
UTM

UNIVERSITI TEKNOLOGI MALAYSIA

# SECJ2013

# Data Structure and Algorithm

# 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 in a hotel.
- To improve understanding and enhance application of data structures: sorting and searching in a real-world scenario.
- To utilize file mechanisms in C++ to read and write data from/ to external files as a simulation of the database of the system.

## Synopsis

The Hotel Booking System is designed for hotel administrators to manage bookings. By using file input operations, the system reads the files containing admin information and booking information. Admin's ID, name, position and phone number are stored in Admin class while check-in date, check-out date, room number, room type and total price of booking are stored in Booking class. Then requiring admin ID from the user to achieve authentication. The verification is carried out through sequential searching of ID input in Admin class. With the menu provided, admin is able to sort bookings ascendingly or descendingly based on check-in date, check-out date, room type and total price with merge sort techniques implemented.

# Design

## Class diagram

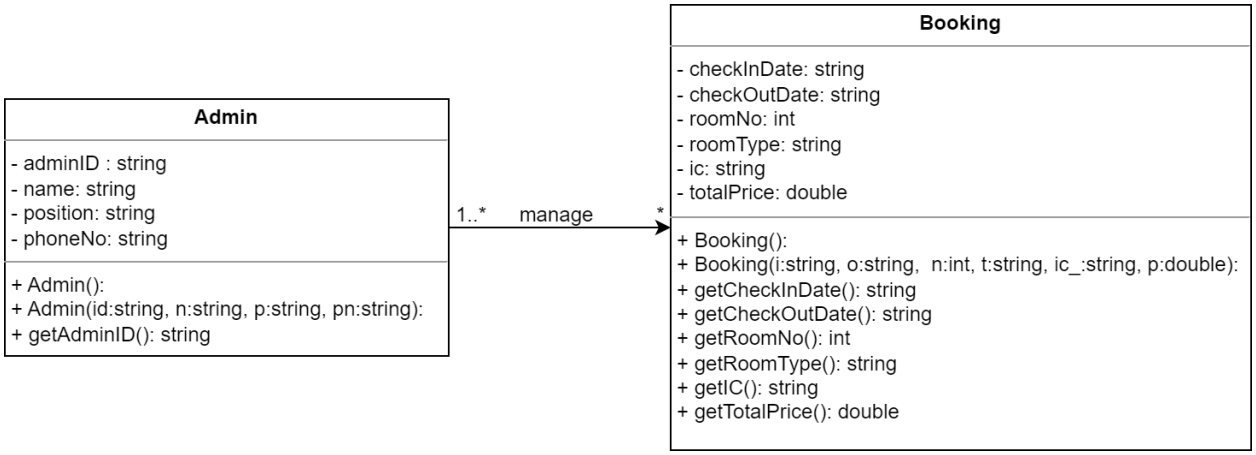


***Figure 1: Class Diagram of Hotel Booking System***

## Flowchart

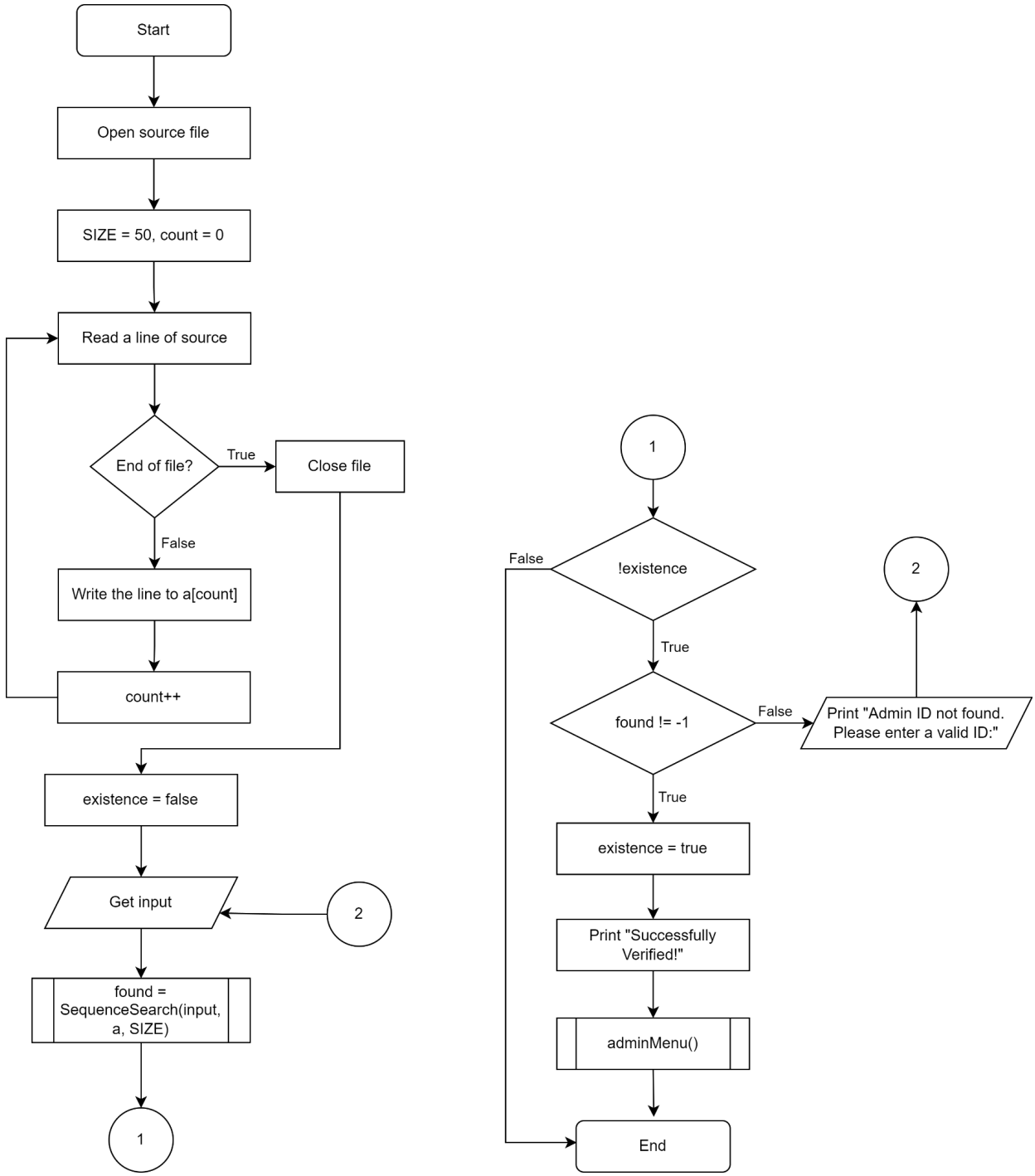


***Figure 2: Flowchart of Main Function***

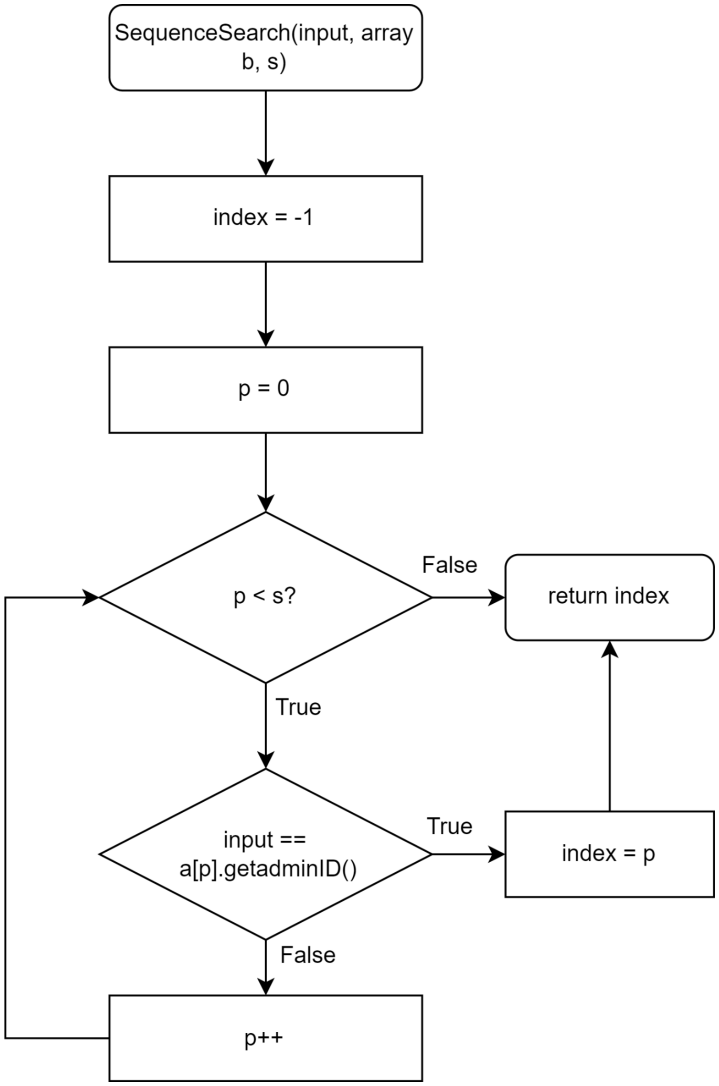


