TABLE OF CONTENTS

1. Abstract
2. Acknowledgements
3. Introduction
4. Problem Analysis
5. Review of Related Lectures
6. Algorithm Development and Flowchart
7. Coding
8. Result and Discussion
9. Conclusion

   Snapshot of Program

ABSTRACT

The mini-project "Librarian: The Library Management Console App" aims to provide a computerized interface to standardize and handle the daily activities of a library. The key objective of this app is to help the user easily access the study materials and keep track of the library history.

As a library consists of a large number of books, to locate the required book can be quite tedious and time-consuming for the reader while keeping records of issued books and members is rather difficult for the librarian. This project plans to overcome these everyday problems with a book-map for book location and database management system for keeping the account of operation records. The app is a demonstration of how a programming language can be used to help solve common problems and reduce human efforts.The fact that the project is a console app suggests that it is designed to be operated via a text-only computer interface and takes input and displays output at a command line console.

Overall, this project intends to lend a hand to the students as well as library staffs to maintain the library in the best way possible reducing human efforts.

ACKNOWLEDGEMENTS

We would like to extend our sincere gratitude to everyone who directly or indirectly supported us and provided helpful assistance with this project. First, we would like to thank **Ms. Anku Jaiswal**, our teacher of Computer programming-I, who provided us with the insights and expertise of C-programming, which is the basis for our entire project. Together, we would like to acknowledge the frequent helps from our lab teacher **Mr. Biswas Pokharel.** Also we would like to thank all our dear friends and seniors for their advice, encouragement, support and help.

We would also like to thank the authors of various national and international books, which proved to be of great help for the preparation of the project.

1. R.D Bhatta and B.R. Dawadi- A Textbook of C programming.

2. D.S Baral, S.K. Ghimire- The Secrets of C- Programming language.

3. Byron S. Gotterfried- Programming With C.

4. Balagurusamy- Programming In ANSI C.

INTRODUCTION

Librarian: The Library Management Console App is created entirely with the help of C- programming language. This program has abundant use of various aspects of C-programming such as user-defined function, structure, formatted input/output system, file handling and many more. The primary focus lies on implementation of maximum knowledge and ideas gained throughout learning C-programming in the project. Taking into consideration the difficulties and boredom in manual library management, this app hopes to facilitate systematized library management.

3.1 **Background and problem statements:**

On a daily basis, a library has numerous check-ins and has to deal with a number of book transactions. With hundreds of books, the readers have to go through each section of library to find the interested book, which is troublesome in itself. In addition, to keep the track of issuance, renewal and return of book manually is much difficult for the librarian. In order to effectively resolve this issue electronically, **Librarian** app exhibits simple codes and logic demonstrating the utility of programming language to ease the workload. Furthermore, it also improves library presentation and greatly helps organization of library.

This project can be broken down into two main parts:

a. **Book map**: The book map provides exact location of the book sought by the user. When the detail of the book is entered, the book map locates the section and shelf of the desired book. With the help of library management software and well-organized books store, the user can easily search and find the books. This solves the problem of having to look through each bookshelf just to find one book.
b. **Database Management System:** The records of the book issued, renewed and returned are stored in a file that can be accessed anytime. It reduces the manual paperwork as proper information of books has been recorded electronically. Each member can login to his own account and check for the books that have been issued by him which ultimately helps him make optimum use of the books. The information can be timely accessed and the library becomes more systematic as it should be. Moreover, admin (librarian) can keep update the information of books and manage availability and arriver record of the book.

3.2 **Objectives**

a. To make best use of user defined functions in order to make the program significantly easy to understand and maintain as well as to benefit multiple use of a single function.

b. To learn about various library functions included in different header files.

c. To use structure and file handling simultaneously to safely record information.

d. To create a user-friendly environment in a console-based interface.

e. To practice work division to in turn improve teamwork with enhancement of problem solving skills.

3.3 **Limitations**

a. The program does not make use of graphics, which is less eye-catching to the user.

b. The user records cannot be accessed if the user forgets the password for the username.

PROBLEM ANALYSIS

4.1 **Understanding the problem**

For programming a well-functioning library management system, the following questions were must to be fulfilled:

a. How should user access to his book records?
b. How can a user locate the required book without having to look for it in every shelf?
c. What if a new user arrives for library membership?
d. How can the project be made user friendly with only the use of keyboards for data entry and access?
e. How to enter record for the book issued by the user?

4.2 **Input requirements**

a. Username and password for an existing user trying to access his account.
b. New username and password for creating a new user account.
c. Details of the book whose location is to be found on book-map.
d. Check for issuance, renewal and return of book.
e. User interaction to proceed to next page or exit to previous page.

The information about book transaction and username-password is stored in a separate file through file handling.

4.3 **Output requirements**

The output requirements for this program are quite simple. As the program is only an interface between the user and the computer, the user is displayed the message of success or failure of the operation. The output of book map is demonstrated by locating the exact position of the book shown in the display screen. All major works are done within the file.

4.4 **Processing requirements**

The processing requirement should be clearly defined to convert the given input data to the required output. The coding is done on 64-bit OS using TDM-GCC as a complier. The header file <windows.h> is used in the source code which is only compatible with Windows OS, so the code should be run in a computer with windows OS, or the code involving this header file shall be omitted for any other OS.

4.5 **Technical feasibility**

In field, this program can be used in every library to systematize the library management. It does not require any special features and can be used whenever electric supply is provided. It is cost-effective and the codes are also quite simple to understand and use.

REVIEW OF RELATED LITERATURES

What is C?

C is a general-purpose, structured programming language. Its instructions consists terms that resemble algebraic expression, augmented by certain English keywords such as if, else, for, do and while, etc. C contains additional features that allow it to be used at a lower level, thus bridging the gap between machine language and the more conventional high level language. This flexibility allows C to be used for system programming (e.g. for writing operating systems as well as for applications programming such as for writing a program to solve mathematical equation or for writing a program to bill customers).

