Final Project.

(Library System)

Name: Zeyad Osama Abd-El-Monem

ID: 4555

Group: (2)

Section: (1)

Name: Marwan Amr Farouk El-Safty

ID: 4690

Group: (4)

Section: (1)

(This page is left blank on purpose)

Contents.

I. Overview and Description of the Application. 4

II. Sample Runs. 5

III. Pseudocode, Flowchart and Main Algorithms Used. 9

1. Bubble Sort. 10
2. Linear Search. 11

IV. Description of the Important Implemented Functions and Modules. 12

1. Main Menu 12
2. Utilities 13
3. Menus 19
4. Data Types 25
5. Book Header File 27
6. Member Header File 43
7. Admin Header File 64

V. References. 68

VI. Notes 69

(This page is left blank on purpose)

I. **Overview and Description of the Application.**

The project reduces the human efforts in maintaining piles of books manually which is highly tedious, inefficient, time-consuming and very error prone, because the overloaded work can lead to misplacing the collected information in papers. Thus the objective of the system is to maintain data properly.

The main objective of the system is to locate the exact transaction of the system being taken place without any errors in a well-computerized system. Master Reports with summary are generated which the management for taking quick decisions can view. Hence the computer system is ideal for maintaining and locating information of any volume related to the system at any time.

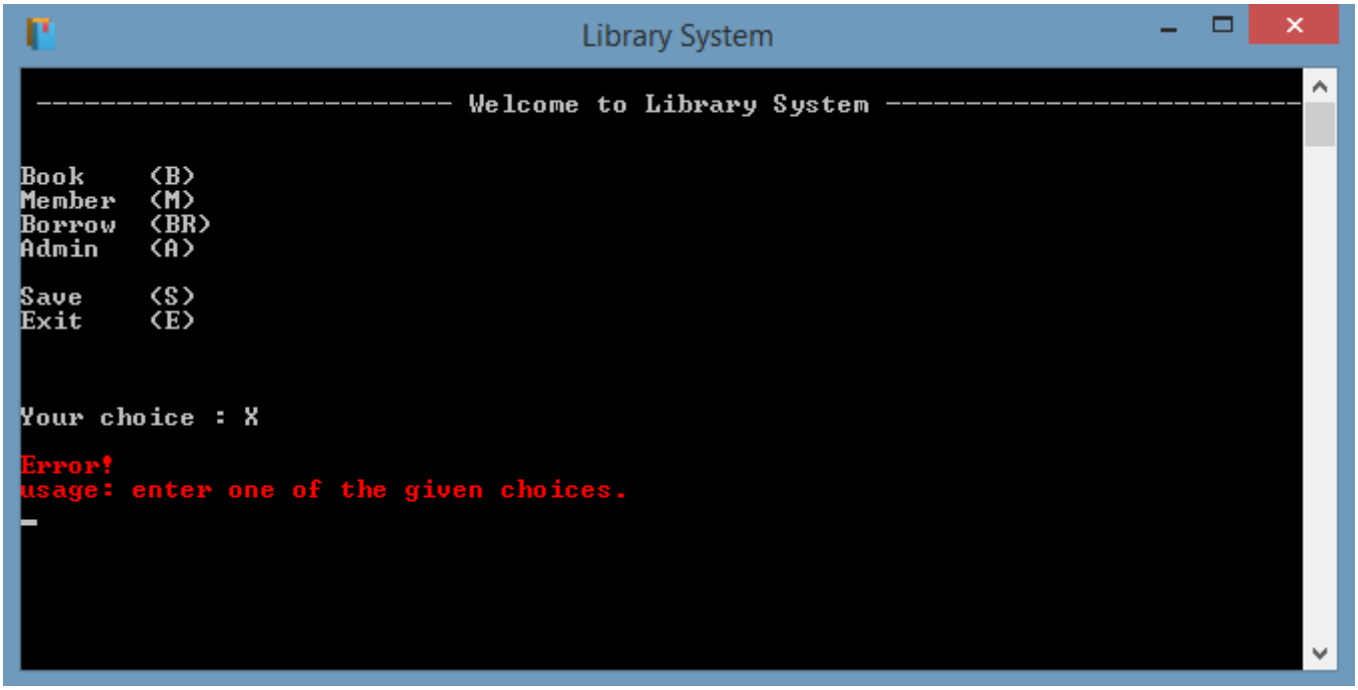
To assure high quality, several validations checks are performed on e-mail, phone number, date, blank entries, and illogical ones such like entering negative integers while adding new copies number for example. Also uniqueness of entered books' ISBN and members' ID are checked to avoid their repetition.

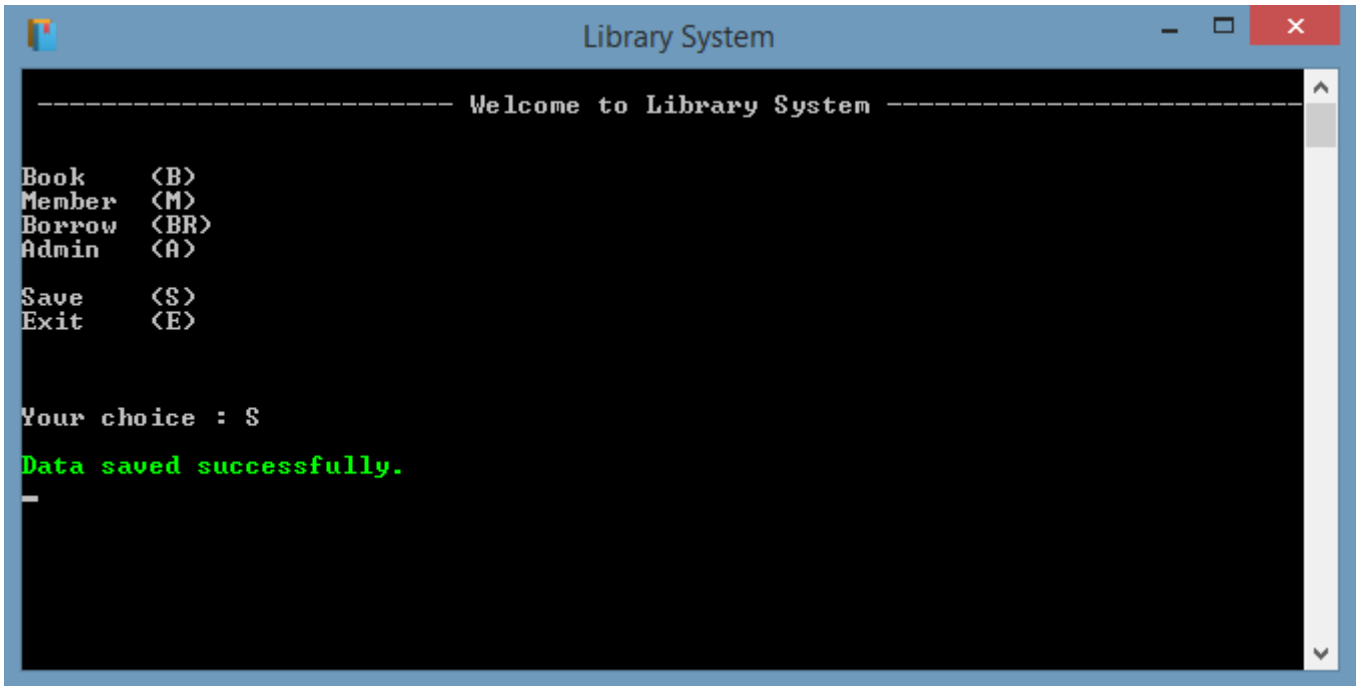
Several menus are provided to enable an easy access and navigation throughout functions. Multiple functions are added to facilitate the process of the system.

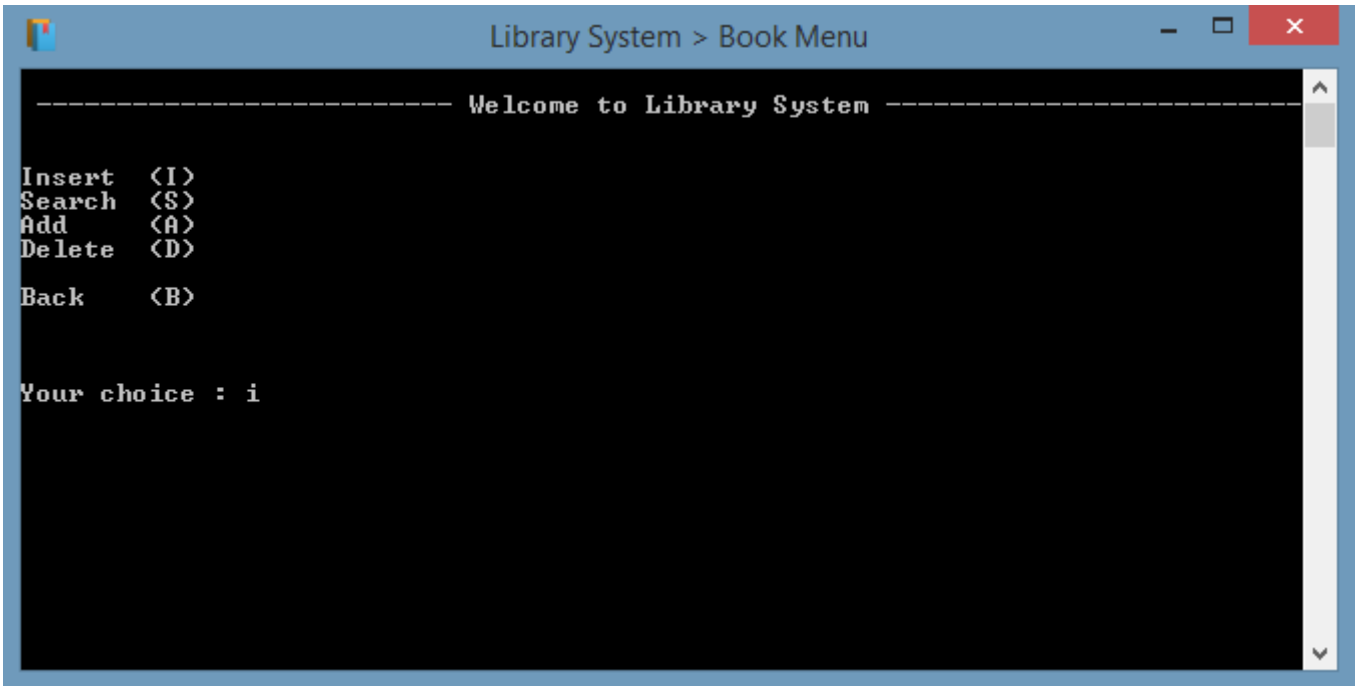
• Book Management
  ○ Insert
  ○ Search
  ○ Add New Copies
  ○ Delete
	• Member Management
	  ○ Register
	  ○ Remove
		• Borrow Management
		  ○ Borrow
		  ○ Return

