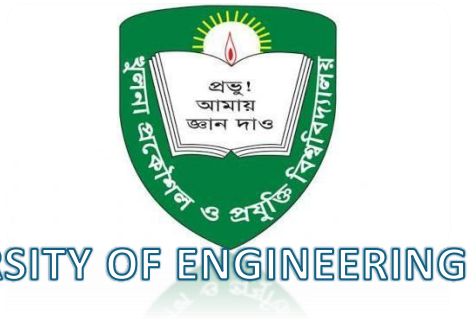

KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

Project REPORT

Department: Electronics and Communications Engineering

Course No.: CSE 2202

Course Title: Data Structure and Algorithm Laboratory

Topic: Library Book Management System.

DATE:03-12-2024

Submitted By:

Muntasir Billah Nakeeb

Roll: 2109016

Objectives:

1. To create a project using C++ programming and data structures.
2. Develop a basic Library Book Management System that utilizes data structures.
3. To implement features like linked list, class, tree, array etc.
4. To be familiar with Data structures.
5. To make the program easy while it is running.
6. To be able to make a project using Data structures and algorithm.

Introduction:

Library Book Management System project was Developed in C++ programming language and is a console application with the help of Data structures concept. This system was built as a straightforward side project in the Code::Blocks IDE using the GCC compiler. The Library Book Management System console software is a simple tool with no graphics. Like how Library Book Management System works on websites of school or university. Members can view their personal details and their book details. A simple project is Library Book Management System in C++. The basis of a Library Book Management System is the idea of creating organizational records of Members, adding this data and updating it. Here, admin can add members information securely and quickly. The system makes it easy to keep records of each person. The entire project was developed using the "C++" programming language and various variables and class. Consumers will find this little project easy to use and understand.

Motivation and Background:

Administrator: This requires an authority password to maintain Members' information. Such as:

- Book Cataloging: The system stores and manages book details such as title, author, ISBN, and publication year. Users can add, delete, and view books in the catalog.

- Book Search: The system allows users to search for books based on their title or ISBN, using sorting and searching algorithms for quick and efficient retrieval.
- Book Issuing and Returning: The system keeps track of books issued to library members and manages the return process, utilizing a stack to handle recently returned books and a queue for managing book requests.
- Member Management: The system manages library members by storing their details in a linked list, linking issued books to specific members and ensuring accurate tracking.
- Library Section Hierarchy: The system represents the organizational structure of the library using a tree data structure, categorizing books into different sections such as Fiction, Non-Fiction, and Science.
- Graph-Based Relationships (Optional): The system can also represent relationships between various library sections (e.g., related categories) using a graph data structure, allowing for advanced categorization and navigation.

Member: Members can view their information, issue book, return book, by giving their ID and password.

Hardware and Software Requirements:

1. **Code::Blocks IDE**
2. **GCC compiler**

Description:

Let's discuss the code part by part. At first let's see class, global variables, and functions and header files.

```
#include<bits/stdc++.h>
#include<iostream>
#include<windows.h>
#include<stdbool.h>
#include <conio.h>
#include <vector>
#include <string>
#include <stack>
#include <queue>
```

Figure 1: Header file

```
///GLOBAL VARIABLES;
vector<BOOK> books;
MEMBER *start=NULL;
RETURN_ST rtrns;
REQUEST_Q rqstq;
SECTION *root=NULL ;
int ex=0;
```

Figure 2: GLOBAL Variables.

```
class BOOK
{

class MEMBER
{

///MANAGEMENT CLASS of Recently returned book
class RETURN_ST
{

///MANAGEMENT CLASS of Requested book
class REQUEST_Q
{


class SECTION ///LYBRARY SECTION HIERARCHY
{

///BOOK MANAGEMENT CLASS(ADD, DELETE, DISPLAY)
class BOOK_CATALOG
{

///BOOK MANAGEMENT CLASS(ADD, DELETE, DISPLAY)
class MEMBER_CTALOG
{

/// MANAGEMENT CLASS (SEARCH , SORT)
class SORT_SEARCH
{

///ADMIN PANNEL INHERITS BOOK CATALOG, MEMBER CATALOG,SORT_SEARCH
class ADMIN: public BOOK_CATALOG, public MEMBER_CTALOG, public SORT_SEARCH
{
```

Figure 3: ALL class

```
///USE TO ISSUE A BOOK BY ISBN
///REQUEST MEMBER TO LOG IN LOGIN
void book_issue()
{    void book_issue()
void add_request()
{

///USE TO RETURN BOOK FROM MEMBER ISSUED BOOK
///REQUEST MEMBER TO LOG IN LOGIN
void book_deposit()
{

///REQUEST MEMBER TO LOG IN LOGIN
/// TO SHOW DETAILS
void my_details()
{

///CHEKCING PASSWORD TO LOGIN;
bool password(string pass)
{

///GET INPUT AND GIVE UPPER CASE;
string upperl()
{

///FOR BEAUTY
void front_page()
{
void starting()
{
```

Figure 4: All declared function

Here is the breakdown of the header file:

1. #include<bits/stdc++.h>: This is a non-standard header often used in competitive programming. It includes almost all C++ standard library headers (such as iostream, vector, string, algorithm, etc.). It can lead to longer compilation times and is not recommended for production code.
2. #include<iostream>: This header is used for input and output stream operations. It contains the definitions for std::cin, std::cout, std::cerr, and std::clog.
3. #include<windows.h>: This is a Windows-specific header, which provides access to Windows API functions, such as creating windows, handling messages, performing file operations, and managing memory and processes in Windows.
4. #include<stdbool.h>: This is a C header file that defines the bool, true, and false keywords. It is included for boolean operations in C programs. C++ already supports bool natively, so this is generally not needed in C++ code.
5. #include<conio.h>: This is another C-specific header that includes functions for console input/output operations, such as getch() for capturing keypress events and clrscr() for clearing the screen. It is not part of the C++ standard library and is considered outdated and non-portable (it works primarily in Turbo C++ or similar older compilers).
6. #include<vector>: This header is used for the std::vector class, which is a dynamic array. It provides an efficient way of storing and manipulating sequences of data.
7. #include<string>: This header defines the std::string class, which is a standard library class for handling strings in C++.
8. #include<stack>: This header defines the std::stack container, which is a standard container adapter that provides a LIFO (Last In, First Out) data structure.
9. #include<queue>: This header defines the std::queue container, which provides a FIFO (First In, First Out) data structure.

