National University of Computer and Emerging Sciences

Lahore Campus

Data Structures

Assignment #2

Date: 24-10-2025

Course code: CS2001

Instructor

Mr. Muhammad Naveed

Total marks:

Questions:

TA

Menahil Amir

_____ _____ _____

Roll No Section Student Signature

CLO 02: Demonstrate basic concepts of Data Structures

Attempt all the questions.

QUESTION 1: (30 Marks)

Whenever we visit any place where we need to be served, the serving is often on first come first serve basis. Suppose you are to implement such a system of a Hospital's OPD, where patients are registered and enqueued for their medical examination. You need to take the following information from the patients:

1. Name
2. Age
3. Condition (Normal/Critical/Intense)
4. /Priority *(Derived)*

Your program should be able to perform the following functionalities:

- Enqueue Patients
- Specify Condition and Priority level
- Update Information
- View Current Queue (each patient detail)
- Prioritize Patients with severe condition

We, living in Pakistan, know that our system isn't rigid on rules. There are people with strong references that need to be served first, your program should be able to enqueue such people on high priority and make sure that the interface never witnesses it.

Senior Citizens are not to be made waiting in accordance to general ethics. Your program should be able to make such people wait for MAXIMUM 3 people to be served before them. The program should be able to handle exceptions in a decent manner. Following exceptions may occur:

- Incomplete Information
- Improper derived priority
- Delayed serving

Use queue and stack with linked list. You're not allowed to iterate and access a random middle value, only the front of queue and top or stack should be accessed.

Question 2: (90 Marks)

This question requires implementing a library management system using Binary Search Trees (BSTs), AVL Trees, and MinHeaps. Each data structure plays a critical role in efficiently managing books, users, and user demand. The goal is to create an interconnected system that allows for easy addition, tracking, and management of library resources.

1. Binary Search Tree (BST) for Book Records

The BST organizes book records in the library by ISSN (International Standard Serial Number), ensuring efficient insertion, search, update, and deletion operations. It keeps book records sorted, which allows for fast lookup and enables other operations to work in O(log n) time complexity.

Node Structure:

- **ISSN**: A unique identifier for each book.
- **Title**: The title of the book.
- **List of Author Names**: Names of all authors of the book.
- **Reference Copies**: The number of reference copies available in the library (these cannot be borrowed).
- **Issuable Copies**: The number of copies available for issuance.
- **Heap Pointer**: A pointer to a MinHeap associated with the book. This heap tracks user demand based on borrowing requests, prioritizing those who requested earliest.

Each node in the BST will represent a book, organized by ISSN. The BST will manage books by ISSN to ensure unique records, making it easy to find and manage books efficiently.

2. AVL Tree for User Accounts

The AVL Tree manages user accounts in a balanced manner, ensuring that adding, finding, and updating user records are done in O(log n) time. The AVL Tree automatically rebalances itself after insertions or deletions to maintain efficient access.

Node Structure:

- **User ID:** A unique identifier for each user.
- **Name**: The name of the user.
- **Currently Borrowed Books:** A list of ISSNs for books currently borrowed by the user.

• **Fine Amount:** The current fine amount owed by the user, which increases if the user fails to return books on time.

Each node in the AVL Tree represents a user, with User ID as the sorting key. The AVL Tree ensures that user data, such as borrowed books and fines, is stored efficiently and accessible with minimal delay.

3. MinHeap for Tracking Book Demand

Each book has an associated MinHeap to track user demand by request date. The MinHeap prioritizes users who requested to borrow the book earlier, ensuring fairness and efficient management of issuable copies.

Node Structure:

• **User ID:** The ID of the user who requested the book.

• **Request Date:** The date when the user requested to borrow the book. The heap uses this date as the sorting key, with the earliest requests at the root.

For each book in the BST, the MinHeap will store a sorted list of user requests, prioritizing users based on their request date. When a copy becomes available, the request at the root (earliest) is served first.

TASKS

1. Add a Book

Insert a new book into the BST, organized by ISSN. If the same title appears with a different ISSN, treat it as a unique record. Initialize a MinHeap for each book upon insertion, which will be used to track user requests.

Procedure:

1. Create a node with the book details and insert it into the BST based on ISSN. **2**. Allocate a new MinHeap for the book to manage future user requests.

Outcome: The BST organizes books by ISSN, and each book is equipped with a MinHeap to track user demand.

Input:

> Add Book

Enter ISSN: 101

Enter Title: Pakistani Literature

Enter Authors (comma separated): Ahmed Ali

Enter Reference Copies: 2

Enter Issuable Copies: 3

Output:

Book added: ISSN: 101, Title: Pakistani Literature

2. Register a User

Add a new user to the AVL Tree, sorted by User ID. Each user is initialized with an empty borrowed list and a fine amount of zero.

Procedure:

1. Create a node for the new user with the given details and insert it into the AVL Tree. **2**. Ensure the tree rebalances as needed to maintain efficiency.

Outcome: The AVL Tree now contains a node for the new user, organized by User ID, ready to track borrowed books and fines.

Input:

> Register User

Enter User ID: 1001

Enter Name: Sara

> Register User

Enter User ID: 1002

Enter Name: Hassan

Output:

User registered: User ID: 1001, Name: Sara

User registered: User ID: 1002, Name: Hassan

3. Borrow Book

When a user requests to borrow a book, their User ID and request date are added to the MinHeap of the requested book. If there are issuable copies available, the book is issued to the user with the earliest request.