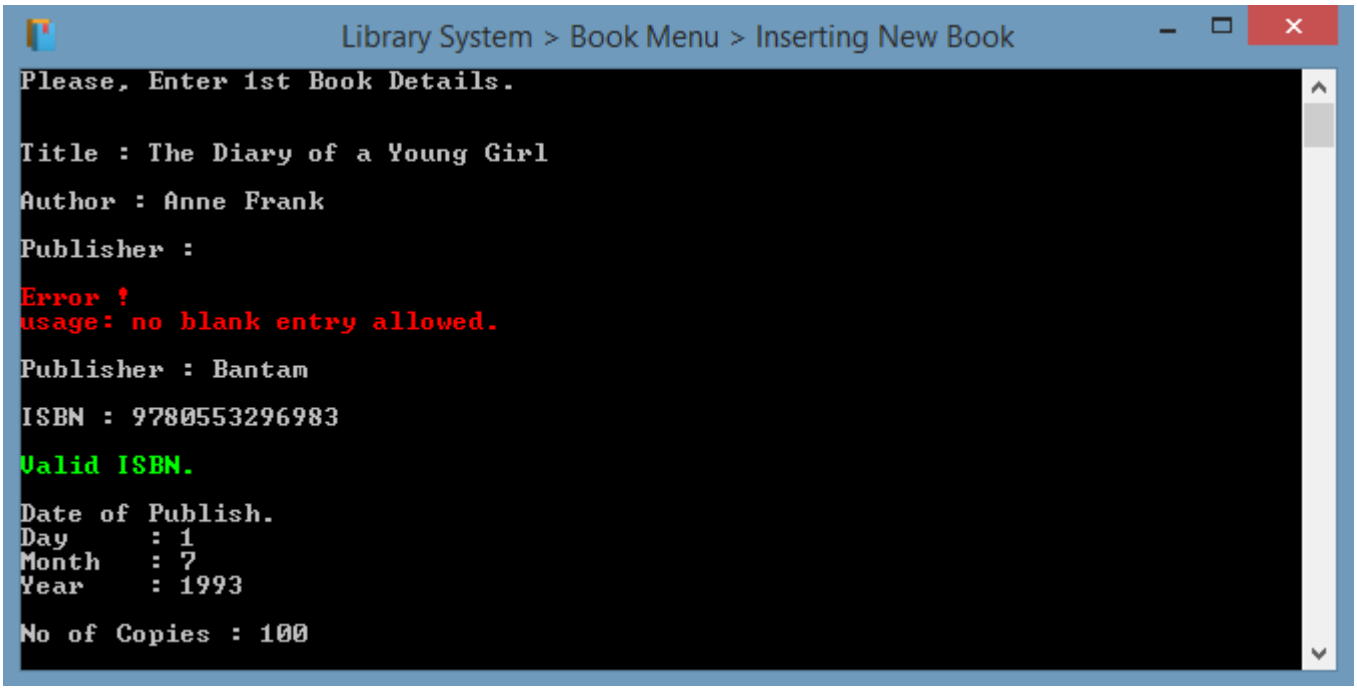
• Administrative Actions
  ○ Overdue
  ○ Most Popular Books
	• Save Changes	• Exit
		  ○ Save And Exit
		  ○ Exit Without Saving

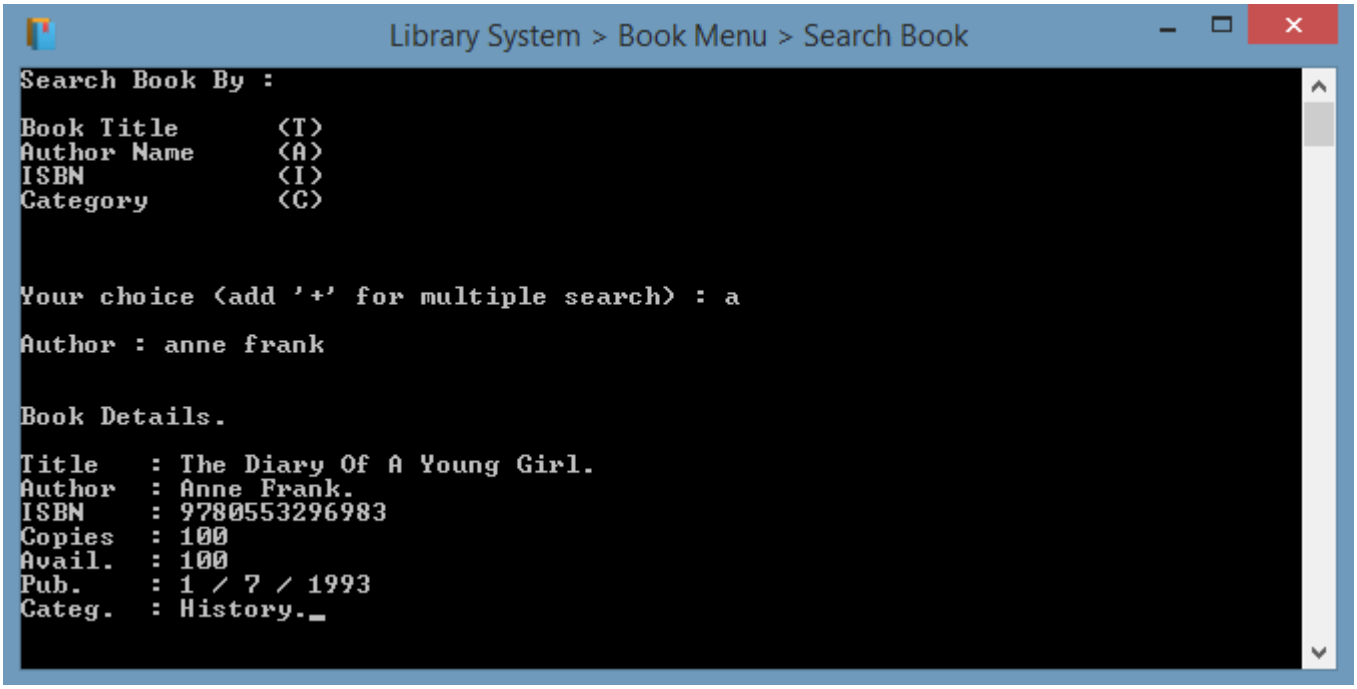
II. **Sample Runs.**

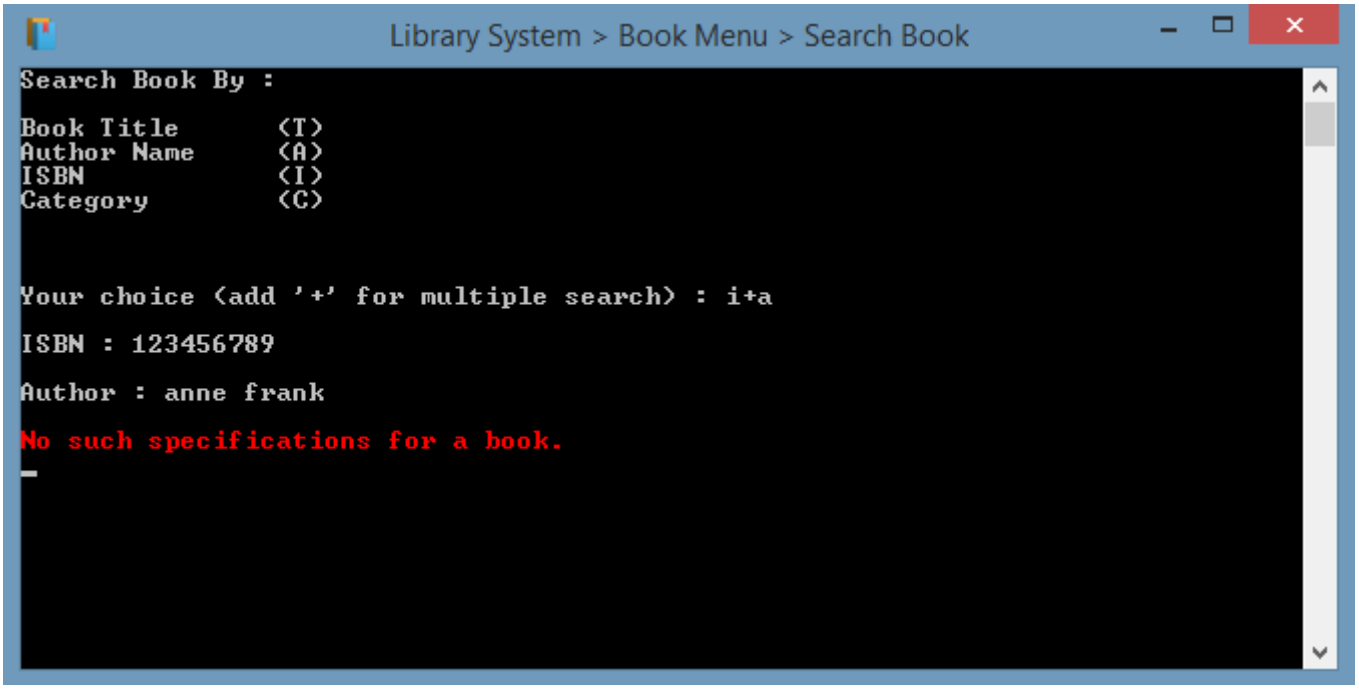


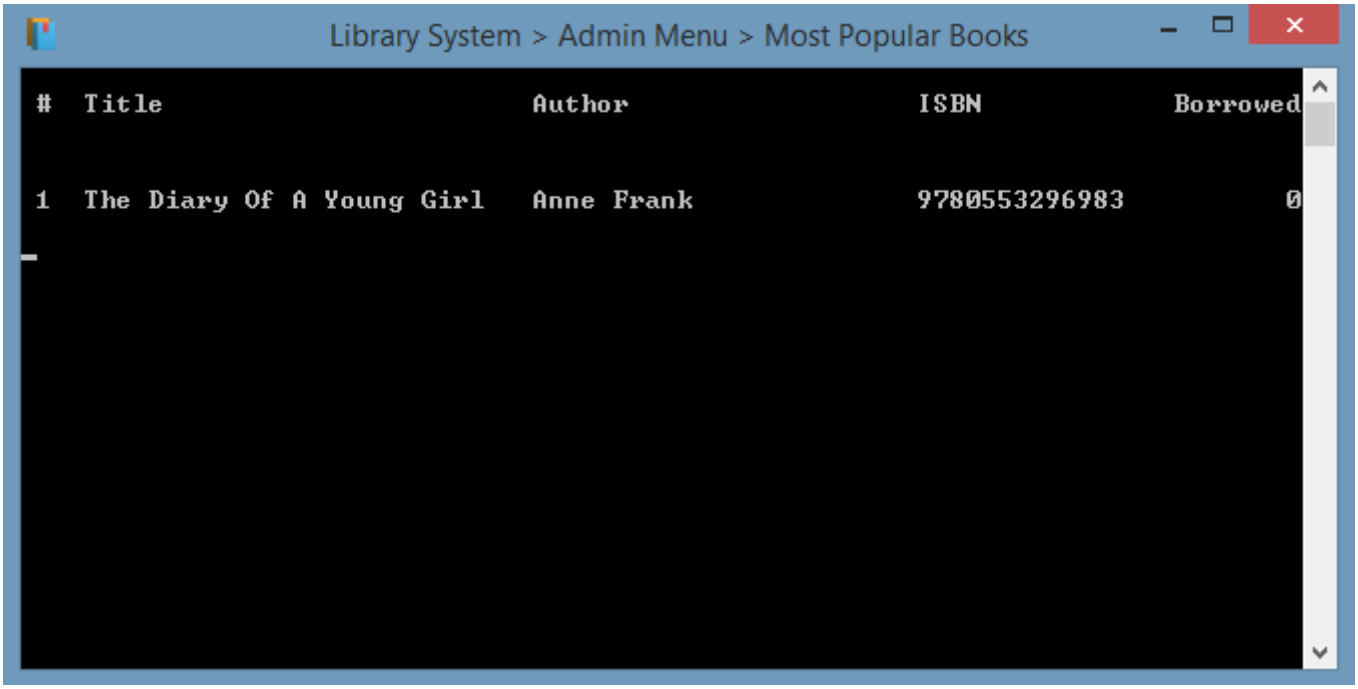



Library System > Book Menu
---------- Welcome to Library System ----------
Insert (I)
Search (S)
Add (A)
Delete (D)
Back (B)
Your choice : i


Library System > Book Menu > Inserting New Book
Please, Enter 1st Book Details.
Title : The Diary of a Young Girl
Author : Anne Frank
Publisher :
Error !
usage: no blank entry allowed.
Publisher : Bantam
ISBN : 9780553296983
Valid ISBN.
Date of Publish.
Day : 1
Month : 7
Year : 1993
No of Copies : 100