Here is the breakdown of BOOK class:

The BOOK class models a book with four attributes: title, author, ISBN, and the number of copies. It has a constructor that allows initialization of these attributes when an object is created.

```
class BOOK
{
public:
    string title, author, ISBN;
    int no;
    BOOK(string t, string a, string isbn, int n):
        title(t),author(a),ISBN(isbn),no(n) {}
};
```

Figure 5: class BOOK

Here is the breakdown of the MEMBER class:

1. Data Members:
   - string name, id, pass: These store the member's name, ID, and password.
   - vector<BOOK> issued_book: A list of books that the member has currently issued.
   - vector<BOOK> requested_book: A list of books that the member has requested (but not yet issued).
   - int availabe: The number of books the member can still issue or request. It starts with 5, meaning the member can issue or request up to 5 books.
   - int top: This keeps track of the number of requested books (requests that haven't been fulfilled yet). It is initialized to 0.
   - MEMBER* next: This is a pointer to the next member in a linked list, presumably used if members are stored in a linked list.
2. Member Functions:
   - bool issue(BOOK& bk):
     - This function handles the issuing of a book to the member. It first checks if the member has available slots (availabe > 0).If the

member has the capacity to issue a book, the book is added to issued_book, and the available slots (availabe) decrease by 1.

- If the member is issuing a book and there are books in the requested_book list, the most recent requested book is removed (i.e., a request is prioritized), and the top value is decremented.
- Returns true if the book is successfully issued, otherwise false.

- int issue_no(): This function returns the number of books the member has currently issued
- int request_no(): This function returns the number of books the member has requested (i.e., the top value).

```
class MEMBER
{
public:
    string name, id, pass;
    vector<BOOK>issued_book;
    vector<BOOK>requested_book;
    int availabe=5;///Can issue or request
    int top=0; /// requested book no.
    MEMBER* next;

    ///TAKE A BOOK & RETURN IS ISSUING OR NOT
    bool issue(BOOK& bk)
    {

    int issue_no()///RETURN NO. OF ISSUED BOOK
    {

    int request_no()///RETURN NO. OF REQUESTED BOOK
    {


    bool request(BOOK& bk)///request for out of stock book
    {

    ///Return tittle for increasing available copy
    string deposit(int serial)
    {

    ///ASSOCIATED WITH display_member(), deposit(), my_details()
    void display_me()///DISPLAY MEMBER INFORMATION
    {
```

Figure 6:Member class

- bool request(BOOK& bk): This function allows the member to request a book that is out of stock.If the member has not yet exceeded their request

limit (top < availabe), the book is added to the requested_book list. Returns true if the request is successful, otherwise false.

- string deposit(int serial):
  - This function handles the return of a book (deposit).
  - The member specifies the book to return using the index serial from the issued_book list.
  - The book is removed from the issued_book list, and the available slots (availabe) are incremented.
  - Returns the title of the book being returned.
- void display_me(): This function displays the member's details: ID, name, the number of books issued, and a list of the books issued (with their titles and authors).It also displays the number of requested books and the list of requested books

Here is the breakdown of RETURN_ST class:

```
///MANAGEMENT CLASS of Recently returned book
class RETURN_ST
{     class RETURN_ST {...}
private:
    stack<BOOK> return_stack;///Store Recently return Book:

public:

    void add_return(const BOOK& book)
    {
        return_stack.push(book);
    }

    ///ASSOCIATED WITH ADMIN
    void print_return()///PRINT THE RECENTLY RETURNED BOOK
    {
        int i=0;
        cout<<"\tRecently returned book: "<<endl;
        if(return_stack.empty())
        {
            cout<<"\tNo Book..."<<endl;
            return;
        }
        stack<BOOK> temp = return_stack;
        while (!temp.empty())
        {
            BOOK book = temp.top();
            cout << "\t["<<i<<"].Book: " << book.title << "\tby " << book.author << endl;
            temp.pop();
            i++;
        }
    }
};
```

Figure 7: RETURN_ST CLASS

1. Data Members:
   - stack<BOOK> return_stack: A stack that stores books that have been recently returned.
2. Member Functions:
   - void add_return(const BOOK& book): Adds a returned book to the return_stack.
   - void print_return():
     - Displays the list of recently returned books.
     - If the stack is empty, it shows a message "No Book...".
     - It uses a temporary stack to avoid modifying the original return_stack while displaying its contents.

