPDATION

Many a times , you may forget your password , thus the main menu also provides an option for password updation. One might have to enter the username , after that password change can occur. After the task is done , the application will ask to enter a key – again back to the main menu.

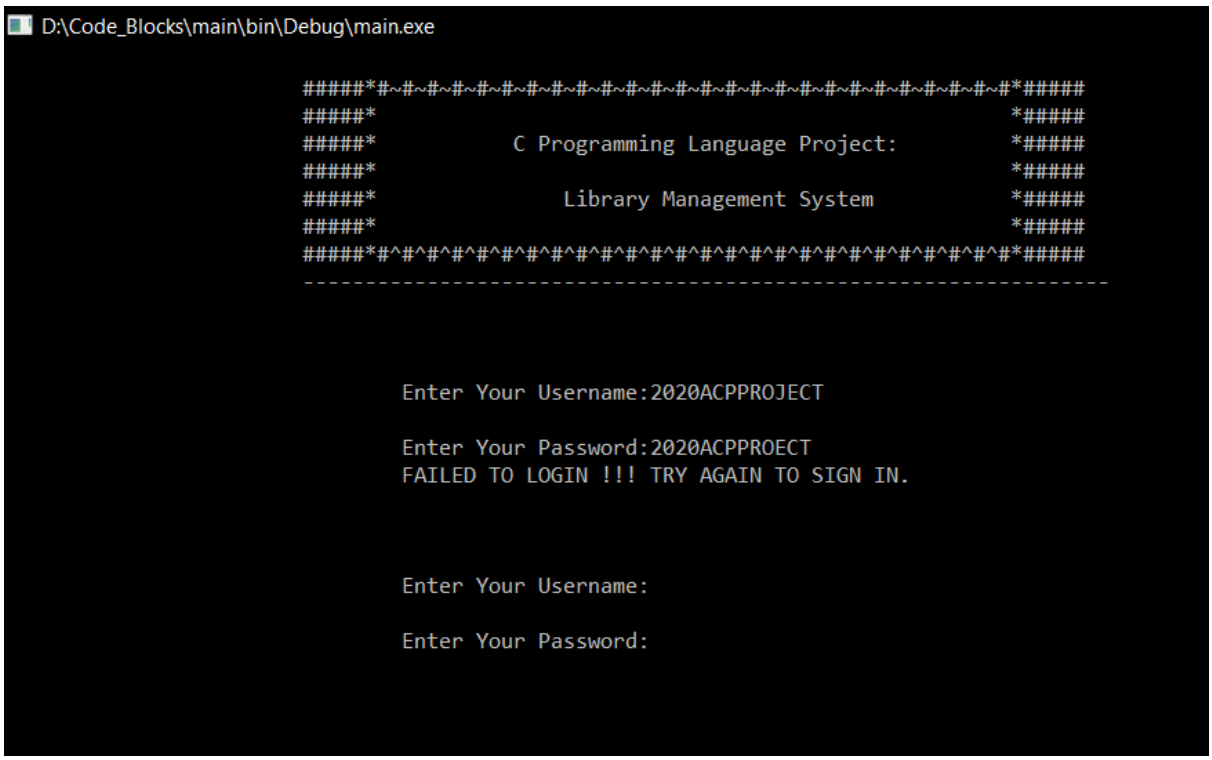
MODULE – 9 : EXIT THE APPLICATION

Last but not the Least , one might have to exit from the application after his/her task is done – this option , will end your access to the application , but you can come back via re-login.