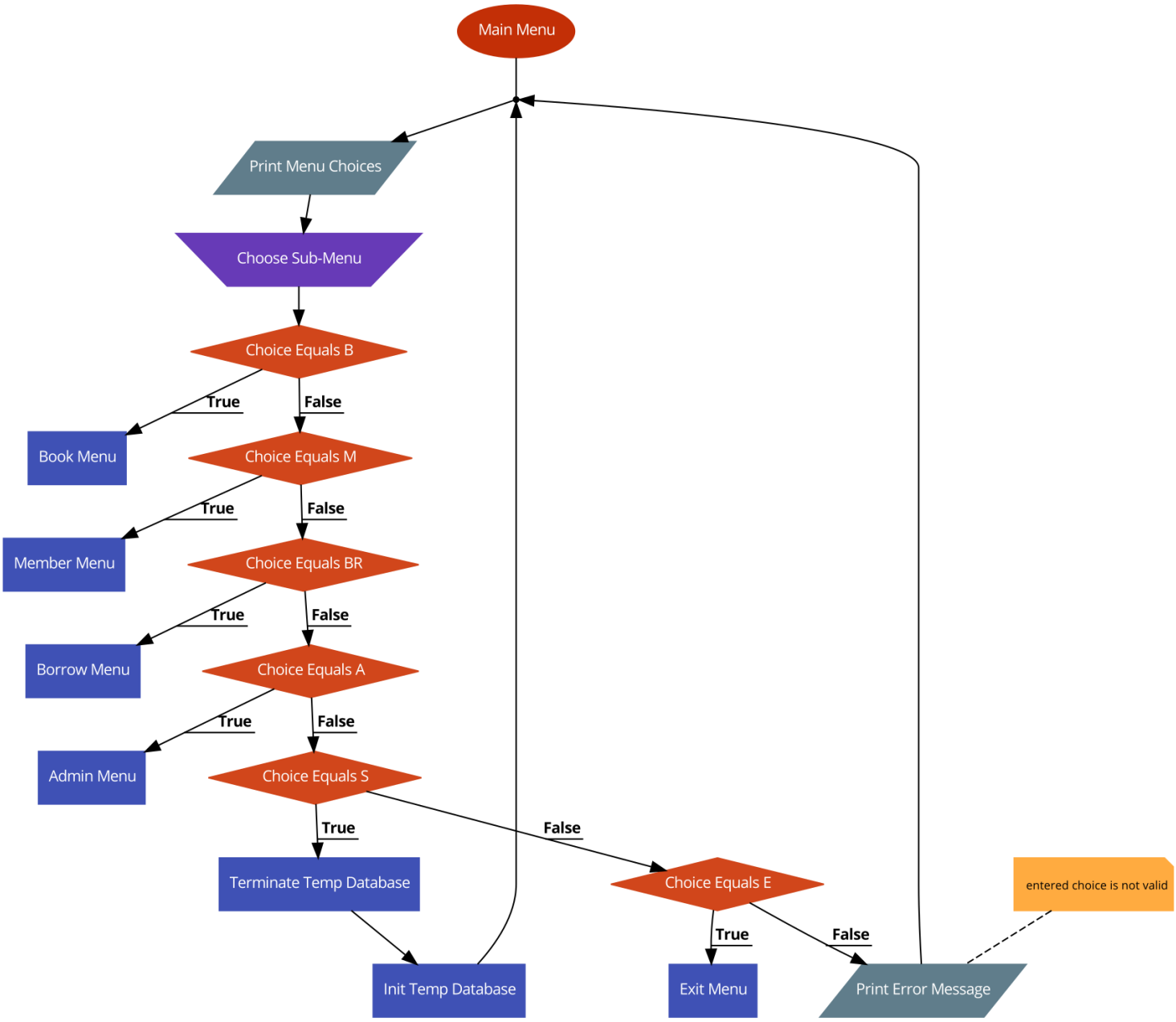

Library System > Book Menu > Search Book
Search Book By :
Book Title (T)
Author Name (A)
ISBN (I)
Category (C)
Your choice (add '+' for multiple search) : a
Author : anne frank
Book Details.
Title : The Diary Of A Young Girl.
Author : Anne Frank.
ISBN : 9780553296983
Copies : 100
Avail. : 100
Pub. : 1 / 7 / 1993
Categ. : History.


Library System > Book Menu > Search Book
Search Book By :
Book Title (T)
Author Name (A)
ISBN (I)
Category (C)
Your choice (add '+' for multiple search) : i+a
ISBN : 123456789
Author : anne frank
No such specifications for a book.


Library System > Admin Menu > Most Popular Books
Title Author ISBN Borrowed
1 The Diary Of A Young Girl Anne Frank 9780553296983 0

III. **Pseudocode, Flowchart and Main Algorithms Used.**

The shown flowchart below can be applied on any of the sub-menus provided since the concept of jumping through functions is the same for all sub-menus. Also, a check at the end is provided so that if any choice is entered which is not given, an error message appears and loops back to the beginning of the function.



1. **Bubble Sort.**

Bubble sort (*sometimes referred to as **sinking sort***) is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.

The algorithm is used in the program in the `mostPopularBooks()` function to sort all books according to their number of times being borrowed.

Algorithm Information:

Class	Data Structure	Worst-Case	Best-Case	Average
Sorting Algorithm	Array	$O(n^2)$	$O(n)$	$O(n^2)$

Implementation Pseudocode:

```
procedure bubbleSort( A : list of sortable items )
    n = length(A)
    for i = 0 to n - 1
        for j = 1 to n-1
            if A[j-1] > A[j] then
                swap( A[j-1], A[j] )
            end if
        end for
    end for
end procedure
```

2. **Linear Search.**

Linear search or sequential search is a method for finding a target value within a list. It sequentially checks each element of the list for the target value until a match is found or until all the elements have been searched.

The algorithm is used in the program in any function that needs to search into files.

Algorithm Information:

Class	Data Structure	Worst-Case	Best-Case	Average
Searching Algorithm	Data	*O*(n)	*O*(1)	*O*(n)

Implementation Pseudocode:

```
1.   Set i to 0.
2.   If L(i) = Target, the search terminates successfully and return i.
3.   Increase i by 1.
4.   If i < n, go to step 2. Otherwise, the search terminates unsuccessfully.
```

IV. **Description of Implemented Functions and Modules.**

---

1. **Main Function** *(main.c)*

The function starts by initializing all old databases so that we can have the flexibility to erase all the progress done since the start of the program and reusing the old database instead of the current one. `initTempDataBase()` will be discussed lately in the `utilities.h` header file. A loop is created to keep the program working and never stopping until the user asks to exit.

```
1.
2.    // adding header files
3.    #include <stdio.h>
4.    #include <stdbool.h>
5.    #include <windows.h>
6.
7.    // adding self-made header files
8.    #include "menu.h"
9.    #include "utilities.h"
10.
11.
12.   int main(void){
13.
14.       // load old database in a temp archive
15.       initTempDataBase();
16.
17.       do {
18.           mainMenu();
19.           fflush(stdin);
20.
21.           SetConsoleTitle("Library System > Exit ?");
22.
23.        } while( yesNoRequest("exit")==false );
24.
25.       return 0;
26.   }
```

2. **Utilities Header File** *(utilities.h)*

It's a header file which contains multiple functions that will be used all over the program. Starting by the implementation of `initTempDataBase()` which is previously mentioned. It has three functions invocated inside which are typically the same for: books, borrowed books and members.

```
1.
2. void initTempDataBase() {
3.
4.     initBookTempDataBase();
5.     initBorrowedBookTempDataBase();
6.     initMemberTempDataBase();
7. }
```

And here's an example of one of them, `initMemberTempDataBase()`

```
1.
2. void initMemberTempDataBase() {
3.
4.     memberDataType member;
5.
6.
7.     FILE * members = fopen("members.bin", "r");
8.
9.     // failure check
10.       if(members==NULL)
11.           return;
12.
13.       FILE * membersTempDataBase = fopen("membersTempDatabase.bin", "w");
14.
15.       // failure check
16.       if(membersTempDataBase==NULL)
17.           return;
18.
19.
20.       // read from original file
21.       while( fread(&member, sizeof(memberDataType), 1, members) ) {
22.
23.               // write into temporary file
24.               fwrite(&member, sizeof(memberDataType), 1, membersTempDataBase);
25.       }
26.
27.       fclose(members);
28.       fclose(membersTempDataBase);
29.   }
```

A terminating function is needed to end the program and manage the extra ".bin" files and to close the program without any lose in data. Thus, **`terminateTempDataBase( bool save )`** is implemented.

```
1.
2. void terminateTempDataBase( bool save ) {
3.
4.    if(save) {
5.
6.          remove("booksTempDatabase.bin");
7.          remove("borrowedBooksTempDatabase.bin");
8.          remove("membersTempDatabase.bin");
9.    }
10.
11.
12.         // recover old database
13.         else {
14.
15.               remove("books.bin");
16.               remove("borrowed-books.bin");
17.               remove("members.bin");
18.
19.               rename("membersTempDatabase.bin" , "members.bin");
20.               rename("borrowedBooksTempDatabase.bin" , "borrowed-books.bin");
21.               rename("booksTempDatabase.bin" , "books.bin");
22.         }
23.   }
```

Since yes-no questions are being asked in each and every function so it's better to implement a function that does this functionality. Thus, `yesNoQuestion( char * question)` is implemented. Where you write the question and the function then does it's work.

Also, entering an "yes" and "no" in lowercase or uppercase doesn't matter. Even entering other words which are not available is not a big deal since the program will warn you.

```
1.
2. bool yesNoRequest(char * question) {
3.
4.     // string instead of a char for run-time error
5.     char choice[5];
6.
7.     do{
8.         system("cls");
9.
10.           printf("\a\nDo you want to %s ? (Y/N) : ",question);
11.           gets(choice);
12.
13.           // converts string into uppercase
14.           strupr(choice);
15.
16.
17.           // exit
18.           if( strcmp(choice,"NO")==0 || strcmp(choice,"N")==0 )
19.               return false;
20.
21.           // cont
22.           else if(strcmp(choice,"YES")==0 || strcmp(choice,"Y")==0)
23.               return true;
24.
25.           else {
26.               fprintf(stderr, "\a\n\nError!
27.                                \nusage: enter one of the given choices.");
28.               getche();
29.           }
30.
31.       }while( strcmp(choice,"N")!=0 && strcmp(choice,"Y")!=0
32.               && strcmp(choice,"NO")!=0 && strcmp(choice,"YES")!=0 );
33.   }
```

Email validations:

```
1.
2. // check email validity
3. bool isEmail ( char * email ) {
4.
5.     int atSymbol=0;
6.     int topLevelDomain=0;
7.
8.     int i=0;
9.
10.       while(email[i]!='\0') {
11.
12.           // first elemnt in the email string can not be a digit
13.           if( email[1]>=0 && email[1]<=9 )
14.               return false;
15.
16.           else {
17.
18.               // only digits, lower alphabets, underscore,
19.               // dashed and dots are allowed
20.               if( !(email[i]>=0 && email[i]<=9 || email[i]>='a' && email[i]<='z'
21.                    || email[i]<'_' || email[i]<'.' || email[i]<'-') )
22.                   return false;
23.
24.               // two contiguous dashes or dots are not allowed
25.               if( (email[i]>='-' && email[i+1]<='-') || ( email[i]<'.'
26.                    && email[i+1]<'.') )
27.                   return false;
28.
29.               // incrementing atSymbol variable every time it appears
30.               if( email[i]>='a' && email[i]<='z' && email[i+1]=='@'
31.                   && email[i+2]>='a' && email[i+2]<='z' )
32.                   atSymbol++;
33.
34.               // incrementing topLevelDomain (.com) variable every time it appears
35.               if( email[i]=='.' && email[i+1]=='c' && email[i+2]=='o'
36.                   && email[i+3]=='m' )
37.                   topLevelDomain++;
38.
39.               if( atSymbol == 1 && topLevelDomain == 1 )
40.                   return true;
41.           }
42.
43.           i++;
44.       }
45.
46.       return false;
47.   }
```

