HW2_113550084_李博凱

Library Management System Report

Introduction

This report presents the implementation of a Library Management System that uses AVL trees as the main data structure. The system enables efficient management of books, supporting operations such as adding, searching, checking out, returning, and listing books. All operations are designed to achieve optimal time complexity.

Requirements Fulfillment

Basic Operations

1. **Add a new book**: Users can add books with detailed information including title, author, publication year, ISBN, genre, and the number of copies.
2. **Search for a book by published year**: The system retrieves all books published in a specified year, sorted lexicographically by title.
3. **Check out a book**: Users can borrow books. The system tracks borrower information and reduces the count of available copies.
4. **Return a book**: Returned books are processed by increasing the available copies count.
5. **List all books**: The system displays all books sorted lexicographically by title.
6. **Remove a book**: Users can remove books entirely from the library.

Enhanced Features

1. **Detailed book attributes**: Each book includes title, author, publication year, ISBN, genre, total copies, and available copies.
2. **Borrow history tracking**: The system tracks all users who have borrowed each book.

3. **Dual AVL trees**: Two AVL trees are maintained—one sorted by title and the other by publication year. All operations can be performed in O(log n) time.

Data Structure Implementation

Book Information Storage

Books are stored in a `Book` class with the following attributes:

```
class Book {
  private:
    string title;
    string author;
    int year;
    string isbn;
    string genre;
    int totalCopies;
    int availableCopies;
    vector<string> borrowHistory;
    int borrowCount;
  public:
    // Methods for accessing and manipulating book data
};
```

This structure encapsulates all relevant data and status, and tracks borrow history using a vector of strings.

Library Structure

The library maintains two AVL trees and a global borrow history:

```
struct AVLNode {
  Book book;
  AVLNode* left;
  AVLNode* right;
  int height;
  vector<pair<string, string>> borrowHistory;
```

```
    AVLNode(const Book& b) : book(b), left(nullptr), right(nullptr), height(1) {}
};

class Library {
    private:
        AVLNode* titleRoot;
        AVLNode* yearRoot;
        vector<pair<string, string>> borrowHistory;
    public:
        // Public methods for performing library operations
};
```

Sorting Algorithm

The AVL trees ensure that books are stored in sorted order:

1. **Title-based sorting**: The `titleRoot` tree keeps books sorted lexicographically by title. Listing books via in-order traversal automatically results in sorted output.
2. **Year-based sorting**: The `yearRoot` tree organizes books by publication year, with secondary sorting by title when years are equal.

Sorting logic during insertion:

```
// Title-based insertion
if (book.getTitle() < node->book.getTitle()) {
    node->left = insertByTitle(node->left, book);
} else if (book.getTitle() > node->book.getTitle()) {
    node->right = insertByTitle(node->right, book);
}

// Year-based insertion
if (book.getYear() < node->book.getYear() ||
    (book.getYear() == node->book.getYear() && book.getTitle() < node->book.
getTitle())) {
    node->left = insertByYear(node->left, book);
} else if (book.getYear() > node->book.getYear() ||
        (book.getYear() == node→book.getYear() && book.getTitle() > node→bo
ok.getTitle())) {
    node→right = insertByYear(node→right, book);
}
```

This comparison logic ensures the trees are correctly sorted.

Search Algorithm

Two main search functionalities are provided:

1. **Search by title**: A recursive binary search on the title-based AVL tree:

```
AVLNode* Library::searchByTitle(AVLNode* node, const string& title) {
    if (node == nullptr || node→book.getTitle() == title) {
        return node;
    }
    if (title < node→book.getTitle()) {
        return searchByTitle(node→left, title);
    }
    return searchByTitle(node→right, title);
}
```

1. **Search by year**: A recursive traversal to find all books published in a specific year:

```
void Library::searchByYear(AVLNode* node, int year, vector<Book>& results)
{
    if (node == nullptr) return;

    if (node→book.getYear() < year) {
        searchByYear(node→right, year, results);
    } else if (node→book.getYear() > year) {
        searchByYear(node→left, year, results);
    } else {
        searchByYear(node→left, year, results);
        results.push_back(node→book);
        searchByYear(node→right, year, results);
    }
}
```

