SECJ2013
DATA STRUCTURE AND ALGORITHM

PROJECT

SECTION 02

LECTURER: Ms. Lizawati binti Mi Yusuf

GROUP NAME: LogiCode

NAME	MATRIC NO.
Ong Yi Yan	A22EC0101
Tang Yan Qing	A22EC0109
Koh Su Xuan	A22EC0060

Table of Contents

Objective

- To simulate an actual Hotel Booking System in a way that administrators can use it to manage the room bookings of customers in a hotel.
- To improve understanding and enhance application of data structure: Binary Search Tree in a real-world scenario.
- To utilize file mechanism in C++ to read data from external files as a simulation of the database of the system.

Synopsis

The Hotel Booking System is designed for hotel administrators to manage room bookings in a hotel. By using file input operation, the system reads the files containing admin information which are admin name, position, username and password and booking information which are check-in date, check-out date, room number, room type, identification card number and total price of bookings and stores in AdminNode class and BookingNode class respectively.

The administrators are then required to enter their username and password to achieve the purpose of Authentication of the Hotel Booking System. The verification is carried out through the searching function implemented by the binary search tree on the username and password inputted in the AdminNode class.

With the administrator menu provided, the administrator is able to insert, delete, search, view the booking information and number of bookings stored in the BookingNode class. The menu is operated and implemented with pointer-based Binary Search Tree data structure concept and technique.

Project Overview/ Problem Analysis

In this project, the case study carried out is the Hotel Booking System, which is described to be a system to manage room bookings in a hotel. The primary users targeted for the system are the administrators of the hotel while the secondary users targeted are the staff of the hotel. There are multiple main features that were implemented in the system of this project, including,

I. Authentication and Security System
   - The user is required to enter their username and password before accessing the Hotel Booking System. The entered information is then verified by the system with the implementation of Binary Search Tree to store admin information which are admin name, position, username and password retrieved from the file (database) and search for matching information to authenticate the administrator.

II. Booking Information Management System
   - The booking information is stored by using Binary Search Tree after retrieving from the file (database) in a similar way as Authentication and Security System. The booking details include check-in date, check-out date, room number, room type, identification card number of customer and total price are then used in collaboration to implement functions to insert, delete, search, view booking information and number of bookings in the system.

III. File Handling as the Simulation of Database of System
   - By using the external files which store the administrator and booking information, the database of the real world scenario is successfully simulated. The data in the files are read and stored into the system using file operations and Binary Search Tree in C++ respectively.

IV. User Interface
   - In order to enhance user experience, an interactive menu for administrators to perform various operations is created with structured arrangement and design. The administrators

menu options include clear instructions for each operation such as inserting a booking, deleting a booking, searching for a booking, viewing all bookings and viewing the number of bookings in the system.

By addressing the main features in the Hotel Booking System, a well-structured system that can manage room bookings in a hotel is achieved at the same time as developing an understanding of data structures of Binary Search Tree and file handling in C++.

Design

Class diagram

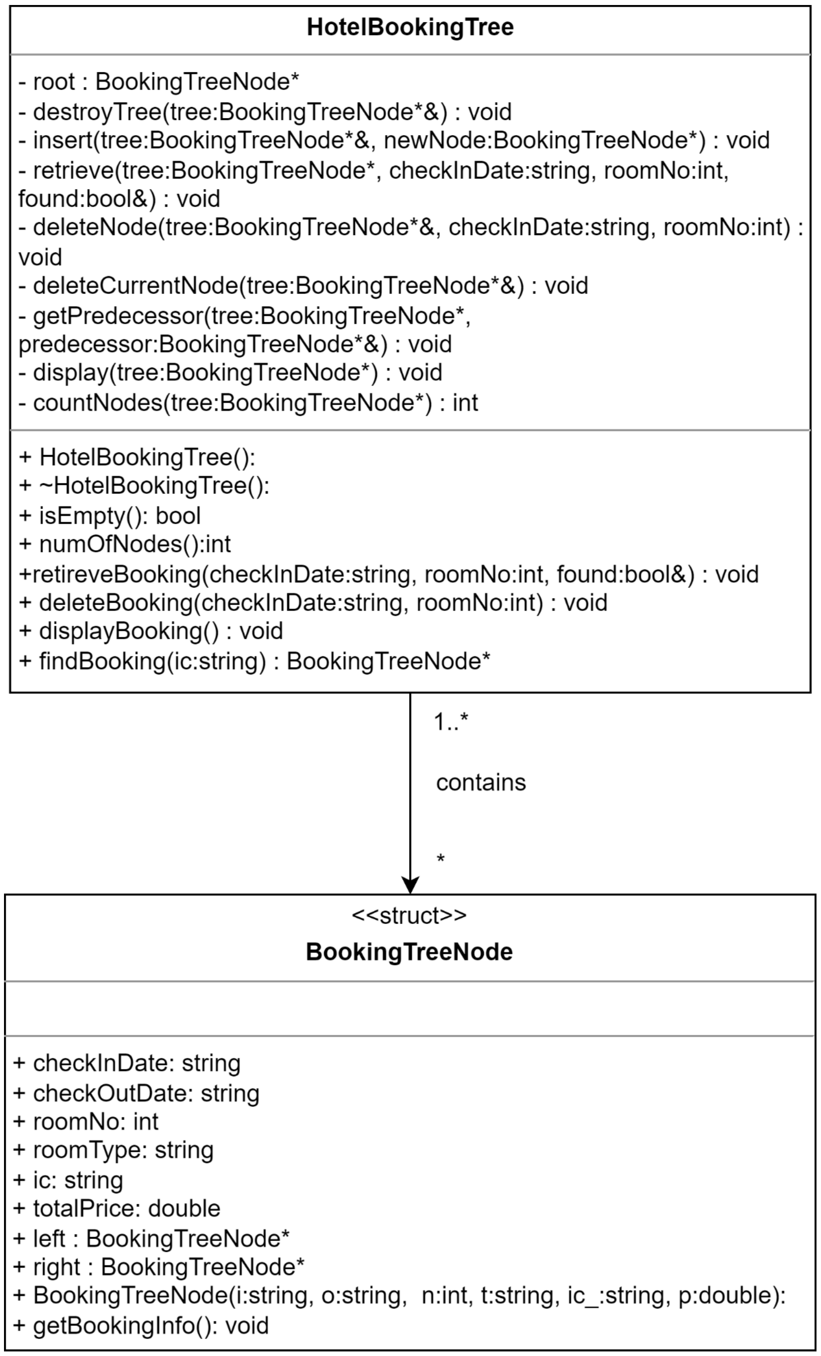

HotelBookingTree
- root : BookingTreeNode*
- destroyTree(tree:BookingTreeNode*&) : void
- insert(tree:BookingTreeNode*&, newNode:BookingTreeNode*) : void
- retrieve(tree:BookingTreeNode*, checkInDate:string, roomNo:int, found:bool&) : void
- deleteNode(tree:BookingTreeNode*&, checkInDate:string, roomNo:int) : void
- deleteCurrentNode(tree:BookingTreeNode*&) : void
- getPredecessor(tree:BookingTreeNode*, predecessor:BookingTreeNode*&) : void
- display(tree:BookingTreeNode*) : void
- countNodes(tree:BookingTreeNode*) : int
+ HotelBookingTree():
+ ~HotelBookingTree():
+ isEmpty(): bool
+ numOfNodes():int
+retireveBooking(checkInDate:string, roomNo:int, found:bool&) : void
+ deleteBooking(checkInDate:string, roomNo:int) : void
+ displayBooking() : void
+ findBooking(ic:string) : BookingTreeNode*
1..*
contains
*
<<struct>>
BookingTreeNode
+ checkInDate: string
+ checkOutDate: string
+ roomNo: int
+ roomType: string
+ ic: string
+ totalPrice: double
+ left : BookingTreeNode*
+ right : BookingTreeNode*
+ BookingTreeNode(i:string, o:string, n:int, t:string, ic_:string, p:double):
+ getBookingInfo(): void

Figure : Class Diagram of Hotel Booking System

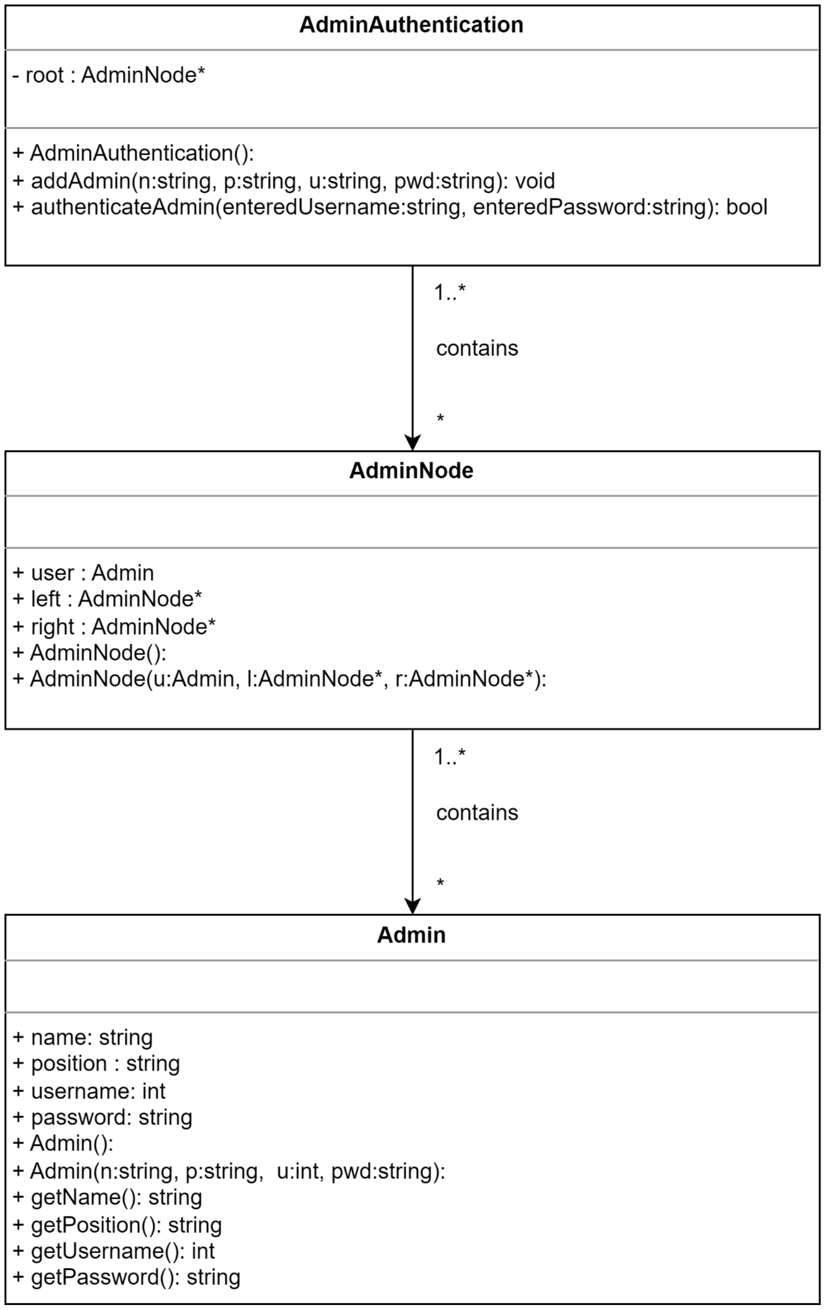


Figure : Class Diagram of Hotel Booking System

Flowchart

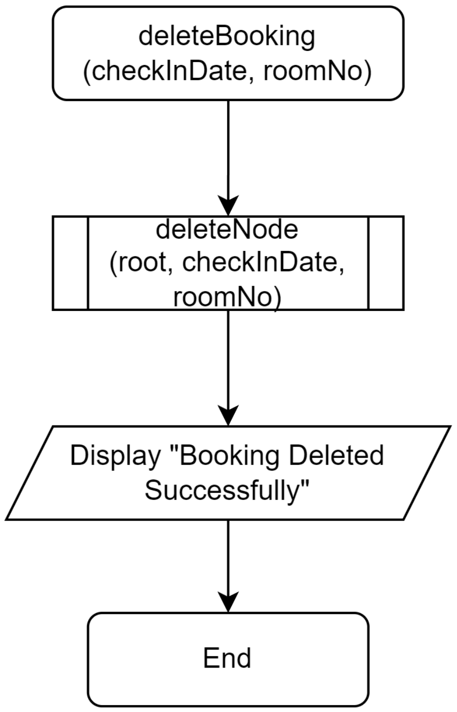

deleteBooking (checkInDate, roomNo)
deleteNode (root, checkInDate, roomNo)
Display "Booking Deleted Successfully"
End