Here is the breakdown of REQUEST_Q

It has two data member one will stores requested book which are out of stack and other will stores unavailable book. Some function, void add_request(const BOOK& book) will add unavailable book, void stock_request(const BOOK& book will add out of stock book, void print_request() will print the request queue

```
///MANAGEMENT CLASS of Requested book
class REQUEST_Q
{
private:
    queue<BOOK> request_queue;///STORE requested book that are out of stock:
    queue<BOOK> request_to_add;///STORE requested book that are not available:
public:

    void add_request(const BOOK& book)/// ADD UNAVAILABLE BOOK
    {

    void stock_request(const BOOK& book)///ADD OUT OF STOCK
    {

    ///ASSOCIATED WITH ADMIN
    void print_request()
    {
};
```

Figure 8:REQUEST_Q

SOURCE CODE:

```
///FINISHED: 30-11-22024
///DEVELOPED BY
///      -> NAKEEB(2109016)
///      -> MAHI  (2109025)
///      -> RAJESH(2109026)
///      -> SHOWMO(2109028)
///      -> SHUVO (2109029)

#include<bits/stdc++.h>
#include<iostream>
#include<windows.h>
#include<stdbool.h>
#include <conio.h>
#include <vector>
#include <string>
#include <stack>
#include <queue>
using namespace std;

bool password(string pass);
string upperl();

class BOOK
{
public:
    string title, author, ISBN;
    int no;
    BOOK(string t, string a, string isbn, int n):
        title(t),author(a),ISBN(isbn),no(n) {}
};

class MEMBER
{
public:
    string name, id, pass;
    vector<BOOK>issued_book;
    vector<BOOK>requested_book;
    int availabe=5;///Can issue or request
    int top=0; /// requested book no.
    MEMBER* next;

    ///TAKE A BOOK & RETURN IS ISSUING OR NOT
    bool issue(BOOK& bk)
    {
        if(availabe>0)///CHECKING ISSUE ABILITY
        {
            issued_book.push_back(bk);
            availabe--;
            if(top>availabe)///issuing get the priority
            {
                requested_book.pop_back();
                top--;
            }
            return true;
        }
        else
            return false;
    }
```

```
 59
 60      int issue_no()///RETURN NO. OF ISSUED BOOK
 61      {
 62          return 5-availabe;
 63      }
 64
 65      int request_no()///RETURN NO. OF REQUESTED BOOK
 66      {
 67          return top;
 68      }
 69
 70
 71      bool request(BOOK& bk)///request for out of stock book
 72      {
 73          if(top<availabe)
 74          {
 75              requested_book.push_back(bk);
 76              return true;
 77          }
 78          else
 79              return false;
 80      }
 81
 82      ///Return tittle for increasing available copy
 83      string deposit(int serial)
 84      {
 85          string t;
 86          t=issued_book[serial].title;
 87          issued_book.erase(issued_book.begin()+serial);
 88          availabe++;
 89          return t;
 90      }
 91
 92      ///ASSOCIATED WITH display_member(), deposit(), my_details()
 93      void display_me()///DISPLAY MEMBER INFORMATION
 94      {
 95          cout<<"\tID: "<<id<<"\t\tName: "<<name<<endl;
 96          cout<<"\tNo. of Issued book: "<<issue_no()<<endl;
 97          int i=0;
 98          for(auto bk:issued_book)
 99          {
100              cout<<"\t["<<i++<<"]. TITLE: "<<bk.title<<"\t\tAUTHOR:
"<<bk.author<<endl;
101          }
102          cout<<endl;
103          cout<<"\tNo. of requested book: "<<request_no()<<endl;
104          for(auto bk:requested_book)
105          {
106              cout<<"\tTITLE: "<<bk.title<<"\t\tAUTHOR:
"<<bk.author<<endl;
107          }
108          cout<<endl<<endl;
109      }
110  };
111
112  ///MANAGEMENT CLASS of Recently returned book
113  class RETURN_ST
114  {
115  private:
116      stack<BOOK> return_stack;///Store Recently return Book:
117
```

```
118  public:
119
120      void add_return(const BOOK& book)
121      {
122          return_stack.push(book);
123      }
124
125      ///ASSOCIATED WITH ADMIN
126      void print_return()///PRINT THE RECENTLY RETURNED BOOK
127      {
128          int i=0;
129          cout<<"\tRecently returned book: "<<endl;
130          if(return_stack.empty())
131          {
132              cout<<"\tNo Book..."<<endl;
133              return;
134          }
135          stack<BOOK> temp = return_stack;
136          while (!temp.empty())
137          {
138              BOOK book = temp.top();
139              cout << "\t["<<i<<"].Book: " << book.title << "\tby " <<
book.author << endl;
140              temp.pop();
141              i++;
142          }
143      }
144  };
145
146  ///MANAGEMENT CLASS of Requested book
147  class REQUEST_Q
148  {
149  private:
150      queue<BOOK> request_queue;///STORE requested book that are out of
stock:
151      queue<BOOK> request_to_add;///STORE requested book that are not
available:
152  public:
153
154      void add_request(const BOOK& book)/// ADD UNAVAILABLE BOOK
155      {
156          request_to_add.push(book);
157      }
158
159      void stock_request(const BOOK& book)///ADD OUT OF STOCK
160      {
161          request_queue.push(book);
162      }
163
164      ///ASSOCIATED WITH ADMIN
165      void print_request()
166      {
167          int i=0;
168          queue<BOOK>temp = request_queue;
169          cout<<"\tOUT OF STOCK: "<<endl;
170          if(request_queue.empty())
171          {
172              cout<<"\tNo Book..."<<endl;
173              return;
174          }
175          while (!temp.empty())///PRINTING OUT OF STOCK
```

```
 176          {
 177              BOOK book = temp.front();
 178              cout << "\t["<<i<<"].Book: " << book.title << "\tby " <<
book.author << endl;
 179              temp.pop();
 180              i++;
 181          }
 182
 183          temp=request_to_add;
 184          i=0;
 185          cout<<"\tNOT AVAILABLE: "<<endl;
 186          if(request_to_add.empty())
 187          {
 188              cout<<"\tNo Book..."<<endl;
 189              return;
 190          }
 191          while (!temp.empty())///PRINTING NOT AVAILABLE
 192          {
 193              BOOK book = temp.front();
 194              cout<< "\t["<<i<<"].Book: " << book.title << "\tby " <<
book.author << endl;
 195              temp.pop();
 196              i++;
 197          }
 198      }
 199  };
 200
 201
 202  class SECTION ///LYBRARY SECTION HIERARCHY
 203  {
 204  public:
 205      string name;
 206      vector<SECTION*> subs;
 207      void add_section(SECTION* sub)
 208      {
 209          subs.push_back(sub);
 210      }
 211  };
 212
 213  ///GLOBAL VARIABLES;
 214  vector<BOOK> books;
 215  MEMBER *start=NULL;
 216  RETURN_ST rtrns;
 217  REQUEST_Q rqstq;
 218  SECTION *root=NULL ;
 219  int ex=0;
 220
 221  ///BOOK MANAGEMENT CLASS(ADD, DELETE, DISPLAY)
 222  class BOOK_CATALOG
 223  {
 224  public:
 225
 226      void add_book()///ADD BOOK BY PUSHING IN GLOBAL VARIABLE books
 227      {
 228          string title, author, ISBN;
 229          int no;
 230          cin.ignore();
 231          cout<<"\tEnter title: ";
 232          title=upperl();
 233          cin.ignore();
