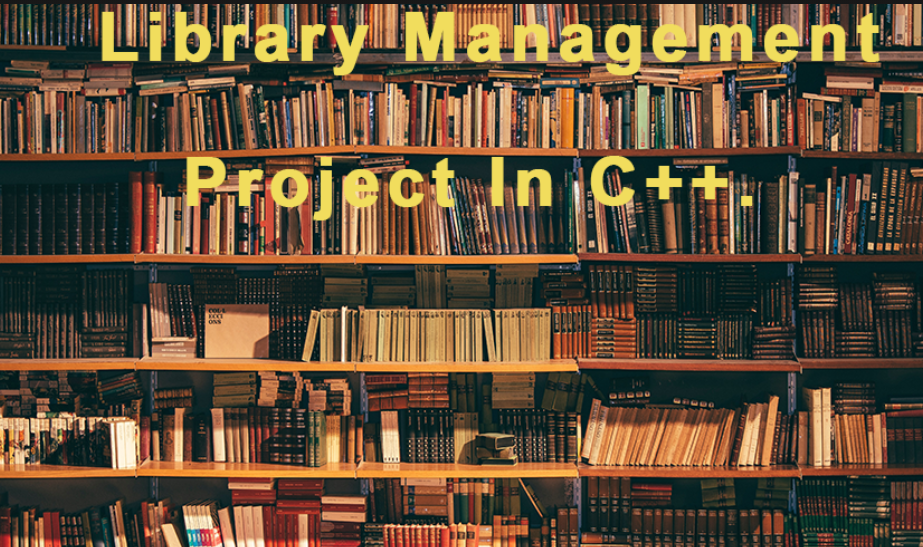
**DSA ASSIGNMENT**

**On**

**“LIBRARY MANAGEMENT SYSTEM”**

**Project Report**

Submitted in partial fulfillment of the requirements for the award of degree of

**B TECH**





Submitted by :

NAME : **Sakshi Kumari**

REGD.NO : ***12214983***

Email.ID: **[sakshi62992@gmail.com](mailto:sakshi62992@gmail.com)**

**In this report, the C++ implementation of a**

**simple library management system (LMS) is**

**discussed. Students can issue and return books**

**through the system, which also lets librarians**

**keep track of users and books. Getting experience with fundamental C++ data structures and algorithms is the main aim.**

**Code:** 

#include <iostream>

using namespace std;

#include <string>

// Define the structure for a book

struct Book {

int id;

string title;

string author;

bool isIssued;

string issuedTo;

};

// Define the structure for a linked list node

struct Node {

Book data;

Node\* next;

};

// Define the Library class

class Library {

private:

Node\* head; // Pointer to the head of the linked list

public:

Library() : head(nullptr) {

// Initialize the library with sample data

initializeLibrary();

}

void addBook(int id, const std::string& title, const std::string& author);

void searchBookById(int id);

void searchBookByTitle(const std::string& title);

void issueBook(int id, const std::string& student);

void returnBook(int id);

void listAllBooks();

void deleteBook(int id);

// Utility functions

Node\* findBookById(int id);

void sortBooks();

void initializeLibrary();

};

// Add a new book to the library

void Library::addBook(int id, const std::string& title, const std::string& author) {

Node\* newNode = new Node();

newNode->data.id = id;

newNode->data.title = title;

newNode->data.author = author;

newNode->data.isIssued = false;

newNode->data.issuedTo = "";

newNode->next = head;

head = newNode;

}

// Search for a book by ID

void Library::searchBookById(int id) {

Node\* bookNode = findBookById(id);

if (bookNode) {

cout << "Book found: " << bookNode->data.title << " by " << bookNode->data.author << "\n";

} else {

cout << "Book not found.\n";

}

}

// Search for a book by title

void Library::searchBookByTitle(const std::string& title) {

Node\* current = head;

while (current) {

if (current->data.title == title) {

cout << "Book found: " << current->data.id << " by " << current->data.author << "\n";

return;