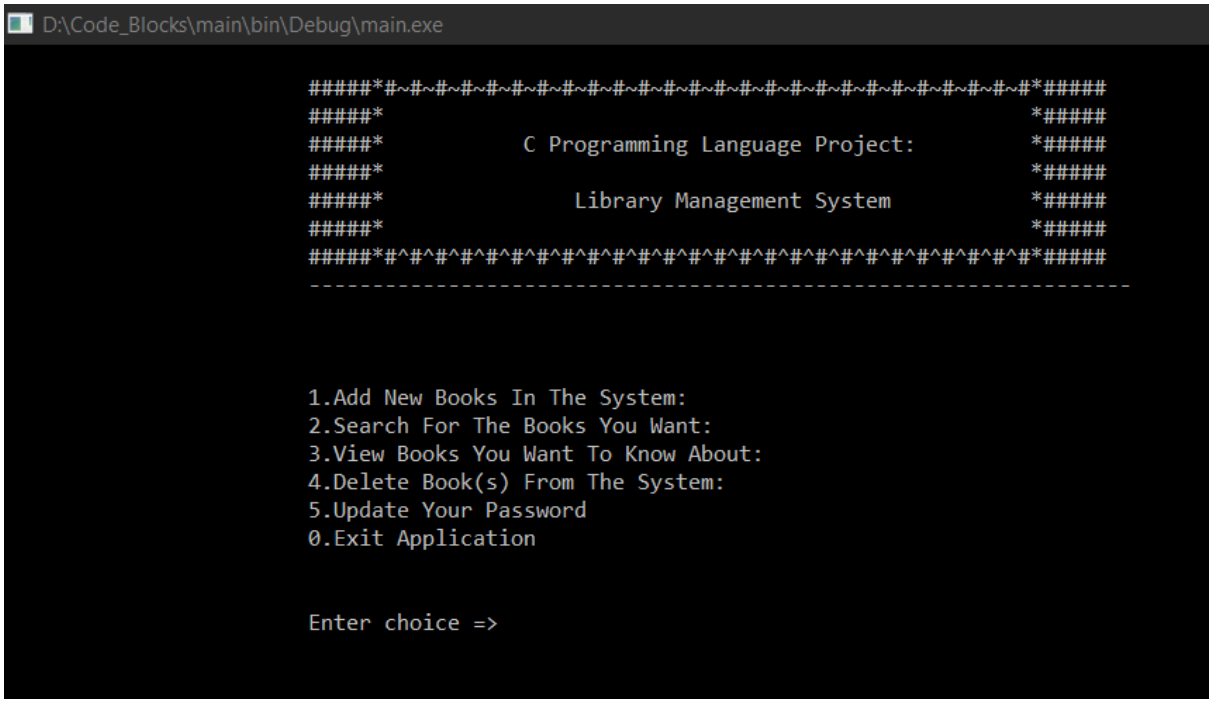
OUTPUT SCREENSHOTS



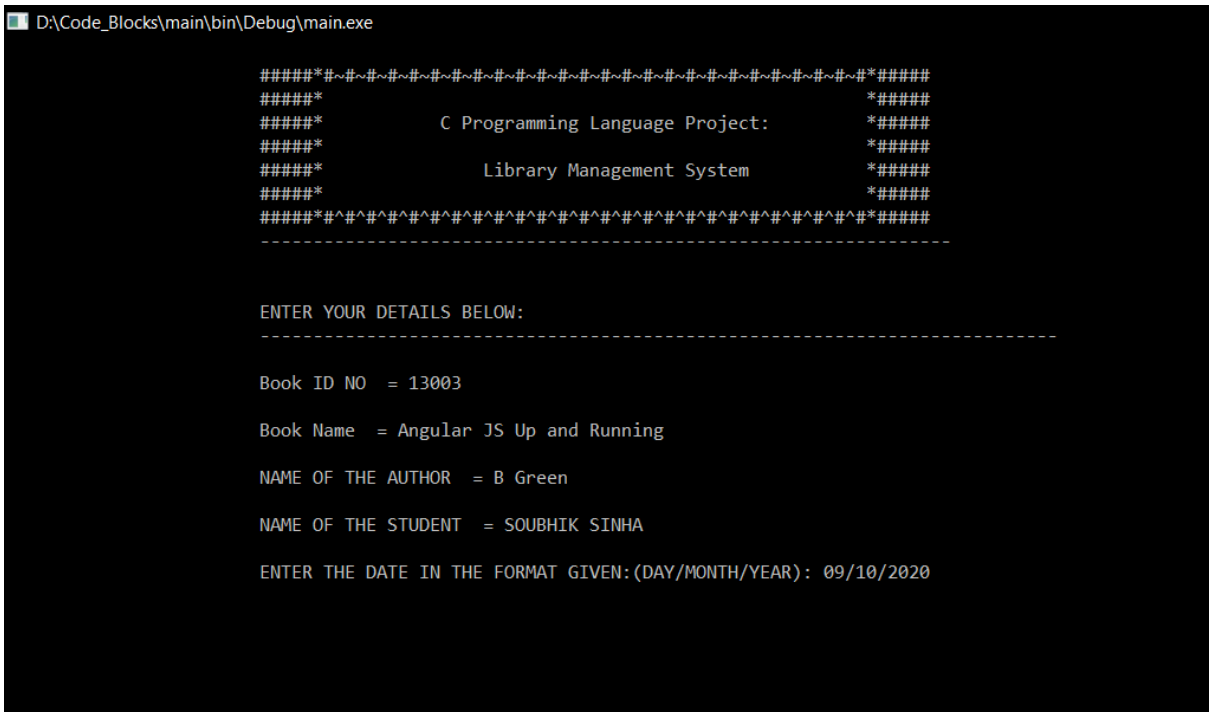
WELCOME SCREEN WITH MESSAGE



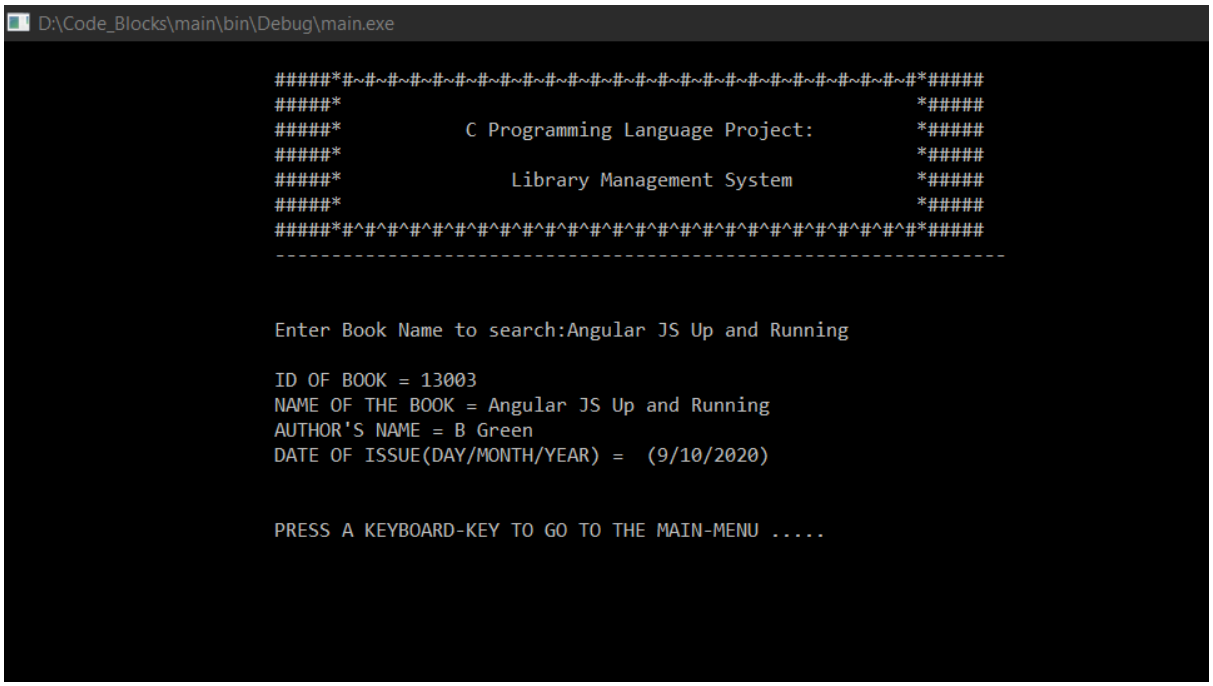
USER ACCOUNT AUTHENTICATION (LOGIN)



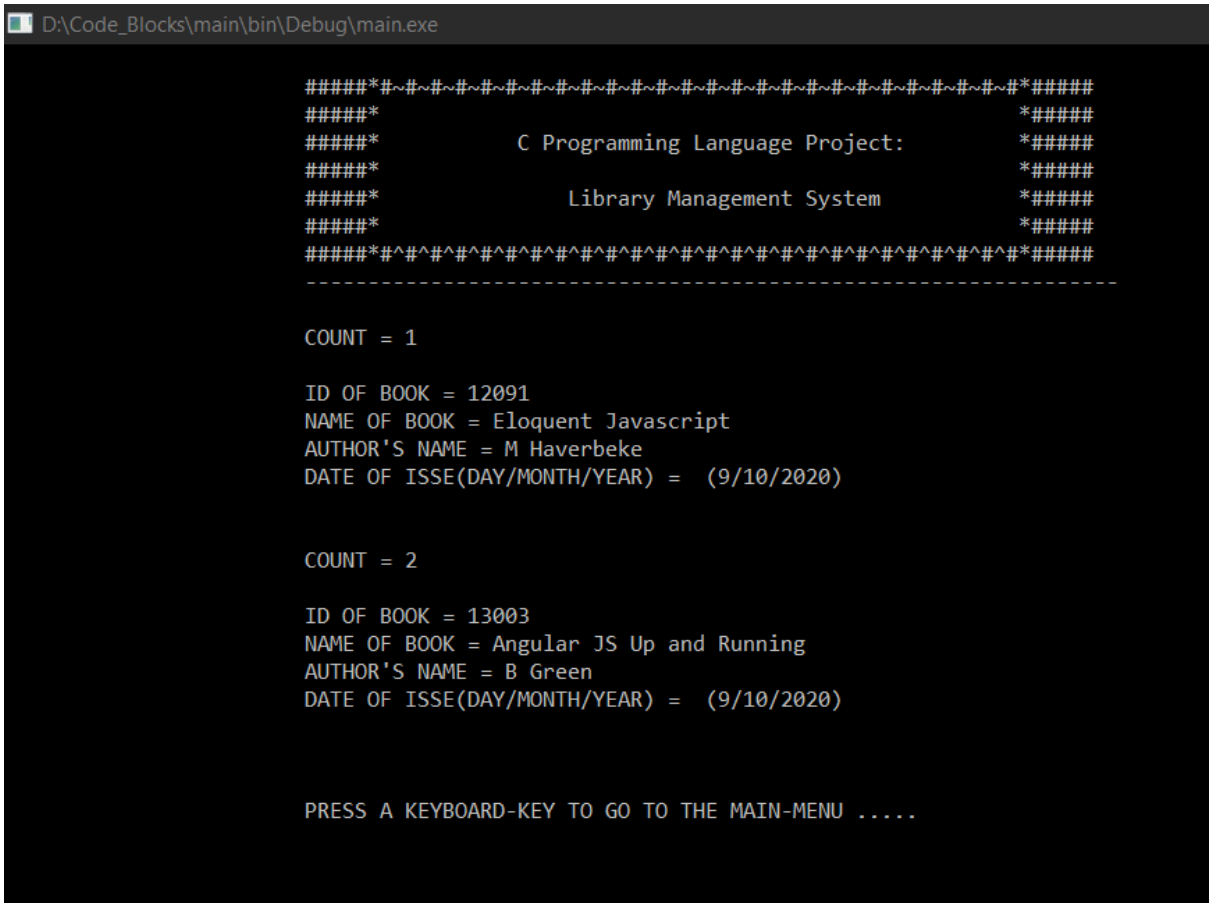
MAIN MENU



ADDING A BOOK INFORMATION



SEARCHING FOR A BOOK



VIEWING VARIOUS BOOK INFORMATION ENETERED IN THE SYSTEM

PROJECT

LIBRARY MANAGEMENT SYSTEM

CODE :

```
/*
  TOPIC ==> LIBRARY MANAGEMENT SYSTEM
  DATE ==> 04/10/2020
*/

#include <stdio.h>
#include <time.h>
#include <string.h>
#define YEAR_MIN  1900 //minimum range of year taken
#define YEAR_MAX  9999 //maximum range of the year taken
#define PWD_MAX_SIZE  20 //maximum password size
#define USERNAME_MAX_SIZE 30 //maximum user-name size
#define FILE_NAME  "Libmngtsys.bin"

//Book Information related Macro(s)
#define BOOKNAME_SIZE_MAX   60 //maximum size of the Book Name
#define AUTHORNAME_SIZE_MAX 60 //maximum size of the Author Name
#define STUNAME_MAX_SIZE 60 //maximum size of the Student Name
```

```
#define STUADDR_MAX_SIZE 200 //maximum size of the Student Address
#define FILE_HEADER_SIZE  sizeof(credential)

//Date Acceptance || Storage Structures
typedef struct  // structure for date
{
   int yyyy; //four digit Year Format
   int mm;  //two digit Month Format
   int dd; //two digit Date Format
} Date;
typedef struct // structure for inputting login credentials
{
   char username[USERNAME_MAX_SIZE];
   char password[PWD_MAX_SIZE];
} credential;
typedef struct// calling within the program (from any function)
{
   unsigned int books_id; // declaring integer data type