Header file

A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files. There are two types of header files: the files that the programmer writes and the files that come with the compiler.

The various header files involved in this project are:

- stdio,h : contain declaration of many functions and Macros which required to get input from input devices and show output on output screen of C program.
- conio.h: provides console input/output.
- stdlib.h: This header defines several general purpose functions including dynamic memory management, random number generation, communication with the environment, integer arithmetics, searching, sorting and converting.
- windows.h: it is a Windows-specific header file for the C and C++ programming languages which contains declarations for all of the functions in the Windows API, all the common macros used by Windows programmers, and all the data types used by the various functions and subsystems.
- time.h: This header file contains definitions of functions to get and manipulate date and time information.

File Inclusion

The #include directive inserts the contents of a specified file into the text stream delivered to the compiler. Usually, standard headers and global definitions are included in the program stream with the #include directive. This directive has two forms:

#include "*filename*" *newline*
#include <*filename*> *newline*

The format of *filename* is platform-dependent. If the *filename* is enclosed in quotation marks, the search for the named file begins in the directory where the file containing the #include directive resides. If the file is not found there, or if the file name is enclosed in angle brackets (< >), the file search follows platform-defined search rules. In general, the quoted form of #include is used to include files written by users, while the bracketed form is used to include standard library files.

In this project #include has been used to include header files and functions that are not pre-defined in C. #include is used for including predefined functions to the program. This includes all the predefined syntaxes that can be used in programming.

E.g. #include "bookfinder.c"

#include "login.c"

#include "bookData.c"

Control Statements

The statements which alter flow of execution of a program are known as control statements. In the absence of control statements, the instructions or statements are executed in same order in which they appear in source program. Such type of construct is known as sequential construct. Sometimes tasks are performed on the basis of certain logic test where output comes according to true or false tests or conditions. Similarly , sometimes it is necessary to perform repeated actions or skip some statements from execution. For these operations, control statements are needed. They control flow of program so that it is not necessary to execute statements in the same order in which they appear in the program.

Selective structures are used when we have a number of situations where we need to change the order of execution of statements based on certain condition. The selective statements make a decision to take the right path before changing the order of execution.

C provides the following statements for selective structure:

i. if statements
ii. switch statements

if statements: The if statement is a powerful decision making statement and it is used to control the flow of execution of statements. It is a two way statement and is used in conjunction with an expression. If statement allows the computer to evaluate the expression first and then on depending whether the value of the expression is true or false it transfer the control to the particular statement.
At this point of the program has two paths to follow: one for true condition and other for false condition.
The types of if statements are explained below:

Simple if statement: The simple if statement is used to conditionally excite a block of code based on whether a test condition is true or false. If the condition is true the block of code is executed, otherwise it is skipped. The syntax of if statement is given below:

```
if(test expression)
{
statement-block;
}
statement-x;
```

if else statement:

The if…else statement extends the idea of the if statement by specifying another section of code that should be executed only if the condition is false i.e. conditional branching. True- block statements are to be executed only if the test expression is true and false block statements to be executed only if the condition is false.
The syntax of if else statement is given below:

```
 if(test expression)
{
 true block statement;
}
 else
{
 false block statement;
}
```

Switch statement :

C has built a multi way decision statements known as switched, that tests the value of an expression against a list of case values (integer or character constants). When a match is found, the statements associated with that case is executed.

Syntax:

```
switch (expression)
{
case constant1:
block of case constant1;
break;
case constant2:
block of case constant2;
break;
case constant3:
block of case constant3;
break;
.....
.....
default:
default block;

}
next statement(s);
```

Looping:

Loop causes a section of code to be repeated for a specified number of times or until some condition holds true. When a condition becomes false, the loop terminates and control passes to statement below loop.

Different types of loops are discussed below with their major characteristics and syntax used in C:

while loop:

The "while loop" specifies that a section of code should be executed while a certain condition holds true. The syntax of while loop is given below:

```
while(test expression)
{
 body of loop
 ( statements block)
}
```

For loop:

The for loop is used to execute a block of code for a fixed number of repetitions. Initialization is generally an assignment statement used to set loop control variable. Test expression is a relational expression that determines when loop exits. Update expression defines how the loop variable changes each time when the loop is repeated. The syntax of for loop is given below:

```
for(initialization expression;test expression;update expression)
{
body of loop;
 }
```

break statement:

The break statement is used to jump out of loop. The break statement terminates the execution of the nearest enclosing loop. Control passes to the

statement that follows the terminated statement. In a switch break statement causes the program to execute the next statement after switch.

break;

Function:

A function is a self-contained program segment that carries out some specific well defined task. Every c program consists of one or functions. Execution of program always begins by carrying out instruction in main. Function makes program significantly easier to understand and maintain. A well written function may be reused in multiple programs. Program are easier to design, debug and maintain.

User defined function :

These are the function declared and defined by the user according to their program requirements. These functions are available only for the current program in which it is defined. It can be used by the program in which it is defined as well as the related files of the program. But it can't be used as library function in all program. When a function is called in the block or any other function, the execution control will jump to the called function; it will execute the statements in the function and return back to the called block/function with/without some values.

```
#include<stdio.h>

#include<conio.h>

void function_name(){

------------

------------

}

void main(){

function_name();

--------------------

}
```

Structure

It is a collection of variables (may be of different types) under a single name. A structure provides a means of grouping variables under a single name for easier handling and identification. It can be defined as a new named type, thus extending the number of available types. It is a heterogeneous user defined data type. It is also called constructed data type. It may contain different data types. Structure can also store non-homogeneous data type into a single collection. Structure may contain pointers, array or even other structures other than common data type (such as int, float, long int etc). It is a convenient way of grouping several pieces of related information together. Complex hierarchies can be created by nested structures. Structures may be copied to and assigned. They are also useful in passing groups of logically related data into functions.