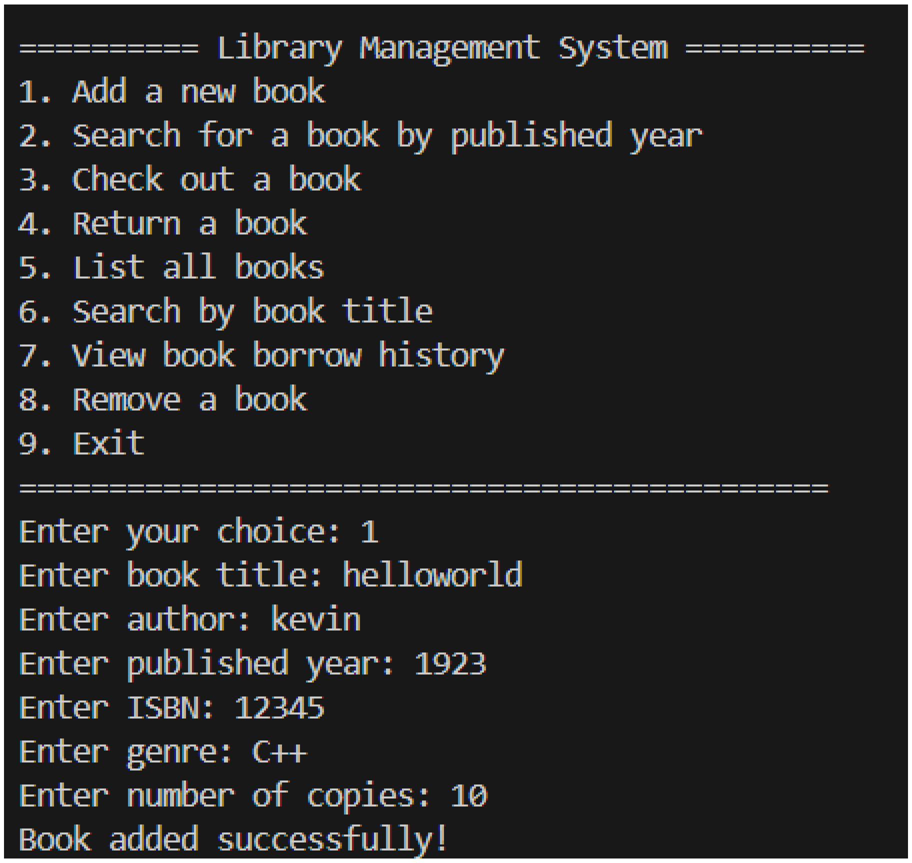
Due to the AVL tree's balance, both search algorithms run in O(log n) time on average.

Time Complexity Analysis

Operation	Time Complexity	Explanation
Add Book	O(log n)	Insertion into two AVL trees, each taking O(log n)
Remove Book	O(log n)	Deletion from both AVL trees
Search by Title	O(log n)	Binary search in a balanced AVL tree
Search by Year	O(log n + k)	O(log n) to locate, plus O(k) to collect k matching books
Check Out/Return	O(log n)	Search and update operations are both logarithmic
List All Books	O(n)	In-order traversal returns all books in sorted order