Figure 1: Flowchart of deleteBooking function

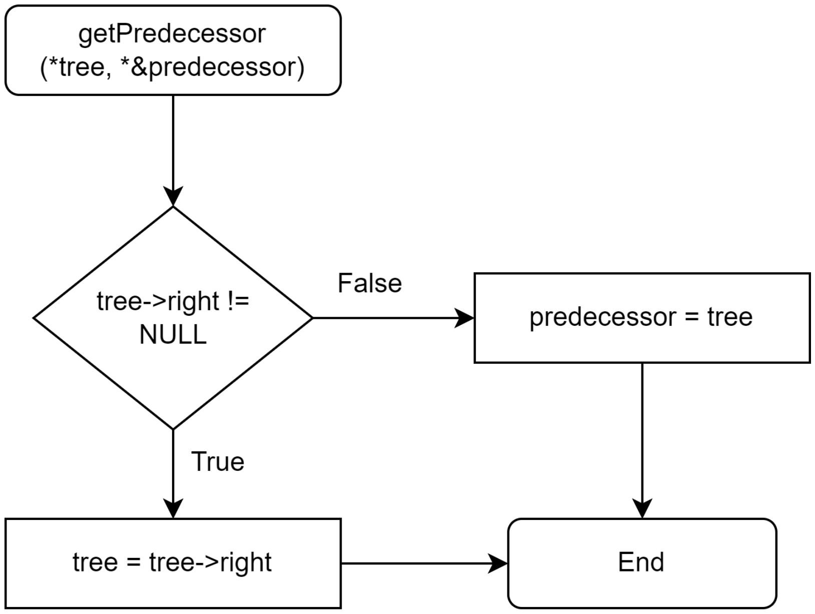


Figure 2: Flowchart of getPredecessor function

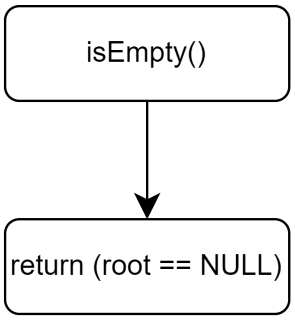


Figure 3: Flowchart of isEmpty function

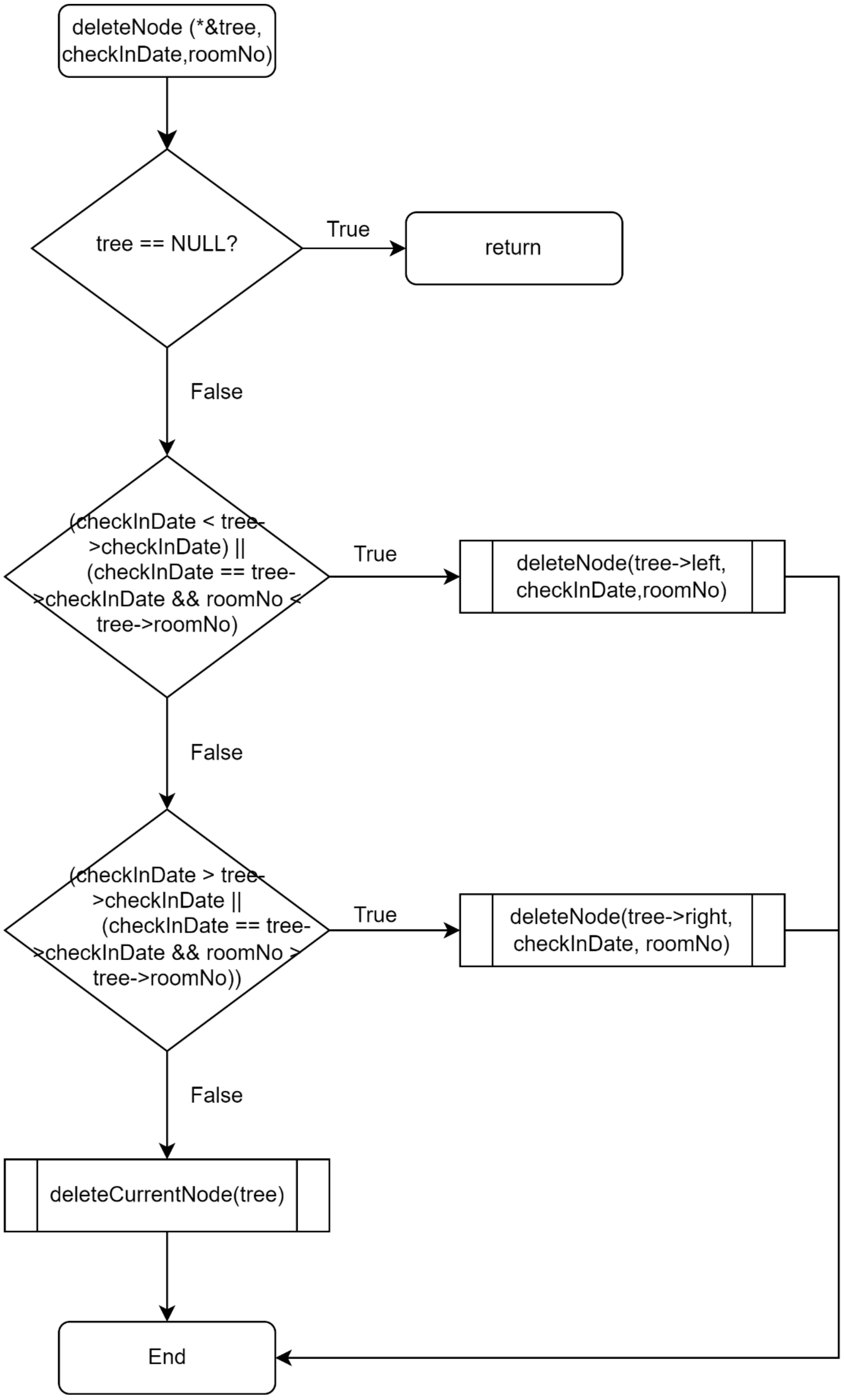

deleteNode (*&tree, checkInDate,roomNo)
tree == NULL?
True
return
False
(checkInDate < tree->checkInDate) || (checkInDate == tree->checkInDate && roomNo < tree->roomNo)
True
deleteNode(tree->left, checkInDate,roomNo)
False
(checkInDate > tree->checkInDate || (checkInDate == tree->checkInDate && roomNo > tree->roomNo))
True
deleteNode(tree->right, checkInDate, roomNo)
False
deleteCurrentNode(tree)
End

Figure 4: Flowchart of deleteNode function

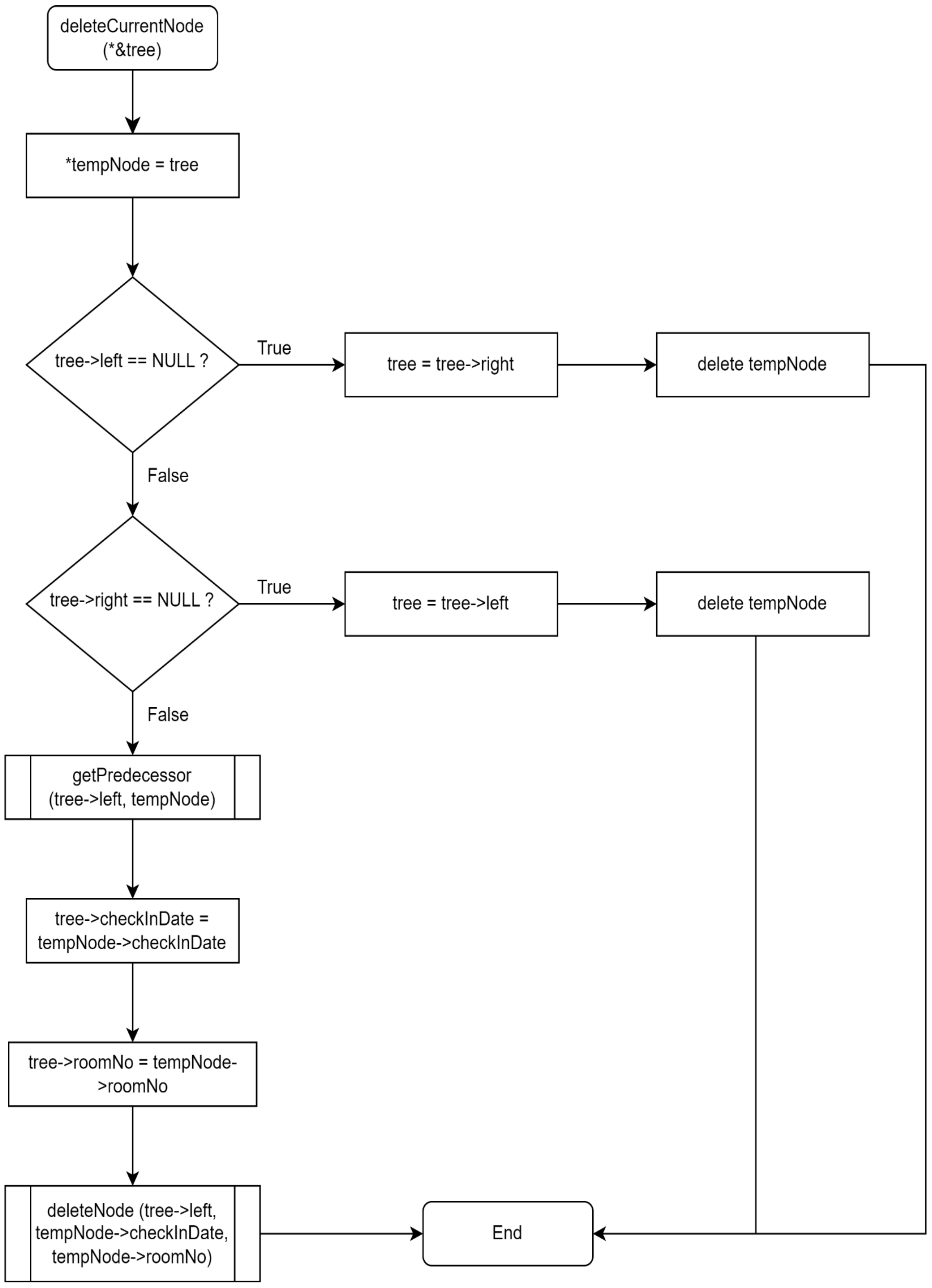

deleteCurrentNode (*&tree)
*tempNode = tree
tree->left == NULL ?
True
tree = tree->right
delete tempNode
False
tree->right == NULL ?
True
tree = tree->left
delete tempNode
False
getPredecessor (tree->left, tempNode)
tree->checkInDate = tempNode->checkInDate
tree->roomNo = tempNode->roomNo
deleteNode (tree->left, tempNode->checkInDate, tempNode->roomNo)
End