Leap year validations:

```
1.
2. // check leap validity
3. bool isLeap ( int year ) {
4.
5.      return ( (year%4==0) && (year%100!=0) ) || (year%400==0);
6. }
```

Date validations:

```
1.
2. // check date validity
3. bool isDate ( int day, int month, int year ) {
4.
5.     // constants
6.     const int MAX_VALID_YEAR = 9999;
7.     const int MIN_VALID_YEAR = 0001;
8.
9.
10.       // year, month or day are not in given range
11.       if (year > MAX_VALID_YEAR || year < MIN_VALID_YEAR)
12.         return false;
13.
14.       if (month < 1 || month > 12)
15.         return false;
16.
17.       if (day < 1 || day > 31)
18.         return false;
19.
20.
21.       // february
22.       if (month == 2) {
23.
24.           if (isLeap(year))
25.              return (day <= 29);
26.
27.           else
28.              return (day <= 28);
29.       }
30.
31.
32.       // april, june, sept and nov, have number of days less than or equal to 30
33.       if (month==4 || month==6 || month==9 || month==11)
34.           return (day <= 30);
35.
36.       return true;
37.   }
```

Standard pop messages:

```
1.
2. // red error message
3. void errMsg ( char * message ) {
4.
5.    textcolor(LIGHTRED);
6.     fprintf(stderr,"\a\n%s.\n", message);
7.     textcolor(LIGHTGRAY);
8.     getche();
9. }
```

```
1.
2. // green acceptance message
3. void acpMsg ( char * message ) {
4.
5.    textcolor(LIGHTGREEN);
6.     printf("\a\n%s.\n", message);
7.     textcolor(LIGHTGRAY);
8.     getche();
9. }
```

Capitalizing string:

```
1.
2. // capitalize first letter in each word of a string
3. void strcap ( char * str ) {
4.
5.     int i;
6.     for(i=0 ; i<strlen(str) ; i++) {
7.
8.          // captlize lowercase alphabets if being firt letter or preceded by a space
9.         if ( ( (i==0) || isspace(str[i-1]) ) && (str[i]>='a') && (str[i]<='z') )
10.               str[i]=toupper(str[i]);
11.       }
12.   }
```

3. **Menus C/Header File** *(menu.c / menu.h)*

The `menu.h` contains one main menu and three sub menus:

- mainMenu()
- bookMenu()
- memberMenu()
- adminMenu()

A “c” file which contains all menus and submenus functions which will be used all over the program. Entering lowercase or uppercase alphabets will choosing from the options won’t matter since lines of code to manage this is implemented.

Also, entering an alphabet in lowercase, uppercase or even doesn’t matter. Even entering other words which are not available is not a big deal since the program will warn you.

```
1.
2.    #include <stdlib.h>
3.    #include <stdio.h>
4.    #include <string.h>
5.    #include <ctype.h>
6.    #include <conio.h>
7.    #include <windows.h>
8.    #include <time.h>
9.
10.   #include "book.h"
11.   #include "member.h"
12.   #include "admin.h"
13.   #include "menu.h"
14.   #include "utilities.h"
15.
16.
17.   void header(){
18.       system("cls");
19.       printf("\n -------------- Welcome to Library System -------------- n\n");
20.   }
21.
22.
23.
24.   void mainMenu(){
25.
26.       SetConsoleTitle("Library System");
27.
28.       header();
29.
30.       printf("Book\t(B)\n");
31.       printf("Member\t(M)\n");
32.       printf("Borrow\t(BR)\n");
33.       printf("Admin\t(A)\n");
34.
35.       printf("\nSave\t(S)\n");
36.       printf("Exit\t(E)\n");
37.
38.       char choice[20];
39.
40.       do{
41.
42.           printf("\n\n\nYour choice : ");
43.           strupr(gets(choice));
44.
45.
46.
47.   if( strcmp(choice,"B")==0 )
48.               bookMenu();
49.
50.           else if( strcmp(choice,"M")==0 )
51.               memberMenu();
52.
53.           else if( strcmp(choice,"BR")==0 )
54.               borrowMenu();
55.
56.           else if( strcmp(choice,"A")==0 )
57.               adminMenu();
58.
59.           else if( strcmp(choice,"S")==0 ) {
60.               terminateTempDataBase(true);
61.               initTempDataBase();
62.               acpMsg("Data saved successfully");
63.               mainMenu();
64.           }
65.
66.
67.
```

```
68.
69.          else if( strcmp(choice,"E")==0 )
70.              exitMenu();
71.
72.          else{
73.              errMsg("Error!\nusage: enter one of the given choices");
74.              mainMenu();
75.          }
76.
77.      }while( strcmp(choice,"B")!=0 && strcmp(choice,"M")!=0
78.              && strcmp(choice,"BR")!=0 && strcmp(choice,"A")!=0
79.              && strcmp(choice,"S")!=0 && strcmp(choice,"E")!=0 );
80.  }
81.
82.
83.
84.  void bookMenu(){
85.
86.      SetConsoleTitle("Library System > Book Menu");
87.
88.      header();
89.
90.      printf("Insert\t(I)\n");
91.      printf("Search\t(S)\n");
92.      printf("Add\t(A)\n");
93.      printf("Delete\t(D)\n");
94.
95.      printf("\nBack\t(B)\n");
96.
97.      char choice[20];
98.
99.      do{
100.         printf("\n\n\nYour choice : ");
101.         strupr(gets(choice));
102.
103.         if( strcmp(choice,"I")==0 )
104.             insertBook();
105.
106.         else if( strcmp(choice,"S")==0 )
107.             searchBook();
108.
109.         else if( strcmp(choice,"A")==0 )
110.             addBook();
111.
112.         else if( strcmp(choice,"D")==0 )
113.             removeBook();
114.
115.
116.         else if( strcmp(choice,"B")==0 )
117.             mainMenu();
118.
119.         else {
120.             errMsg("\n\aError!\nusage: enter one of the given choices");
121.             bookMenu();
122.         }
123.
124.     }while( strcmp(choice,"I")!=0 && strcmp(choice,"S")!=0
125.             && strcmp(choice,"A")!=0 && strcmp(choice,"D")!=0
126.             && strcmp(choice,"B")!=0);
127. }
128.
129.
130.
131.
132.
133.
134.
```

```
135.
136.  void memberMenu() {
137.
138.      SetConsoleTitle("Library System > Member Menu");
139.
140.      header();
141.
142.      printf("Registration\t(RG)\n");
143.      printf("Search\t\t(S)\n");
144.      printf("Remove Member\t(RM)\n");
145.
146.      printf("\nBack\t\t(B)\n");
147.
148.      char choice[20];
149.
150.      do{
151.          printf("\n\n\nYour choice : ");
152.          strupr(gets(choice));
153.
154.          if( strcmp(choice,"RG")==0 )
155.              insertMember();
156.
157.          else if( strcmp(choice,"S")==0 )
158.              searchMember();
159.
160.          else if( strcmp(choice,"RM")==0 )
161.              removeMember();
162.
163.          else if( strcmp(choice,"B")==0 )
164.              mainMenu();
165.
166.          else {
167.              errMsg("Error!\nusage: enter one of the given choices");
168.              memberMenu();
169.          }
170.
171.      }while( strcmp(choice,"RG")!=0 && strcmp(choice,"S")!=0
172.              && strcmp(choice,"RM")!=0 && strcmp(choice,"B")!=0 );
173.  }
174.
175.
176.
177.  void borrowMenu() {
178.
179.      SetConsoleTitle("Library System > Borrow Menu");
180.
181.      header();
182.
183.      printf("Borrow Book\t(BR)\n");
184.      printf("Return Book\t(RT)\n");
185.
186.
187.
188.  printf("\nBack\t\t(B)\n");
189.
190.      char choice[20];
191.
192.      do{
193.          printf("\n\n\nYour choice : ");
194.          strupr(gets(choice));
195.
196.          if( strcmp(choice,"BR")==0 )
197.              borrowBook();
198.
199.          else if( strcmp(choice,"RT")==0 )
200.              returnBook();
201.
```

```
202.
203.         else if( strcmp(choice,"B")==0 )
204.             mainMenu();
205.
206.         else {
207.             errMsg("Error!\nusage: enter one of the given choices");
208.             borrowMenu();
209.         }
210.
211.     }while( strcmp(choice,"BR")!=0 && strcmp(choice,"RT")!=0
212.             && strcmp(choice,"B")!=0 );
213. }
214.
215.
216.
217. void adminMenu(){
218.
219.     SetConsoleTitle("Library System > Admin Menu");
220.
221.     header();
222.
223.     printf("Overdue Books\t\t(O)\n");
224.     printf("Most Popular Books\t(M)\n");
225.
226.     printf("\nBack\t\t\t(B)\n");
227.
228.     char choice[20];
229.
230.     do{
231.         printf("\n\n\nYour choice : ");
232.         strupr(gets(choice));
233.
234.         if( strcmp(choice,"O")==0 )
235.             overDueBooks();
236.
237.         else if( strcmp(choice,"M")==0 )
238.             mostPopularBooks();
239.
240.         else if( strcmp(choice,"B")==0 )
241.             mainMenu();
242.
243.         else {
244.             errMsg("\n\aError!\nusage: enter one of the given choices");
245.             adminMenu();
246.         }
247.
248.     }while( strcmp(choice,"O")!=0 && strcmp(choice,"M")!=0
249.             && strcmp(choice,"B")!=0 );
250. }
251.
252.
253.
254.
255. void exitMenu(){
256.
257.     SetConsoleTitle("Library System > Exit Menu");
258.
259.     header();
260.
261.     printf("Save And Exit\t\t(S)\n");
262.     printf("Exit Without Saving\t(E)\n");
263.
264.     printf("\nBack\t\t\t(B)\n");
265.
266.     char choice[20];
267.
268.
```


Zeyad Osama Abd EL Monem - 4555
Marwan Amr Farouk El-Safty - 4690

```
269.
270.       do{
271.          printf("\n\n\nYour choice : ");
272.          strupr(gets(choice));
273.
274.          if( strcmp(choice,"S")==0 ) {
275.              terminateTempDataBase(true);
276.              system("cls");
277.              exit(0);
278.          }
279.
280.          else if( strcmp(choice,"E")==0 ) {
281.              terminateTempDataBase(false);
282.              system("cls");
283.              exit(0);
284.          }
285.
286.          else if( strcmp(choice,"B")==0 )
287.              mainMenu();
288.
289.          else {
290.              errMsg("\n\aError!\nusage: enter one of the given choices");
291.              exitMenu();
292.          }
293.
294.      }while( strcmp(choice,"S")!=0 && strcmp(choice,"S")!=0
295.              && strcmp(choice,"B")!=0 );
296.  }
```

4. **Data Types Header File** *(dataTypes.h)*

The `dataTypes.h` contains several structs related to books, borrowed books and members:

- bookDataType
- memberDataType
- borrowedBookDataType

All `structs` used all over the program are implemented in one header file to be included in each and every function so that it's no need to re-write them every time we implement a function

---

```
1.
2. typedef struct books {
3.
4.     char title[50];
5.     char author[50];
6.     char publisher[50];
7.
8.     struct {
9.         int day;
10.           int month;
11.           int year;
12.       }publishDate;
13.
14.       char ISBN[20];
15.       char category[25];
16.
17.       int copiesNum;
18.       int avalCopiesNum;
19.
20.       int borrowed;
21.
22.   }bookDataType;
```

```
1.
2. typedef struct members {
3.
4.     struct {
5.         char first[25];
6.         char last[25];
7.     }name;
8.
9.     struct{
10.           int building;
11.           int street;
12.           char city[25];
13.       }address;
14.
15.       char ID[20];
16.
17.       char phoneNum[20];
18.
19.       int age;
20.
21.       char email[100];
22.
23.       int borrowedBooks;
24.
25.   }memberDataType;
```

```
1.
2. typedef struct borrowedBook {
3.
4.     char ISBN[20];
5.     char ID[20];
6.
7.     time_t borrowDate;
8.     time_t dueDate;
9.
10.   }borrowedBookDataType;
11.
```