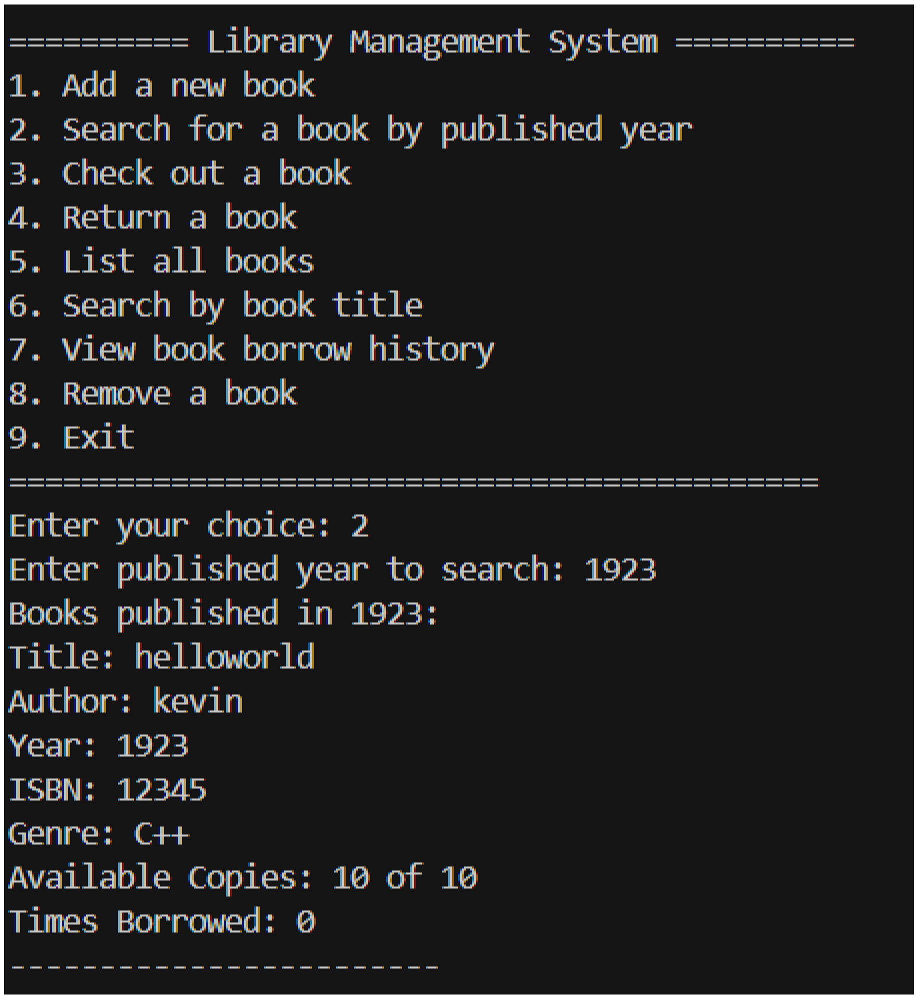
Snapshots of all required operations

1. Add Book



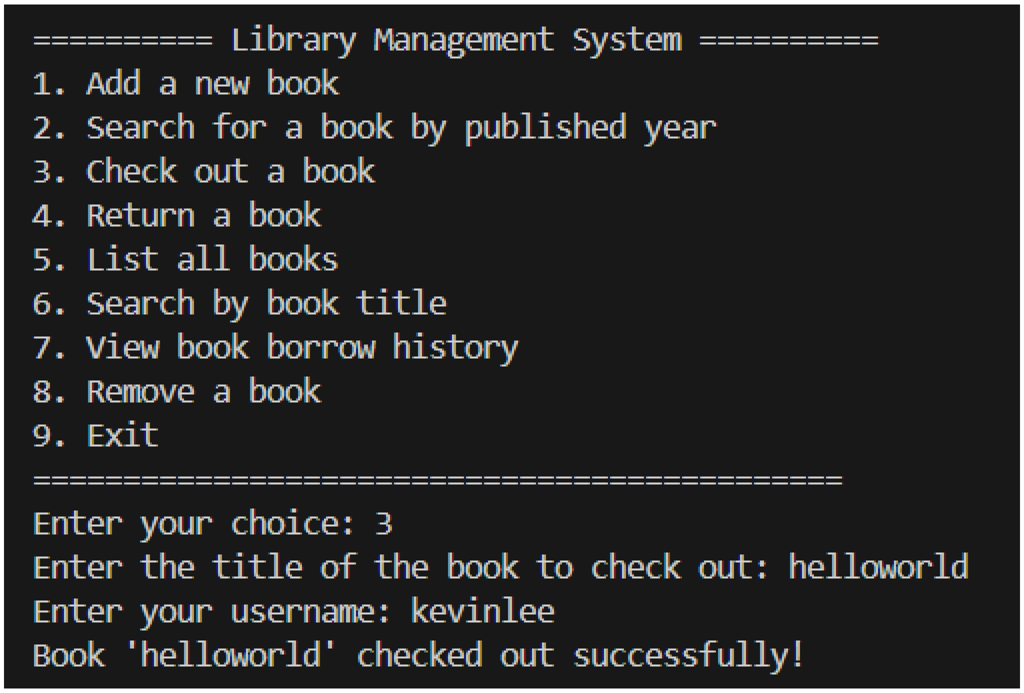
```
========== Library Management System ==========
1. Add a new book
2. Search for a book by published year
3. Check out a book
4. Return a book
5. List all books
6. Search by book title
7. View book borrow history
8. Remove a book
9. Exit
=============================================
Enter your choice: 1
Enter book title: helloworld
Enter author: kevin
Enter published year: 1923
Enter ISBN: 12345
Enter genre: C++
Enter number of copies: 10
Book added successfully!
```