   char bookName[BOOKNAME_SIZE_MAX];// declaring character data type
   char authorName[AUTHORNAME_SIZE_MAX];// declaring character data type
   char studentName[STUNAME_MAX_SIZE];// declaring character data type
```

```
   char studentAddr[STUADDR_MAX_SIZE];// declaring character
data type
   Date bookIssueDate;// declaring integer data type
} BooksInfo;
void messageOP(const char* msg) //msg => message
{
   int len =0;
   int pos = 0;
   //space needed for printing has to be specified
   len = (78 - strlen(msg))/2;
   printf("\t\t\t");
   for(pos =0 ; pos < len ; pos++)
   {
      //space
      printf(" ");
   }
   //message
   printf("%s",msg);
}
void messageTop(const char *msg)
{
   system("cls");
   printf("\n\t\t\t# +#*                          #~#~#~#~#~#~#~#~#~#*#+
#");
```

```
   printf("\n\t\t\t## #*                                    *# ##");
   printf("\n\t\t\t##+ *       ADVANCED C PROGRAMMING - PROJECT
2020      * +##");
   printf("\n\t\t\t## #*                                    *# ##");
   printf("\n\t\t\t# +#*            Library Management System          *#+
#");
   printf("\n\t\t\t## #*                                    *# ##");
   printf("\n\t\t\t## #*#^#^#^#^#^#^#^#^#                          *
+##");
   printf("\n\t\t\t- - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n");
}
void welcomemsg()
{
   messageTop("Print it !!");
   printf("\n\n");
   printf("\n\t\t\t\t*-*-*-*-*-*-*-*                    *");
   printf("\n\t\t\t\t-             WELCOME              *");
   printf("\n\t\t\t\t*               TO               *");
