COP 5536 Fall 2023

Programming Project

Due – Nov 18th 2023, 11:59 pm EST

GatorLibrary Management System

Problem Description:

GatorLibrary is a fictional library that needs a software system to efficiently manage its books, patrons, and borrowing operations.

The system should utilize a Red-Black tree data structure to ensure efficient management of the books. Implement a priority-queue mechanism using Binary Min-heaps as a data structure for managing book reservations in case a book is not currently available to be borrowed. Each book will have its own min-heap to keep track of book reservations made by the patrons.

Each node in the Red-Black tree will represent a book and will have the following structure:

- **BookId** // Integer ID
- **BookName** //Name of the book
- **AuthorName** //Name of the Author
- **AvailabilityStatus** //To indicate whether it is currently borrowed
- **BorrowedBy** //ID of the Patron who borrowed the book
- **ReservationHeap**: Implement Binary Min-heap for managing book reservations and waitlists for the book ordered by the patron's priority which is an integer. (*Priority 1 has precedence over Priority 2 and so on).* Ties need be broken by considering the timestamp at which the reservation was made *(first come first serve basis)*. Every node of the Min-heap should contain (patronID, priorityNumber, timeOfReservation)

Note:*

- *Assume that each waitlist is limited to 20.*
- *While taking timestamps, ensure the precision is high enough.*

The system should support the following operations:

1. PrintBook(bookID): Print information about a specific book identified by its unique bookID (e.g., title, author, availability status).
Note: If not found, Print "*BookID not found in the Library*"*

2. PrintBooks(bookID1, bookID2): Print information about all books with bookIDs in the range [bookID1, bookID2].

3. InsertBook(bookID, bookName, authorName, availabilityStatus, borrowedBy, reservationHeap): Add a new book to the library. BookID should be unique, and availability indicates whether the book is

available for borrowing. BorrowBy and reservationHeap is empty intially when a book is inserted and gets updated when the book is borrowed.
Note: There is only one copy of a book i.e. all books are unique.*

4. BorrowBook(patronID, bookID, patronPriority): Allow a patron to borrow a book that is available and update the status of the book. If a book is currently unavailable, create a reservation node in the heap as per the patron's priority (patronPriority).
**Note: A patron should not be allowed to reserve the book that they already have in possession.*

5. ReturnBook(patronID, bookID): Allow a patron to return a borrowed book. Update the book's status and assign the book to the patron with highest priority in the Reservation Heap. *(if there's a reservation)*.

6. DeleteBook(bookID): Delete the book from the library and notify the patrons in the reservation list that the book is no longer available to borrow.

7. FindClosestBook(targetID) : Find the book with an ID closest to the given ID (checking on both sides of the ID). Print all the details about the book. In case of ties, print both the books ordered by bookIDs.

8. ColorFlipCount(): GatorLibrary's Red-Black tree structure requires an analytics tool to monitor and analyze the frequency of color flips in the Red-Black tree. Track the occurrence of color changes in the Red-Black tree nodes during tree operations, such as insertion, deletion, and rotations.
Note: Only color flips should be counted i.e. when black changes to red and vice versa.*

Programming Environment
You may use either Java, C++ for this project. Your program will be tested using the Java or g++ compiler on the thunder.cise.ufl.edu server. So, you should verify that it compiles and runs as expected on this server, which may be accessed via the Internet. Your submission must include a makefile that creates an executable file named **gatorLibrary**.

Your program should execute using the following
For c/c++:
$ ./ gatorLibrary file_name
For java:
$ java gatorLibrary file_name
Where file_name is the name of the file that has the input test data.

Input and Output Requirements:
- **Read input from a text file where input_filename is specified as a command-line argument.**
- All Output should be written to a text file having filename as concatenation of **input_filename** + "_" + "**output_file.txt**".

(eg. inputFilename = '*test1.txt'* , outputFilename = '*test1_output_file.txt')*
- The program should terminate when the operation encountered in the input file is ***Quit()***.
- While Printing Reservation Heap, only print the PatronIDs as ordered in the Heap. *(Example 3)*

Input Format

```
InsertBook(bookID, bookName, authorName, availabilityStatus)
PrintBook(bookID)
PrintBooks(bookID1, bookID2)
BorrowBook(patronID, bookID, patronPriority)
ReturnBook(patronID, bookID)
Quit()
```

Example 1:

Input

```
InsertBook(1, "Book1", "Author1", "Yes")
PrintBook(1)
BorrowBook(101, 1, 1)
Insert(2, "Book2", "Author2", "Yes")
BorrowBook(102, 1, 2)
PrintBooks(1, 2)
ReturnBook(101, 1)
ReturnBook(102, 2)
Quit()
```

Output:

```
BookID = 1
Title = "Book1"
Author = "Author1"
Availability = "Yes"
BorrowedBy =
Reservations = []

Book 1 Borrowed by Patron 101

Book 2 Borrowed by Patron 102


BookID = 1
Title = "Book1"
Author = "Author1"
Availability = "No"
BorrowedBy = 101
Reservations = [102]

BookID = 2
Title = "Book2"
Author = "Author2"
Availability = "Yes"
BorrowedBy =
Reservations = []

Book 1 Returned by Patron 101
```

```
Book 1 Allotted to Patron 102

Program Terminated!!
```

Example 2:

Input

```
InsertBook(1, "Book1", "Author1", "Yes")
BorrowBook(101, 1, 1)
BorrowBook(102, 1, 2)
BorrowBook(102, 1, 1)
BorrowBook(106, 1, 4)
BorrowBook(505, 1, 5)
ReturnBook(101, 1)
Quit()
```