 234          cout<<"\tEnter author: ";
```

```
235             author=upperl();
236             cin.ignore();
237             cout<<"\tEnter ISBN: ";
238             getline(cin,ISBN);
239             cin.ignore();
240             cout<<"\tNumber of Copy: ";
241             cin>>no;
242             BOOK newBook(title, author, ISBN, no);
243             books.push_back(newBook);
244             cout<<endl<<"\tA new Book is added SUCCESSFULLY...";
245         }
246
247         bool delete_bookisbn()///DELETE A BOOK USING ISBN
248         {
249             string ISBN;
250             cin.ignore();
251             cout<<"\tEnter ISBN no.: ";
252             getline(cin,ISBN);
253             int i=0;
254             for (auto bks:books)
255             {
256                 if (bks.ISBN == ISBN)
257                 {
258                     books.erase(books.begin()+i);
259                     return true;
260                 }
261                 i++;
262             }
263             return false;
264         }
265
266         bool delete_booktitle()///DELETE A BOOK USING ISBN
267         {
268             string title;
269             cin.ignore();
270             cout<<"\tEnter Title: ";
271             title=upperl();
272             int i=0;
273             for (auto bks:books)
274             {
275                 if (bks.title == title)
276                 {
277                     books.erase(books.begin()+i);
278                     return true;
279                 }
280                 i++;
281             }
282             return false;
283         }
284
285         void display_books()
286         {
287             cout<<"\tALL BOOKS: "<<endl;
288             for (auto bks : books)
289             {
290                 cout<<"\t**Title: "<<bks.title<<"\t\t**Author:
"<<bks.author<<endl
291                     <<"\t**ISBN: "<< bks.ISBN<<"\t\t**Available copy:
"<<bks.no<<endl<<endl;
292             }
293         }
```

```
294
295       void display_abook()
296       {
297           string title;
298           cin.ignore();
299           cout<<"\tEnter title: ";
300           title=upperl();
301           cout<<"\tBOOK: "<<endl;
302           for (auto bks : books)
303           {
304               if(bks.title==title)
305               {
306                   cout<<"\t**Title: "<<bks.title<<"\t\t**Author:
"<<bks.author<<endl
307                       <<"\t**ISBN: "<< bks.ISBN<<"\t\t**Available copy:
"<<bks.no<<endl<<endl;
308                   return;
309               }
310           }
311           cout<<"\tEnter wrong title..."<<endl;
312           return;
313       }
314   };
315
316   ///BOOK MANAGEMENT CLASS(ADD, DELETE, DISPLAY)
317   class MEMBER_CTALOG
318   {
319   public:
320
321       ///ADD MEMBER, checking,sorting
322       void add_member()
323       {
324           MEMBER *srt=start, *srt1, *ptr=new MEMBER();
325
326           cin.ignore();
327           cout<<"\tEnter member Name: ";
328           ptr->name=upperl();
329           cin.ignore();
330           cout<<"\tEnter ID: ";
331           ptr->id=upperl();
332           ptr->pass=ptr->id;
333           ptr->next=NULL;
334
335           ///Duplicate checking
336           if(check_dup(ptr->name,ptr->id))
337           {
338               cout<<"\tAlready Exist..."<<endl;
339               return;
340           }
341           else
342           {
343               if(start==NULL)
344               {
345                   start=ptr;
346               }
347               else if(start->id > ptr->id)
348               {
349                   ptr->next=start;
350                   start=ptr;
351               }
352               else
```

```
353             {
354                 srt1=srt;
355                 srt=srt->next;
356                 while(srt!=NULL)
357                 {
358                     if(srt->id>ptr->id)
359                     {
360                         ptr->next=srt;
361                         srt1->next= ptr;
362                         break;
363                     }
364                     srt1=srt;
365                     srt=srt->next;
366                 }
367                 if(srt==NULL)
368                 {
369                     srt1->next=ptr;
370                 }
371
372             }
373             cout<<"\tMEMBER ADDED SUCCESSFULLY..."<<endl;
374         }
375
376     }
377
378     ///Check duplicate member
379     bool check_dup(string name, string id)
380     {
381         MEMBER *srt=start;
382         while(srt!=NULL)
383         {
384             if(srt->name==name && srt->id==id)
385             {
386                 return true;
387                 ///break;
388             }
389             srt=srt->next;
390         }
391         return false;
392         /* if(srt->name==name && srt->id==id)
393          {
394              return true;
395          }
396          else
397
398          {
399              return false;
400          }*/
401
402     }
403
404     ///DELETE MEMBER BY ID
405     void delete_member()
406     {
407         string id;
408         MEMBER *ptr=new MEMBER();
409         MEMBER *srt=start;
410         cin.ignore();
411         cout<<"\tEnter studnet ID: ";
412         getline(cin,id);
413         while(srt->id!=id && srt->next!=NULL)
```

```
414             {
415                 ptr=srt;
416                 srt=srt->next;
417             }
418             if(srt->id!=id)
419             {
420                 cout<<"\tMember doesn't Exist..."<<endl;
421                 getch();
422                 return;
423             }
424             ptr->next=srt->next;
425             cout<<"\tMember DELETED Successfully..."<<endl;
426         }
427
428         ///DISPLAY ALL MEMBER
429         ///ASSOCIATED WITH MEMBER::display_me()
430         void display_member()
431         {
432             MEMBER *srt=start;
433             while(srt->next!=NULL)
434             {
435                 srt->display_me();
436                 srt=srt->next;
437             }
438             srt->display_me();
439         }
440     };
441
442     /// MANAGEMENT CLASS (SEARCH , SORT)
443     class SORT_SEARCH
444     {
445     public:
446
447         ///PARTITION FOR TITLE
448         int partition1( int low, int high)
449         {
450             string pivot = books[high].title;
451             int i = low - 1;
452             for (int j = low; j < high; j++)
453             {
454                 if (books[j].title < pivot)
455                 {
456                     i++;
457                     swap(books[i], books[j]);
458                 }
459             }
460             swap(books[i + 1], books[high]);
461             return i + 1;
462         }
463
464         ///PARTITION FOR ISBN
465         int partition2( int low, int high)
466         {
467             string pivot = books[high].ISBN;
468             int i = low - 1;
469             for (int j = low; j < high; j++)
470             {
471                 if (books[j].ISBN < pivot)
472                 {