   printf("\n\t\t\t\t-             LIBRARY              *");
   printf("\n\t\t\t\t*             MANAGEMENT             *");
   printf("\n\t\t\t\t-              SYSTEM              *");
   printf("\n\t\t\t\t*                    *-*-*-*-*-*-*-*");
   printf("\n\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\n");
   printf("\n\n\n\t\t\t Press the 'ENTER' Key to Continue.....");
   getch();
```

```
}
int nameValidation(const char *name)
{
   int NameValid = 1;
   int len = 0;
   int i = 0;
   len = strlen(name);
   for(i = 0; i<len ; ++i)
   {
      if(!(isalpha(name[i])) && (name[i] != '\n') && (name[i] != ' '))
      {
         NameValid = 0;
         break;
      }
   }
   return NameValid;
}

//Leap Year Validation / Checking
int  yearLeap(int yr)
{
   return (((yr % 4 == 0) &&
          (yr % 100 != 0)) ||
         (yr % 400 == 0));
```

```
}

//Date Validation (for the correct format)
int dateValidation(Date *valid)
{
   if (valid->yyyy > YEAR_MAX ||
         valid->yyyy < YEAR_MIN)
      return 0;
   if (valid->mm < 1 || valid->mm > 12)
      return 0;
   if (valid->dd < 1 || valid->dd > 31)
      return 0;

   if (valid->mm == 2)  //The month of February get 29 days on Leap
Years
   {
      if (yearLeap(valid->yyyy))
         return (valid->dd <= 29);
      else
         return (valid->dd <= 28);
   }
   if (valid->mm == 4 || valid->mm == 6 ||
         valid->mm == 9 || valid->mm == 11)
      return (valid->dd <= 30);
```

```
    return 1;
}

//Adding the information of the Books to the system
void bookAddition()
{
    int days;
    BooksInfo dbaddbook = {0};//database created for books