Structure definition(Structure template declaration) :

It creates a format that may be used to declare structure variables. It can be at the beginning of the program file, before any variables or function are defined. They may also appear before the main as a global definition and can be used by other function as well. struct is the keyword used for creating structure.

Syntax:

```
struct structure_name
{
data_type member 1;
data_type member 2;
.....
.....
data_type member n
};
```

example:

```
struct student
{
char name[20];
```

int rollno;

float marks;

};

File:

A collection of data which is stored on a secondary device like a hard disk is called as a file. A file is generally used as real-life applications that contain a large amount of data. Therefore, the file is a place on the disk where a group of related data is stored.

Opening a file:

Before performing any input/output operation, file must be opened. While opening file, the following must be specified:

i. The name of file
ii. ii. The manner in which it should be opened (that for reading ,writing ,both reading and writing ,appending at the end of file, overwriting the file)

Syntax for opening a file:

ptr_variable=fopen(file_name,file_mode);

This associates a file name with the buffer and specifies howthe data file will be utilized(File Opening Mode).

The function fopen() returns a pointer to the beginning of the buffer area associated with the file.

iii. When working with a stream oriented data file ,the first step is to establish a buffer area, where information is temporary stored while being transferred between the computer's memory and data file .the buffer area is established by writing: FILE *ptr_variable;

Here, FILE is a special structure declared in header file stdio.h.

ALGORITHM DEVELOPMENT AND FLOWCHART

An **algorithm** is step by step description of activities or methods to be processed to get the desired output from a given input. In other words, an algorithm is step-by-step description of the procedure written in human understandable language for solving given problems.

A brief algorithm for our project:

. Algorithm for main function:
Step 1: Start
Step 2: Goto function welcome
Step 3: Stop

Algorithm for function welcome
Step 1: Start
Step 2: Display Welcome Statement and name of project team members
Step 3: Goto function mainMenu
Step 4: Stop

Algorithm for function mainMenu
Step 1: Start
Step 2: Ask for the user to LogIn , SignUp or open Bookfinder
Step 3: Open respective functions
Step 4: Stop

Algorithm for bookfinder
Step 1: Start
Step 2: Ask for user to look at the map of the library or point out a certain book in the map
Step 3: Show respective maps
Step 4: Stop

Algorithm for function signUp
Step 1: Start
Step 2: Ask for username and password and save it as a structure
Step 3: Store the structure in "users.dat" file
Step 4: Ask if user wants to signUp another account
    4.1: If yes, go to step 2
    4.2: If no,go to function mainMenu
Step 5: Stop

Algorithm for function logIn
Step 1: Start

Step 2: Ask for username and password
Step 3: Scan for username and password in "users.dat" file
  3.1: If match is found, go to step 4
  3.2: If not, goto show error message and go to function mainMenu
Step 4: Create a file named same as the username
Step 5: Ask for entry or transaction
Step 6: If asked for entry, go to function student, else go to function transaction
Step 7: Stop

Algorithm for student
Step 1: Start
Step 2: Ask for the name of book
Step 3: Is the name of book found in "book.dat"
    3.1 If yes, show success and then show the time of transaction
    3.2: Else, display error message
Step 4: Go to function mainMenu
Step 5: Stop

Algorithm for function transaction:
Step 1: Start
Step 2: Scan the file of the user and display all the transactions
Step 3: Stop

.

.

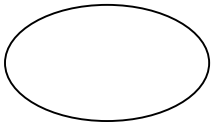
A **flowchart** is a pictorial representation of an algorithm that uses boxes of different shapes to denote different types of instructions. The boxes are connected by solid lines having arrow heads to represent the actual direction of the flow of execution of program.

The graphical symbols below are the major constituents of a flowchart:

- ✓ Arrow

used to connect flowchart symbols and directions indicating the flow of logic.

- ✓ Oval (start/stop)

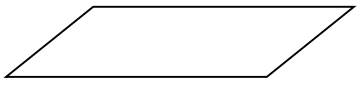


used to indicate beginning and end of task.

- ✓ Rectangle

used for arithmetic and data manipulation operations.

- ✓ Parallelogram (Input/output)

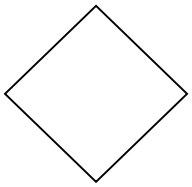


used for input and output.

- ✓ Connectors

used to connect different flow lines and remote part of flowchart.

- ✓ Decision



used for decision making and branching operations that have two options.

- ✓ Function call

used wherever a function is called.

CODING

As we used file inclusion , we had different codes in different files:

Following code is in main.c

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <Windows.h>
#include <time.h>

#include "bookfinder.c"
#include "login.c"
#include "bookData.c"



static int x,y; // the co-ordinate points

void lineVertical(int x,int y1,int y2)
{
   for(int i=y1;i<=y2; i++){
      gotoxy(x,i);
      printf("*");
   }
```

```
}

void lineHorizontal(int x1,int x2,int y)
{
   for(int i=x1;i<=x2; i++){
      gotoxy(i,y);
      printf("*");
   }
}

void gotoxy(int i, int j) // to put the cursor in a certain co-ordinate
{
   x = i;
   y = j;
   COORD c;
   c.X = x;
   c.Y = y;
   SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),c);
}

void br(int l) // line break. arguement refers no of lines to be skipped.
{
   gotoxy(x,y+l);
}
```

```
void squareTemplate()
{
   for(int p=10;p<=100;p++)
   {
      for(int q=2;q<=30;q++)
      {
         if(p==10||p==100||q==2||q==30)
         {
            gotoxy(p,q);
            printf("*");
         }
      }
   }
}


void welcome() // the welcome page
{
   system("cls");
   system("COLOR 2E"); // change background and text color
   squareTemplate();
   gotoxy(38,8);
   printf("\t         LIBRARIAN");
   br(2);
   printf("The Library Management Console App");
```