2. Search for a book by published year



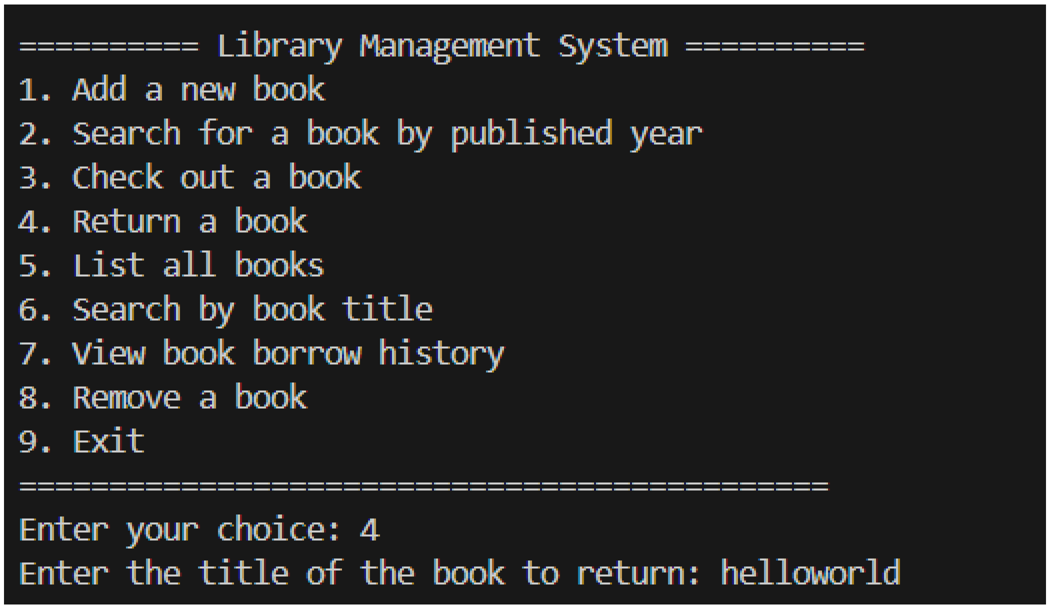
```
========== Library Management System ==========
1. Add a new book
2. Search for a book by published year
3. Check out a book
4. Return a book
5. List all books
6. Search by book title
7. View book borrow history
8. Remove a book
9. Exit
==============================================
Enter your choice: 2
Enter published year to search: 1923
Books published in 1923:
Title: helloworld
Author: kevin
Year: 1923
ISBN: 12345
Genre: C++
Available Copies: 10 of 10
Times Borrowed: 0
------------------------
```