information
    FILE *fp = NULL;
    int status = 0;
    fp = fopen(FILE_NAME,"ab+");
    if(fp == NULL)
    {
        printf("FILE UNABLE TO OPEN !!\n");
        exit(1);
    }
    messageTop("ENTER (1) TO ADD NEW BOOKS");
    printf("\n\n\t\t\tENTER THE REQUIRED DETAILS BELOW:");
    printf("\n\t\t\t^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^\n");
    printf("\n\t\t\tBook ID NO.  = ");
    fflush(stdin);
    scanf("%u",&dbaddbook.books_id);
    do
```

```
   {
      printf("\n\t\t\tBOOK NAME  = ");
      fflush(stdin);
      fgets(dbaddbook.bookName,BOOKNAME_SIZE_MAX,stdin);
      status = nameValidation(dbaddbook.bookName);
      if (!status)
      {
         printf("\n\t\t\tINVALID CHARACTER IN THE NAME !! PLEASE
ENTER AGAIN");
      }
   }
   while(!status);
   do
   {
      printf("\n\t\t\tNAME OF THE AUTHOR  = ");
      fflush(stdin);
      fgets(dbaddbook.authorName,AUTHORNAME_SIZE_MAX,stdin);
      status = nameValidation(dbaddbook.authorName);
      if (!status)
      {
         printf("\n\t\t\tINVALID CHARACTER IN THE NAME !! PLEASE
ENTER AGAIN");
      }
   }
```

```
   while(!status);
   do
   {
      printf("\n\t\t\tNAME OF THE STUDENT  = ");
      fflush(stdin);
      fgets(dbaddbook.studentName,STUNAME_MAX_SIZE,stdin);
      status = nameValidation(dbaddbook.studentName);
      if (!status)
      {
         printf("\n\t\t\tINVALID CHARACTER IN THE NAME !! PLEASE
ENTER AGAIN");
      }
   }
   while(!status);
   do
   {
      printf("\n\t\t\tENTER THE DATE IN THE FORMAT
GIVEN:(DAY/MONTH/YEAR) - DD/MM/YYYY: "); //date in Important
!!!

