**Objectives:**

To make a project using Data Structure elements such as linked list, tree, graph etc.

**Introduction:**

We developed a Library Management System using Data Structure.

A data structure is a way of organizing and storing data in a computer to efficiently use, manage, and manipulate it. It provides a systematic way to access and perform operations on the data. Data structures can be linear, like arrays, lists, stacks, queues, or non-linear, like trees, graphs, and hash tables. Each structure has its own characteristics, advantages, and suitable applications based on how the data needs to be stored, accessed, and modified. Choosing an appropriate data structure is crucial for optimizing algorithms and improving the efficiency of programs.

There are numerous types of data structures, each with its own characteristics, purposes, and applications. Here's an overview of some common types:

**1. Arrays**: An array is a collection of elements stored at contiguous memory locations. It allows for easy access to elements using indexing but has a fixed size.

**2. Linked Lists:** Linked lists are made up of nodes where each node contains data and a reference to the next node (and sometimes the previous node in a doubly linked list). They offer dynamic size and efficient insertion/deletion but lack random access.

**3. Stacks:** A stack is a Last-In-First-Out (LIFO) data structure where elements are inserted and removed from the same end (top). Operations like push (insert) and pop (remove) are fundamental.

**4. Queues:** A queue is a First-In-First-Out (FIFO) data structure where elements are inserted at the rear and removed from the front. Enqueue (insert) and dequeue (remove) are primary operations.

**5. Trees:** Trees are hierarchical data structures with nodes arranged in a hierarchical order. They have a root node, internal nodes, and leaf nodes. Common types include binary trees, AVL trees, red-black trees, etc.

**6. Graphs:** Graphs consist of nodes (vertices) connected by edges. They can be directed or undirected and have various representations (adjacency matrix, adjacency list) depending on the application.

Each data structure has its strengths and weaknesses, making them suitable for different scenarios and tasks.

**Project Description:**

We endeavored to create a library management system designed for efficiently handling various tasks related to books within a library setting. Our system utilizes linked lists for its implementation and encompasses the following features:

1. Creating a new record for a book

2. Displaying a comprehensive list of all books

3. Modifying existing book information

4. Facilitating the rental of a book

5. Managing the return process for borrowed books

6. Performing searches for specific books

7. Deleting book records

8. Saving changes and exiting the system **Elements of Data Structure Used in the project:**

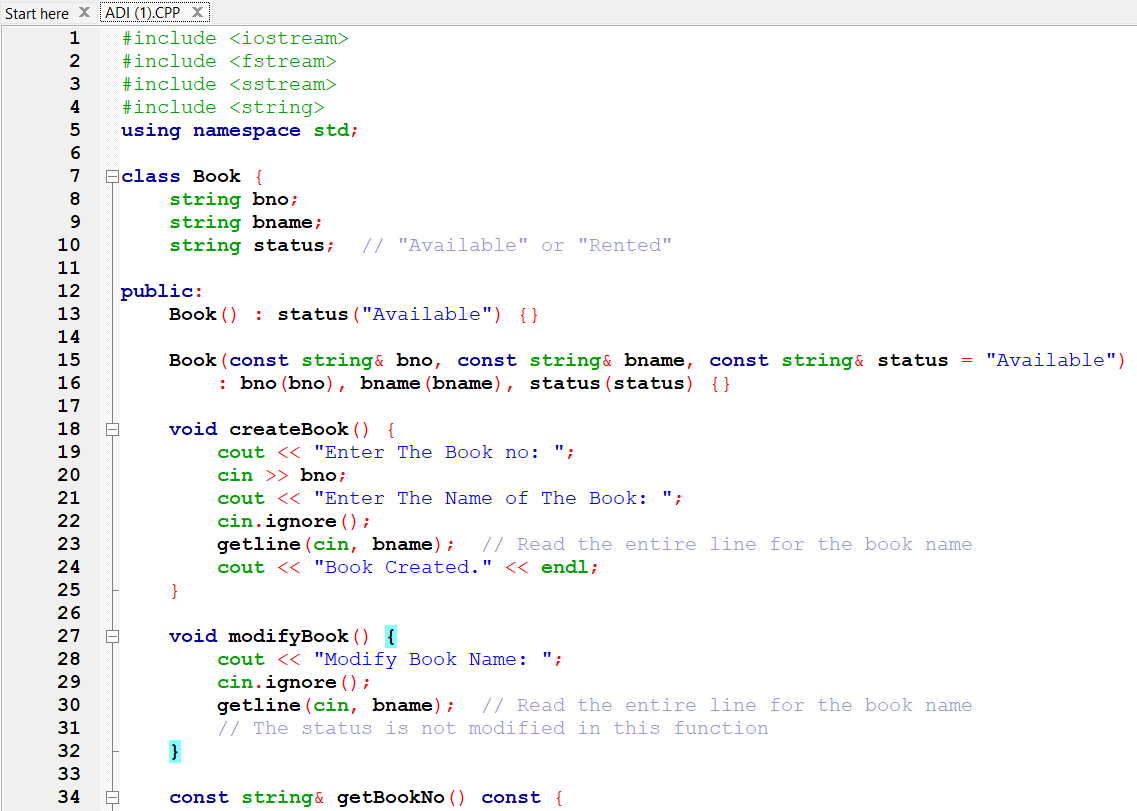
**1. Traversal:** The process of traversing a linked list entails sequentially visiting each node in the list. Starting from the head node, one moves through the list by following the pointers or references from one node to the next until reaching the end, usually indicated by a null reference.