Figure 5: Flowchart of deleteCurrentNode function

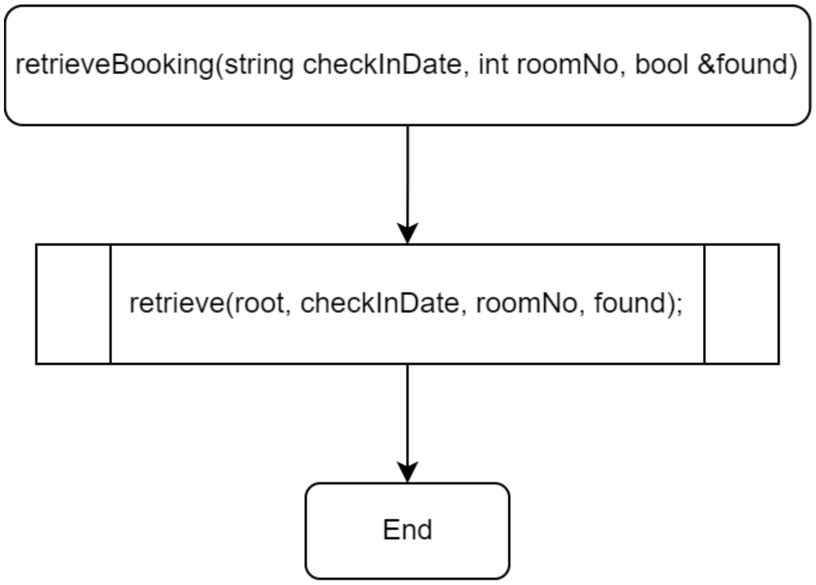

retrieveBooking(string checkInDate, int roomNo, bool &found)
retrieve(root, checkInDate, roomNo, found);
End

Figure 6: Flowchart of retrieveBooking Function

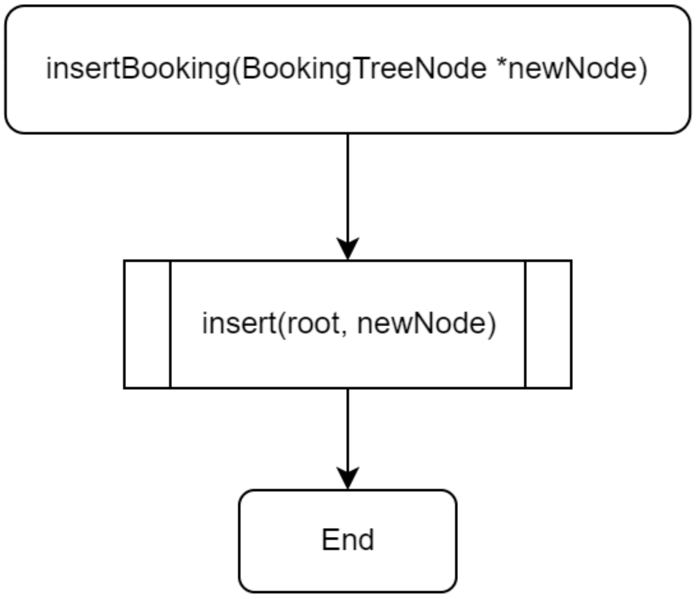


Figure 7: Flowchart of insertBooking Function

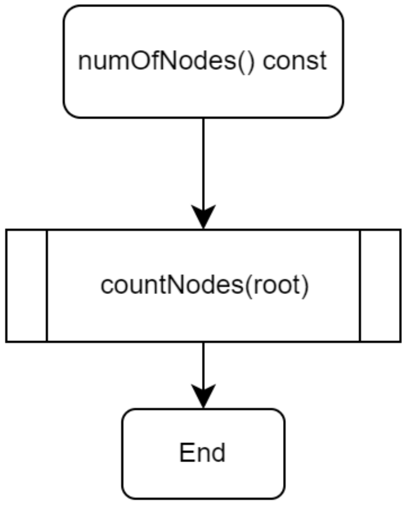


Figure 8: Flowchart of numOfNodes Function

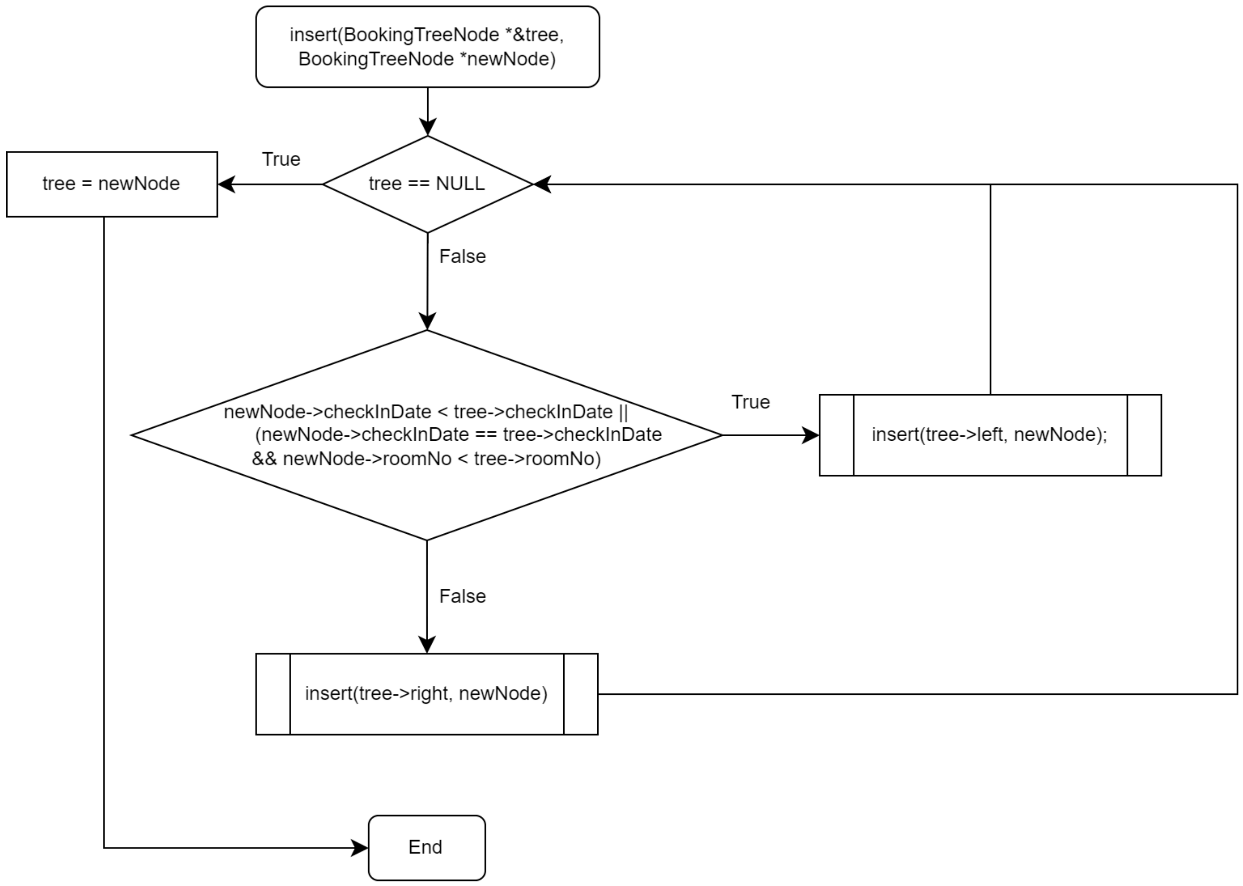

insert(BookingTreeNode *&tree, BookingTreeNode *newNode)
tree == NULL
True
tree = newNode
False
newNode->checkInDate < tree->checkInDate || (newNode->checkInDate == tree->checkInDate && newNode->roomNo < tree->roomNo)
True
insert(tree->left, newNode);
False
insert(tree->right, newNode)
End

Figure 9: Flowchart of insert Function

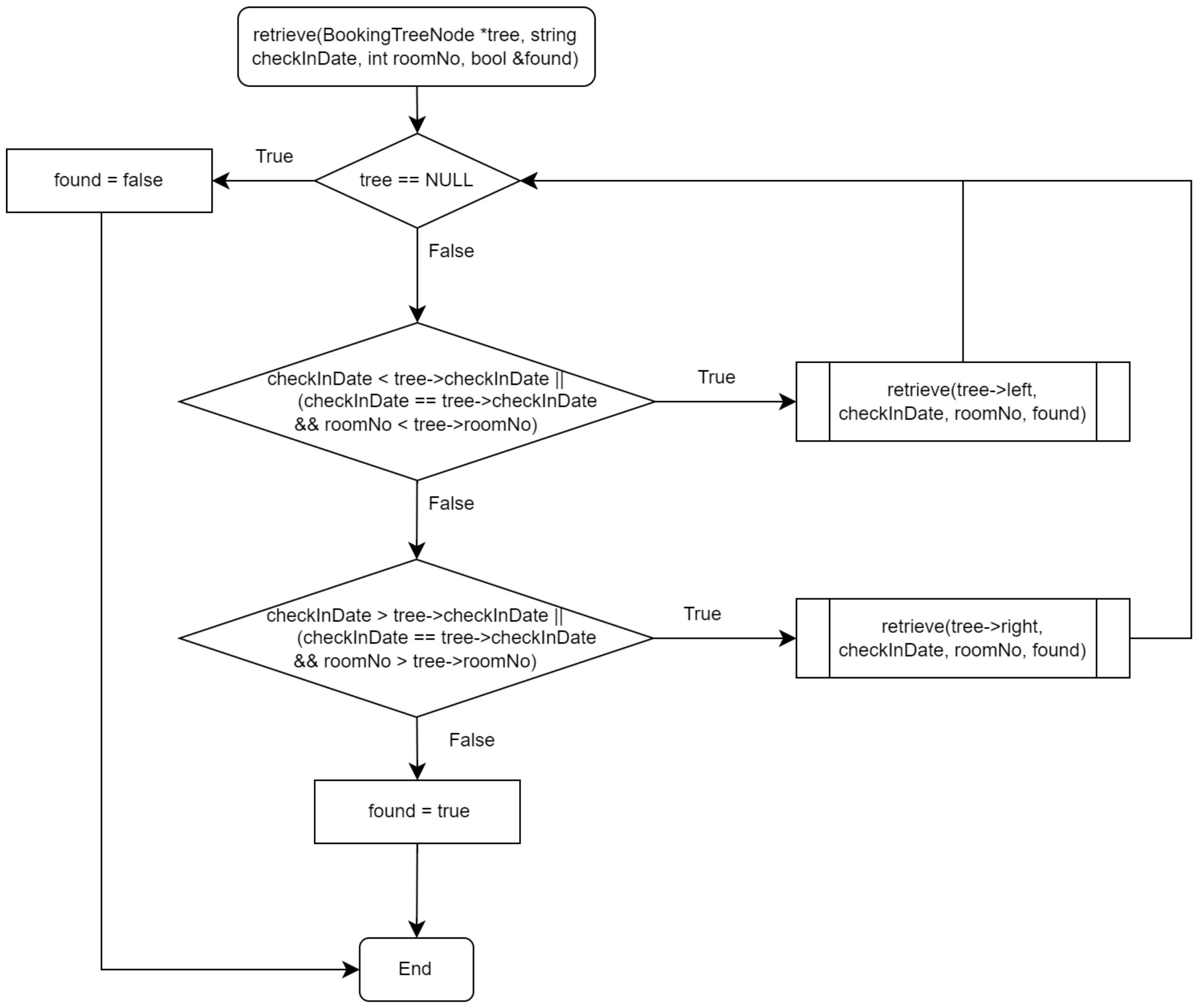

retrieve(BookingTreeNode *tree, string checkInDate, int roomNo, bool &found)
tree == NULL
True
found = false
False
checkInDate < tree->checkInDate || (checkInDate == tree->checkInDate && roomNo < tree->roomNo)
True
retrieve(tree->left, checkInDate, roomNo, found)
False
checkInDate > tree->checkInDate || (checkInDate == tree->checkInDate && roomNo > tree->roomNo)
True
retrieve(tree->right, checkInDate, roomNo, found)
False
found = true
End