```
    gotoxy(25,20);
    printf("Made By:");
    br(1);
    printf("1. Rujal Acharya");
    br(1);
    printf("2. Prayag Man Mane");
    br(1);
    printf("3. Sanjay K.C.");
    br(1);
    printf("4. Asmin Silwal");
    gotoxy(38,14);
    printf("     Press any key to continue ");
    char i;
    //scanf("%c",&i);
    i = getche();
    mainMenu();
    gotoxy(25,7);

}

void mainMenu()// The Main Menu
{
    system("cls");
    squareTemplate();
    gotoxy(48,8);
```

```
printf("MENU");
gotoxy(30,11);
printf("1. Log In (if you already have an account)");
br(1);
printf("2. Sign Up (if you don't have an account)");
br(1);
printf("3. Book Searcher");
br(1);
printf("4. Exit");
gotoxy(25,20);
printf("Enter your choice  ");
char i = getche();
switch(i)
{
   case '1':
      logIn();
      break;
   case '2':
      signUp();
      break;
   case '3':
      bookFinder();
      break;
   case '4':
      exitPage();
```

```
            break;
        default:
            mainMenuError();
    }
}

void mainMenuError()
{
    system("cls");
    squareTemplate();
    gotoxy(26,8);
    printf("You have entered invalid information");
    br(2);
    printf("Do you want to return to the main menu(Y/N)? ");
    char ch = getche();
    switch(ch)
    {
        case 'Y':
        case 'y':
            mainMenu();
            break;
        case 'N':
        case 'n':
            exitPage();
            break;
```

```
        default:
            mainMenuError();
    }
}

void exitPage()
{
    system("cls");
    squareTemplate();
    gotoxy(26,8);
    printf("Thank you for using LIBRARIAN -> The Library Management Console
     App");
    br(1);
    printf("\t\t   See You Soon");
    getche();
    blankPage();
}

void blankPage()
{
    system("cls");
}

void main()
{
```

```
    welcome();
}
```

Similarly, following code is saved as login.c

```
FILE *fp;
char uName[10], pwd[10];int i;char c;


struct user{
    char username[10];
    char password[10];
}*userp;



void logIn()
{
    system("cls");
    squareTemplate();
    userp=(struct user *)malloc(sizeof(struct user));


    if ( ( fp=fopen("user.dat", "r+")) == NULL) {
            if ( ( fp=fopen("user.dat", "w+")) == NULL) {
```

```
        printf ("Could not open file\n");
        exit ( 1);
    }
}
gotoxy(38,8);
printf("Username: ");
scanf("%9s",uName);
gotoxy(38,9);
printf("Password: ");
scanf("%9s",pwd);
while ( fread (userp, sizeof(struct user), 1, fp) == 1) {
    if( strcmp ( userp->username, uName) == 0) {
        br(1);
        printf ("username Matched\n");
        if( strcmp ( userp->password, pwd) == 0) {
            br(1);
            printf ("password Matched\n");
            getche();
             system("cls");
             squareTemplate();
             gotoxy(30,11);
             br(1);
            printf("Enter your choice");
            br(3);
            /*UPDATED*/
```

```
                printf("Issue book");
                br(1);
                printf("Transactions");
                char l=getche();
                switch(l)
                {
                case '1':
                   student(userp->username);
                   break;
                case '2':
                   transaction(userp->username);
                   break;
                default:
                   printf("Invalid");
                }

              }
            }

        }
        free (userp);
   fclose(fp);
}

void signUp()
```

```
{

   userp=(struct user *)malloc(sizeof(struct user));

   do
         {
            system("cls");
            squareTemplate();
            if ( ( fp=fopen("user.dat", "a+")) == NULL) {
               if ( ( fp=fopen("user.dat", "w+")) == NULL) {
                  gotoxy(38,8);
                  printf ("Could not open file\n");
                  exit (1);
               }
            }
            gotoxy(38,8);
            printf("Choose A Username: ");
            scanf("%9s",userp->username);
            gotoxy(38,9);
            printf("Choose A Password: ");
            scanf("%9s",userp->password);
            fwrite (userp, sizeof(struct user), 1, fp);
            br(1);
            printf("Add another account? (Y/N): ");
            c = getche();
```

```
        }while(c=='Y'||c=='y');
        free (userp);
    fclose(fp);
    mainMenu();

}
```

Similarly, following code is saved as bookfinder.c:

```
void bookFinder()
{
    system("cls");
    squareTemplate();
    system("COLOR F6");
    gotoxy(46,8);
    printf("BOOKFINDER");
    gotoxy(36,10);
    printf("Find the location of the books");
    gotoxy(40,14);
    printf("Press any key to continue  ");
    char i = getche();
    bookFinderMenu();
}
```

```
void bookFinderMenu()
{
   system("cls");
   squareTemplate();
   gotoxy(48,8);
   printf("BOOKFINDER");
   gotoxy(36,10);
   printf("1. Map of library");
   br(1);
   printf("2. Search for a particular book");
   br(1);
   printf("3. Exit from BOOKFINDER");
   gotoxy(40,20);
   printf("Enter your choice  ");
   char i = getche();
   switch(i)
   {
      case '1':
         fillBlankMap();
         break;
      case '2':
         bookOptions();
         break;
      case '3':
         system("COLOR 2E");
```

```
            mainMenu();
            break;
        default:
            bookFinderMenuError();
    }
}

void bookFinderMenuError()
{
    system("cls");
    squareTemplate();
    gotoxy(26,8);
    printf("You have entered invalid information");
    br(2);
    printf("Do you want to return to the BOOKFINDER(Y/N)? ");
    char ch = getche();
    switch(ch)
    {
        case 'Y':
        case 'y':
            bookFinderMenu();
            break;
        case 'N':
        case 'n':
            mainMenu();
```