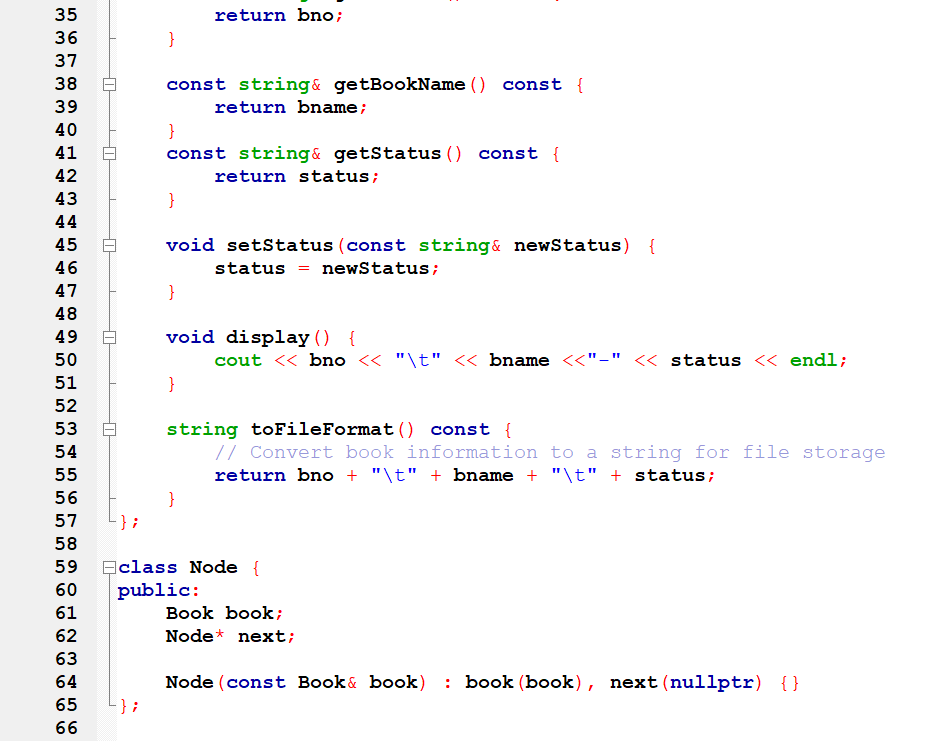
**2. Searching:** In a linked list, searching involves seeking a specific element or value within the list. Commencing at the head node, the list is traversed, and the target value is compared with the data in each node until the target value is located or the end of the list is reached.

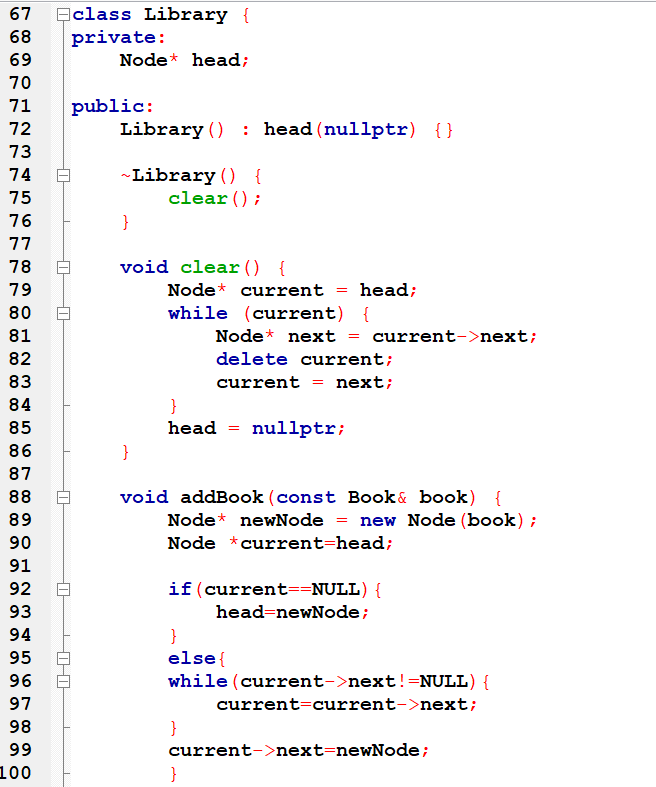
**3. Insertion:** Inserting a node into a linked list requires adding a new node at a designated position within the list. The insertion process may involve updating references or pointers of neighboring nodes, depending on whether the new node is being added to the beginning, end, or middle of the list.