Figure 10: Flowchart of retrieve Function

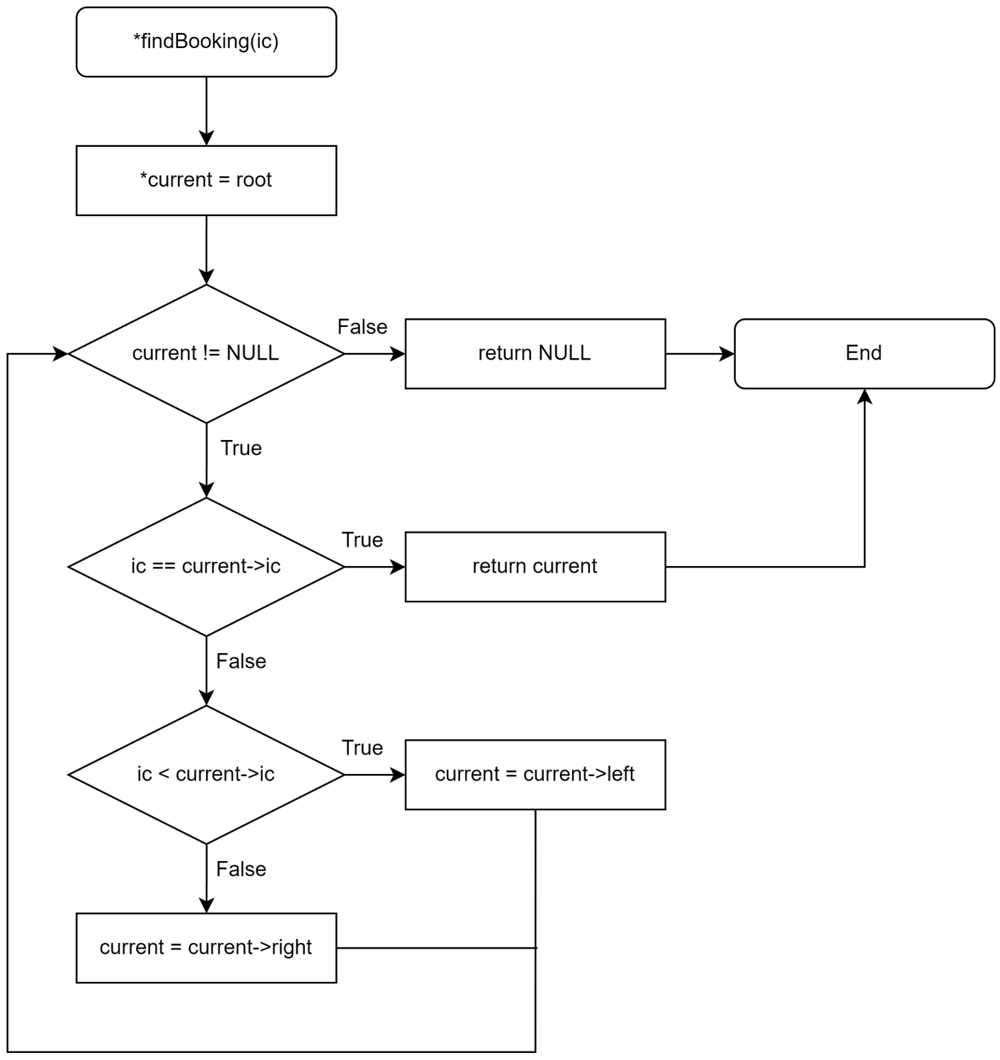


Figure 11: Flowchart of findBooking function

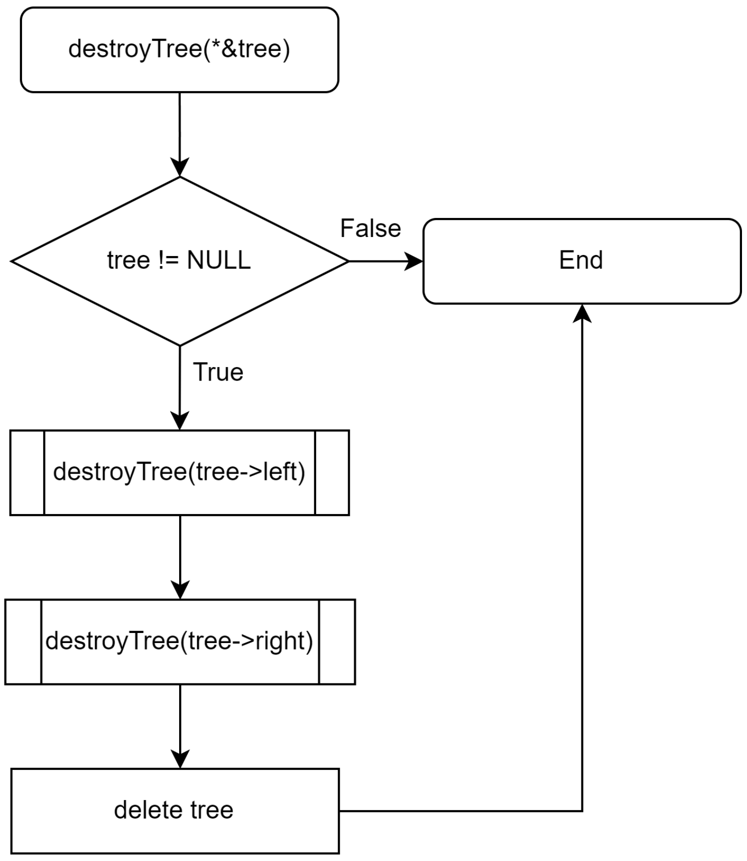


Figure 12: Flowchart of destroyTree function

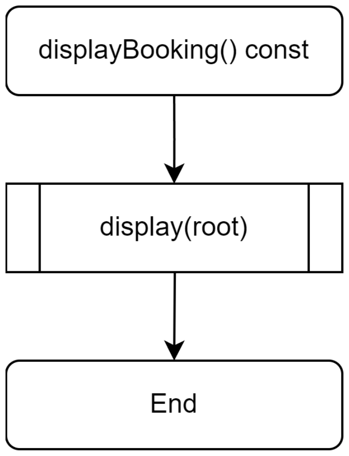


Figure 13: Flowchart of displayBooking function

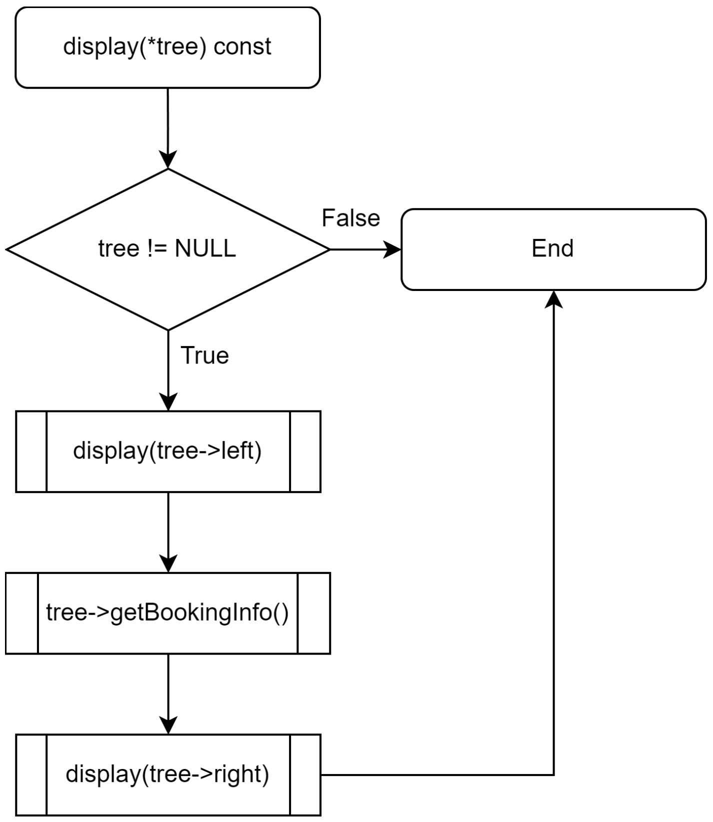


Figure 14: Flowchart of display function

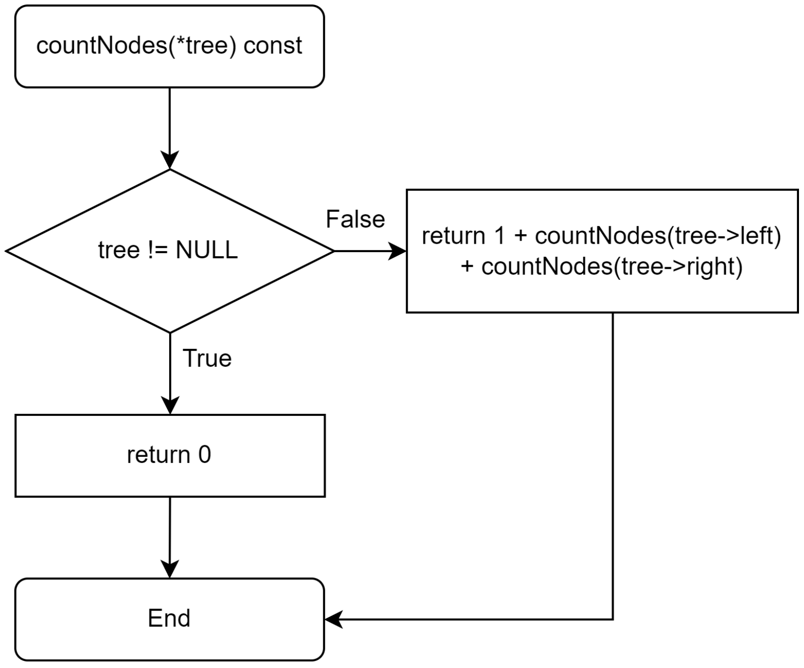


Figure 15: Flowchart of countNodes function

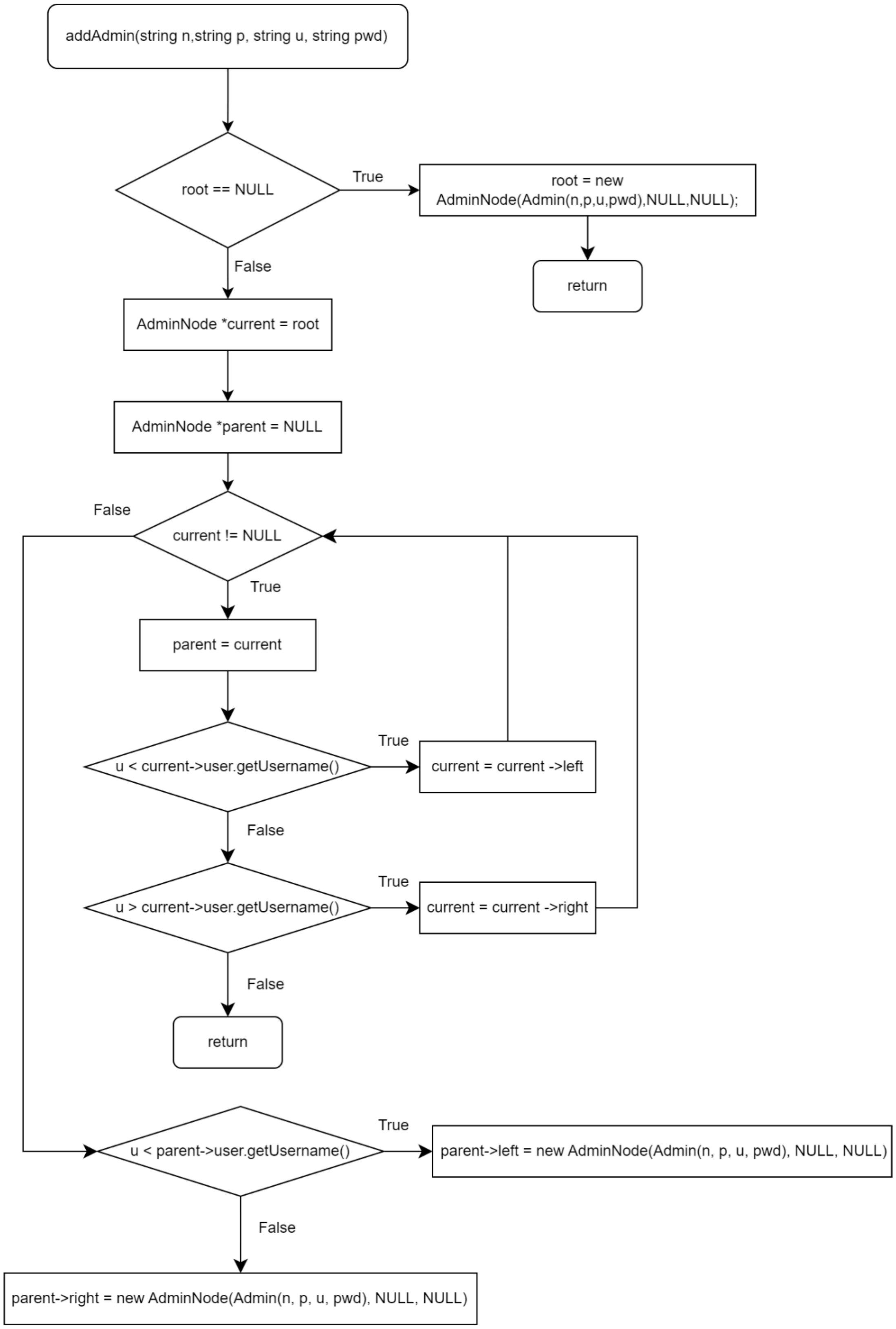

addAdmin(string n,string p, string u, string pwd)
root == NULL
True
root = new AdminNode(Admin(n,p,u,pwd),NULL,NULL);
return
False
AdminNode *current = root
AdminNode *parent = NULL
False
current != NULL
True
parent = current
u < current->user.getUsername()
True
current = current ->left
False
u > current->user.getUsername()
True
current = current ->right
False
return
u < parent->user.getUsername()
True
parent->left = new AdminNode(Admin(n, p, u, pwd), NULL, NULL)
False
parent->right = new AdminNode(Admin(n, p, u, pwd), NULL, NULL)

Figure 16: Flowchart of addAdmin Function

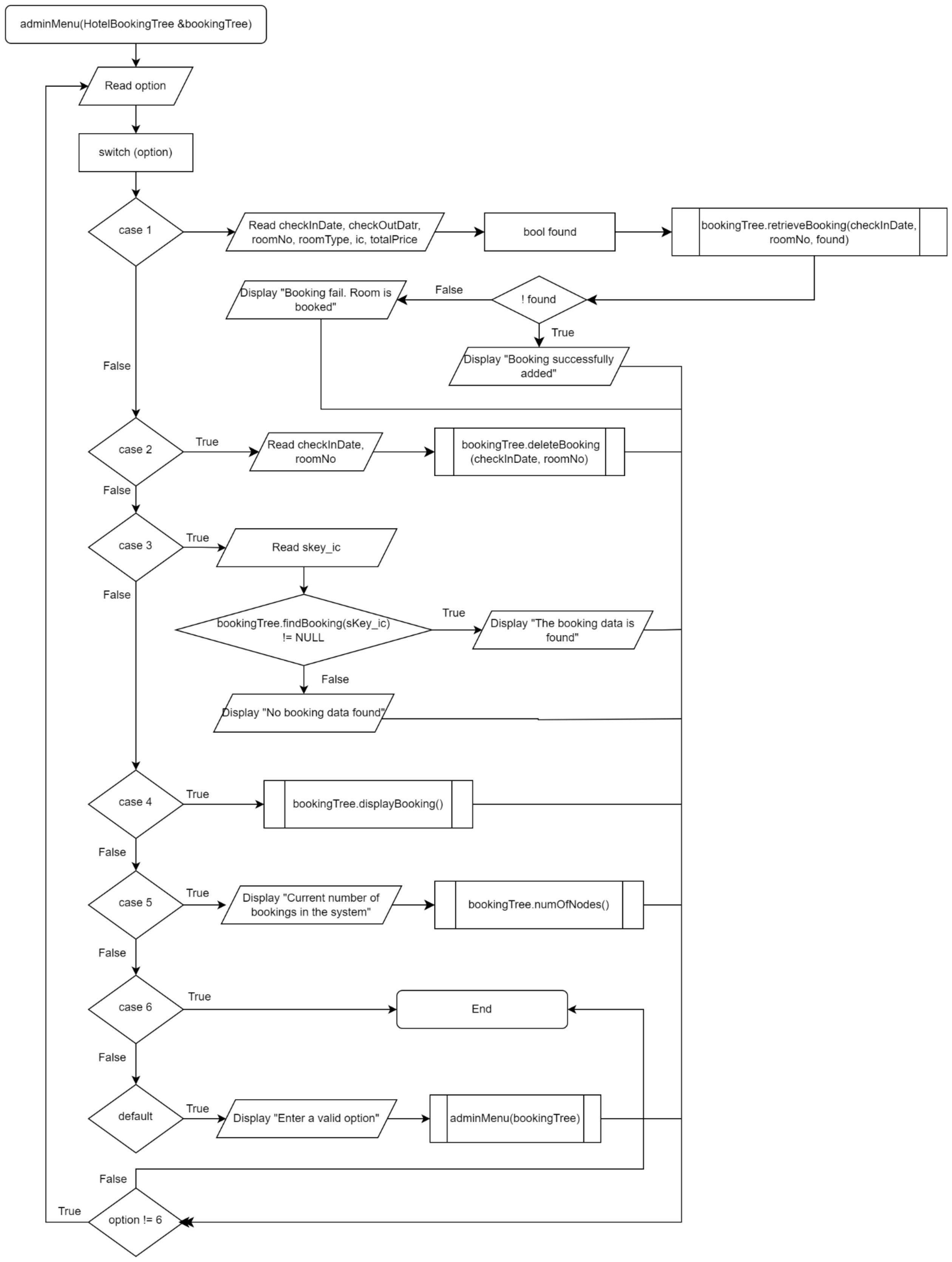

adminMenu(HotelBookingTree &bookingTree)