3. Check out a book



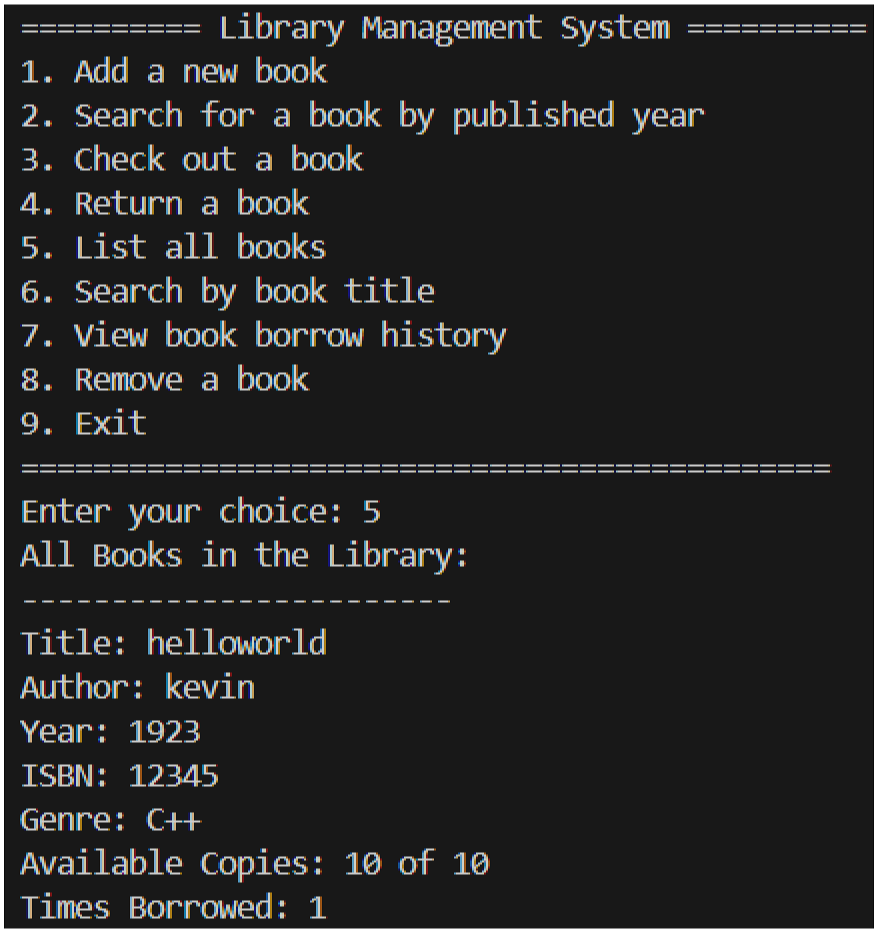
```
========== Library Management System ==========
1. Add a new book
2. Search for a book by published year
3. Check out a book
4. Return a book
5. List all books
6. Search by book title
7. View book borrow history
8. Remove a book
9. Exit
=============================================
Enter your choice: 3
Enter the title of the book to check out: helloworld
Enter your username: kevinlee
Book 'helloworld' checked out successfully!
```

4. Return a book



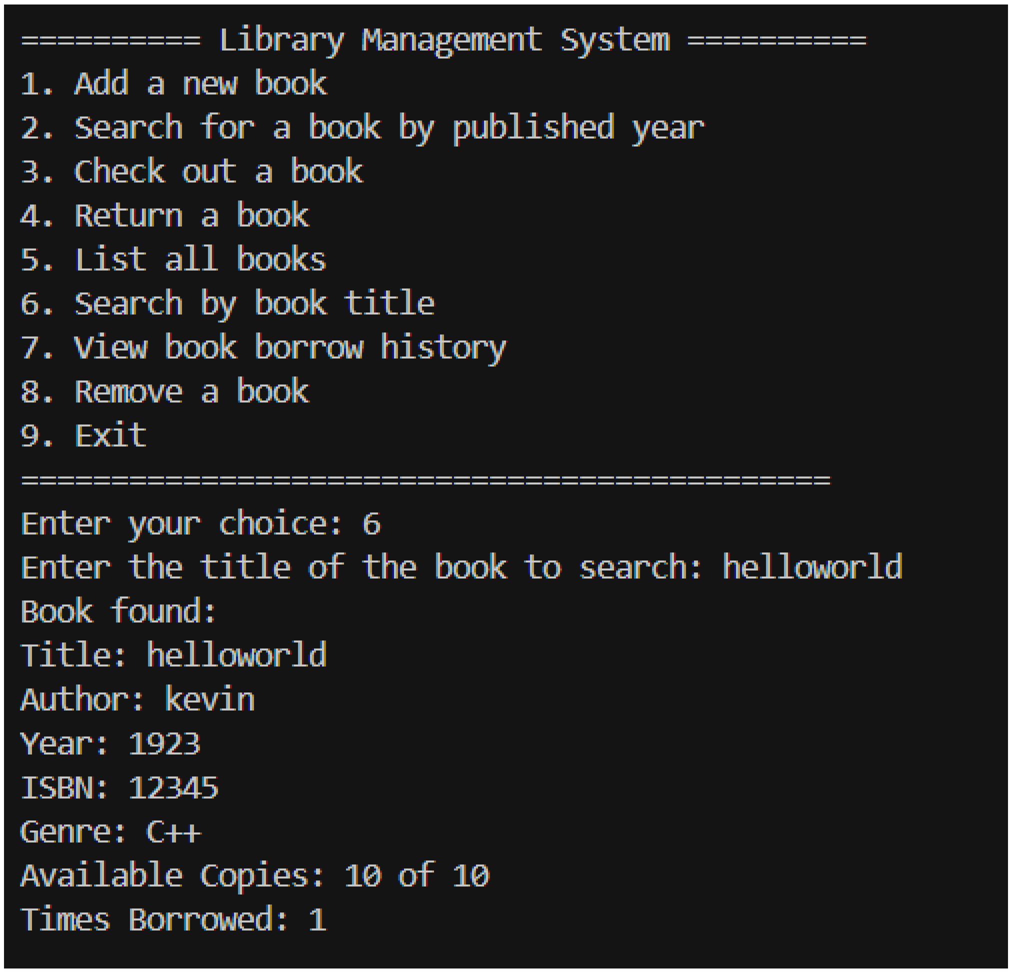
```
========== Library Management System ==========
1. Add a new book
2. Search for a book by published year
3. Check out a book
4. Return a book
5. List all books
6. Search by book title
7. View book borrow history
8. Remove a book
9. Exit
=============================================
Enter your choice: 4
Enter the title of the book to return: helloworld
```