}

current = current->next;

}

cout << "Book not found.\n";

}

// Issue a book to a student

void Library::issueBook(int id, const std::string& student) {

Node\* bookNode = findBookById(id);

if (bookNode && !bookNode->data.isIssued) {

bookNode->data.isIssued = true;

bookNode->data.issuedTo = student;

cout << "Book issued to " << student << "\n";

} else {

cout << "Book is already issued or not found.\n";

}

}

// Return a book

void Library::returnBook(int id) {

Node\* bookNode = findBookById(id);

if (bookNode && bookNode->data.isIssued) {

bookNode->data.isIssued = false;

bookNode->data.issuedTo = "";

cout << "Book returned.\n";

} else {

cout << "Book is not issued or not found.\n";

}

}

// List all books in the library

void Library::listAllBooks() {

sortBooks();

Node\* current = head;

while (current) {

cout << current->data.id << ": " << current->data.title << " by " << current->data.author;

if (current->data.isIssued) {

cout << " (Issued to " << current->data.issuedTo << ")";

}

cout << "\n";

current = current->next;

}

}

// Delete a book from the library

void Library::deleteBook(int id) {

Node\* current = head;

Node\* prev = nullptr;

while (current && current->data.id != id) {

prev = current;

current = current->next;

}

if (current) {

if (prev) {

prev->next = current->next;

} else {

head = current->next;

}

delete current;

cout << "Book deleted.\n";

} else {

cout << "Book not found.\n";

}

}

// Find a book by ID

Node\* Library::findBookById(int id) {

Node\* current = head;

while (current) {

if (current->data.id == id) {

return current;

}

current = current->next;

}

return nullptr;

}

// Sort books by ID

void Library::sortBooks() {

if (!head || !head->next) return;

Node\* sorted = nullptr;

while (head) {

Node\* current = head;

head = head->next;

if (!sorted || sorted->data.id >= current->data.id) {

current->next = sorted;

sorted = current;

} else {

Node\* temp = sorted;

while (temp->next && temp->next->data.id < current->data.id) {

temp = temp->next;

}

current->next = temp->next;

temp->next = current;

}

}

head = sorted;

}

// Initialize the library with sample data

void Library::initializeLibrary() {

addBook(101, "The White Tiger", " Aravind Adiga");

addBook(102, "Five Point Someone", "Chetan Bhagat");

addBook(103, "The God of Small Things", "Arundhati Roy");

addBook(104, "The Inheritance of Loss ", "Kiran Desai");

addBook(105, "The Discovery of India ", "Jawaharlal Nehru");

}

// Main function to demonstrate the functionality

int main() {

Library library;

int choice, id;

string title, author, student;

while (true) {

cout << "\nLibrary Management System\n";

cout << "1. Add Book\n";

cout << "2. Search Book by ID\n";

cout << "3. Search Book by Title\n";

cout << "4. Issue Book\n";

cout << "5. Return Book\n";

cout << "6. List All Books\n";

cout << "7. Delete Book\n";

cout << "8. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter book ID: ";

cin >> id;

cin.ignore(); // To ignore the newline character left by std::cin

cout << "Enter book title: ";

getline(std::cin, title);

cout << "Enter book author: ";

getline(std::cin, author);

library.addBook(id, title, author);

cout << "Book added successfully.\n";

break;

case 2:

cout << "Enter book ID: ";

cin >> id;

library.searchBookById(id);

break;

case 3:

cin.ignore(); // To ignore the newline character left by std::cin

cout << "Enter book title: ";

getline(std::cin, title);

library.searchBookByTitle(title);

break;

case 4:

cout << "Enter book ID: ";

cin >> id;

cin.ignore(); // To ignore the newline character left by std::cin

cout << "Enter student name: ";

getline(std::cin, student);

library.issueBook(id, student);

break;

case 5:

cout << "Enter book ID: ";

cin >> id;

library.returnBook(id);

break;

case 6:

library.listAllBooks();

break;

case 7:

cout << "Enter book ID: ";

cin >> id;

library.deleteBook(id);

break;

case 8:

cout << "Exiting...\n";

return 0;

default:

cout << "Invalid choice. Please try again.\n";