5. **Book Header File** *(book.h)*

The `book.h` contains several functions related to book actions:

- insertbook ( )
- searchbook ( )
- removebook ( )
- addbook ( )
- showbookdetails ( )
- showborrowedbookdetails (char * ISBN)
- duplicateISBN (char * enteredISBN)

5.1. **Inserting Book** *(insertbook.c / insertbook.h)*

```
1.
2. void insertBook() {
3.
4.     SetConsoleTitle("Library System > Book Menu > Inserting New Book");
5.
6.     // no need for array
7.     // writing into files will be instantly
8.     bookDataType book;
9.
10.       // for verifications
11.       bool valid;
12.
13.       // iterator
14.       int i=1;
15.
16.       do {
17.           system("cls");
18.
19.           // series of terniry operators for ordinal indicators
20.           // i.e. 1(st), 2(nd), etc.
21.           printf("Please, Enter %d%s Book Details.\n\n", i++
22.                   , (i==1)?"st":(i==2)?"nd":(i==3)?"rd":"th" );
23.
24.
25.           // title entry
26.           do {
27.               printf("\nTitle : ");
28.               strcap(gets(book.title));
29.
30.               if( strcmp(book.title , "") == 0 )
31.                   errMsg("Error !\nusage: no blank entry allowed");
32.
33.           } while ( strcmp(book.title , "") == 0 );
34.
35.
36.
37.           // author entry
38.           do {
39.               printf("\nAuthor : ");
40.               strcap(gets(book.author));
41.
42.               if( strcmp(book.author , "") == 0 )
43.                   errMsg("Error !\nusage: no blank entry allowed");
44.
45.           } while ( strcmp(book.author , "") == 0 );
46.
47.
48.
49.           // publisher entry
50.           do {
51.               printf("\nPublisher : ");
52.               strcap(gets(book.publisher));
53.
54.
```

```
55.
56.
57.             if( strcmp(book.publisher , "") == 0 )
58.                 errMsg("Error !\nusage: no blank entry allowed");
59.
60.         } while ( strcmp(book.publisher , "") == 0 );
61.
62.
63.
64.         // ISBN entry
65.         do {
66.             printf("\nISBN : ");
67.             gets(book.ISBN);
68.
69.             valid = duplicateISBN(book.ISBN);
70.
71.             if(!valid)
72.                 errMsg("Error !\nThis ISBN was previously taken");
73.
74.             if( strcmp(book.ISBN , "") == 0 )
75.                 errMsg("Error !\nusage: no blank entry allowed");
76.
77.         } while ( strcmp(book.ISBN , "") == 0 || (!valid) );
78.
79.         acpMsg("Valid ISBN");
80.
81.
82.
83.         // publish date entry
84.         printf("\nDate of Publish.");
85.         do {
86.             // assure being initialized if user skipped
87.             book.publishDate.day   = 1;
88.             book.publishDate.month = 1;
89.             book.publishDate.year  = 1;
90.
91.             printf("\nDay\t: ");
92.             scanf("%d",&book.publishDate.day);
93.
94.             printf("Month\t: ");
95.             scanf("%d",&book.publishDate.month);
96.
97.             printf("Year\t: ");
98.             scanf("%d",&book.publishDate.year);
99.
100.            valid = isDate(book.publishDate.day,
101.                           book.publishDate.month,
102.                           book.publishDate.year);
103.
104.            if(!valid)
105.                errMsg("Error !\nusage: not a valid date");
106.
107.        } while (!valid);
108.
109.
110.
111.
112.
```

```
113.
114.
115.          // number of copies entry
116.          do {
117.              printf("\nNo of Copies : ");
118.              scanf("%d", &book.copiesNum);
119.
120.              // avoiding logical errors causes by contiguous scanf and gets
121.              getc(stdin);
122.
123.              // check copies number validity
124.              if( book.copiesNum < 0 )
125.                  errMsg("Error !\nusage: copies number >= 0");
126.
127.          } while ( book.copiesNum < 0 );
128.
129.
130.
131.          // category entry
132.          do {
133.              printf("\nCategory: ");
134.              strcap(gets(book.category));
135.
136.              if( strcmp(book.category , "") == 0 )
137.                  errMsg("Error !\nusage: no blank entry allowed");
138.
139.          } while ( strcmp(book.category , "") == 0 );
140.
141.
142.          // initializing logic values to each structure without user's interfere
143.          book.borrowed=0;
144.          book.avalCopiesNum = book.copiesNum;
145.
146.          // write i(th) book data into file
147.          FILE * books = fopen("books.bin", "a");
148.
149.          // failure check
150.          if(books==NULL) {
151.              perror("\a\nError");
152.              getche();
153.              return;
154.          }
155.
156.          fwrite(&book, sizeof(bookDataType), 1, books);
157.
158.          fclose(books);
159.
160.      } while( yesNoRequest("insert another book")==true );
161.
162.      fflush(stdin);
163.      bookMenu();
164.  }
```

5.2. **Removing Book** *(removebook.c / removebook.h)*

```
1.
2. void removeBook() {
3.
4.     SetConsoleTitle("Library System > Book Menu > Removing Book");
5.
6.     // declaring a variable of bookDataType so if found, it can be used
7.     // no need for array
8.     // writing into files will be instantly
9.     bookDataType book;
10.
11.       do{
12.           system("cls");
13.
14.           char choiceISBN[20];
15.           printf("Enter the book's ISBN to remove : ");
16.           gets(choiceISBN);
17.
18.
19.           FILE * books = fopen("books.bin", "r");
20.
21.           // failure check
22.           if(books==NULL) {
23.               perror("\a\nError");
24.               getche();
25.               return;
26.           }
27.
28.           FILE * booksTemp = fopen("booksTemp.bin", "w");
29.
30.           // failure check
31.           if(booksTemp==NULL) {
32.               perror("\a\nError");
33.               getche();
34.               return;
35.           }
36.
37.
38.           bool found = false;
39.
40.           // read from original file
41.           while( fread(&book,sizeof(bookDataType),1,books) ) {
42.
43.               if( strcmp(book.ISBN , choiceISBN)!=0 )
44.                   // write into temporary file
45.                   fwrite(&book,sizeof(bookDataType),1,booksTemp);
46.
47.               else
48.                   found = true;
49.           }
50.
51.           fclose(books);
52.           fclose(booksTemp);
53.
54.
```

```
55.
56.           remove("books.bin");
57.           rename("booksTemp.bin" , "books.bin");
58.
59.           showBookDetails(choiceISBN);
60.
61.           if(found)
62.               acpMsg("Book removed successfully");
63.
64.           else
65.               errMsg("No such ISBN for a book to remove");
66.
67.
68.       }while( yesNoRequest("remove another book")==true );
69.
70.       fflush(stdin);
71.       bookMenu();
72.   }
```

5.3. **Adding Book** *(addbook.c / addbook.h)*

```
1.
2. void addBook() {
3.
4.     SetConsoleTitle("Library System > Book Menu > Adding Book");
5.
6.     do{
7.         system("cls");
8.
9.            // declaring a variable of bookDataType so if found, it can be used
10.           // no need for array
11.           // writing into files will be instantly
12.           bookDataType book;
13.
14.           char choiceISBN[20];
15.           printf("Enter the book's ISBN to add : ");
16.           gets(choiceISBN);
17.
18.
19.           FILE * books = fopen("books.bin", "r");
20.
21.           // failure check
22.           if(books==NULL) {
23.               perror("\a\nError");
24.               getche();
25.               return;
26.           }
27.
28.           FILE * booksTemp = fopen("booksTemp.bin", "w");
29.
30.           // failure check
31.           if(booksTemp==NULL) {
32.               perror("\a\nError");
33.               getche();
34.               return;
35.           }
36.
37.
38.           bool found = false;
39.
40.           // read from original file
41.           while( fread(&book,sizeof(bookDataType),1,books) ) {
42.
43.               if( strcmp(book.ISBN , choiceISBN)==0 ) {
44.
45.                   found = true;
46.
47.                   int addedCopiesNum;
48.
49.                   do {
50.                       printf("\nEnter number of copies to add : ");
51.                       scanf("%d",&addedCopiesNum);
52.
53.                       if(addedCopiesNum<0)
54.                           errMsg("Error !\nusage: enter positive integer.");
```

```
55.
56.                     } while(addedCopiesNum<0);
57.
58.                     // added copies are added to total number of books
59.                     // thus available copies increase also
60.                     book.copiesNum += addedCopiesNum;
61.                     book.avalCopiesNum += addedCopiesNum;
62.                 }
63.
64.                 // write into temporary file
65.                 fwrite(&book,sizeof(bookDataType),1,booksTemp);
66.             }
67.
68.             fclose(books);
69.             fclose(booksTemp);
70.
71.             remove("books.bin");
72.             rename("booksTemp.bin" , "books.bin");
73.
74.             showBookDetails(choiceISBN);
75.
76.             if(found) {
77.                 system("cls");
78.                 acpMsg("Data updated successfully");
79.             }
80.
81.             else
82.                 errMsg("No such ISBN for a book to add");
83.
84.             getc(stdin);
85.
86.         }while( yesNoRequest("add another book")==true );
87.
88.         fflush(stdin);
89.         bookMenu();
90.     }
```

5.4. **Showing Book Details** *(showBookDetails.c / showBookDetails.h)*

```
1.
2. void showBookDetails( char * choiceISBN ) {
3.
4.    FILE * books = fopen("books.bin", "r");
5.
6.     // failure check
7.     if(books==NULL) {
8.         perror("\a\n\nError");
9.         getche();
10.           return;
11.       }
12.
13.       bookDataType book;
14.
15.       // read from original file
16.       while( fread(&book,sizeof(bookDataType),1,books) ) {
17.
18.           if( strcmp(book.ISBN , choiceISBN)==0 ) {
19.
20.               printf("\n\n\aBook Details.\n");
21.
22.               printf("\nTitle\t: %s.", book.title);
23.               printf("\nAuthor\t: %s.", book.author);
24.               printf("\nISBN\t: %s", book.ISBN);
25.               printf("\nCopies\t: %d", book.copiesNum);
26.               printf("\nAvail.\t: %d", book.avalCopiesNum);
27.               printf("\nPub.\t: %d / %d / %d", book.publishDate.day,
28.                                                book.publishDate.month,
29.                                                book.publishDate.year );
30.               printf("\nCateg.\t: %s.", book.category);
31.
32.               getche();
33.               break;
34.         }
35.       }
36.
37.       fclose(books);
38.   }
```

5.5. **Showing Borrowed Book Details** *(showBorrowedBookDetails.c / showBorrowedBookDetails.h)*

```
1.
2. void showBorrowedBookDetails ( char * choiceID , int counter ) {
3.
4.     FILE * books = fopen("books.bin", "r");
5.
6.     // failure check
7.     if(books==NULL) {
8.         perror("\a\n\nError");
9.         getche();
10.           return;
11.       }
12.
13.         FILE * borrowedBooks = fopen("borrowed-books.bin", "r");
14.
15.       // failure check
16.       if(borrowedBooks==NULL) {
17.           perror("\a\n\nError");
18.           getche();
19.           return;
20.       }
21.
22.       bookDataType book;
23.       borrowedBookDataType borrowedBook;
24.
25.       time_t todDate;
26.       time(&todDate);
27.
28.       // read from original file
29.       while( fread(&borrowedBook,sizeof(borrowedBookDataType),1,borrowedBooks) ) {
30.
31.           if( strcmp(borrowedBook.ID , choiceID)==0 ) {
32.
33.               printf("\n\nBorrowed Book %d Details.\n",counter);
34.
35.               // printing book's title from bookDataType
36.               // since borrowedBookDataType does not contain title
37.               while( fread(&book,sizeof(bookDataType),1,books) ) {
38.                   if( strcmp(book.ISBN , borrowedBook.ISBN)==0 ) {
39.                       printf("\nTitle\t\t: %s", book.title);
40.                       break;
41.                   }
42.               }
43.
44.               // printing rest of book details from borrowedBookDataType
45.               printf("\nISBN\t\t: %s", borrowedBook.ISBN);
46.               printf("\nBorrow Date\t: %s", ctime(&borrowedBook.borrowDate));
47.               printf("Due Date\t: %s %s", ctime(&borrowedBook.dueDate),
48.                    (difftime(todDate,borrowedBook.dueDate)>=0)? "(Overdue)" : "" );
49.
50.               getche();
51.               break;
52.         }
53.       }
54.
```

```
55.
56.         fclose(books);
57.         fclose(borrowedBooks);
58.     }
```