5. List all books

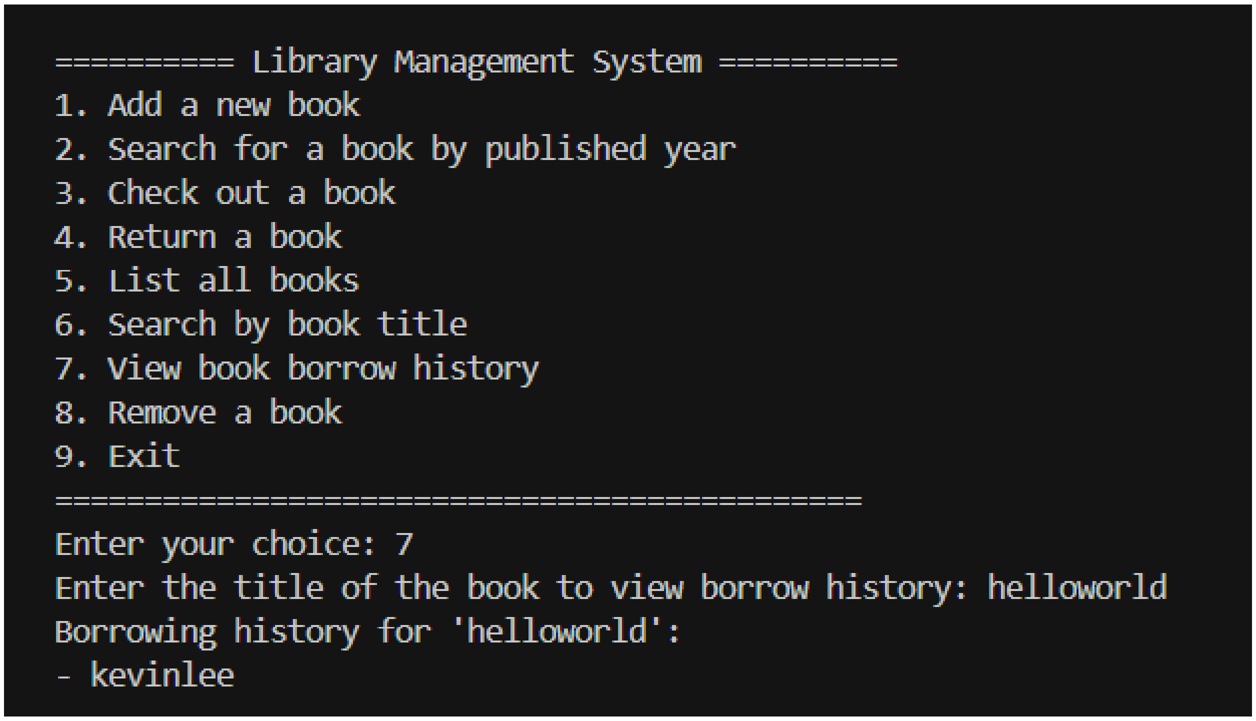


```
========== Library Management System ==========
1. Add a new book
2. Search for a book by published year
3. Check out a book
4. Return a book
5. List all books
6. Search by book title
7. View book borrow history
8. Remove a book
9. Exit
=============================================
Enter your choice: 5
All Books in the Library:
------------------------
Title: helloworld
Author: kevin
Year: 1923
ISBN: 12345
Genre: C++
Available Copies: 10 of 10
Times Borrowed: 1
```