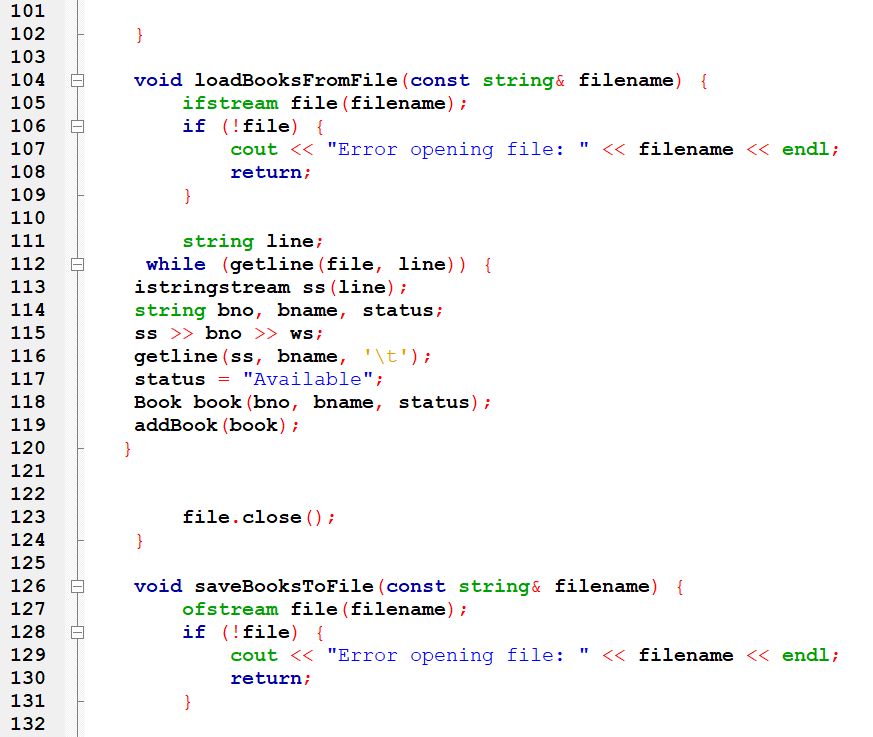
**4. Deletion:** Deleting a node from a linked list involves removing a node from the list based on a specified condition, such as a particular value or position. This process typically includes adjusting pointers or references of neighboring nodes to ensure the continued integrity of the list after the removal.