Output:

```
Book 1 Borrowed by Patron 101

Book 1 Reserved by Patron 102

Book 1 Already Reserved by Patron 102

Book 1 Reserved by Patron 106

Book 1 Reserved by Patron 505

Book 1 Returned by Patron 101

Book 1 Allotted to Patron 102

Program Terminated!!
```

Example 3:

Input

```
InsertBook(4, "Book4", "Author1", "Yes")
InsertBook(2, "Book2", "Author1", "Yes")
BorrowBook(2001, 2, 3)
InsertBook(5, "Book5", "Author3", "Yes")
BorrowBook(3002, 2, 1)
PrintBook(2)
BorrowBook(3002, 5, 1)
BorrowBook(1003, 2, 4)
PrintBook(4)
BorrowBook(2010, 4, 2)
PrintBooks(2, 5)
BorrowBook(2010, 2, 2)
BorrowBook(1004, 2, 4)
ReturnBook(2001, 2)
```

```
ReturnBook(2010, 4)
FindClosestBook(3)
InsertBook(3, "Book3", "Author4", "Yes")
FindClosestBook(3)
DeleteBook(2)
ColorFlipCount()
Quit()
PrintBook(4)
BorrowBook(2)
ReturnBook(1003, 2)
```

Output:

```
Book 2 Borrowed by Patron 2001

Book 2 Reserved by Patron 3002

BookID = 2
Title = "Book2"
Author = "Author1"
Availability = "No"
BorrowedBy = 2001
Reservations = [3002]

Book 5 Borrowed by Patron 3002

Book 2 Reserved by Patron 1003

BookID = 4
Title = "Book4"
Author = "Author1"
Availability = "Yes"
BorrowedBy = None
Reservations = []

Book 4 Borrowed by Patron 2010

BookID = 2
Title = "Book2"
Author = "Author1"
Availability = "No"
BorrowedBy = 2001
Reservations = [3002, 1003]

BookID = 4
Title = "Book4"
Author = "Author1"
Availability = "No"
BorrowedBy = 2010
Reservations = []

BookID = 5
Title = "Book5"
Author = "Author3"
Availability = "No"
BorrowedBy = 3002
```

```
Reservations = []

Book 2 Reserved by Patron 2010

Book 2 Reserved by Patron 1004

Book 2 Returned by Patron 2001

Book 2 Allotted to 3002

Book 4 Returned by Patron 2010

BookID = 2
Title = "Book2"
Author = "Author1"
Availability = "No"
BorrowedBy = 3002
Reservations = [2010, 1003, 1004]

BookID = 4
Title = "Book4"
Author = "Author1"
Availability = "Yes"
BorrowedBy = None
Reservations = []

BookID = 3
Title = "Book3"
Author = "Author4"
Availability = "Yes"
BorrowedBy = None
Reservations = []

Book 2 is no longer available. Reservations made by Patrons 2010,1003,1004 have
been cancelled!

Colour Flip Count: 5

Program Terminated!!
```

Submission Requirements:

- Include a makefile for easy compilation.
- Provide well-commented source code.
- Submit a PDF report that includes project details, function prototypes, and explanations.
- Follow the input/output and submission requirements as described in the reference project.

Do not use nested directories. All your files must be in the first directory that appears after unzipping.

You must submit the following:

1. Makefile: You must design your makefile such that ‘make’ command compiles the source code and produces an executable file. (For java class files that can be run with java command)
2. Source Program: Provide comments.
3. REPORT:
   - The report should be in PDF format.

- The report should contain your basic info: Name, UFID and UF Email account
- Present function prototypes showing the structure of your programs. Include the structure of your program.

To submit, please compress all your files together using a zip utility and submit to the Canvas system.
You should look for the Assignment Project for the submission.
Your submission should be named LastName_FirstName.zip.

Please make sure the name you provided is the same as the same that appears on the Canvas system. ***Please do not submit directly to a TA.*** *All email submissions will be ignored without further notification. Please note that the due date is a hard deadline. No late submission will be allowed. Any submission after the deadline will not be accepted.*

Grading Policy:
Grading will be based on the correctness and efficiency of algorithms. Below are some details of the grading policy.

- Correct implementation and execution: 70%
- Comments and readability: 15%
- Report: 15%

Important: Your program will be graded based on the produced output. You must make sure to produce the correct output to get points. There will be a threshold for the running time of your program. If your program runs slow, we will assume that you have not implemented the required data structures properly. You will get ***negative points*** *if you do not follow the* ***input/output or submission requirements*** *above.* ***Following is the clear guidance of how your marks will be deducted.***

- Source files are not in a single directory after unzipping: -5 points
- Incorrect output file name: -5 points
- Error in make file : -5 points
- Make file does not produce an executable file that can be run with one of the following commands: -5 points
  ./ gatorLibrary file_name
  java gatorLibrary file_name
- Hard coded input file name instead of taking as an argument from the command prompt: -5 points
- Not following the Output formatting specified in examples: -5 points
- Any other input/output or submission requirement mentioned in the document: -3 points

Also, we may ask you to fix the above problems and demonstrate your projects.

Miscellaneous:
Implement Red-Black tree and Binary min-heap from scratch, **without using built-in libraries**.
Your implementation should be your own. You have to work by yourself for this assignment (discussion is allowed). **Your submission will be checked for plagiarism!!**