6. Search by book title

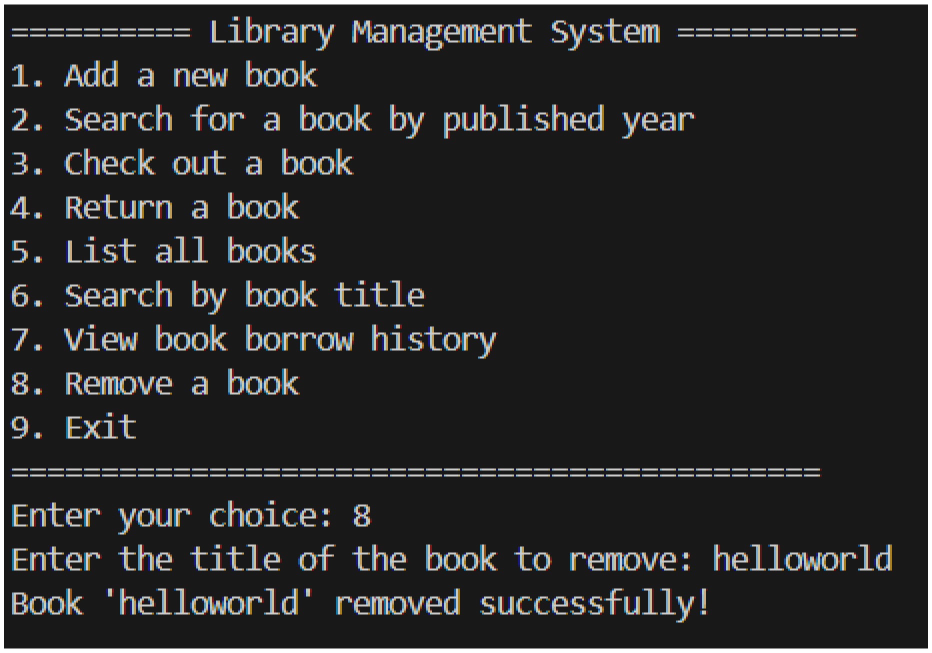

```
========== Library Management System ==========
1. Add a new book
2. Search for a book by published year
3. Check out a book
4. Return a book
5. List all books
6. Search by book title
7. View book borrow history
8. Remove a book
9. Exit
=============================================
Enter your choice: 6
Enter the title of the book to search: helloworld
Book found:
Title: helloworld
Author: kevin
Year: 1923
ISBN: 12345
Genre: C++
Available Copies: 10 of 10
Times Borrowed: 1
```