473                     i++;
474                     swap(books[i], books[j]);
```

```
475             }
476         }
477         swap(books[i + 1], books[high]);
478         return i + 1;
479     }
480
481     /// Quick Sort BY TITLE
482     void sort_title(int low, int high)
483     {
484         if (low < high)
485         {
486             /// Partitioning index
487             int pi = partition1(low, high);
488             sort_title(low, pi - 1);  /// Left sub-array
489             sort_title(pi + 1, high); /// Right sub-array
490         }
491     }
492
493     ///QUICK SORTING BY ISBN
494     void sort_isbn(int low, int high)
495     {
496         if (low < high)
497         {
498             /// Partitioning index
499             int pi = partition2(low, high);
500             sort_title(low, pi - 1);  /// Left sub-array
501             sort_title(pi + 1, high); /// Right sub-array
502         }
503
504     }
505
506     ///BINARY SEARCH BY TITLE
507     BOOK* search_title(const string& title)
508     {
509         int left = 0, right = books.size() - 1;
510         sort_title(left,right);
511         while (left <= right)
512         {
513             int mid = left + (right - left) / 2;
514
515             if (books[mid].title == title)
516             {
517                 return &books[mid];
518             }
519             else if (books[mid].title < title)
520             {
521                 left = mid + 1;
522             }
523             else
524             {
525                 right = mid - 1;
526             }
527         }
528         return nullptr;///NOT FOUND LOGIC
529     }
530
531     ///Normal search By ISBN
532     BOOK* search_isbn(string ISBN)
533     {
534         for (auto& book : books)
535         {
```

```
536                 if (book.ISBN == ISBN)
537                 {
538                     return &book;
539                 }
540             }
541             return nullptr;
542         }
543 
544     };
545 
546     ///ADMIN PANNEL INHERITS BOOK CATALOG, MEMBER CATALOG,SORT_SEARCH
547     class ADMIN: public BOOK_CATALOG, public MEMBER_CTALOG, public
SORT_SEARCH
548     {
549         string pass="2210";
550     public:
551 
552         ///Allow to find the desired node
553         ///RETURN parent with ADDRESS FOR ADDING
554         SECTION* finds(SECTION* node,string name)
555         {
556             if (node->name == name)
557             {
558                 return node;
559             }
560             for (auto& sub: node->subs)
561             {
562                 SECTION* found = finds(sub, name);
563                 if (found)
564                 {
565                     return found;
566                 }
567             }
568             return nullptr;
569         }
570 
571         /// ALLOW ADDING A SECTION BY TAKING PARENT
572         void add_hierarchy()
573         {
574             if(root==NULL)
575             {
576                 cin.ignore();
577                 cout<<"\tEnter the ROOT of Hierarchy: ";
578                 root = new SECTION();
579                 root->name=upperl();
580             }
581             else
582             {
583                 string pname;
584                 cout << "\tEnter the name of the parent: ";
585                 pname=upperl();
586                 SECTION* parent= finds(root,pname);///Find the parent
section
587                 SECTION* sub=new SECTION();
588                 if (parent!=nullptr)
589                 {
590                     cout << "\tEnter the name of the sub-section: ";
591                     sub->name=upperl();
592                     parent->add_section(sub);/// Add it to the parent
593 
594                     cout<<"\t "<<sub->name<< " added to section
```

```
"<<pname<<endl;
595                 }
596                 else
597                 {
598                     cout << "\tSection '" <<pname<< "' not found.\n";
599                 }
600             }
601         }
602
603         void print_section(SECTION* node, int level)///PRINT HIERARCHY
604         {
605             if (!node)
606                 return;
607             cout <<"\t\t"<< string(level, ' ') << node->name << endl;
608             for (SECTION* subs : node->subs)
609             {
610                 print_section(subs, level + 1);
611             }
612         }
613
614         ///TO KEEP DATA SAFE
615         void login()
616         {
617             if (password(pass))
618             {
619                 menu();
620             }
621             else
622             {
623                 cout<<"\tWrong Password...";
624                 getch();
625             }
626         }
627
628         ///ADMINSTRATOR MENU
629         void menu()
630         {
631             system("cls");
632             int ch;
633             cout<<"\n\n\n\tADMINISTRATOR MENU";
634             cout<<"\n\n\t[0].BACK TO MAIN MENU.";
635             cout<<"\n\n\t[1].CREATE BOOK.";
636             cout<<"\n\n\t[2].DELETE BOOK BY ISBN OR TITLE.";
637             cout<<"\n\n\t[3].DISPLAY ALL BOOKS.";
638             cout<<"\n\n\t[4].SPECIFIC BOOK INFORMATION.";
639             cout<<"\n\n\t[5].SORT BOOK BY ISBN OR TITLE.";
640             cout<<"\n\n\t[6].ADD MEMBER RECORD.";
641             cout<<"\n\n\t[7].DELETE MEMBER RECORD.";
642             cout<<"\n\n\t[8].DISPLAY MEMBERS RECORD.";
643             cout<<"\n\n\t[9].SHOW RECENTLY RETURNED BOOK.";
644             cout<<"\n\n\t[10].SHOW REQUESTED BOOK.";
645             cout<<"\n\n\t[11].ADD SECTION OF THE HIERARCHY.";
646             cout<<"\n\n\t[12].PRINT SECTION HIERARCHY.";
647             cout<<"\n\n\tPlease Enter Your Choice (0-12): ";
648             cin>>ch;
649             switch(ch)
650             {
651             case 0:
652             {
653                 system("cls");
654                 cout<<endl<<endl;
```

```
655             return ;
656             break;
657         }
658 
659         case 1:
660         {
661 
662             system("cls");
663             cout<<endl<<endl;
664             add_book();
665             getch();
666             break;
667         }
668 
669         case 2:
670         {
671             system("cls");
672             cout<<endl<<endl;
673             int choice;
674             cout<<"\tDelete BOOK BY->";
675             cout<<"\n\t[1].ISBN\n\t[2].TITLE."<<endl;
676             cout<<"\tEnter your choice(1-2): ";
677             cin>>choice;
678 
679             if(choice==1)
680             {
681                 if(delete_bookisbn())
682                 {
683                     cout<<"\tBook is Deleted."<<endl;
684                 }
685                 else
686                 {
687                     cout<<"\tThere is no book of this ISBN."<<endl;
688                 }
689             }
690             else if(choice==2)
691             {
692                 if(delete_booktitle())
693                 {
694                     cout<<"\tBook is Deleted."<<endl;
695                 }