Read option
switch (option)
case 1
Read checkInDate, checkOutDatr, roomNo, roomType, ic, totalPrice
bool found
bookingTree.retrieveBooking(checkInDate, roomNo, found)
! found
False
Display "Booking fail. Room is booked"
True
Display "Booking successfully added"
False
case 2
True
Read checkInDate, roomNo
bookingTree.deleteBooking (checkInDate, roomNo)
False
case 3
True
Read skey_ic
bookingTree.findBooking(sKey_ic) != NULL
True
Display "The booking data is found"
False
Display "No booking data found"
False
case 4
True
bookingTree.displayBooking()
False
case 5
True
Display "Current number of bookings in the system"
bookingTree.numOfNodes()
False
case 6
True
End
False
default
True
Display "Enter a valid option"
adminMenu(bookingTree)
False
True
option != 6

Figure 17: Flowchart of adminMenu Function

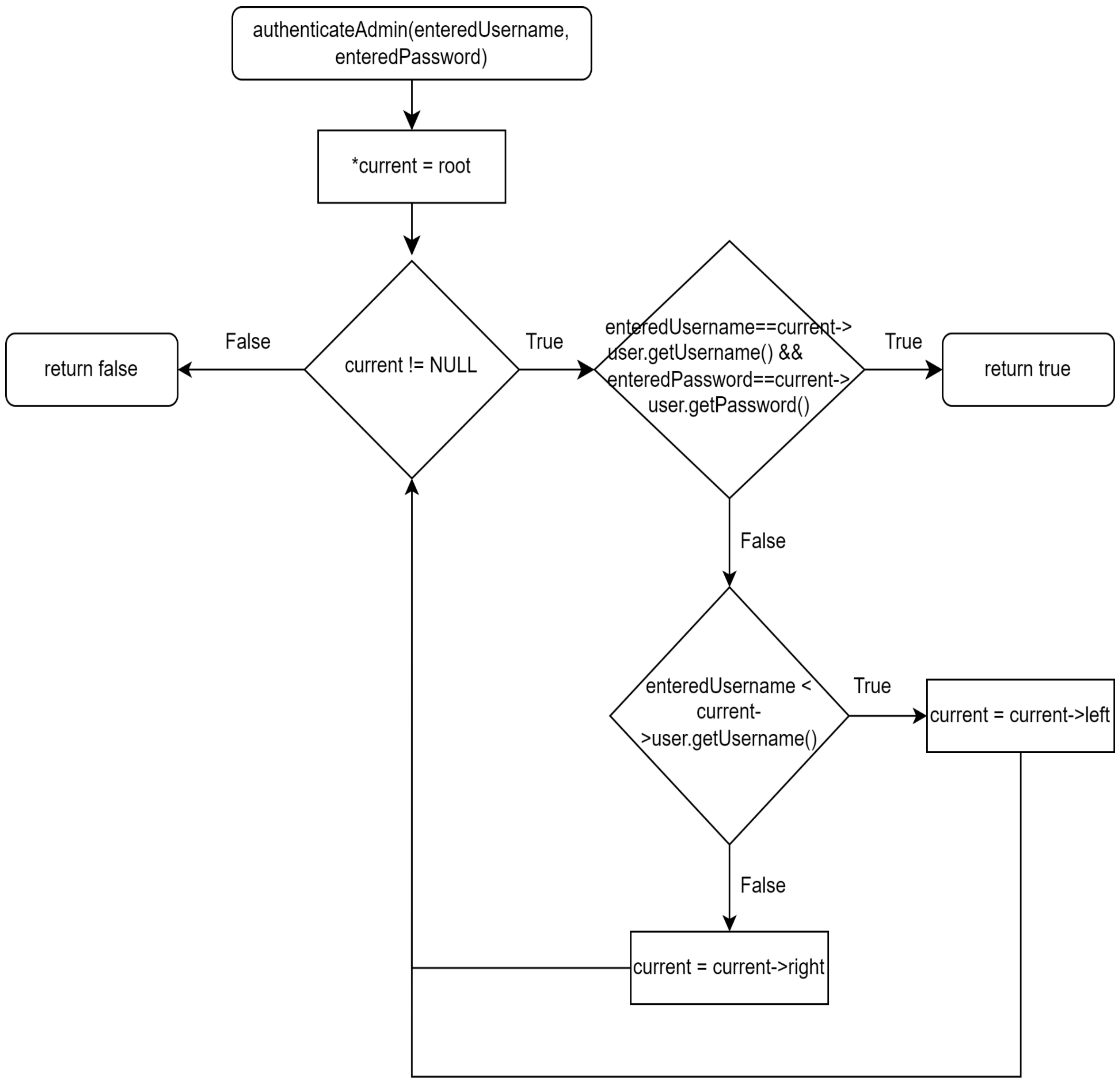

authenticateAdmin(enteredUsername, enteredPassword)
*current = root
current != NULL
False
return false
True
enteredUsername==current->user.getUsername() && enteredPassword==current->user.getPassword()
True
return true
False
enteredUsername < current->user.getUsername()
True
current = current->left
False
current = current->right