7. View book borrow history



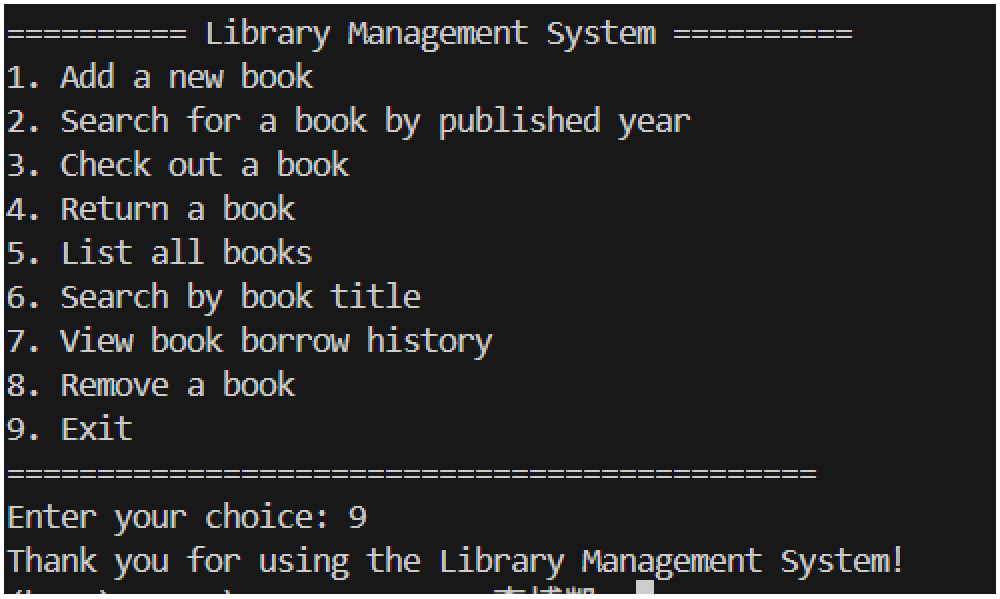
```
========== Library Management System ==========
1. Add a new book
2. Search for a book by published year
3. Check out a book
4. Return a book
5. List all books
6. Search by book title
7. View book borrow history
8. Remove a book
9. Exit
=============================================
Enter your choice: 7
Enter the title of the book to view borrow history: helloworld
Borrowing history for 'helloworld':
- kevinlee
```

8. Remove a book



```
========== Library Management System ==========
1. Add a new book
2. Search for a book by published year
3. Check out a book
4. Return a book
5. List all books
6. Search by book title
7. View book borrow history
8. Remove a book
9. Exit
=============================================
Enter your choice: 8
Enter the title of the book to remove: helloworld
Book 'helloworld' removed successfully!
```

9. Exit



```
========== Library Management System ==========
1. Add a new book
2. Search for a book by published year
3. Check out a book
4. Return a book
5. List all books
6. Search by book title
7. View book borrow history
8. Remove a book
9. Exit
=============================================
Enter your choice: 9
Thank you for using the Library Management System!
```

Defensive Programming Techniques

The system employs various defensive programming strategies to ensure robustness, reliability, and to prevent runtime errors.

Book Class Error Prevention

1. **Availability Check for Checkout**

   Before allowing a user to check out a book, the system verifies that there are available copies. This prevents users from borrowing books that are out of stock and ensures the internal data remains consistent.

   ```
   cpp
   複製程式碼
   bool Book::checkOut(const string& username){
       if(availableCopies > 0){
           availableCopies--;
           borrowHistory.push_back(username);
           borrowCount++;
           return true;
       }
       return false;
   ```

```
}
```

2. **Copies Limit Validation for Returns**

   When a user returns a book, the system checks that the number of available copies does not exceed the original total. This prevents returning more copies than were actually checked out.

```
cpp
複製程式碼
bool Book::returnBook(){
    if(availableCopies < totalCopies){
        availableCopies++;
        return true;
    }
    return false;
}
```

Library Class Safeguards

1. **Null Reference Protection**

   When searching for a book by title, the system checks whether the current node is `nullptr` before accessing its content. This avoids null pointer exceptions during AVL tree traversal.

```
cpp
複製程式碼
AVLNode* Library::searchByTitle(AVLNode* node, const string& title){
    if(node == nullptr || node->book.getTitle() == title){
        return node;
    }
    // ...
}
```

2. **Safe Search Results Handling**

   When retrieving the borrow history of a specific book, the system first searches for the book and returns an empty list if the book is not found. This avoids returning null and allows the calling code to handle results safely.

   ```
   cpp
   複製程式碼
   vector<string> Library::getSpecificBookBorrowHistory(const string& title){
       AVLNode* node = searchByTitle(titleRoot, title);
       return node ? node->book.getBorrowHistory() : vector<string>();
   }
   ```

3. **Operation Validation Before Deletion**

   When attempting to remove a book, the system ensures that the book exists before trying to delete it. This prevents unnecessary computation or accidental errors from invalid operations.

   ```
   cpp
   複製程式碼
   bool Library::removeBook(const string& title){
       AVLNode* node = searchByTitle(titleRoot, title);
       if(node == nullptr) return false;
       // ...
   }
   ```