696                 else
697                 {
698                     cout<<"\tThere is no book of this ISBN."<<endl;
699                 }
700             }
701             else
702             {
703                 cout<<"Invalid choice. TRY again...";
704                 getch();
705                 break;
706             }
707             getch();
708             break;
709         }
710 
711         case 3:
712         {
713             system("cls");
714             cout<<endl<<endl;
715             display_books();
```

```
716             getch();
717             break;
718         }
719
720         case 4:
721         {
722             system("cls");
723             cout<<endl<<endl;
724             display_abook();
725             getch();
726             break;
727         }
728
729         case 5:
730         {
731             system("cls");
732             cout<<endl<<endl;
733             int choice;
734             cout<<"\tSORT BOOK BY->";
735             cout<<"\n\t[1].ISBN\n\t[2].TITLE."<<endl;
736             cout<<"\tEnter your choice(1-2): ";
737             cin>>choice;
738
739             if(choice==1)
740             {
741                 sort_isbn(0,books.size()-1);
742             }
743             else if(choice==2)
744             {
745                 sort_title(0,books.size()-1);
746             }
747             else
748             {
749                 cout<<"Invalid choice. TRY again...";
750                 getch();
751                 break;
752             }
753
754             string print="\tBOOKS are sorting...";
755             for(auto ch:print)
756             {
757                 cout<<ch;
758                 Sleep(40);
759             }
760             getch();
761             break;
762         }
763
764         case 6:
765         {
766             system("cls");
767             cout<<endl<<endl;
768             add_member();
769             getch();
770             break;
771         }
772
773         case 7:
774         {
775             system("cls");
776             cout<<endl<<endl;
```

```
777                 delete_member();
778                 getch();
779                 break;
780             }
781
782             case 8:
783             {
784                 system("cls");
785                 cout<<endl<<endl;
786                 display_member();
787                 getch();
788                 break;
789             }
790
791             case 9:
792             {
793                 system("cls");
794                 cout<<endl<<endl;
795                 rtrns.print_return();
796                 getch();
797                 break;
798             }
799
800             case 10:
801             {
802                 system("cls");
803                 cout<<endl<<endl;
804                 rqstq.print_request();
805                 getch();
806                 break;
807             }
808
809             case 11:
810             {
811                 system("cls");
812                 cout<<endl<<endl;
813                 add_hierarchy();
814                 getch();
815                 break;
816             }
817
818             case 12:
819             {
820                 system("cls");
821                 cout<<endl<<endl;
822                 print_section(root,0);
823                 getch();
824                 break;
825             }
826
827             default:
828                 cout<<"\t\t";
829             }
830             menu();
831         }
832
833     };
834
835     ///USE TO ISSUE A BOOK BY ISBN
836     ///REQUEST MEMBER TO LOG IN LOGIN
837     void book_issue()
```

```
838  {
839      string id;
840      MEMBER *srt=start;
841      BOOK *bk;
842      string ISBN;
843      SORT_SEARCH bro;
844
845      cin.ignore();
846      cout<<"\tEnter your ID: ";
847      getline(cin,id);
848
849      while(srt->id!=id)///checking ID
850      {
851          srt=srt->next;
852      }
853      if(srt->id!=id)
854      {
855          cout<<"\tSorry. You are not in the Member list."<<endl;
856          return;
857      }
858
859      cin.ignore();
860      cout<<"\tEnter ISBN: ";
861      getline(cin,ISBN);
862      bk= bro.search_isbn(ISBN);
863
864      if(bk==nullptr)///CHECKING BOOK
865      {
866          cout<<"\tYou enterd wrong ISBN."<<endl;
867          getch();
868          return;
869      }
870      else if(bk->no>0)///CHECKING BOOK STOCK AVAILABLE OR NOT
871      {
872          if(password(srt->pass))///CHECKING USER OR NOT
873          {
874              if(srt->issue(*bk))///CHECKING ABILITY FOR ISSUING
875              {
876                  cout<<"\tIssued successfully..."<<endl;
877                  bk->no--;
878              }
879              else
880              {
881                  cout<<"\tYou are not able to issue..."<<endl;
882              }
883
884          }
885          else
886          {
887              cout<<"\tYou entered wrong password..."<<endl;
888          }
889      }
890      else
891      {
892          cout<<"\tNot available..."<<endl;
893          if(srt->request(*bk))
894          {
895              rqstq.stock_request(*bk);
896          }
897      }
898  }
```

```
899
900  void add_request()
901  {
902      string title, author, ISBN;
903      cin.ignore();
904      cout<<"\tEnter title: ";
905      title=upperl();
906      cin.ignore();
907      cout<<"\tEnter author: ";
908      author=upperl();
909      cin.ignore();
910      cout<<"\tEnter ISBN: ";
911      getline(cin,ISBN);
912      cin.ignore();
913      BOOK newBook(title, author, ISBN, 0);
914      rqstq.add_request(newBook);
915  }
916
917  ///USE TO RETURN BOOK FROM MEMBER ISSUED BOOK
918  ///REQUEST MEMBER TO LOG IN LOGIN
919  void book_deposit()
920  {
921      string id,title;
922      int serial;
923      MEMBER *srt=start;
924      cin.ignore();
925      cout<<"\tEnter your ID: ";
926      getline(cin,id);
927      while(srt->id!=id)
928      {
929          srt=srt->next;
930      }
931      if(srt->id!=id)
932      {
933          cout<<"\tSorry. You are not in the Member list."<<endl;
934          return;
935      }
936      if(password(srt->pass))
937      {
938          if(srt->availabe<5)
939          {
940              srt->display_me();
941              cout<<"Enter the seial no: ";
942              cin>>serial;
943              title=srt->deposit(serial);
944              for(auto &bk:books)
945              {
946                  if(bk.title==title)
947                  {
948                      bk.no--;
949                      rtrns.add_return(bk);
950                      cout<<"Returned SUCCESSFULLY..."<<endl;
951                  }
952              }
953          }
954          else
955          {
956              cout<<"You have no issued Book..."<<endl;
957          }
958      }
959      else
```

```
960         {
961             cout<<"\tYou entered wrong password..."<<endl;
962         }
963     }
964
965     ///REQUEST MEMBER TO LOG IN LOGIN
966     /// TO SHOW DETAILS
967     void my_details()
968     {