Figure 18: Flowchart of authenticateAdmin function

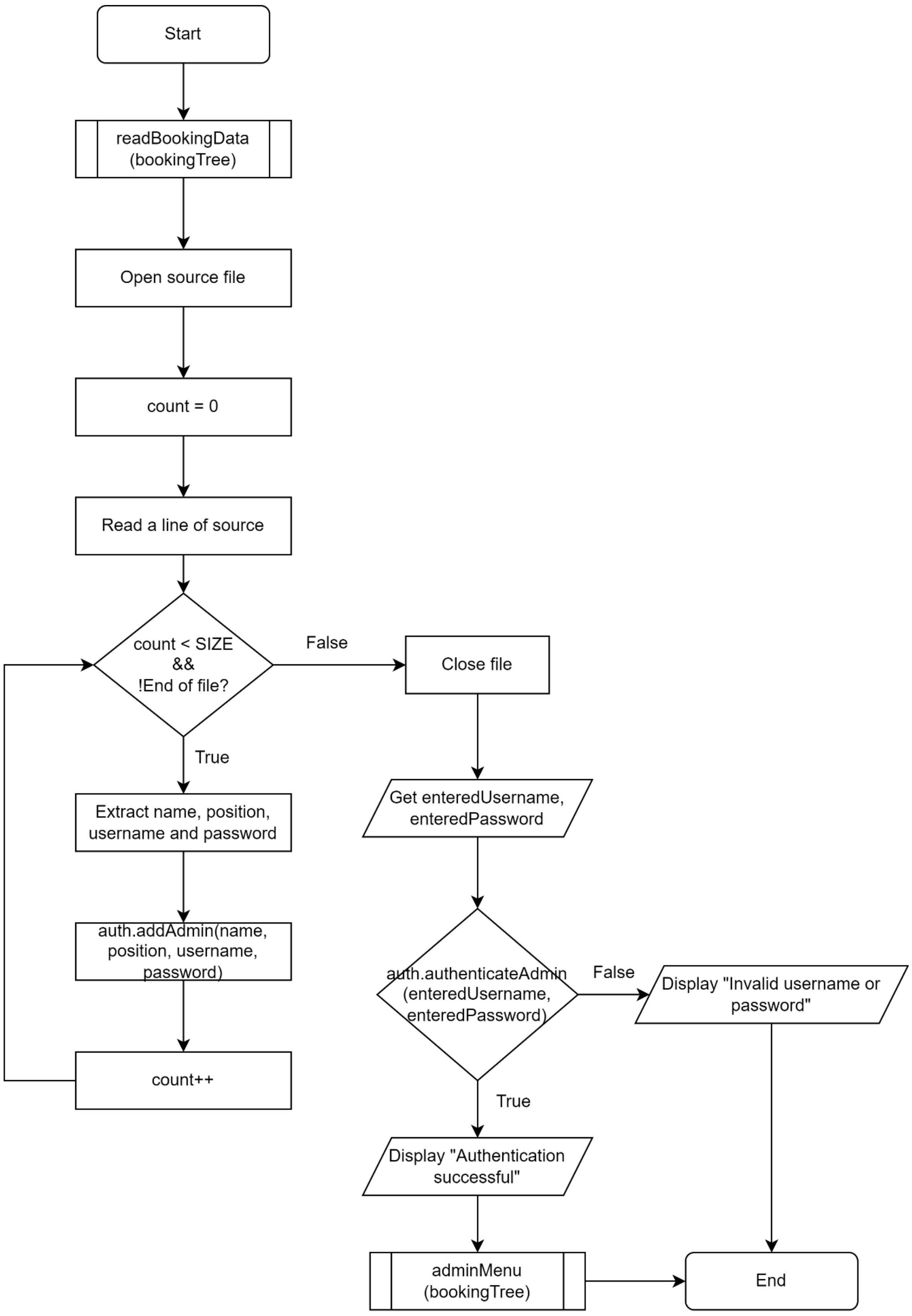

Start
readBookingData (bookingTree)
Open source file
count = 0
Read a line of source
count < SIZE && !End of file?
False
True
Close file
Extract name, position, username and password
Get enteredUsername, enteredPassword
auth.addAdmin(name, position, username, password)
count++
auth.authenticateAdmin (enteredUsername, enteredPassword)
False
Display "Invalid username or password"
True
Display "Authentication successful"
adminMenu (bookingTree)
End

Figure 19: Flowchart of main function

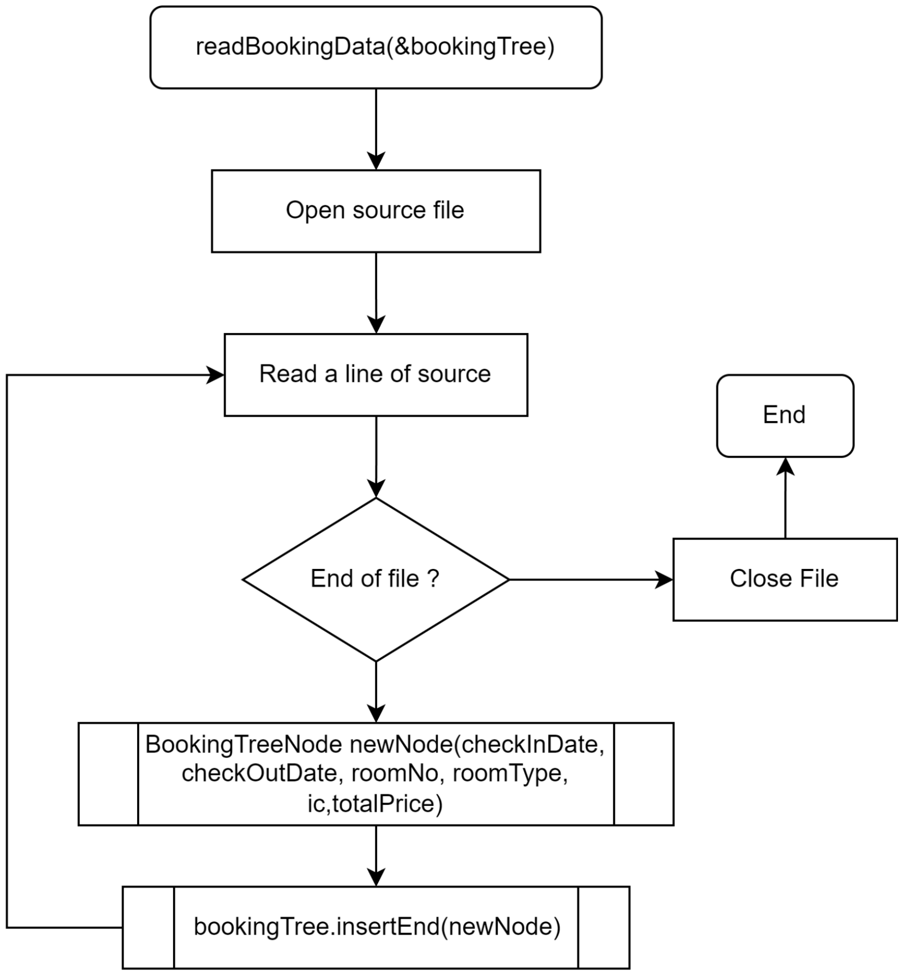

readBookingData(&bookingTree)
Open source file
Read a line of source
End of file ?
Close File
End
BookingTreeNode newNode(checkInDate, checkOutDate, roomNo, roomType, ic,totalPrice)
bookingTree.insertEnd(newNode)

Figure 20: Flowchart of readBookingData Function

Description of how to implement data structure operation:

Binary Search Tree

In the Hotel Booking System, we have implemented Binary Search Tree - Inserting, Deleting, Searching, Displaying and Counting in the form of pointer-based to define functions in managing the room bookings in a hotel. This data structure operation works by having two main declarations which are Node declaration and Tree declaration.

1. Inserting

For inserting a new node into the tree, insert() function and insertBooking() functions are used. In our case, the tree is called bookingTree and the node is called bookingTreeNode. inserBooking() function is called in the main function to pass newNode (a bookingTreeNode object) and root (starting node of bookingTree) to the insert() function. insert() function first checks whether the bookingTree is empty. If empty, then the newNode will become the root. If not, it will compare the check-in date and room number of the currentNode and newNode. newNode will be inserted to the left child nodes of currentNode if its check-in date is earlier than the root. If both currentNode and newNode contain equal check-in date, newNode will be inserted to the left child nodes of currentNode if its room number is smaller. The process continues until newNode is placed in the correct position.

2. Deleting

In the Binary Search Tree - Deleting, we have defined two delete functions which are deleteNode() and deleteCurrentNode(). The delete operations are implemented by deciding and locating the desirable data to be deleted. The deleteNode() function will compare the check-in date and room number data entered by the user. If the check-in date or room number entered is smaller than the check-in date or room number in the node pointed by the tree pointer, the tree pointer will move to the left subtree. On the other hand, if the check-in date or room number entered is larger than the check-in date or room number in the node pointed by the tree pointer, the tree pointer will move to the right subtree. The moving process will continue until the check-in date and room number entered are equal to the data in the node pointed by the tree pointer. Then, the deleteCurrentNode() function will be called. In the deleting process, if the unwanted booking data has no children at the left side, the unwanted node will be replaced by the booking data at the right side of the unwanted node. If the unwanted booking data has no children at the right side, the unwanted node will be replaced by the booking data at the left side of the unwanted node. However, if the unwanted booking data has two children at both side, the getPredecessor() function will be called to find the booking data located at the right most of the left side of the tree, and the booking data in that node will replace the node with the unwanted booking data. Lastly, the memory used by the unwanted booking data will be released.

3. Searching

retrieve() function and retrieveBooking() function are utilized for searching a booking. By taking the check-in date, room number and a boolean variable as parameters, retrieve() function will check whether the input check-in date and room number are less than current node's check-in date and room number, if so, then recursively call the function on the left subtree of the starting node. If not, then search it on the right subtree. If the booking is found, the boolean variable will be set as true. retrieveBooking() function is called in the main function to pass the user input to the retrieve() function.

Besides, the authenticateAdmin() function is used to find the username and password in the tree. The current pointer that is initially point to the root will move to the node at left subtree if the entered username is smaller than the username pointed by the pointer and move to the node at right subtree if the entered username is larger than the username pointed by the pointer until the username and password that same with that one that the user entered is found. If not, the function will return false.

4. Displaying

In the system, the displayBooking() function and display() function collaborate with getBookingInfo() function to display the list of room bookings in the hotel which contains check-in date, check-out date, room number, room type, customer IC and total price.

The displayBooking() function is first called by the option chosen by the administrator to view the booking data in the system. Once called, the function will then call the display() function. The latter then checks whether the argument passed is pointing to NULL. Recursive functions are called if the argument is pointing to a node in which it will call display() function and pass the pointer pointed by the left pointer of the original argument, collaborate with getBookingInfo() to display the current argument room booking data and ultimately call again the display() function and pass the pointer pointed by the right pointer of the original argument. This process continues until the tree points to NULL which means that the tree is a leaf. As a result, the room booking data is printed in inorder traversal.