```
            break;
        default:
            mainMenuError();
    }
}

void blankMap()
{
    system("cls");
    system("COLOR 0A");
    for(int p=5;p<=100;p++)
    {
        for(int q=3;q<=25;q++)
        {
            if(p==5||p==100||q==3||q==25)
            {
                gotoxy(p,q);
                printf("*");
            }
        }
    }
    // The map starts here
    lineHorizontal(5,100,8);
    lineVertical(75,8,25);
    lineVertical(25,8,15);
```

```
    lineHorizontal(5,75,15);
    lineVertical(36,15,25);
}

void fillBlankMap()
{
    blankMap();
    gotoxy(50,1);
    printf("LIBRARY");
    gotoxy(50,5);
    printf("Section A");
    gotoxy(10,12);
    printf("Section B");
    gotoxy(47,12);
    printf("Section C");
    gotoxy(82,16);
    printf("Section D");
    gotoxy(12,20);
    printf("Section E");
    gotoxy(47,20);
    printf("Section F");
    gotoxy(12,26);
    getche();
    system("COLOR F6");
    bookFinderMenu();
```

```
}

void bookOptions()
{
   system("cls");
   squareTemplate();
   gotoxy(26,8);
   printf("1. Physics");
   br(1);
   printf("2. Chemistry");
   br(1);
   printf("3. Mathematics");
   br(1);
   printf("4. Computer Programming");
   br(2);
   printf("Enter Your Choice ");
   char ch = getche();
   switch(ch)
   {
   case '1':
      physicsMap();
   case '2':
      chemistryMap();
   case '3':
      mathMap();
```

```
    case '4':
        cMap();
    default:
        bookFinderMenuError();
    }
}

void physicsMap()
{
    blankMap();
    gotoxy(50,1);
    printf("PHYSICS");
    gotoxy(50,16);
    printf("++");
    br(1);
    printf("++");
    gotoxy(12,26);
    getche();
    system("COLOR F6");
    bookFinderMenu();
}

void chemistryMap()
{
    blankMap();
```

```
    gotoxy(50,1);
    printf("CHEMISTRY");
    gotoxy(12,17);
    printf("++");
    br(1);
    printf("++");
    gotoxy(12,26);
    getche();
    system("COLOR F6");
    bookFinderMenu();
}

void mathMap()
{
    blankMap();
    gotoxy(50,1);
    printf("MATHEMATICS");
    gotoxy(13,12);
    printf("++");
    br(1);
    printf("++");
    gotoxy(12,26);
    getche();
    system("COLOR F6");
    bookFinderMenu();
```

```
}

void cMap()
{
   blankMap();
   gotoxy(40,1);
   printf("COMPUTER PROGRMMING");
   gotoxy(80,14);
   printf("++");
   br(1);
   printf("++");
   gotoxy(12,26);
   getche();
   system("COLOR F6");
   bookFinderMenu();
}
```

Following code is saved as bookData.c:

```
#define rtime (+30)



struct book
{
   char name[50],author[50];
   /*int id,qnty;*/
}*b;

void student(char name[])
{
system("cls");
int flag = 0;
 char q;
 char date_sub[50];
 time_t t;
 time(&t);
 struct tm *date_issue;
 char bn[50],bt[50];
 b=(struct book *)malloc(sizeof(struct book));
 FILE *stud;
 FILE *bk;
 bk=fopen("book.dat","r");
```

```
 stud=fopen(name,"a+");
 printf("Enter the name of the book  ");
 scanf("%s",bn);
 while(fread(b,sizeof(struct book),1,bk)){
 if(strcmp(b->name,bn)==0)
 {
    flag = 1;
    date_issue=ctime(&t);
    printf("\ndate issued : %s",date_issue);
    fwrite(b,sizeof(struct book),1,stud);
 }
 }
 if(flag==0)
   printf("\nBook unavailable");
 q=getche();
 fclose(bk);
 fclose(stud);
 mainMenu();

}
void transaction(char name[])
{
   system("cls");
   FILE *bk;
   b=(struct book*)malloc(sizeof(struct book));
```

```
    if(fopen(name,"r")==NULL)
    {
        printf("No transactions available");
        printf("\n Do you want to return to main menu(Y/N)");
        char e=getche();
        if(e=='y'||e=='Y')
            mainMenu();
        else
            exitPage();
    }
    else
    {
        bk=fopen(name,"r");
        while(fread(b,sizeof(struct book),1,bk))
        {
            printf("%s\t%s\n",b->name,b->author);
        }
         fclose(bk);
         char c=getche();
         mainMenu();
    }
}
```

Finally, following code is saved as books.c:

```
#include<stdio.h>
#include<string.h>
struct book
{
   char name[50],author[50];
}*b;
void main()
{
   char c;
   FILE *bk;
   b=(struct book *)malloc(sizeof(struct book));
   do
   {
      if ( ( bk=fopen("book.dat", "a+")) == NULL) {
               if ( ( bk=fopen("book.dat", "w+")) == NULL) {
                  printf ("Could not open file\n");
                  exit ( 1);
               }
            }
      printf("Enter name of book: ");
      scanf("%s",b->name);
      printf("Enter name of author: ");
      scanf("%s",b->author);
```