}

}

return 0;

}



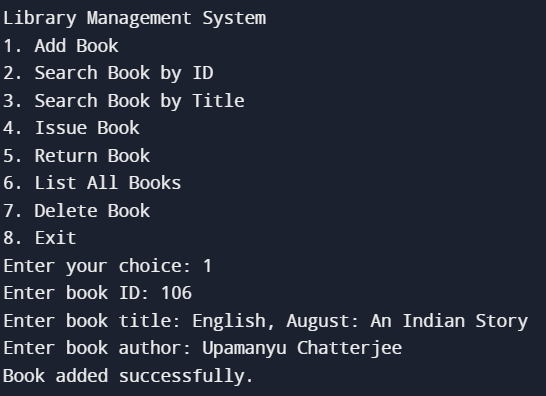




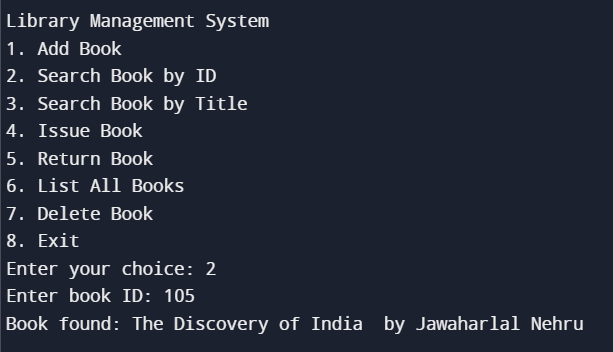
**Output: (Used Online Compiler)**

Website Link :- <https://www.programiz.com/cpp-programming/online-compiler/>

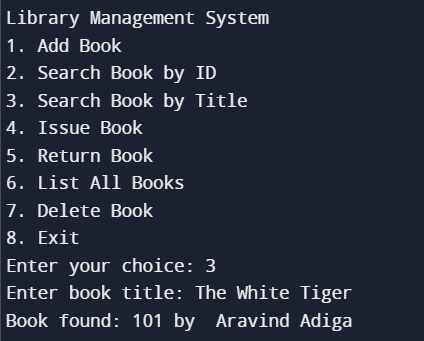
1. **To Add Book :-**



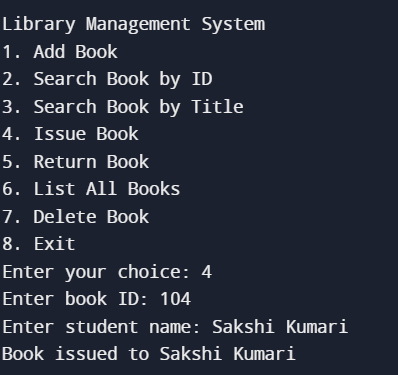
1. **To Search for a Book by ID:-**

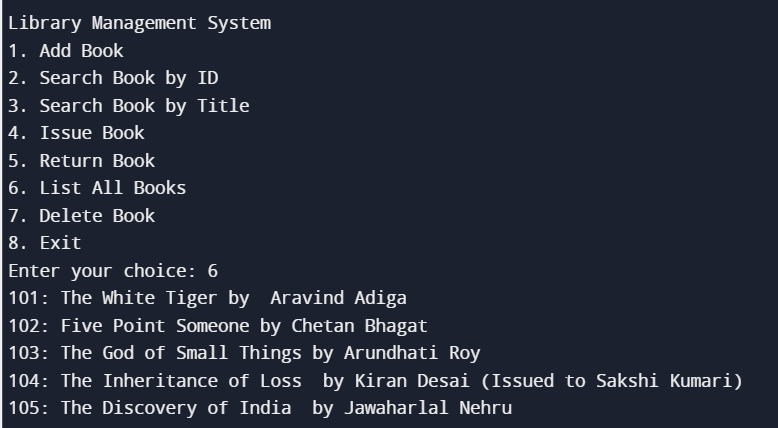


**3.To Search for a Book by Title:-**

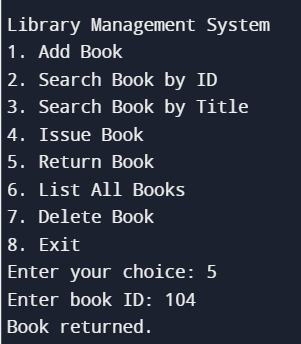


**4.Issue a Book :-**

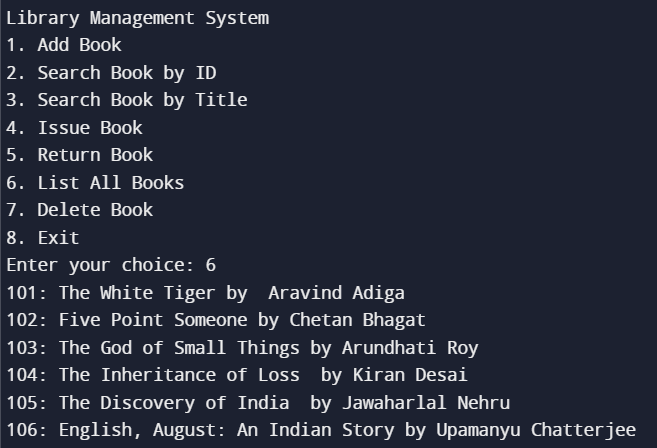