969         string id;
970         MEMBER *srt=start;
971         cin.ignore();
972         cout<<"\tEnter your ID: ";
973         getline(cin,id);
974
975         while(srt!=NULL)///ID CHECKING
976         {
977             if(srt->id==id)
978             {
979                 break;
980             }
981         }
982         if(srt!=NULL)
983         {
984             if(password(id))///USER CHECKING
985             {
986                 system("cls");
987                 cout<<"\tYour INFORMATION: "<<endl<<endl;
988                 srt->display_me();
989             }
990             else
991             {
992                 cout<<"Wrong password..."<<endl;
993             }
994         }
995         else
996         {
997             cout<<"Wrong ID..."<<endl;
998         }
999
1000
1001    }
1002
1003    ///CHEKCING PASSWORD TO LOGIN;
1004    bool password(string pass)
1005    {
1006        string pw;
1007        cin.ignore();
1008        cout<<"\tEnter the pasword: ";
1009        getline(cin, pw);
1010        return pass==pw;
1011    }
1012
1013    ///GET INPUT AND GIVE UPPER CASE;
1014    string upperl()
1015    {
1016        string s;
1017        getline(cin,s);
1018        transform(s.begin(), s.end(), s.begin(), ::toupper);
1019        return s;
1020    }
```

```
1021
1022  ///FOR BEAUTY
1023  void front_page()
1024  {
1025      system("cls");
1026      system("color 20");
1027      cout<<endl<<endl<<endl;
1028      cout<<"\t\t      |---------------------------------------\n";
1029      cout<<"\t\t      |                                       |\n";
1030      cout<<"\t\t      |                                       |\n";
1031      cout<<"\t\t      |     ----------------------------      |\n";
1032      cout<<"\t\t      |             WELCOME TO                |\n";
1033      cout<<"\t\t      |                /-/                    |\n";
1034      cout<<"\t\t      |      LIBRARY MANAGEMENT SYSTEM        |\n";
1035      cout<<"\t\t      |     ----------------------------      |\n";
1036      cout<<"\t\t      |                                       |\n";
1037      cout<<"\t\t      |                                       |\n";
1038      cout<<"\t\t      |            DEVELOPED BY               |\n";
1039      cout<<"\t\t      |          TEAM PROS-CODER              |\n";
1040      cout<<"\t\t      |                                       |\n";
1041      cout<<"\t\t      |---------------------------------------|\n";
1042      cout<<"\n\n\t\t             Press any key to continue:";
1043      getch();
1044  }
1045  void starting()
1046  {
1047      system("cls");
1048      cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<"\t\t\t\t";
1049      char AppStarting[100] = "A p p l i c a t i o n   i s   S t a r
t i n g .........";
1050      for(int i=0;i<strlen(AppStarting);i++)
1051      {
1052          cout<<AppStarting[i];
1053          Sleep(60);
1054      }
1055  }
1056
1057
1058  int main()
1059  {
1060      front_page();
1061      starting();
1062      do
1063      {
1064          system("cls");
1065          system("color 17");
1066          cout<<"\n\n\n\t MAIN MENU";
1067          cout<<"\n\n\t[1]. ADMINISTRATOR MENU";
1068          cout<<"\n\n\t[2]. MY DETAILS";
1069          cout<<"\n\n\t[3]. BOOK ISSUE";
1070          cout<<"\n\n\t[4]. BOOK DEPOSIT";
1071          cout<<"\n\n\t[5]. SEARCH BOOK BY TITLE";
1072          cout<<"\n\n\t[6]. SEARCH BOOK BY ISBN";
1073          cout<<"\n\n\t[7]. REQUEST TO ADD BOOK";
1074          cout<<"\n\n\t[0]. EXIT";
1075          cout<<"\n\n\tPlease Select Your Option (0-7): ";
1076          int ch;
1077          cin>>ch;
1078          switch(ch)
1079          {
1080          case 0:
```

```
1081            {
1082                system("color 40");
1083                ex=1;
1084                getch();
1085                break;
1086            }
1087
1088            case 1:
1089            {
1090                system("cls");
1091                cout<<endl<<endl;
1092                ADMIN admin;
1093                admin.login();
1094                break;
1095            }
1096
1097            case 2:
1098            {
1099                system("cls");
1100                cout<<endl<<endl;
1101                my_details();
1102                getch();
1103                break;
1104            }
1105
1106            case 3:
1107            {
1108                system("cls");
1109                cout<<endl<<endl;
1110                book_issue();
1111                getch();
1112                break;
1113            }
1114
1115            case 4:
1116            {
1117                system("cls");
1118                cout<<endl<<endl;
1119                book_deposit();
1120                getch();
1121                break;
1122            }
1123
1124            case 5:
1125            {
1126                system("cls");
1127                cout<<endl<<endl;
1128                SORT_SEARCH s;
1129                string title;
1130                BOOK *bk;
1131
1132                cin.ignore();
1133                cout<<"\tEnter BOOK TITLE: ";
1134                getline(cin,title);
1135
1136                bk=s.search_title(title);
1137                if(bk==nullptr)
1138                {
1139                    cout<<"\tWRONG INPUT...";
1140                    getch();
1141                    break;
```

```
1142                }
1143                cout<<"\t**Title: "<<bk->title<<"\t**Author: "<<bk-
>author<<endl
1144                    <<"\t**ISBN: "<< bk->ISBN<<"\t**Available copy:
"<<bk->no<<endl<<endl;
1145                getch();
1146                break;
1147            }
1148
1149            case 6:
1150            {
1151                system("cls");
1152                cout<<endl<<endl;
1153                SORT_SEARCH s;
1154                string ISBN;
1155                BOOK *bk;
1156
1157                cin.ignore();
1158                cout<<"\tEnter BOOK ISBN: ";
1159                getline(cin,ISBN);
1160
1161                bk=s.search_isbn(ISBN);
1162                if(bk==nullptr)
1163                {
1164                    cout<<"\tWRONG INPUT...";
1165                    getch();
1166                    break;
1167                }
1168                cout<<"\t**Title: "<<bk->title<<"\t**Author: "<<bk-
>author<<endl
1169                    <<"\t**ISBN: "<< bk->ISBN<<"\t**Available copy:
"<<bk->no<<endl<<endl;
1170                getch();
1171                break;
1172            }
1173
1174            case 7:
1175            {
1176                system("cls");
1177                cout<<endl<<endl;
1178                add_request();
1179                getch();
1180                break;
1181            }
1182
1183            default :
1184            {
1185                system("cls");
1186                cout<<endl<<endl;
1187                cout<<"\tInvalid Choice...";
1188                getch();
1189                break;
1190            }
1191            }
1192        }
1193        while(ex==0);
1194    }
```