```
        fwrite (b, sizeof(struct book), 1, bk);
        printf("Add another book (Y/N): ");
        scanf(" %c",&c);
    }while(c=='Y'||c=='y');
    fclose(bk);
    free (b);
    fclose(bk);
}
```

RESULT AND DISCUSSION

The result of the program is clear-cut. With the user providing the information, the file gets added with new records. The book-map shows position of the searched book as in fig.

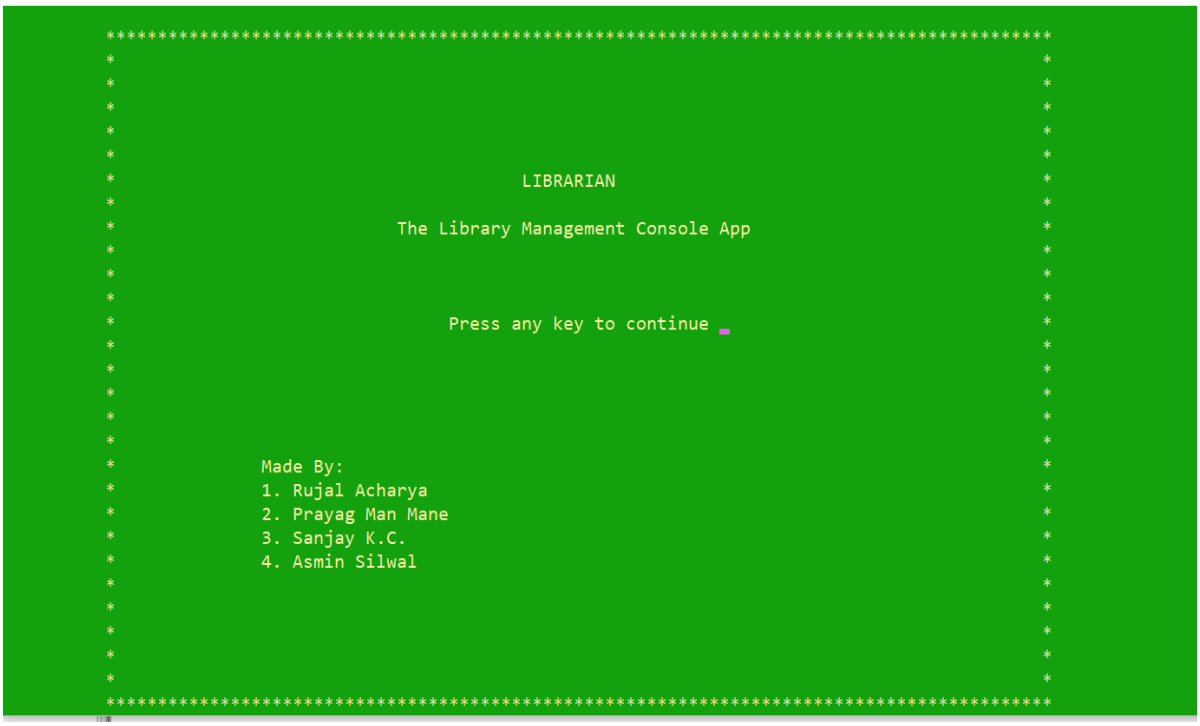
Although this program seems quite simple, special care should be taken in order to code it perfectly. The program failed many times and with number of failed attempts and debugging, the program was finally completed.