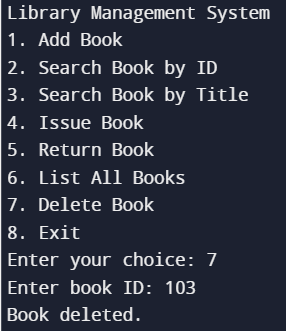
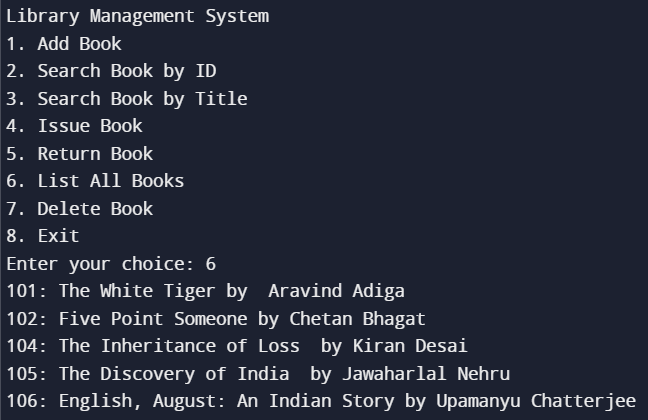
**5.To Return a Book :-**



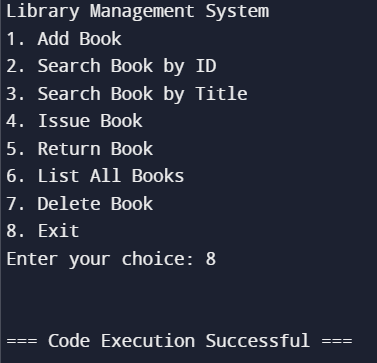
**6.To List the Books :-**



**7.To Delete the book :-**

**8.To Exit the LMS :-**



**SUMMARY**

**This project demonstrates the use of basic data structures and**

**algorithms in implementing a simple Library Management System. It**

**provides a foundational understanding of how these concepts can be**

**applied in real-world applications.**