scanf("%d/%d/%d",&dbaddbook.bookIssueDate.dd,&dbaddbook.bo
okIssueDate.mm,&dbaddbook.bookIssueDate.yyyy);


      //Date Validation
      status = dateValidation(&dbaddbook.bookIssueDate);
```

```
        if (!status)
        {
            printf("\n\t\t\tPLEASE ENTER A VALID DATE !!\n");
        }
    }
    while(!status);
    fwrite(&dbaddbook,sizeof(dbaddbook), 1, fp);
    fclose(fp);
}
// Book Hunting - Mean Searching !!!
void bookSearching()
{
    int found = 0;
    char bookName[BOOKNAME_SIZE_MAX] = {0};
    BooksInfo dbaddbooks = {0};
    FILE *fp = NULL;
    int status = 0;
    fp = fopen(FILE_NAME,"rb");
    if(fp == NULL)
    {
        printf("\n\t\t\tFILE UNABLE TO OPEN\n");
        exit(1);
    }
    messageTop("SEARCH BOOKS");
```

```
    if (fseek(fp,FILE_HEADER_SIZE,SEEK_SET) != 0)
    {
       fclose(fp);
       printf("\n\t\t\tFILE READING ISSUE OCCURED !!\n");
       exit(1);
    }
    printf("\n\n\t\t\tENTER THE BOOK NAME TO BE SEARCHED :");
    fflush(stdin);
    fgets(bookName,BOOKNAME_SIZE_MAX,stdin);
    while (fread (&dbaddbooks, sizeof(dbaddbooks), 1, fp))
    {
       if(!strcmp(dbaddbooks.bookName, bookName))
       {
          found = 1;
          break;
       }
    }
    if(found)
    {
       printf("\n\t\t\tID OF BOOK = %u\n",dbaddbooks.books_id);
       printf("\t\t\tNAME OF THE BOOK =
%s",dbaddbooks.bookName);
       printf("\t\t\tAUTHOR'S NAME = %s",dbaddbooks.authorName);
```

```
      printf("\t\t\tDATE OF ISSUE(DAY/MONTH/YEAR) - DD/MM/YYYY
= (%d/%d/%d)",dbaddbooks.bookIssueDate.dd,
            dbaddbooks.bookIssueDate.mm,
dbaddbooks.bookIssueDate.yyyy);
   }
   else
   {
      printf("\n\t\t\tRECORD NOT FOUND !!!");
   }
   fclose(fp);
   printf("\n\n\n\t\t\tPRESS 'ENTER' FOR RE-DIRECTING TO THE
MAIN-MENU .....");
   getchar();
}
// Book Viewing (for the existing / inputted books in the system)
void viewBooks()
{
   int found = 0;
   char bookName[BOOKNAME_SIZE_MAX] = {0};
   BooksInfo dbaddbooks = {0};
   FILE *fp = NULL;
   int status = 0;
   unsigned int bookcounter = 1;
   messageTop("VIEW BOOKS DETAILS");
   fp = fopen(FILE_NAME,"rb");
```

```
   if(fp == NULL)
   {
      printf("FILE UNABLE TO OPEN.\n");
      exit(1);
   }
   if (fseek(fp,FILE_HEADER_SIZE,SEEK_SET) != 0)
   {
      fclose(fp);
      printf("FILE READING ISSUE OCCURED !!!\n");
      exit(1);
   }
   while (fread (&dbaddbooks, sizeof(dbaddbooks), 1, fp))
   {
      printf("\n\t\t\tCOUNT = %d\n\n",bookcounter); //this will help
to count the no. of existing books in the system
      printf("\t\t\tID OF BOOK = %u",dbaddbooks.books_id);
      printf("\n\t\t\tNAME OF BOOK = %s",dbaddbooks.bookName);
      printf("\t\t\tAUTHOR'S NAME = %s",dbaddbooks.authorName);
      printf("\t\t\tDATE OF ISSE(DAY/MONTH/YEAR) =
(%d/%d/%d)\n\n",dbaddbooks.bookIssueDate.dd,
            dbaddbooks.bookIssueDate.mm,
dbaddbooks.bookIssueDate.yyyy);
      found = 1;
      ++bookcounter;
   }
```

```
    fclose(fp);
    if(!found)
    {