5.6. **Searching Book** *(searchBook.c / searchBook.h)*

```
1.
2. void searchBook() {
3.
4.     SetConsoleTitle("Library System > Book Menu > Search Book");
5.
6.     void choices() {
7.
8.         system("cls");
9.
10.           printf("Search Book By :\n\n");
11.
12.           printf("Book Title\t(T)\n");
13.           printf("Author Name\t(A)\n");
14.           printf("ISBN\t\t(I)\n");
15.           printf("Category\t(C)\n");
16.       }
17.
18.
19.       do {
20.           char choice[20];
21.
22.           char choiceTitle[50];
23.           char choiceAuthor[50];
24.           char choiceISBN[50];
25.           char choiceCategory[20];
26.
27.           bool enteredTitle = false;
28.           bool enteredAuthor = false;
29.           bool enteredISBN = false;
30.           bool enteredCategory = false;
31.
32.           int enteredCount;
33.
34.           bool found = false;
35.
36.           // iterator
37.           // do-while loop exit depends on
38.           int i;
39.           do{
40.               choices();
41.
42.               printf("\n\n\nYour choice (add '+' for multiple search) : ");
43.               strupr(gets(choice));
44.
45.               enteredCount = 0;
46.
47.               for( i=0 ; i<strlen(choice) ; i++) {
48.
49.                   if( choice[i]=='T' || choice[i]=='A' || choice[i]=='I'
50.                       || choice[i]=='C' || choice[i]=='+' ) {
51.
52.                       // title
53.                       if( choice[i]=='T' ) {
54.                           printf("\nTitle : ");
```

```
55.
56.                         strupr(gets(choiceTitle));
57.
58.                         enteredTitle = true;
59.                         enteredCount++;
60.                     }
61.
62.                     // author
63.                     else if( choice[i]=='A' ) {
64.                         printf("\nAuthor : ");
65.                         strupr(gets(choiceAuthor));
66.
67.                         enteredAuthor = true;
68.                         enteredCount++;
69.                     }
70.
71.                     // ISBN
72.                     else if( choice[i]=='I' ) {
73.                         printf("\nISBN : ");
74.                         strupr(gets(choiceISBN));
75.
76.                         enteredISBN = true;
77.                         enteredCount++;
78.                     }
79.
80.                     // category
81.                     else if( choice[i]=='C' ) {
82.                         printf("\nCategory : ");
83.                         strupr(gets(choiceCategory));
84.
85.                         enteredCategory = true;
86.                         enteredCount++;
87.                     }
88.                 }
89.
90.                 else {
91.                     errMsg("Error!\nusage: enter one of the given choices");
92.                     break;
93.                 }
94.             }
95.
96.         } while ( i<(strlen(choice)-1) );
97.
98.
99.         // declaring a variable of bookDataType so if found, it can be used
100.        // no need for array
101.        // writing into files will be instantly
102.        bookDataType book;
103.
104.        FILE * books = fopen("books.bin", "r");
105.
106.        // failure check
107.        if(books==NULL) {
108.            perror("\a\nError");
109.            getche();
110.            return;
111.        }
112.
```

```
113.
114.          // read from original file
115.          while( fread(&book,sizeof(bookDataType),1,books) ) {
116.
117.              int specsCount = 0;
118.
119.              if(enteredTitle)
120.                  if( strcmp( strupr(book.title) , choiceTitle)==0  )
121.                      specsCount++;
122.
123.              if(enteredAuthor)
124.                  if( strcmp( strupr(book.author) , choiceAuthor)==0  )
125.                      specsCount++;
126.
127.              if(enteredISBN)
128.                  if( strcmp( strupr(book.ISBN)   , choiceISBN)==0 )
129.                      specsCount++;
130.
131.              if(enteredCategory)
132.                  if( strcmp( strupr(book.category) , choiceCategory)==0 )
133.                      specsCount++;
134.
135.
136.              if(specsCount == enteredCount){
137.                  found = true;
138.                  showBookDetails(book.ISBN);
139.              }
140.          }
141.
142.          fclose(books);
143.
144.
145.          if(!found)
146.              errMsg("No such specifications for a book");
147.
148.
149.      } while ( yesNoRequest("search another book")==true );
150.
151.      fflush(stdin);
152.      bookMenu();
153.  }
```

5.6. **Searching Book** *(searchBook.c / searchBook.h)*

```
1.
2. bool duplicateISBN ( char * enteredISBN ) {
3.
4.    FILE * books = fopen("books.bin", "r");
5.
6.     // failure check
7.     if(books==NULL)
8.         // no previous files available
9.         // thus no previous ISBN entered
10.           return true;
11.
12.       bookDataType book;
13.
14.       // read from original file
15.       while( fread(&book,sizeof(bookDataType),1,books) ) {
16.
17.           if( strcmp(book.ISBN , enteredISBN)==0 ) {
18.
19.               fclose(books);
20.               return false;
21.           }
22.       }
23.
24.       fclose(books);
25.       return true;
26.   }
```

5.7. **Check ISBN Duplication** *(duplicateISBN.c / duplicateISBN.h)*

```
1.
2. bool duplicateISBN ( char * enteredISBN ) {
3.
4.    FILE * books = fopen("books.bin", "r");
5.
6.     // failure check
7.     if(books==NULL)
8.         // no previous files available
9.         // thus no previous ISBN entered
10.           return true;
11.
12.       bookDataType book;
13.
14.       // read from original file
15.       while( fread(&book,sizeof(bookDataType),1,books) ) {
16.
17.           if( strcmp(book.ISBN , enteredISBN)==0 ) {
18.
19.               fclose(books);
20.               return false;
21.           }
22.       }
23.
24.       fclose(books);
25.       return true;
26.   }
```

6. **Member Header File** *(member.h)*

The `member.h` contains several functions related to member actions:

- insertmember ( )
- searchmember ( )
- borrowbook ( )
- returnbook ( )
- removemember ( )
- showmemberdetails (char * ID)
- duplicateID (char * enteredID)

6.1. **Inserting Member** *(insertMember.c / insertMember.h)*

```
1.
2. void insertMember() {
3.
4.     SetConsoleTitle("Library System > Member Menu > Inserting New Member");
5.
6.     // no need for array
7.     // writing into files will be instantly
8.     memberDataType member;
9.
10.       // for verifications
11.       bool valid;
12.
13.       // iterator
14.       int i=1;
15.
16.       do {
17.           system("cls");
18.
19.           // series of terniry operators for ordinal indicators
20.           // i.e. 1(st), 2(nd), etc.
21.           printf("Please, Enter %d%s Member's Details.\n\n", i++,
22.                           (i==1)?"st":(i==2)?"nd":(i==3)?"rd":"th" );
23.
24.
25.           // first name entry
26.           do {
27.               printf("\nFirst Name : ");
28.               strcap(gets(member.name.first));
29.
30.               if( strcmp(member.name.first , "") == 0 )
31.                   errMsg("Error !\nusage: no blank entry allowed");
32.
33.           } while ( strcmp(member.name.first , "") == 0 );
34.
35.
36.           // last name entry
37.           do {
38.               printf("\nLast Name : ");
39.               strcap(gets(member.name.last));
40.
41.               if( strcmp(member.name.last , "") == 0 )
42.                   errMsg("Error !\nusage: no blank entry allowed");
43.
44.           } while ( strcmp(member.name.last , "") == 0 );
45.
46.
47.
48.           // city entry
49.           do {
50.               printf("\nCity : ");
51.               strcap(gets(member.address.city));
52.
53.               if( strcmp(member.address.city , "") == 0 )
54.                   errMsg("Error !\nusage: no blank entry allowed");
```

```
55.
56.             } while ( strcmp(member.address.city , "") == 0 );
57.
58.
59.
60.             // building entry
61.             do {
62.                 printf("\nBulding : ");
63.                 scanf("%d",&member.address.building);
64.
65.                 if( member.address.building<0 )
66.                     errMsg("Error !\nusage: only positive integers are allowed");
67.
68.             } while ( member.address.building<0 );
69.
70.
71.
72.             // street entry
73.             do {
74.                 printf("\nStreet : ");
75.                 scanf("%d",&member.address.street);
76.
77.                 // avoiding logical errors causes by contiguous scanf and gets
78.                 getc(stdin);
79.
80.                 if( member.address.street<0 )
81.                     errMsg("Error !\nusage: only positive integers are allowed");
82.
83.             } while ( member.address.street<0 );
84.
85.
86.
87.             // ID entry
88.             do {
89.                 printf("\nID : ");
90.                 gets(member.ID);
91.
92.                 valid = duplicateID(member.ID);
93.
94.                 if(!valid)
95.                     errMsg("Error !\nThis ID was previously taken");
96.
97.                 if( strcmp(member.ID , "") == 0 )
98.                     errMsg("Error !\nusage: no blank entry allowed");
99.
100.            } while ( strcmp(member.ID , "") == 0 || !valid );
101.
102.            acpMsg("Valid ID");
103.
104.
105.
106.            // age entry
107.            do{
108.                printf("\nAge : ");
109.                scanf("%d", &member.age);
110.
111.
112.
```

```
113.
114.             // avoiding logical errors causes by contiguous scanf and gets
115.             getc(stdin);
116.
117.             // check age validity
118.             if( member.age<=0 )
119.                 errMsg("Error !\nusage: age must be positive number");
120.
121.         } while ( member.age<=0 );
122.
123.
124.         // phone number entry
125.         do{
126.             printf("\nPhone Number (11 digits) : ");
127.             gets(member.phoneNum);
128.
129.             // check phone number validity
130.             if( strlen(member.phoneNum)!=11 )
131.                 errMsg("Error !\nusage: phone numbers must consist of 11 digit");
132.
133.             else
134.                 acpMsg("Valid phone number");
135.
136.
137.         } while ( strlen(member.phoneNum)!=11 );
138.
139.
140.         // check email validity
141.         do{
142.             printf("\nE-Mail : ");
143.             gets(member.email);
144.
145.             valid = isEmail(member.email);
146.
147.             if(valid)
148.                 acpMsg("Valid e-mail address");
149.             else
150.                 errMsg("Error\ninvalid e-mail address");
151.
152.         } while(!valid);
153.
154.
155.         // initializing logic values to each structure without user's interfere
156.         member.borrowedBooks = 0;
157.
158.
159.         // write i(th) member data into file
160.         FILE * members = fopen("members.bin", "a");
161.
162.         // failure check
163.         if(members==NULL) {
164.             perror("\a\nError");
165.             getche();
166.             return;
167.         }
168.
169.         fwrite(&member, sizeof(memberDataType), 1, members);
170.
```


Zeyad Osama Abd EL Monem - 4555
Marwan Amr Farouk El-Safty - 4690

```
171.
172.          fclose(members);
173.
174.      } while( yesNoRequest("add another member")==true );
175.
176.      fflush(stdin);
177.      memberMenu();
178.  }
```