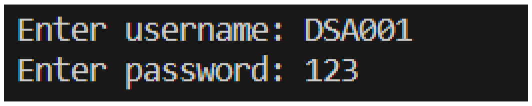
5. Counting

The Counting operation is used to count the number of nodes in the tree in which in our system it is implemented to count the number of bookings having in the system. To achieve the counting purpose, the numberOfNodes() function and countNodes() function operate together when the numberOfNodes() is called due to the administrator choosing to view the number of bookings in the system. It then calls countNodes() function and in the function, the argument passed examines and if it points to NULL, value 0 is returned while other conditions will return the value of 1 + countNodes(tree->left) + countNodes(tree->right). In a simpler manner, the recursive approach makes each node which is each booking data contribute 1 to the count and then the counts of the left and right subtrees are added. Consequently, the number of bookings is counted and returned.

User Guide

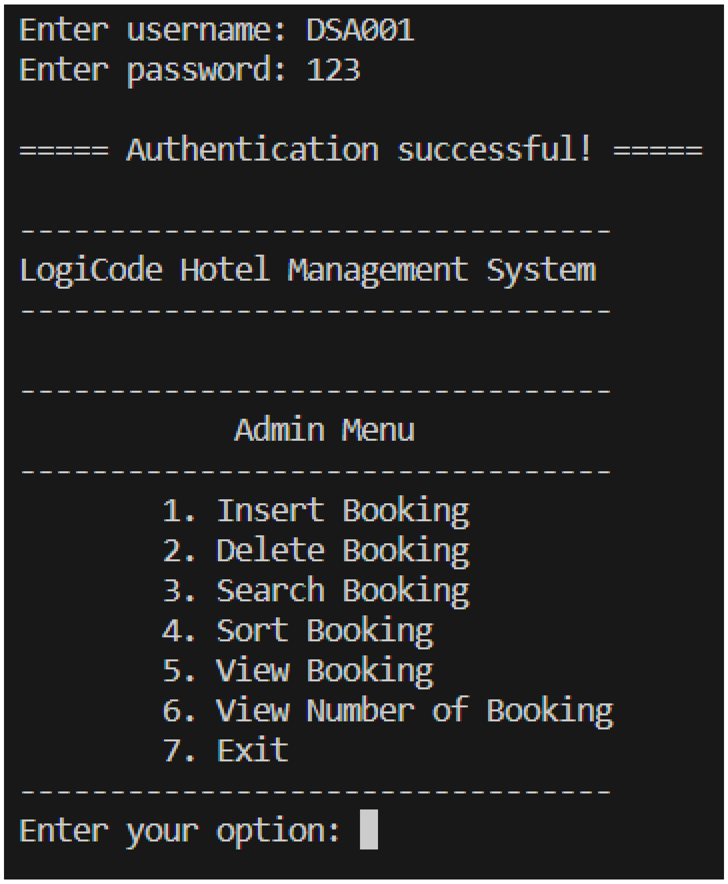
User Authentication

1. The user enters the username and password.



```
Enter username: DSA001
Enter password: 123
```

2. If the username and password are valid, it will show “Authentication successful!” and the admin menu.



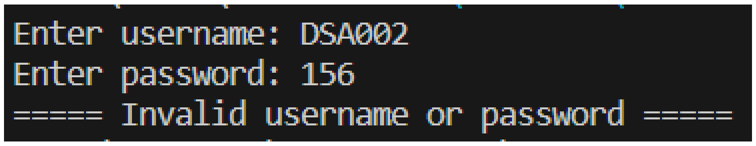
```
Enter username: DSA001
Enter password: 123

===== Authentication successful! =====

---------------------------------
LogiCode Hotel Management System
---------------------------------


---------------------------------
            Admin Menu
---------------------------------
        1. Insert Booking
        2. Delete Booking
        3. Search Booking
        4. Sort Booking
        5. View Booking
        6. View Number of Booking
        7. Exit
---------------------------------
Enter your option: 
```

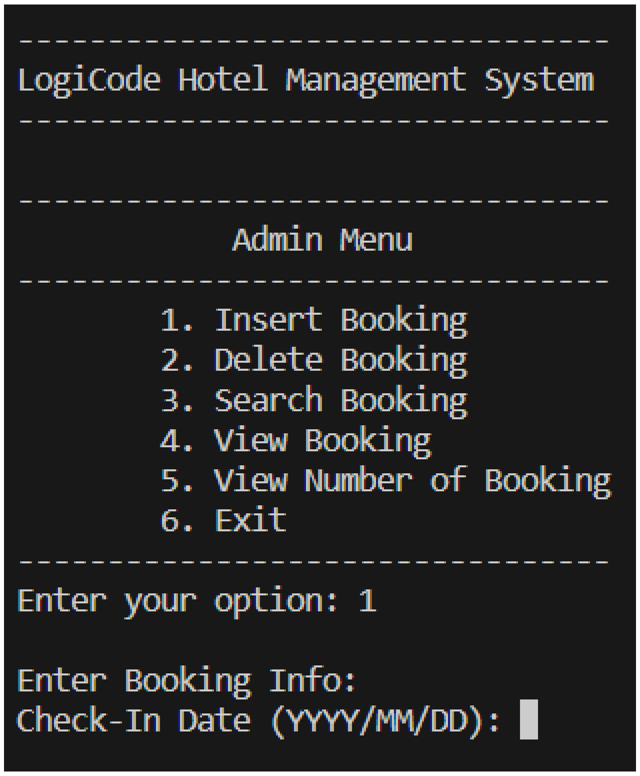
3. If the username and password are invalid, it will show “Invalid username or password”.



```
Enter username: DSA002
Enter password: 156
===== Invalid username or password =====
```

Insert Booking

1. The user enters ‘1’.

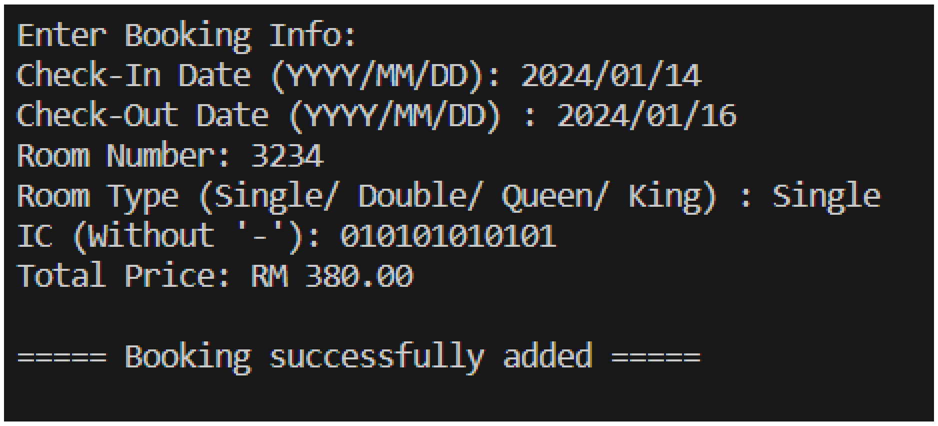


```
---------------------------------
LogiCode Hotel Management System
---------------------------------


---------------------------------
            Admin Menu
---------------------------------
        1. Insert Booking
        2. Delete Booking
        3. Search Booking
        4. View Booking
        5. View Number of Booking
        6. Exit
---------------------------------
Enter your option: 1

Enter Booking Info:
Check-In Date (YYYY/MM/DD): 
```

2. The user enters required information such as check-in date, check-out date,room number, room type, ic and total price.



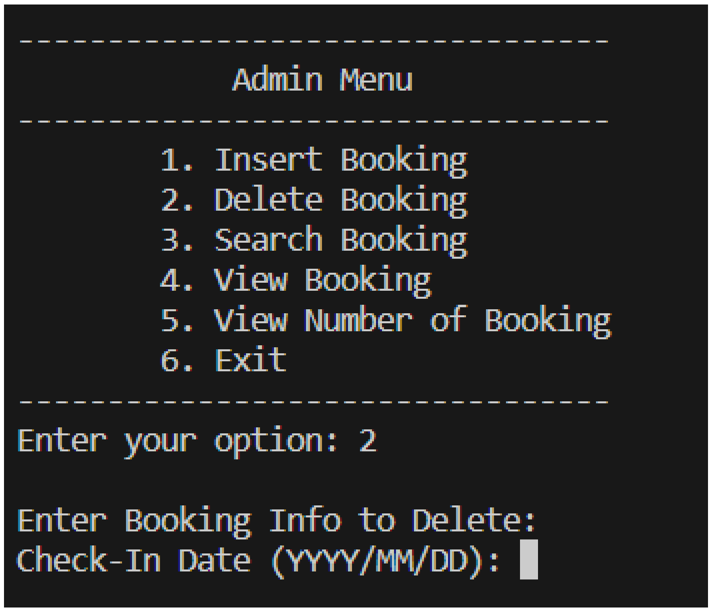
```
Enter Booking Info:
Check-In Date (YYYY/MM/DD): 2024/01/14
Check-Out Date (YYYY/MM/DD) : 2024/01/16
Room Number: 3234
Room Type (Single/ Double/ Queen/ King) : Single
IC (Without '-'): 010101010101
Total Price: RM 380.00

===== Booking successfully added =====
```

3. The booking is successfully added.

Delete booking

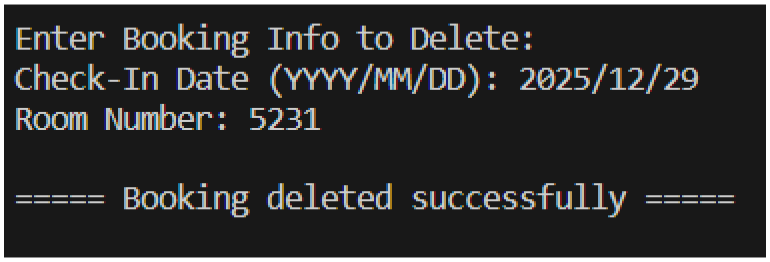
1. The user enters ‘2’.



```
---------------------------------
            Admin Menu
---------------------------------
        1. Insert Booking
        2. Delete Booking
        3. Search Booking
        4. View Booking
        5. View Number of Booking
        6. Exit
---------------------------------
Enter your option: 2

Enter Booking Info to Delete:
Check-In Date (YYYY/MM/DD): 
```

2. The user enters required information such as check-in date and room number.



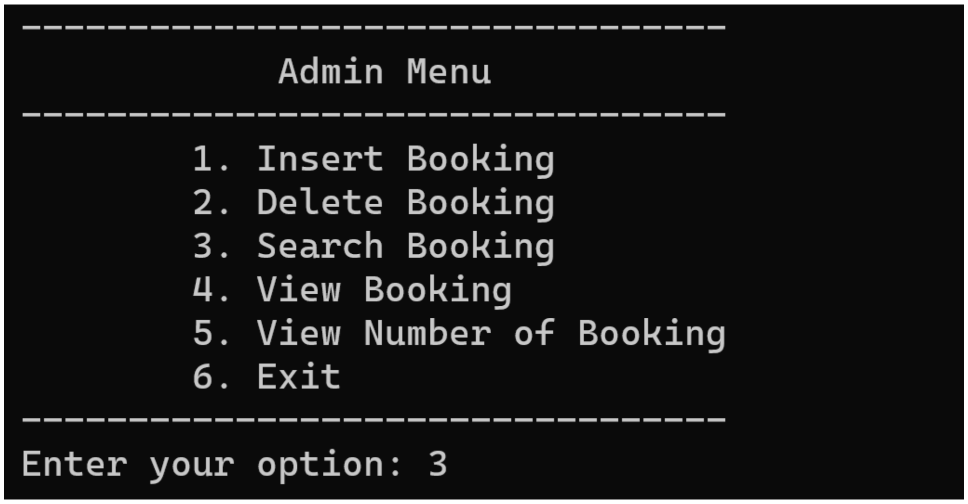
```
Enter Booking Info to Delete:
Check-In Date (YYYY/MM/DD): 2025/12/29
Room Number: 5231

===== Booking deleted successfully =====
```