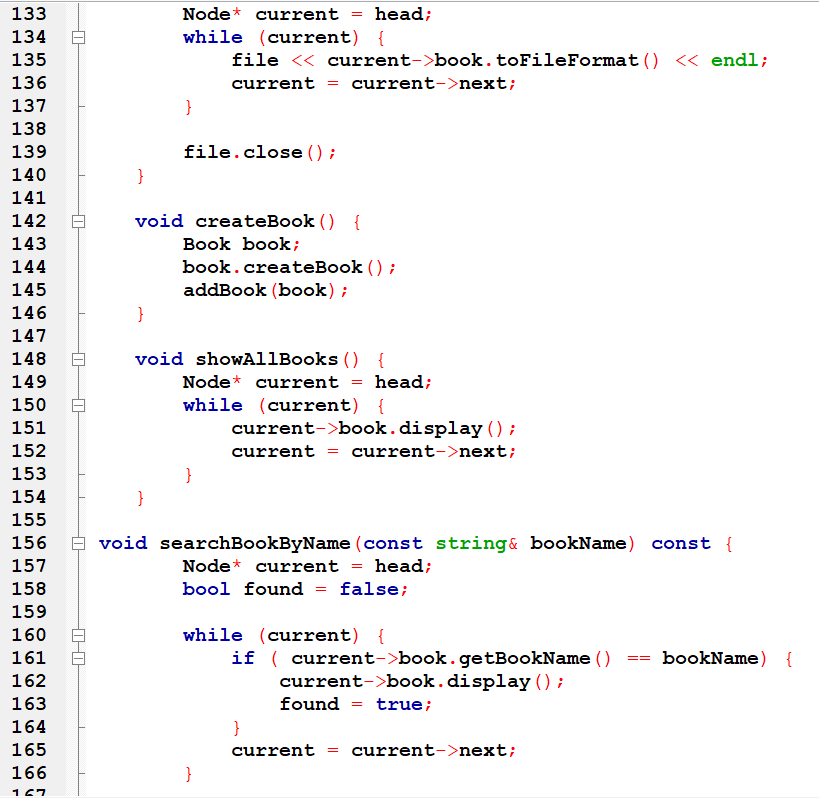
**Code:**

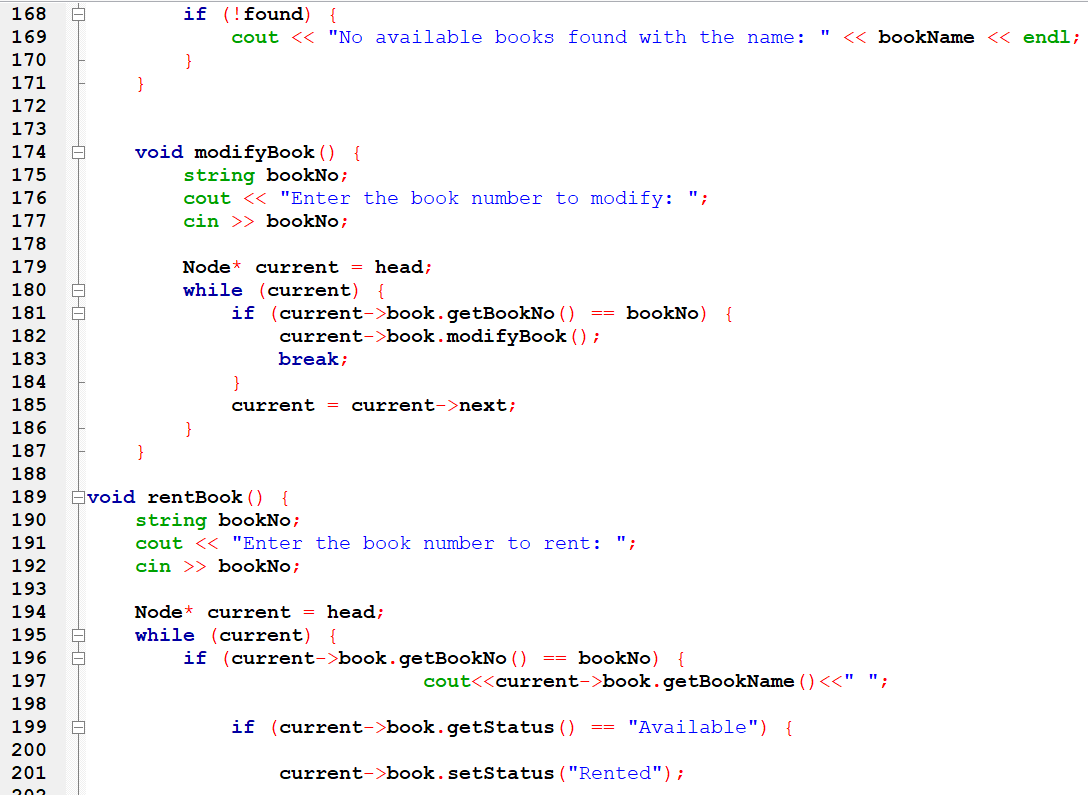


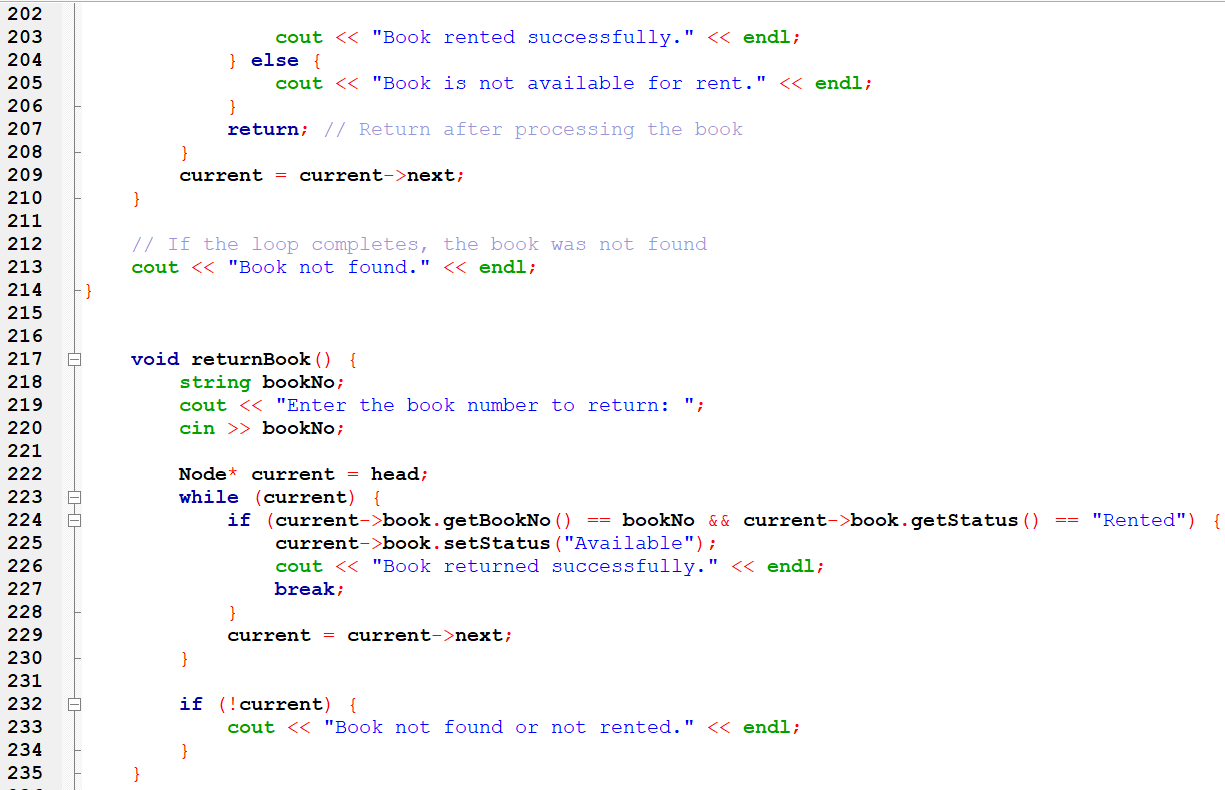


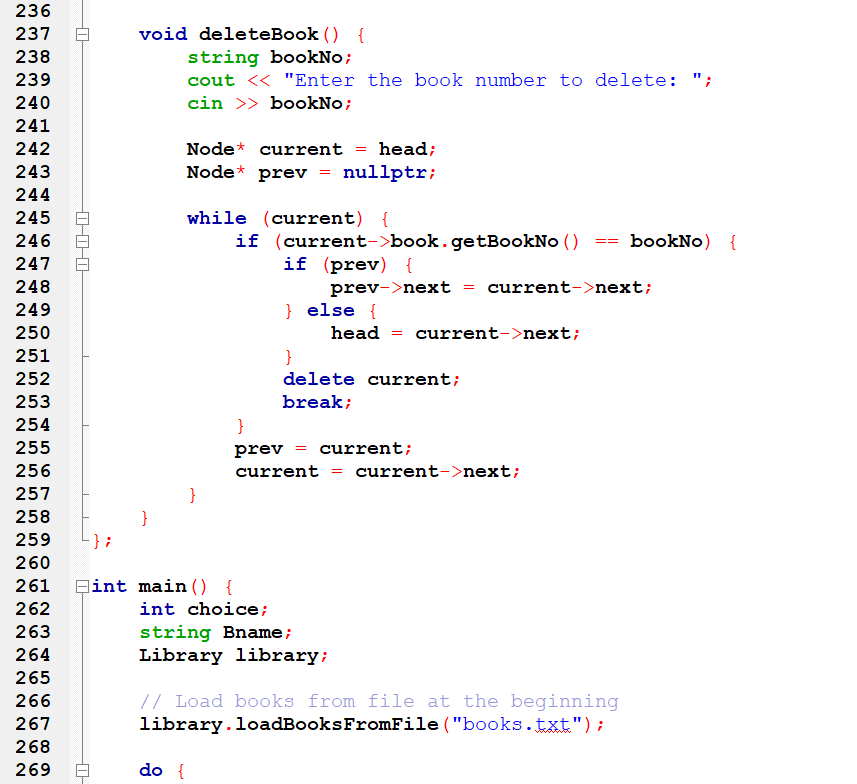


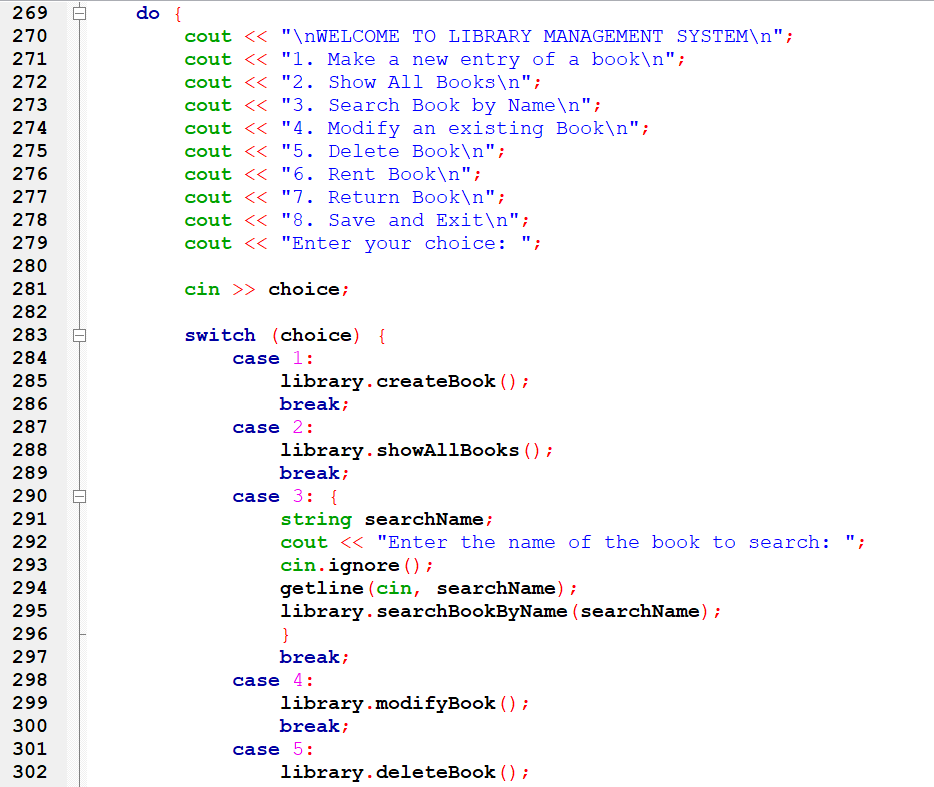


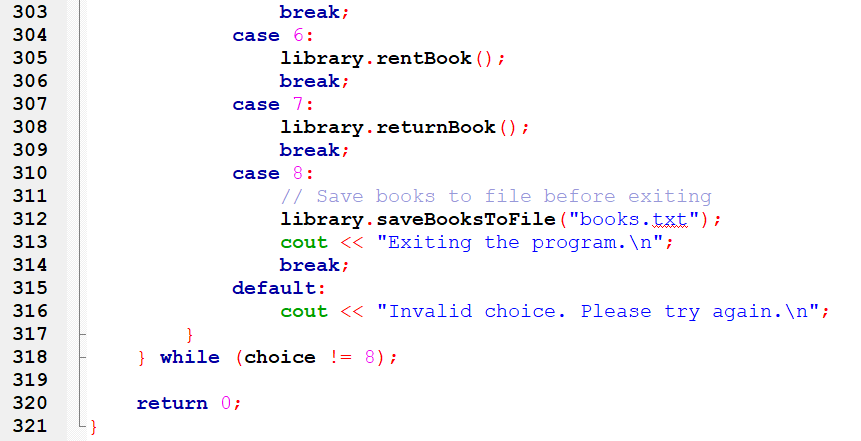






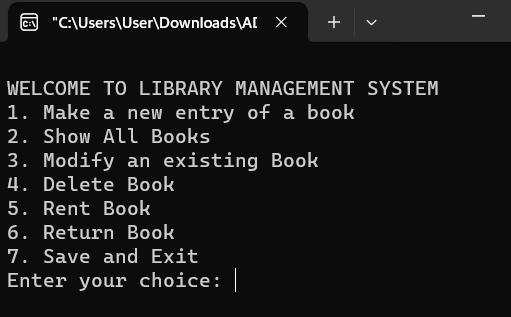




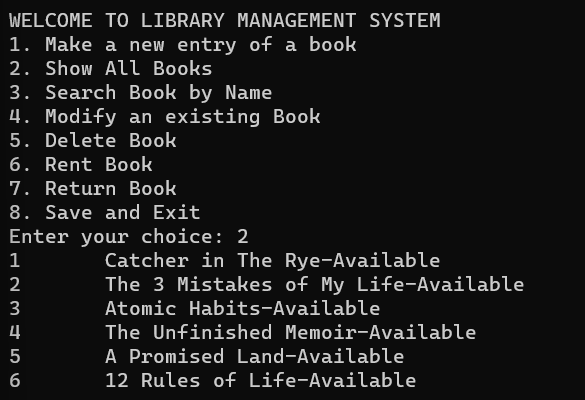


**Output:**

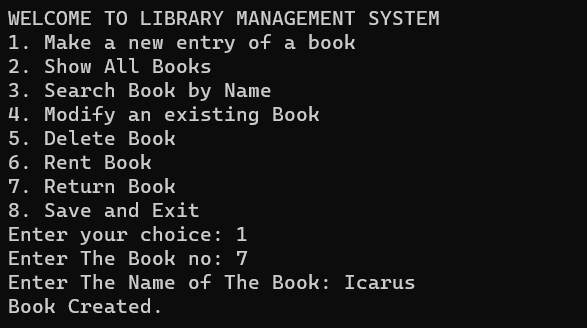
1. **Greeting the user:**

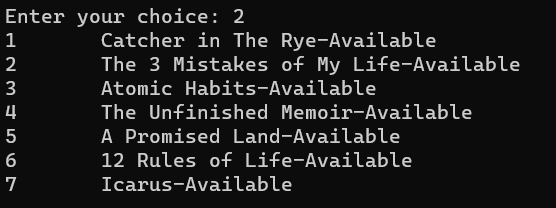


1. **Showing list of all books:**

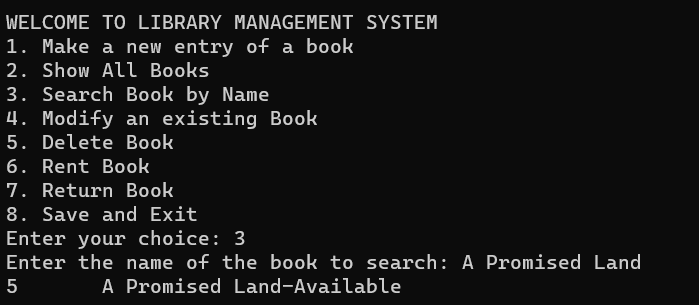


1. **Making a new entry of a book:**