6.2. **Removing Member** *(removeMember.c / removeMember.h)*

```
1.
2. void removeMember() {
3.
4.     SetConsoleTitle("Library System > Member Menu > Removing Member");
5.
6.     // declaring a variable of memberDataType so if found, it can be used
7.     // no need for array
8.     // writing into files will be instantly
9.     memberDataType member;
10.
11.       do{
12.           system("cls");
13.
14.           char choiceID[20];
15.           printf("Enter member's ID to remove : ");
16.           gets(choiceID);
17.
18.
19.           FILE * members = fopen("members.bin", "r");
20.
21.           // failure check
22.           if(members==NULL) {
23.               perror("\a\nError");
24.               getche();
25.               return;
26.           }
27.
28.           FILE * membersTemp = fopen("membersTemp.bin", "w");
29.
30.           // failure check
31.           if(membersTemp==NULL) {
32.               perror("\a\nError");
33.               getche();
34.               return;
35.           }
36.
37.
38.           bool found = false;
39.
40.           // read from original file
41.           while( fread(&member,sizeof(memberDataType),1,members) ) {
42.
43.               if( strcmp(member.ID , choiceID)!=0 )
44.                   // write into temporary file
45.                   fwrite(&member,sizeof(memberDataType),1,membersTemp);
46.
47.               // found member
48.               else {
49.
50.                   // having non-returned borrowed books
51.                   if ( member.borrowedBooks>0 )
52.                       fprintf(stderr, "\a\n\n%s is borrowing %d books.
53.                           Can not remove now until returning them.",
54.                           member.name.first, member.borrowedBooks );
```

```
55.
56.                     else
57.                         found = true;
58.                 }
59.             }
60.
61.             fclose(members);
62.             fclose(membersTemp);
63.
64.             remove("members.bin");
65.             rename("membersTemp.bin" , "members.bin");
66.
67.             showMemberDetails(choiceID);
68.
69.             if(found)
70.                 acpMsg("Member removed successfully");
71.
72.             else
73.                 errMsg("No such ID for a member to remove");
74.
75.
76.         }while( yesNoRequest("remove another member")==true );
77.
78.         fflush(stdin);
79.         memberMenu();
80.     }
```

6.3. **Searching Member** *(searchMember.c / searchMember.h)*

```
1.
2. void searchMember() {
3.
4.     SetConsoleTitle("Library System > Member Menu > Search Member");
5.
6.     void choices() {
7.
8.         system("cls");
9.
10.           printf("Search Member By :\n\n");
11.
12.           printf("First Name\t(N)\n");
13.           printf("ID\t\t(I)\n");
14.           printf("Borrowings\t(B)\n");
15.           printf("Phone Number\t(P)\n");
16.       }
17.
18.
19.       do {
20.           char choice[20];
21.
22.           char choiceName[25];
23.           char choiceID[20];
24.           int choiceBorrowings;
25.           char choicePhone[20];
26.
27.           bool enteredName = false;
28.           bool enteredID = false;
29.           bool enteredBorrowings = false;
30.           bool enteredPhone = false;
31.
32.           int enteredCount;
33.
34.           bool found = false;
35.
36.           // iterator
37.           // do-while loop exit depends on
38.           int i;
39.           do{
40.               choices();
41.
42.               printf("\n\n\nYour choice (add '+' for multiple search) : ");
43.               strupr(gets(choice));
44.
45.               enteredCount = 0;
46.
47.               for( i=0 ; i<strlen(choice) ; i++) {
48.
49.                   if( choice[i]=='N' || choice[i]=='I' || choice[i]=='B'
50.                       || choice[i]=='P' || choice[i]=='+' ) {
51.
52.                       // name
53.                       if( choice[i]=='N' ) {
54.                           printf("\nFirst Name : ");
```

```
55.                         strupr(gets(choiceName));
56.
57.                         enteredName = true;
58.                         enteredCount++;
59.                     }
60.
61.                     // ID
62.                     else if( choice[i]=='I' ) {
63.                         printf("\nID : ");
64.                         strupr(gets(choiceID));
65.
66.                         enteredID = true;
67.                         enteredCount++;
68.                     }
69.
70.                     // borrowings
71.                     else if( choice[i]=='B' ) {
72.                         printf("\nNumber of Borrowings : ");
73.                         scanf("%i",&choiceBorrowings);
74.                         getc(stdin);
75.
76.                         enteredBorrowings = true;
77.                         enteredCount++;
78.                     }
79.
80.                     // phone number
81.                     else if( choice[i]=='P' ) {
82.                         printf("\nPhone Number : ");
83.                         strupr(gets(choicePhone));
84.
85.                         enteredPhone = true;
86.                         enteredCount++;
87.                     }
88.                 }
89.
90.                 else {
91.                     errMsg("Error!\nusage: enter one of the given choices");
92.                     break;
93.                 }
94.             }
95.
96.         } while ( i<(strlen(choice)-1) );
97.
98.
99.         // declaring a variable of memberDataType so if found, it can be used
100.        // no need for array
101.        // reading from files will be instantly
102.        memberDataType member;
103.
104.        FILE * members = fopen("members.bin", "r");
105.
106.        // failure check
107.        if(members==NULL) {
108.            perror("\a\nError");
109.            getche();
110.            return;
111.        }
112.
```

```
113.
114.          // read from original file
115.          while( fread(&member, sizeof(memberDataType), 1, members) ) {
116.
117.              int specsCount = 0;
118.
119.              if(enteredName)
120.                  if( strcmp( strupr(member.name.first) , choiceName)==0  )
121.                      specsCount++;
122.
123.              if(enteredID)
124.                  if( strcmp( strupr(member.ID) , choiceID)==0  )
125.                      specsCount++;
126.
127.              if(enteredBorrowings)
128.                  if( member.borrowedBooks == choiceBorrowings )
129.                      specsCount++;
130.
131.              if(enteredPhone)
132.                  if( strcmp( strupr(member.phoneNum) , choicePhone)==0 )
133.                      specsCount++;
134.
135.
136.              if(specsCount == enteredCount) {
137.                  found = true;
138.                  showMemberDetails(member.ID);
139.              }
140.          }
141.
142.          fclose(members);
143.
144.
145.          if(!found)
146.              errMsg("No such specifications for a member");
147.
148.
149.      } while ( yesNoRequest("search another member")==true );
150.
151.      fflush(stdin);
152.      memberMenu();
153.  }
```

6.4. **Showing Member Details** *(showMemberDetails.c / showMemberDetails.h)*

```
1.
2. void showMemberDetails( char * choiceID ) {
3.
4.    FILE * members = fopen("members.bin", "r");
5.
6.     // failure check
7.     if(members==NULL) {
8.         perror("\a\n\nError");
9.         getche();
10.           return;
11.       }
12.
13.       memberDataType member;
14.
15.       // read from original file
16.       while( fread(&member,sizeof(memberDataType),1,members) ) {
17.
18.           if( strcmp(member.ID , choiceID)==0 ) {
19.
20.               printf("\n\n\aMember Details.\n");
21.
22.               printf("\nFull Name\t: %s %s", member.name.first ,  member.name.last );
23.               printf("\nIdentity (ID)\t: %s", member.ID);
24.               printf("\nE-Mail Address\t: %s", member.email);
25.               printf("\nPhone Number\t: %s", member.phoneNum);
26.               printf("\nBorrowed Books\t: %d", member.borrowedBooks);
27.
28.               getche();
29.               break;
30.         }
31.       }
32.
33.       fclose(members);
34.   }
```

6.5. **Check ID Duplication** *(duplicateID.c / duplicateID.h)*

```
1.
2. bool duplicateID ( char * enteredID ) {
3.
4.     FILE * members = fopen("members.bin", "r");
5.
6.     // failure check
7.     if(members==NULL)
8.         // no previous files available
9.         // thus no previous ID entered
10.           return true;
11.
12.       memberDataType member;
13.
14.       // read from original file
15.       while( fread(&member,sizeof(memberDataType),1,members) ) {
16.
17.           if( strcmp(member.ID , enteredID)==0 ) {
18.
19.               fclose(members);
20.               return false;
21.           }
22.       }
23.
24.       fclose(members);
25.       return true;
26.   }
```

6.6. **Borrowing Book** *(borrowBook.c / borrowBook.h)*

```
1.
2. void borrowBook() {
3.
4.     SetConsoleTitle("Library System > Member Menu > Borrowing Book");
5.
6.     memberDataType member;
7.     bookDataType book;
8.     borrowedBookDataType borrowedBook;
9.
10.       do {
11.           system("cls");
12.
13.
14.           /*
15.           ISBN check
16.           */
17.
18.           char choiceISBN[20];
19.           printf("\nEnter book's ISBN to borrow : ");
20.           gets(choiceISBN);
21.
22.           FILE * books = fopen("books.bin", "r");
23.
24.           // failure check
25.           if(books==NULL) {
26.               perror("\a\nError");
27.               getche();
28.               return;
29.           }
30.
31.
32.           // initializing search attempt to false to modify if found
33.           bool foundBook = false;
34.
35.           // check ISBN availability
36.           while( fread(&book, sizeof(bookDataType) ,1 ,books) ) {
37.
38.               if( strcmp(book.ISBN , choiceISBN)==0 ) {
39.
40.                   foundBook = true;
41.
42.                   if(book.avalCopiesNum==0) {
43.                       errMsg("There's no available copies of this book.");
44.                       getche();
45.                       adminMenu();
46.                   }
47.
48.                   else
49.                       break;
50.               }
51.           }
52.
53.
54.
```

```
55.
56.
57.          // ISBN not found
58.          if(!foundBook) {
59.              fprintf(stderr,"\a\nError.\nNo such ISBN for a book found.");
60.              getche();
61.              memberMenu();
62.          }
63.
64.   ////////////////////////////////////////////////////////////////////////////////
65.
66.          /*
67.          ID check
68.          */
69.
70.          char choiceID[40];
71.          printf("\nEnter member's ID : ");
72.          gets(choiceID);
73.
74.
75.          FILE * members = fopen("members.bin", "r");
76.
77.          // failure check
78.          if(members==NULL) {
79.              perror("\a\nError");
80.              getche();
81.              return;
82.          }
83.
84.          // initializing search attempt to false to modify if found
85.          bool foundMember = false;
86.
87.          // check number of already borrowed books by member
88.          while( fread(&member, sizeof(memberDataType), 1, members) ) {
89.
90.              if( strcmp(member.ID , choiceID)==0 ) {
91.
92.                  foundMember = true;
93.
94.                  if(member.borrowedBooks==3) {
95.                      errMsg("This user has reached the limit
96.                              of borrowed number of books.");
97.                      adminMenu();
98.                  }
99.
100.                 else
101.                     break;
102.             }
103.         }
104.
105.         if(!foundMember) {
106.             errMsg("Error.\nNo such ID for a member found.");
107.             memberMenu();
108.         }
109.
110.
111.
112.
```

```
113.
114.
115.  ////////////////////////////////////////////////////////////////////////////
116.
117.          /*
118.          initializing borrowedBook
119.          */
120.
121.          strcpy(borrowedBook.ISBN , choiceISBN);
122.          strcpy(borrowedBook.ID , choiceID);
123.
124.          int days;
125.
126.          // borrowing days
127.          do  {
128.              printf("\nEnter days to be borrowed ? (21 max) : ");
129.              scanf("%d",&days);
130.
131.              if( days<0 && days>21 ) {
132.                  fprintf(stderr,"\a\nError.\nBorrowing days must range
133.                                  within (0<days<22).");
134.                  getche();
135.              }
136.
137.          }while( days<0 && days>21 );
138.
139.
140.          // instant local date
141.          time_t borrowTime;
142.          borrowedBook.borrowDate = time(&borrowTime);
143.
144.          // due date
145.          // time is taken in seconds
146.          time_t dueTime;
147.          dueTime = borrowTime + (days*24*60*60);
148.          borrowedBook.dueDate = dueTime;
149.
150.
151.          FILE * borrowedBooks = fopen("borrowed-books.bin", "w");
152.
153.          // failure check
154.          if(borrowedBooks==NULL) {
155.              perror("\a\nError");
156.              getche();
157.              return;
158.          }
159.
160.          // save info
161.          fwrite(&borrowedBook, sizeof(borrowedBook), 1, borrowedBooks);
162.
163.          fclose(borrowedBooks);
164.
165.
166.
167.
168.
169.
170.
```

```
171.
172.
173.  ////////////////////////////////////////////////////////////////////////////
174.
175.
176.          /*
177.          changing book info
178.          */
179.
180.          books = fopen("books.bin","r");
181.
182.          // failure check
183.          if(books==NULL) {
184.              perror("\a\nError");
185.              getche();
186.              return;
187.          }
188.
189.          FILE * booksTemp = fopen("booksTemp.bin","w");
190.
191.          // failure check
192.          if(booksTemp==NULL) {
193.              perror("\a\nError");
194.              getche();
195.              return;
196.          }
197.
198.
199.          // decrement available copies and increment borrowed times
200.          while( fread(&book, sizeof(bookDataType), 1, books) ) {
201.
202.              if( strcmp(book.ISBN , borrowedBook.ISBN)==0 ) {
203.                  book.avalCopiesNum--;
204.                  book.borrowed++;
205.              }
206.
207.              fwrite(&book,sizeof(book),1,booksTemp);
208.          }
209.
210.          fclose(books);
211.          fclose(booksTemp);
212.
213.          remove("book.bin");
214.          rename("booksTemp.bin","books.bin");
215.
216.
217.
218.
219.
220.
221.
222.
223.
224.
225.
226.
227.
228.
```

```
229.
230.
231.  ////////////////////////////////////////////////////////////////////////////
232.
233.
234.          /*
235.          changing member info
236.          */
237.
238.          members = fopen("members.bin","r");
239.
240.          // failure check
241.          if(members==NULL) {
242.              perror("\a\nError");
243.              getche();
244.              return;
245.          }
246.
247.          FILE * membersTemp = fopen("membersTemp.bin","w");
248.
249.          // failure check
250.          if(membersTemp==NULL) {
251.              perror("\a\nError");
252.              getche();
253.              return;
254.          }
255.
256.
257.          // increment number of current borrowed books
258.          while( fread(&member, sizeof(memberDataType) ,1 ,members) ) {
259.
260.              if (strcmp(member.ID , borrowedBook.ID)==0 ) {
261.                  member.borrowedBooks++;
262.              }
263.
264.              fwrite(&member, sizeof(memberDataType) ,1 ,membersTemp);
265.          }
266.
267.          fclose(members);
268.          fclose(membersTemp);
269.
270.          remove("members.bin");
271.          rename("membersTemp.bin","members.bin");
272.
273.
274.      } while( yesNoRequest("borrow another book")==true );
275.
276.      fflush(stdin);
277.      memberMenu();
278.  }
```