{{{{{requires edit}}}}}

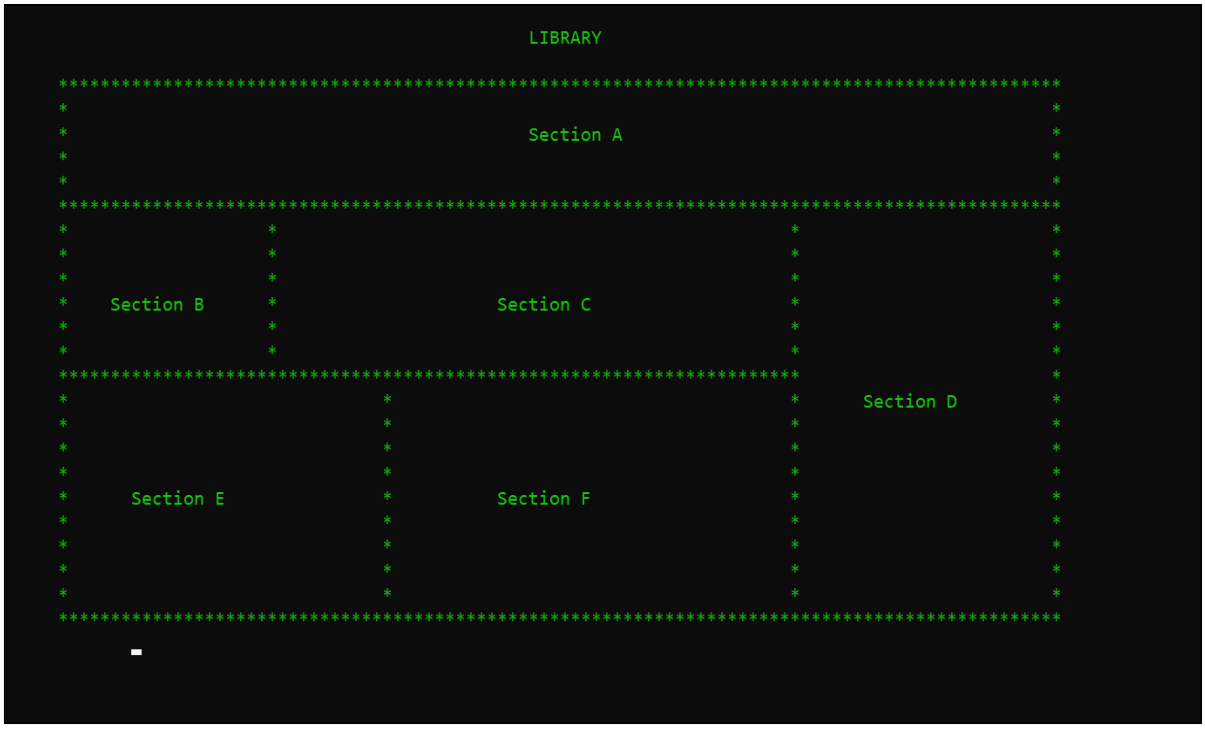
CONCLUSION

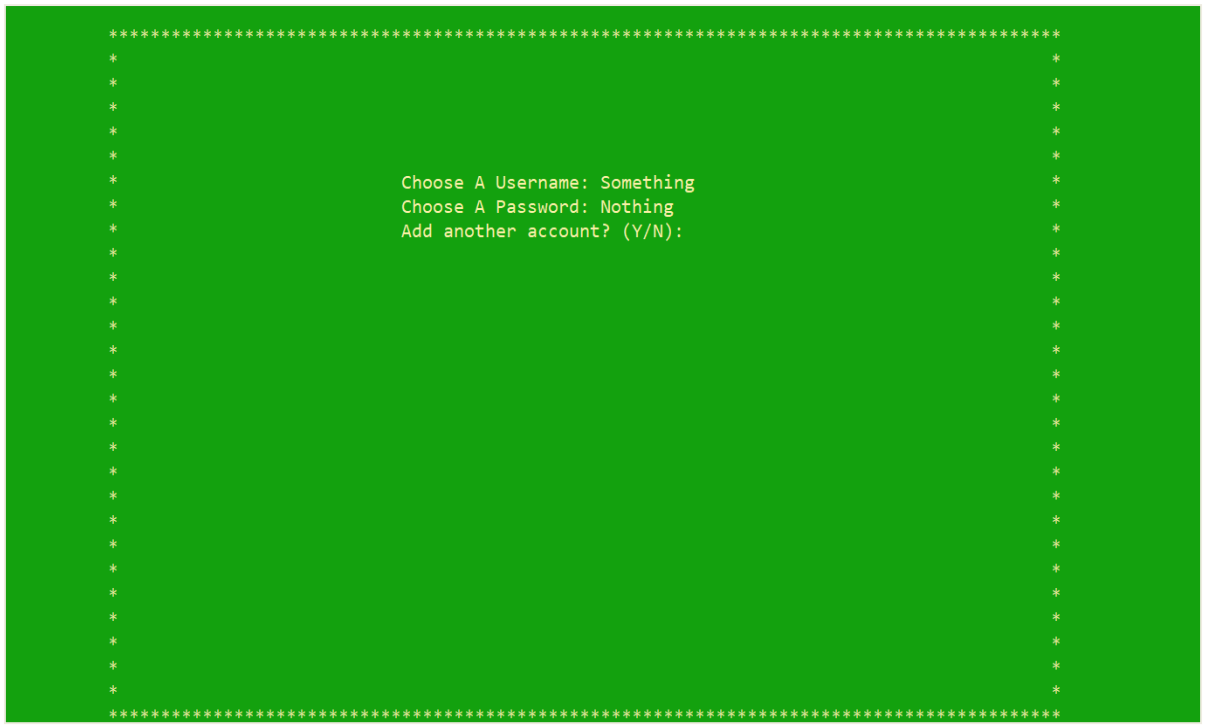
After testing and executing the project, following conclusions were drawn:

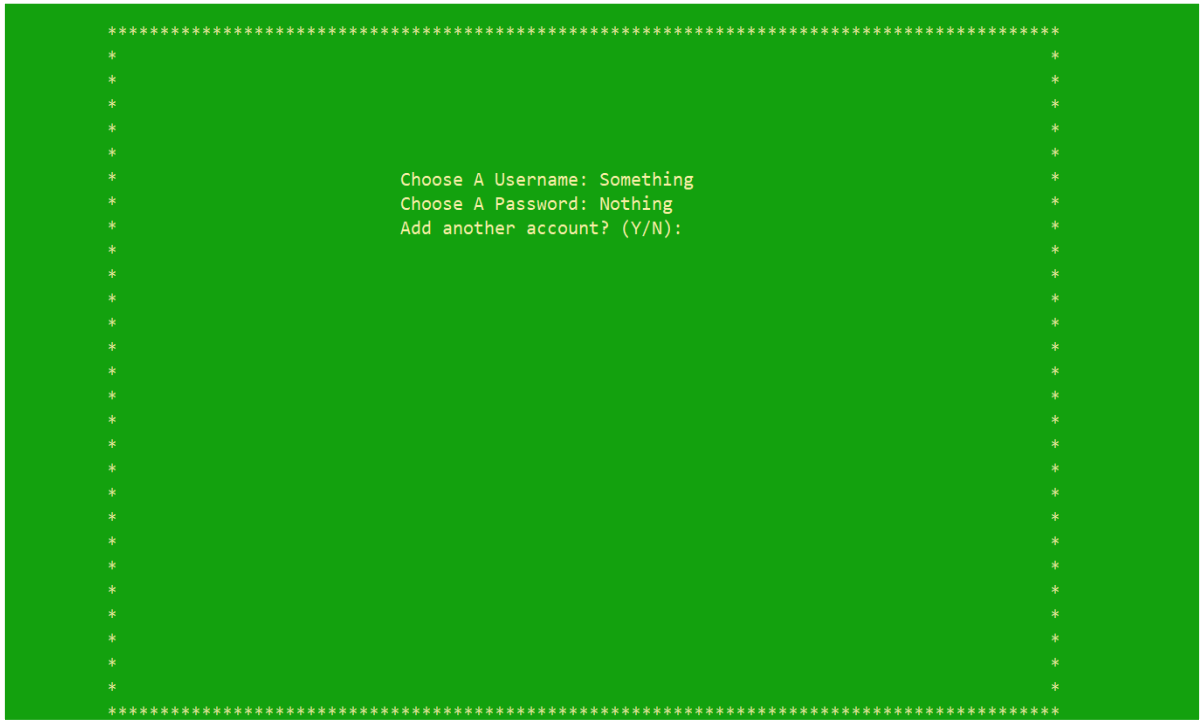
i. This program stores records of transaction of books of user whose username and password is registered.
ii. A user can search for books and find its exact position using book-map without having to search every bookshelf.
iii. This app makes library management more systematic and standard.