Procedure:

1. Locate the book in the BST by ISSN and access its MinHeap.

2. Add the user's ID and request date to the MinHeap.

3. If issuable copies are available, remove the user from the heap's root (earliest request) and issue the book, reducing available copies by one.

4. Update the user's record in the AVL Tree to reflect the borrowed book by adding the book's ISSN to their borrowed list.

Outcome: The user either borrows the book immediately or is added to the queue to await availability. The MinHeap prioritizes the earliest requests, and the AVL Tree updates the user's borrowed books list.

Input:

> Borrow Book

Enter User ID: 1001

Enter ISSN: 101

Output:

Book borrowed: ISSN: 101, Title: Pakistani Literature

4. Return Book

When a user returns a book, the issuable copies count in the BST is increased. If overdue, a fine is added to the user's account in the AVL Tree. If other users are waiting for the book, the next user in the MinHeap queue is issued the book.

Procedure:

1. Locate the book in the BST and increase the count of issuable copies.

2. If the return is overdue, calculate the fine based on the overdue days and update the fine amount in the user's AVL Tree record.

3. Remove the book's ISSN from the user's currently borrowed list in the AVL Tree.

4. If there are pending requests in the MinHeap, serve the book to the next user in line by issuing a copy and updating records accordingly.

Outcome: Issuable copies are updated, fines are calculated for overdue books, and the next user in line receives the book if applicable.

Input:

> Return Book

Enter User ID: 1001

Enter ISSN: 101

Enter Return Date (yyyy-mm-dd): 2024-10-28

Output:

Book returned: ISSN: 101, Title: Pakistani Literature

Input (if overdue):

> Return Book

Enter User ID: 1002

Enter ISSN: 101

Enter Return Date (yyyy-mm-dd): 2024-11-05

Output:

Book returned: ISSN: 101, Title: Pakistani Literature

Fine added: User ID: 1002, Fine: $5

Define a Borrow Period:

• Set a predefined borrow period for all books (e.g., 14 days) or specify different borrow periods for each book type if required.

Store Borrow Dates:

• When a user borrows a book, store the borrow date along with the book's ISSN in the user's record within the **AVL Tree**. This will allow you to keep track of when each book was borrowed by each user. **Return Book Function**:

• When a user returns a book, capture the **return date**.

• Use the **borrow date** (stored in the user's record) and add the **borrow period** to calculate the **due date**.

Compare Return Date with Due Date:

• If the **return date** is later than the **due date**, calculate the overdue period and apply a fine based on the number of overdue days.

Fine Calculation:

• For each overdue day, add a fine amount (e.g., $1 per day) to the user's **fine amount** in the **AVL Tree**.

5. Update Book Information

Update any field of a book record in the BST except the ISSN. Ensure that the MinHeap for book demand remains unaffected unless fields relevant to the demand change.

Procedure:

1. Locate the book in the BST by ISSN.

2. Update the necessary fields (such as title or author names).

Outcome: The book record is updated in the BST, with no changes to the MinHeap or AVL Tree structures.

Input:

> Update Book

Enter ISSN: 101

Enter Field to Update (Title/Authors/Reference Copies/Issuable Copies): Issuable Copies Enter New Value: 5

Output:

Book updated: ISSN: 101, New Issuable Copies: 5

6. Delete a Book

Remove a book from the BST by its ISSN. Clear the associated MinHeap and update user records in the AVL Tree to remove references to this book.

Procedure:

1. Locate and remove the book node from the BST.

2. Deallocate the MinHeap associated with the book, clearing all user requests. **3.** Remove the book from each user's borrowed list in the AVL Tree.

Outcome: The book is fully removed from the system, including all user requests and borrow records.

Input:

> Delete Book

Enter ISSN: 303

Output:

Book deleted: ISSN: 303, Title: History of Pakistan

7. Find Most Demanded Book

Traverse the BST and inspect each book's MinHeap to determine the book with the highest demand (most

pending requests).

Procedure:

1. Traverse the BST, checking the size of each MinHeap.

2. Track the book with the largest heap size (highest demand).

Outcome: Display the title of the most demanded book and the current number of pending requests.

Input:

> Find Most Demanded Book

Output:

Most Demanded Book: Pakistani Literature, ISSN: 101, Pending Requests: 2

8. Display User Record

Given a User ID, display the complete record of the user from the AVL Tree, including their name, list of borrowed books, and any outstanding fine amount.

Procedure:

1. Locate the user in the AVL Tree by User ID.

2. Display the user's full record, including borrowed books and fines.

Outcome: The user's information, including current borrowings and fines, is shown.

Input:

> Display User Record

Enter User ID: 1001

Output:

User ID: 1001, Name: Ali

Borrowed Books: ISSN: 101

Outstanding Fine: $0

9. Calculate Fines for All Users

Traverse the AVL Tree to calculate overdue fines for all users based on books not returned by their due dates. Generate a sorted list of users with outstanding fines.

Procedure:

1. Traverse the AVL Tree.

2. For each user, calculate fines for overdue books.

3. Display a list of users with outstanding fines, sorted by User ID.

Outcome: A list of users with fines is displayed, helping the library manage overdue returns.

Input:

> Calculate Fines

Output:

User ID: 1002, Fine: $5

No fines for User ID: 1001, User ID: 1003