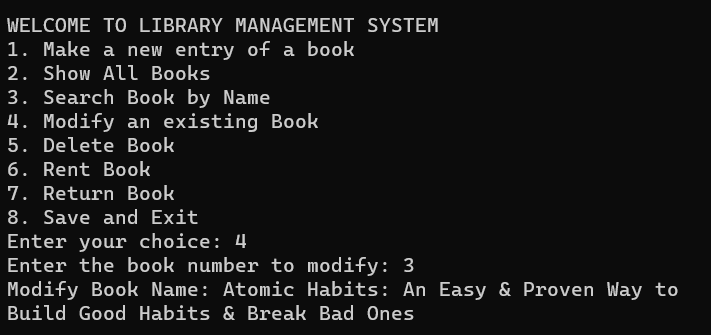


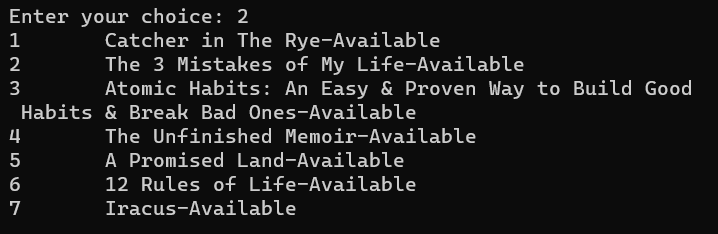


1. **Searching a book:**

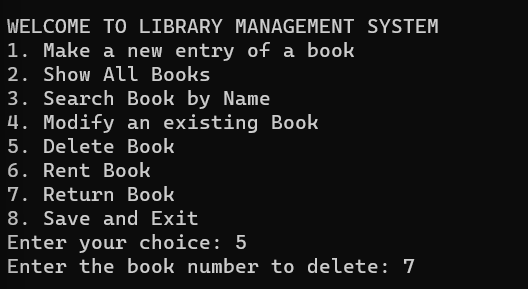


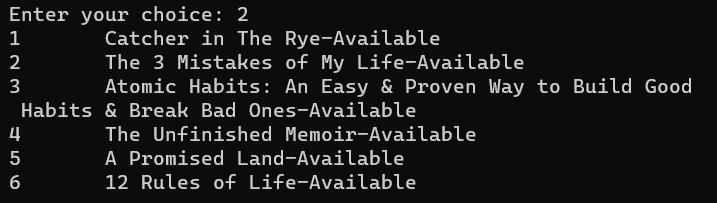
1. **Modifying an existing book:**



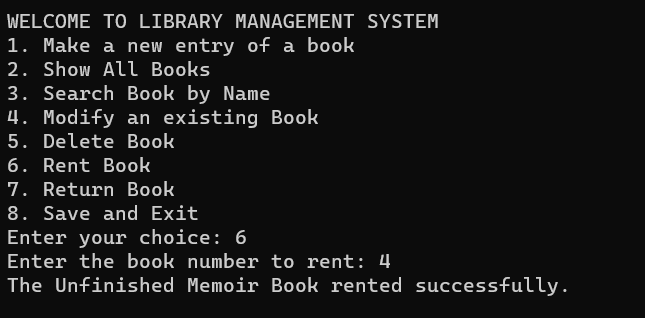


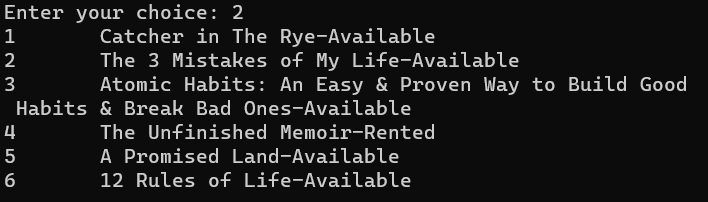
1. **Deleting a book:**



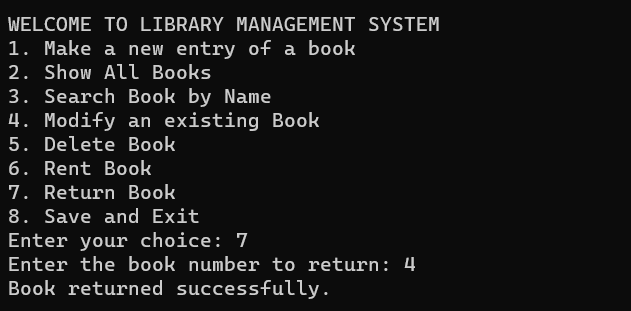


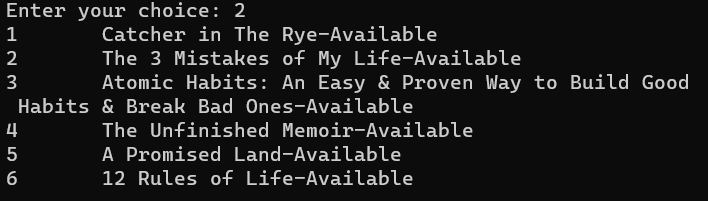
1. **Renting a book:**



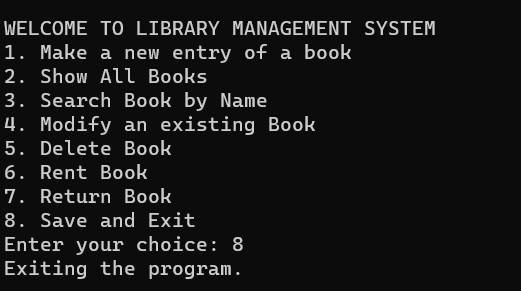


1. **Returning a book:**





1. **Saving and exiting the system:**



**Discussion:**The library management system project excels in its user-friendly design, utilizing a linked list for hierarchical book record management. Essential features, such as book creation, modification, rental, return, search, deletion, and system-saving capabilities, contribute to seamless library operations. Notably, the absence of a late return fine and potential scalability issues underscore areas for refinement. Addressing these limitations will not only improve the system's effectiveness but also pave the way for its adaptation to evolving library needs. In summary, the project showcases successful library management software development, offering insights into user interactions, data structures, and avenues for ongoing enhancement.

**Conclusion:**

In conclusion, the library management system project has achieved its objectives by implementing a linked list data structure, providing a streamlined and user-friendly approach to book record management. With features ranging from book creation and modification to rental, return, search, deletion, and system-saving capabilities, the system greatly enhances the efficiency of library operations. However, recognizing the absence of a late return fine and potential scalability challenges, there exists a clear path for future enhancements. These considerations not only underscore the success of the current project but also pave the way for ongoing improvements, ensuring the system's adaptability to the dynamic landscape of library management.   
  
**Reference :**1. https://www.geeksforgeeks.org/learn-data-structures-and-algorithms-dsa-tutorial/  
2. <https://www.programiz.com/dsa/trees>  
3. https://www.skoolbeep.com/blog/library-management-system/