iv. It is quite easy to handle, so the librarian and user can easily obtain the desired information.

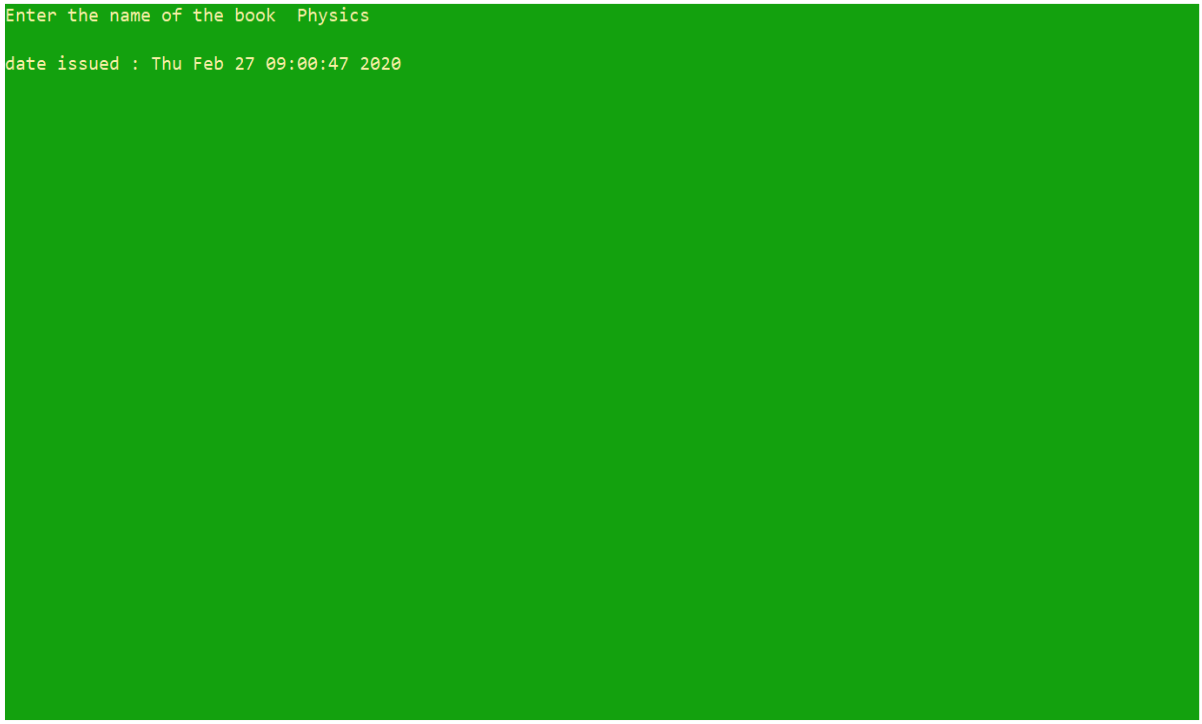
SNAPSHOT OF PROGRAM

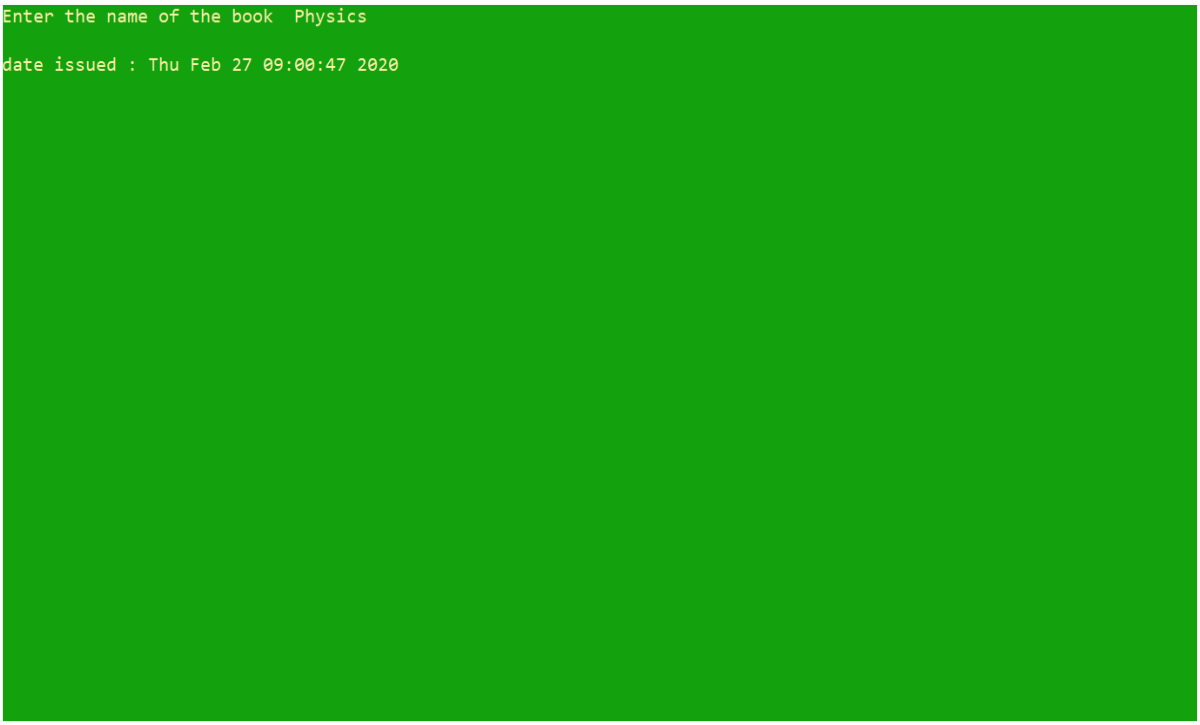



LIBRARY
Section A
Section B
Section C
Section D
Section E
Section F


Choose A Username: Something
Choose A Password: Nothing
Add another account? (Y/N):


Choose A Username: Something
Choose A Password: Nothing
Add another account? (Y/N):


Enter the name of the book Physics
date issued : Thu Feb 27 09:00:47 2020


Enter the name of the book Physics
date issued : Thu Feb 27 09:00:47 2020