Conclusion:

The Library Book Management System provides an efficient, scalable, and automated solution for managing the day-to-day operations of a library. By integrating various data structures such as arrays, stacks, queues, linked lists, trees, and graphs, the system effectively addresses key challenges in cataloging books, searching for them, managing member information, issuing and returning books, and organizing library sections. This project has successfully demonstrated how fundamental data structures can be leveraged to create a system that streamlines library processes, reduces manual errors, and enhances the overall user experience. The implementation of sorting and searching algorithms for book retrieval, stack and queue management for book issuance and returns, and the use of tree and graph structures for organizing and representing library sections highlight the versatility of data structures in solving complex problems in a real-world context.

The Library Book Management System is a step towards modernizing library management, enabling easier tracking of books, quicker access to information, and a more organized workflow for library staff. Furthermore, the system provides an intuitive interface for users, allowing them to easily interact with the library's catalog and services.

In conclusion, this project not only fulfills its objectives of automating and improving library management but also demonstrates a practical application of data structures in software development. The system serves as a solid foundation for further enhancement and expansion, with the potential to integrate additional features such as user authentication, online book reservations, and advanced recommendation algorithms, making it a versatile tool for modern libraries.

Reference:

1. ***https://www.geeksforgeeks.org/dsa-tutorial-learn-data-structures-and-algorithms/***
2. ***https://www.tutorialspoint.com/data_structures_algorithms/index.htm***
3. ***https://www.w3schools.com/dsa/dsa_intro.php***