        printf("\n\t\t\tRECORD NOT FOUND !!!");
    }
    printf("\n\n\t\t\tPRESS THE 'ENTER' KEY FOR RE-DIRECTING TO
THE MAIN-MENU .....");
    fflush(stdin);
    getchar();
}
// Delete Books (which exists in the system)
void bookdeletion()
{
    int found = 0;
    int delbooks = 0;
    credential fileHeaderInfo = {0};
    char bookName[BOOKNAME_SIZE_MAX] = {0};
    BooksInfo dbaddbooks = {0};
    FILE *fp = NULL;
    FILE *tmpFp = NULL;
    int status = 0;
    messageTop("BOOKS DETAIL - DELETE");
    fp = fopen(FILE_NAME,"rb");
    if(fp == NULL)
```

```
{
   printf("FILE UNABLE TO OPEN.\n");
   exit(1);
}
tmpFp = fopen("tmp.bin","wb");
if(tmpFp == NULL)
{
   fclose(fp);
   printf("FILE UNABLE TO OPEN.\n");
   exit(1);
}
fread (&fileHeaderInfo,FILE_HEADER_SIZE, 1, fp);
fwrite(&fileHeaderInfo,FILE_HEADER_SIZE, 1, tmpFp);
printf("\n\t\t\tEnter Book ID NO. for delete:");
scanf("%d",&delbooks);
while (fread (&dbaddbooks, sizeof(dbaddbooks), 1, fp))
{
   if(dbaddbooks.books_id != delbooks)
   {
      fwrite(&dbaddbooks,sizeof(dbaddbooks), 1, tmpFp);
   }
   else
   {
      found = 1;
```

```
        }
    }
    (found)? printf("\n\t\t\tRECORED DELETION SUCCESSFUL
!!!"):printf("\n\t\t\tRECORD NOT FOUND");
    fclose(fp);
    fclose(tmpFp);
    remove(FILE_NAME);
    rename("temp.bin",FILE_NAME);
}

//If you forget your password , you can update that
void updcred(void)
{
    credential fileHeaderInfo = {0};
    FILE *fp = NULL;
    unsigned char userName[USERNAME_MAX_SIZE] = {0};
    unsigned char password[PWD_MAX_SIZE] = {0};
    messageTop("UPDATE CREDENTIAL(S)");
    fp = fopen(FILE_NAME,"rb+");
    if(fp == NULL)
    {
        printf("FILE UNABLE TO OPEN.\n");
        exit(1);
    }
```

```
    fread (&fileHeaderInfo,FILE_HEADER_SIZE, 1, fp);
    if (fseek(fp,0,SEEK_SET) != 0)
    {
       fclose(fp);
       printf("\n\t\t\tPASSWORD UPDATION ISSUE !!\n");
       exit(1);
    }
    printf("\n\n\t\t\tENTER NEW USERNAME:");
    fflush(stdin);
    fgets(userName,USERNAME_MAX_SIZE,stdin);
    printf("\n\n\t\t\tENTER NEW PASSWORD:");
    fflush(stdin);
    fgets(password,PWD_MAX_SIZE,stdin);
    strncpy(fileHeaderInfo.username,userName,sizeof(userName));
    strncpy(fileHeaderInfo.password,password,sizeof(password));
    fwrite(&fileHeaderInfo,FILE_HEADER_SIZE, 1, fp);
    fclose(fp);
    printf("\n\t\t\tPASSWORD CHANGED SUCCESSFUL !!!");
    printf("\n\t\t\tSIGN IN AGAIN - ");
    fflush(stdin);
    getchar();
    exit(1);
}
```

```
//MAIN MENU - includes all the options (functions)
void menu()
{
   int choice = 0;
   do
   {
      messageTop("MAIN MENU");
      printf("\n\n\n\t\t\t1.Update Your Password");
      printf("\n\t\t\t2.Add New Books In The System");
      printf("\n\t\t\t3.Search for the books you want");
      printf("\n\t\t\t4.Delete Book(s) From The System");
      printf("\n\t\t\t5.View books you want to know about");
      printf("\n\t\t\t0.Exit Application");
      printf("\n\n\n\t\t\tEnter your choice ===> ");
      scanf("%d",&choice);
      switch(choice)