6.7. **Returning Book** *(returnBook.c / returnBook.h)*

```
1.
2. void returnBook() {
3.
4.     SetConsoleTitle("Library System > Member Menu > Returning Book");
5.
6.     // declaring a variables so if found, it can be used
7.     // no need for array
8.     // writing into files will be instantly
9.     memberDataType member;
10.       bookDataType book;
11.       borrowedBookDataType borrowedBook;
12.
13.       do {
14.           system("cls");
15.
16.
17.           char choiceID[40];
18.           printf("\nEnter member's ID : ");
19.           gets(choiceID);
20.
21.
22.           FILE * borrowedBooks = fopen("borrowed-books.bin", "r");
23.
24.           // failure check
25.           if(borrowedBooks==NULL) {
26.               perror("\a\nError");
27.               getche();
28.               return;
29.           }
30.
31.
32.           bool foundBorrower = false;
33.           int counter = 1;
34.
35.           // read from original file
36.           while( fread(&borrowedBook,sizeof(borrowedBookDataType),1,borrowedBooks)) {
37.
38.               if( strcmp(borrowedBook.ID , choiceID)==0 ) {
39.
40.                   foundBorrower = true;
41.
42.                   showBorrowedBookDetails( borrowedBook.ISBN , counter );
43.                   counter++;
44.               }
45.           }
46.
47.
48.           // borrower not found
49.           if(!foundBorrower) {
50.               errMsg("Error.\nNo borrowed books with such ID.");
51.               adminMenu();
52.           }
53.
54.
```

```
279.
280.
281.  ////////////////////////////////////////////////////////////////////////////
55.
56.
57.           char choiceReturnISBN[40];
58.           printf("\n\nEnter ISBN to remove : ");
59.           gets(choiceReturnISBN);
60.
61.
62.           FILE * borrowedBooksTemp = fopen("borrowed-booksTemp.bin", "w");
63.
64.           // failure check
65.           if(borrowedBooksTemp==NULL) {
66.               perror("\a\nError");
67.               getche();
68.               return;
69.           }
70.
71.
72.           while( fread(&borrowedBook,sizeof(borrowedBookDataType),1,borrowedBooks)) {
73.
74.               if( strcmp(borrowedBook.ID , choiceReturnISBN)!=0 )
75.
76.                   fwrite(&borrowedBook, sizeof(borrowedBook), 1, borrowedBooksTemp);
77.           }
78.
79.
80.           fclose(borrowedBooks);
81.           fclose(borrowedBooksTemp);
82.
83.           remove("borrowed-books.bin");
84.           rename("borrowed-booksTemp.bin","borrowed-books.bin");
85.
86.
282.  ////////////////////////////////////////////////////////////////////////////
87.
88.           /*
89.           changing book info
90.           */
91.
92.           FILE * books = fopen("books.bin","r");
93.
94.           // failure check
95.           if(books==NULL) {
96.               perror("\a\nError");
97.               getche();
98.               return;
99.           }
100.
101.          FILE * booksTemp = fopen("booksTemp.bin","w");
102.
103.          // failure check
104.          if(booksTemp==NULL) {
105.              perror("\a\nError");
106.              getche();
107.              return;
108.          }
```

```
109.
110.
111.          // increment available copies and increment borrowed times
112.          while( fread(&book, sizeof(bookDataType), 1, books) ) {
113.
114.              if( strcmp(book.ISBN , choiceReturnISBN)==0 ) {
115.                  book.avalCopiesNum++;
116.              }
117.
118.              fwrite(&book,sizeof(book),1,booksTemp);
119.          }
120.
121.          fclose(books);
122.          fclose(booksTemp);
123.
124.          remove("book.bin");
125.          rename("booksTemp.bin","books.bin");
126.
283.
284.  ////////////////////////////////////////////////////////////////////////////
127.
128.          /*
129.          changing member info
130.          */
131.
132.
133.          FILE * members = fopen("members.bin","r");
134.
135.          // failure check
136.          if(members==NULL) {
137.              perror("\a\nError");
138.              getche();
139.              return;
140.          }
141.
142.          FILE * membersTemp = fopen("membersTemp.bin","w");
143.
144.          // failure check
145.          if(membersTemp==NULL) {
146.              perror("\a\nError");
147.              getche();
148.              return;
149.          }
150.
151.
152.          // decrement number of current borrowed books
153.          while( fread(&member, sizeof(memberDataType) ,1 ,members) ) {
154.
155.              if (strcmp(member.ID , choiceID)==0 ) {
156.                  member.borrowedBooks--;
157.              }
158.
159.              fwrite(&member, sizeof(memberDataType) ,1 ,membersTemp);
160.          }
161.
162.          fclose(members);
163.          fclose(membersTemp);
164.
```

```
165.
166.
167.          remove("members.bin");
168.          rename("membersTemp.bin","members.bin");
169.
170.
171.      } while( yesNoRequest("return another book")==true );
172.
173.      fflush(stdin);
174.      bookMenu();
175.  }
```

7. **Admin Header File** *(admin.h)*

The `admin.h` contains several functions related to admin actions:

- overduebooks ( )
- mostpopularbooks ( )

7.1. **Overdue Books** *(overduebooks.c / overduebooks.h)*

```
1.
2. void overDueBooks() {
3.
4.     SetConsoleTitle("Library System > Admin Menu > Over Due Books");
5.
6.     system("cls");
7.
8.        // declaring a variable of bookDataType so if found, it can be used
9.        // no need for array
10.       // reading from files will be instantly
11.       borrowedBookDataType book;
12.
13.
14.       FILE * borrowedBooks = fopen("borrowed-books.bin", "r");
15.
16.       // failure check
17.       if(borrowedBooks==NULL) {
18.           perror("\a\nError");
19.           getche();
20.           return;
21.       }
22.
23.
24.       bool found = false;
25.
26.       time_t todDate;
27.       time(&todDate);
28.
29.       // read from original file
30.       while( fread(&book, sizeof(borrowedBookDataType), 1, borrowedBooks) ) {
31.
32.         if( difftime(todDate,book.dueDate)>=0 ) {
33.                     found = true;
34.               showBookDetails(book.ISBN);
35.         }
36.       }
37.
38.       if(!found)
39.         acpMsg("No overdue books found");
40.
41.
42.       fclose(borrowedBooks);
43.
44.       getche();
45.
46.       fflush(stdin);
47.       adminMenu();
48.   }
```

7.1. **Most Popular Books** *(mostPopularBooks.c / mostPopularBooks.h)*

```
1.
2. void mostPopularBooks() {
3.
4.     SetConsoleTitle("Library System > Admin Menu > Most Popular Books");
5.
6.     system("cls");
7.
8.
9.     FILE * books = fopen("books.bin", "r");
10.
11.        // failure check
12.        if(books==NULL) {
13.            perror("\a\nError");
14.            getche();
15.            return;
16.        }
17.
18.        bookDataType bookBuffer;
19.        bookDataType * book = NULL;
20.
21.        int i=0;
22.
23.        // read from original file
24.        while( fread(&bookBuffer,sizeof(bookDataType),1,books) ) {
25.
26.            book = realloc(book, (i+1)*sizeof(bookDataType) );
27.            book[i] = bookBuffer;
28.
29.            i++;
30.        }
31.
32.        fclose(books);
33.
34.
35.        bookDataType temp;
36.
37.        // bubble sort
38.        int j,k;
39.        for(j=0 ; j<(i-1) ; j++)
40.            for(k=0 ; k<(i-1) ;k++)
41.                if( book[k].borrowed < book[k+1].borrowed ) {
42.                    temp = book[k];
43.                    book[k] = book[k+1];
44.                    book[k+1] = temp;
45.                }
46.
47.
48.        // table header
49.        printf("\n #  ");
50.
51.        printf("Title\t\t\t");
52.        printf("Author\t\t\t");
53.        printf("ISBN\t\t");
54.        printf("Borrowed\n\n\n");
```

```
55.
56.       int top=5;
57.       int rank=1;
58.
59.       // table printing
60.       for(j=1 ; j<=top , j<=i ; j++) {
61.
62.           // to avoid not taking duplication
63.           // i.e. 5th and 6th are same at value
64.           if( (j>=5) && (book[j-1].borrowed == book[j].borrowed) )
65.               top++;
66.
67.           printf("%2d  ", (j==1) ? 1 : (book[j-1].borrowed == book[j].borrowed) ?
68.                                                                   rank : ++rank );
69.
70.           printf("%.50s\t", book[j-1].title );
71.           printf("%.30s\t\t", book[j-1].author );
72.           printf("%.20s\t", book[j-1].ISBN );
73.           printf("%8d\n", book[j-1].borrowed );
74.       }
75.
76.       free(book);
77.
78.       getche();
79.
80.       fflush(stdin);
81.       adminMenu();
82.   }
```

V. **References.**

- http://www.cplusplus.com/reference/
- https://reference.cs50.net/
- https://stackoverflow.com/questions/
- https://www.quora.com/
- http://wiki.codeblocks.org/index.php/Main_Page/

(This page is left blank on purpose)

Notes.

Notes.