3. The booking is deleted successfully.

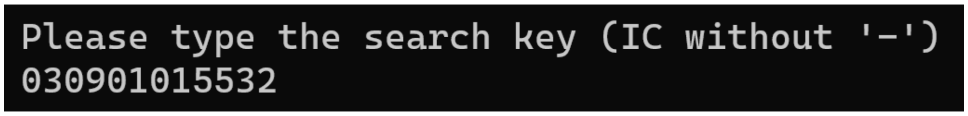
Search Booking

1. The user enters ‘3’.



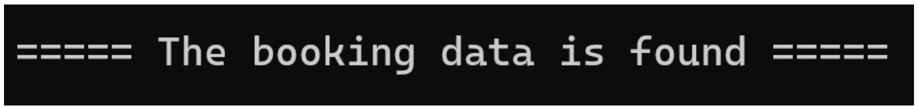
```
---------------------------------
            Admin Menu
---------------------------------
        1. Insert Booking
        2. Delete Booking
        3. Search Booking
        4. View Booking
        5. View Number of Booking
        6. Exit
---------------------------------
Enter your option: 3
```

2. The user enters the customer IC of the booking they want to search.



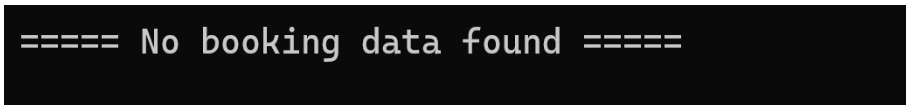
```
Please type the search key (IC without '-')
030901015532
```

3. If the customer has booked a room, then the system displays “The booking data is found”.



```
===== The booking data is found =====
```

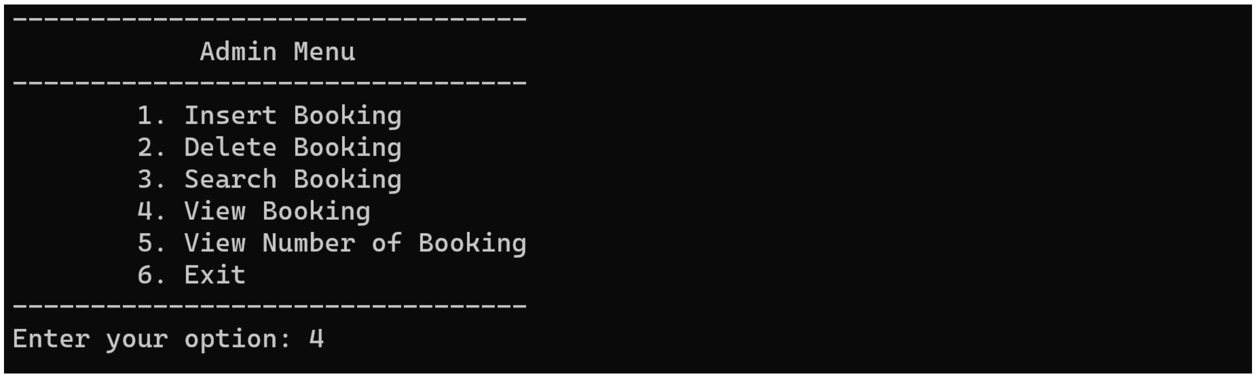
4. If the customer did not book a room, then the system displays “No booking data found”.



```
===== No booking data found =====
```

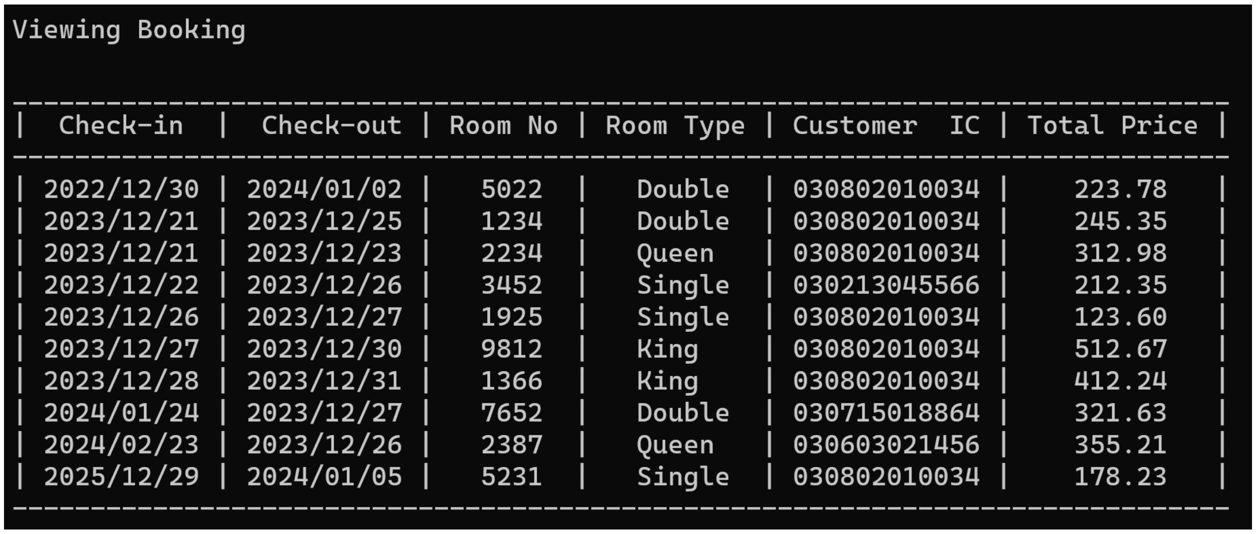
View Booking

1. The user enters ‘4’.



```
---------------------------------
            Admin Menu
---------------------------------
        1. Insert Booking
        2. Delete Booking
        3. Search Booking
        4. View Booking
        5. View Number of Booking
        6. Exit
---------------------------------
Enter your option: 4
```

2. Then the system displays all bookings.

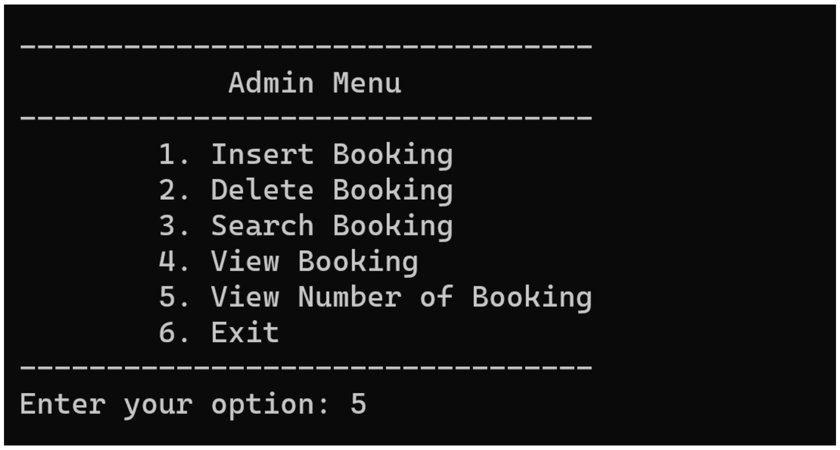


```
Viewing Booking


-------------------------------------------------------------------------------
|  Check-in  |  Check-out | Room No | Room Type | Customer  IC | Total Price |
-------------------------------------------------------------------------------
| 2022/12/30 | 2024/01/02 |   5022  |   Double  | 030802010034 |    223.78   |
| 2023/12/21 | 2023/12/25 |   1234  |   Double  | 030802010034 |    245.35   |
| 2023/12/21 | 2023/12/23 |   2234  |   Queen   | 030802010034 |    312.98   |
| 2023/12/22 | 2023/12/26 |   3452  |   Single  | 030213045566 |    212.35   |
| 2023/12/26 | 2023/12/27 |   1925  |   Single  | 030802010034 |    123.60   |
| 2023/12/27 | 2023/12/30 |   9812  |   King    | 030802010034 |    512.67   |
| 2023/12/28 | 2023/12/31 |   1366  |   King    | 030802010034 |    412.24   |
| 2024/01/24 | 2023/12/27 |   7652  |   Double  | 030715018864 |    321.63   |
| 2024/02/23 | 2023/12/26 |   2387  |   Queen   | 030603021456 |    355.21   |
| 2025/12/29 | 2024/01/05 |   5231  |   Single  | 030802010034 |    178.23   |
-------------------------------------------------------------------------------
```

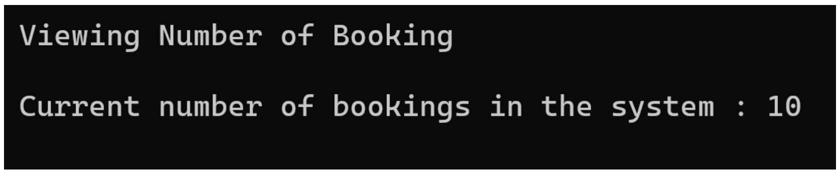
View Number of Booking

1. The user enters ‘5’.



```
----------------------------------
            Admin Menu
----------------------------------
        1. Insert Booking
        2. Delete Booking
        3. Search Booking
        4. View Booking
        5. View Number of Booking
        6. Exit
----------------------------------
Enter your option: 5
```

2. Then the system displays the total number of bookings.

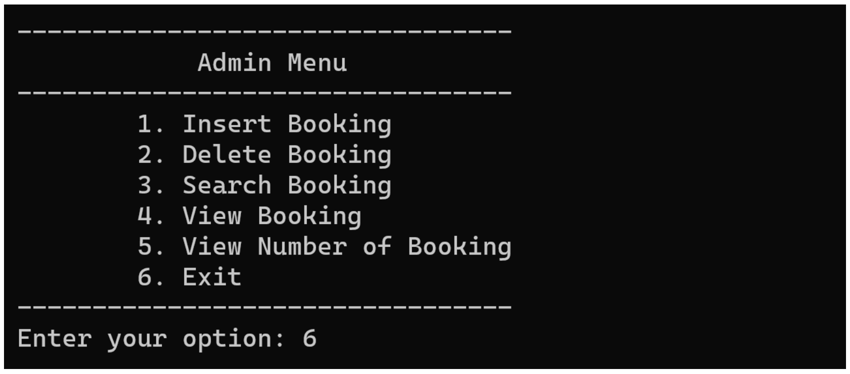


```
Viewing Number of Booking

Current number of bookings in the system : 10
```

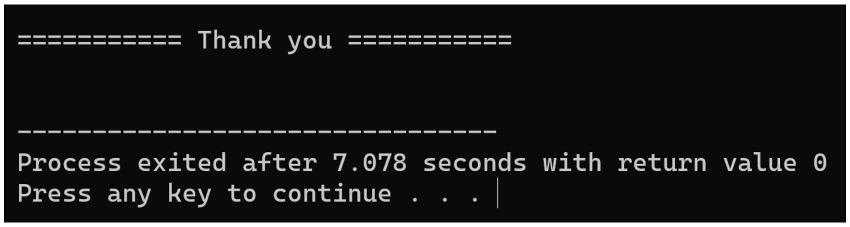
Exit

1. The user enters ‘6’.



```
---------------------------------
            Admin Menu
---------------------------------
        1. Insert Booking
        2. Delete Booking
        3. Search Booking
        4. View Booking
        5. View Number of Booking
        6. Exit
---------------------------------
Enter your option: 6
```

2. The program will stop running.



```
=========== Thank you ===========


--------------------------------
Process exited after 7.078 seconds with return value 0
Press any key to continue . . .
```