      {
      case 1:
         updcred(); //updation of credencials (username / password)
         break;
      case 2:
         bookAddition();
         break;
      case 3:
```

```
            bookSearching();
            break;
        case 4:
            viewBooks();
            break;
        case 5:
            bookdeletion();
            break;
        case 0:
            printf("\n\n\n\t\t\t\tTHANK YOU FOR USING THE
SYSTEM!!!\n\n");
            printf("\t\t\t\t\tHAVE A NICE DAY !!!\n\n\n\n\n");
            exit(1); //exit from the application
            break;
        default:
            printf("\n\n\n\t\t\t'ERROR OCCURED'-INVALID INPUT!!! TRY
AGAIN.....");
        } //Ending of the Switch Statement
    }
    while(choice!=0); //Ending of the Do-While Loop
}
//Login credentials (Password)
void login()
{
```

```
   unsigned char userName[USERNAME_MAX_SIZE] = {0};
   unsigned char password[PWD_MAX_SIZE] = {0};
   int L=0; //'L' here means the number of times Login will be done
   credential fileHeaderInfo = {0};
   FILE *fp = NULL;
   messageTop("Login");
   fp = fopen(FILE_NAME,"rb");
   if(fp == NULL)
   {
      printf("FILE NOT OPENED.\n");
      exit(1);
   }
   fread (&fileHeaderInfo,FILE_HEADER_SIZE, 1, fp);
   fclose(fp);
   do
   {
      printf("\n\n\n\t\t\t\tENTER YOUR USERNAME :");
      fgets(userName,USERNAME_MAX_SIZE,stdin);
      printf("\n\t\t\t\tENTER YOUR PASSORD :");
      fgets(password,PWD_MAX_SIZE,stdin);
      if((!strcmp(userName,fileHeaderInfo.username)) &&
(!strcmp(password,fileHeaderInfo.password)))
      {
         menu();
```

```
        }
        else
        {
            printf("\t\t\t\t\FAILED TO LOGIN !!! TRY AGAIN TO SIGN
IN.\n");
            L++;
        }
    }
    while(L<=3); //Only Three times the Login will be allowed (if it gets
fail)
    if(L>3)
    {
        messageTop("LOGIN FAILED");
        printf("\t\t\t\tSORRY !!! YOU ARE AN UNKNOWN USER.");
//When thrice you tried for login but , unsuccessful
        getch();
        system("cls");
    }
}
int filepresence(const char *path)
{
    // Opening File
    FILE *fp = fopen(path, "rb");
    int status = 0;
    //No Existence of File
```

```
    if (fp != NULL)
    {
        status = 1;
        //Close the File if it exists
        fclose(fp);
    }
    return status;
}
void cred()
{
    FILE *fp = NULL;
    int status = 0;
    const char defaultUsername[] ="ACPPROJECT2020\n";
    const char defaultPassword[] ="ACPPROJECT2020\n";
    credential fileHeaderInfo = {0};
    status = filepresence(FILE_NAME);
    if(!status)
    {
        fp = fopen(FILE_NAME,"wb");
        if(fp != NULL)
        {

strncpy(fileHeaderInfo.password,defaultPassword,sizeof(defaultPass
word));
```

```
strncpy(fileHeaderInfo.username,defaultUsername,sizeof(defaultUse
rname));
            fwrite(&fileHeaderInfo,FILE_HEADER_SIZE, 1, fp);
            fclose(fp);
        }
    }
}
int main()
{
    cred();
    welcomemsg();
    login();
    return 0;
}
```

==========