***Figure 3: Flowchart of SequenceSearch Function***

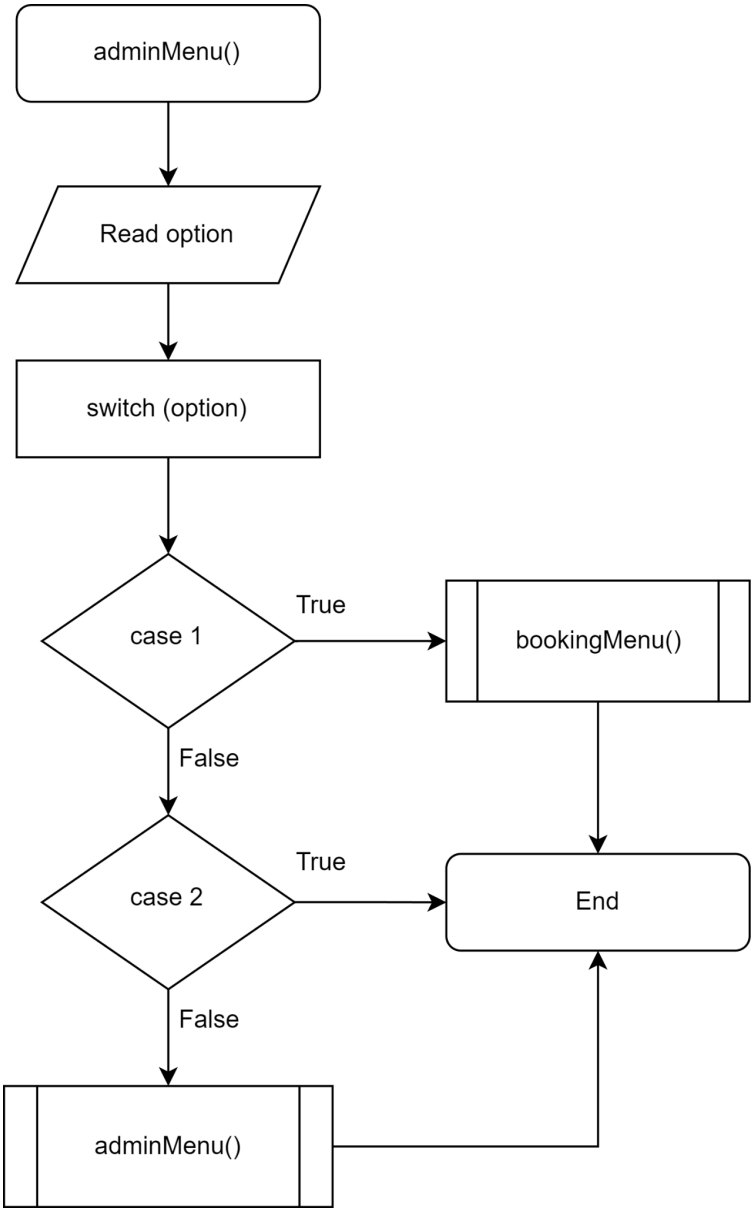


***Figure 4: Flowchart of adminMenu Function***

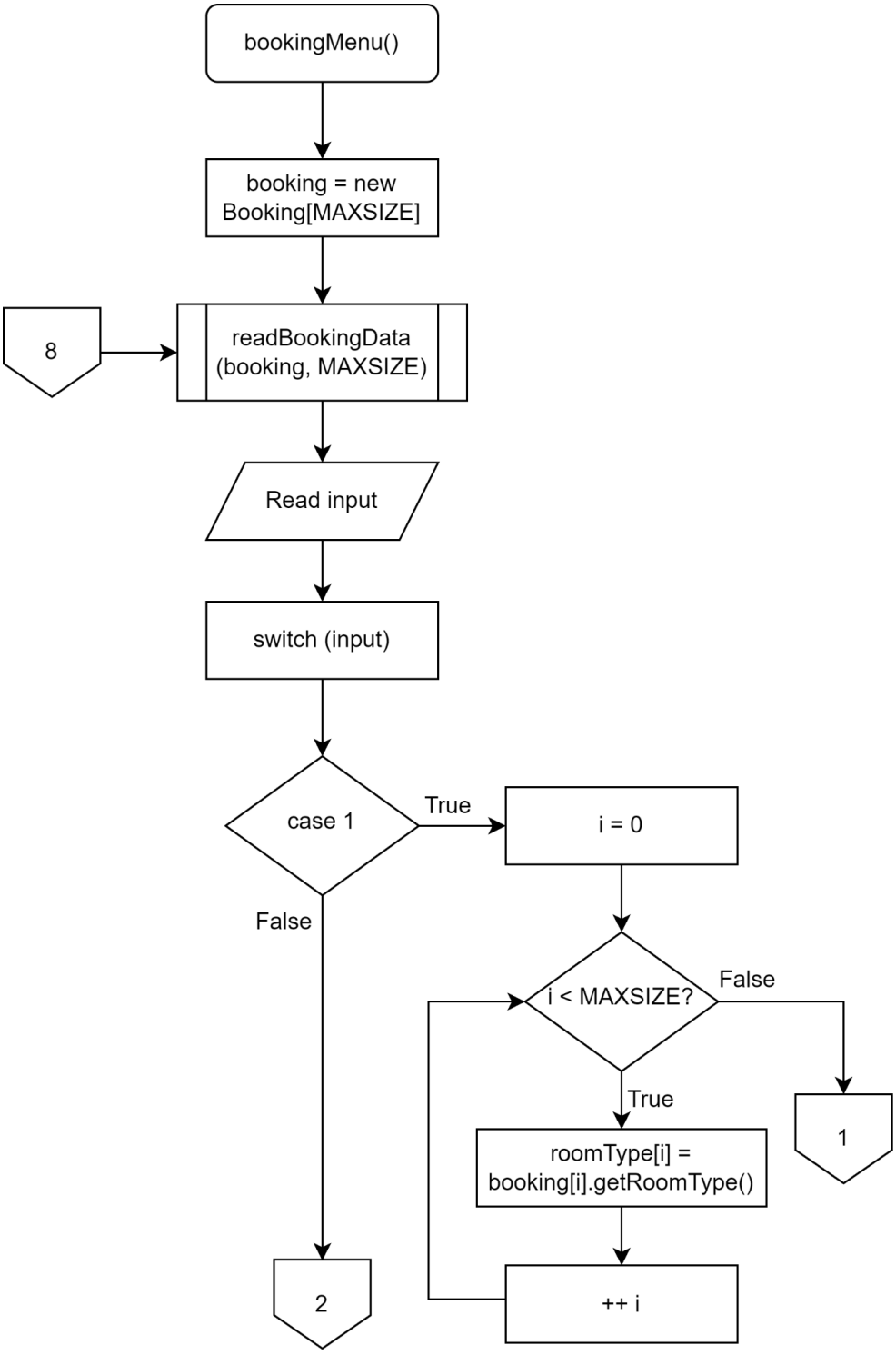


***Figure 5.1: Flowchart of bookingMenu Function***

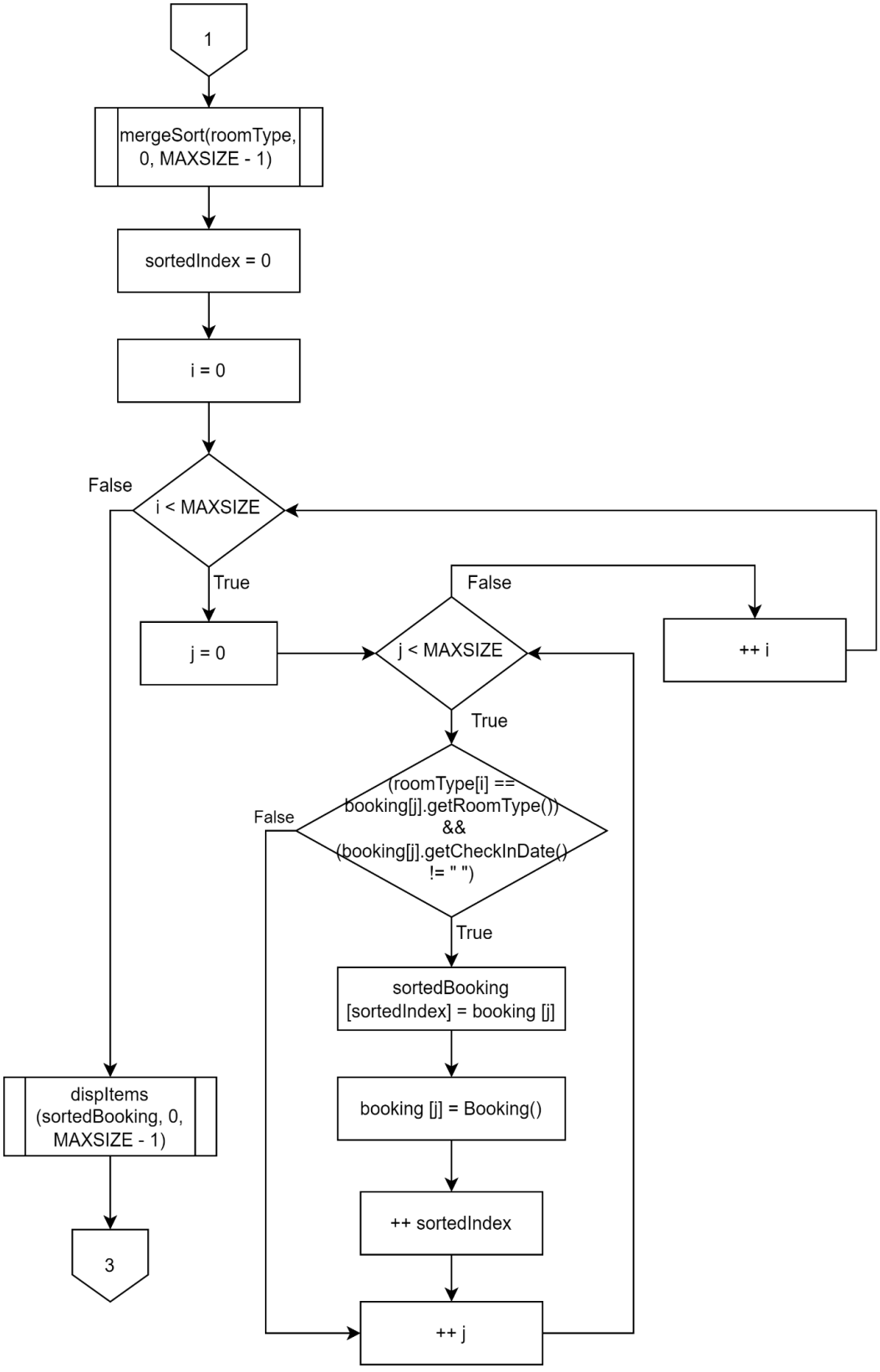


***Figure 5.2: Flowchart of bookingMenu Function***

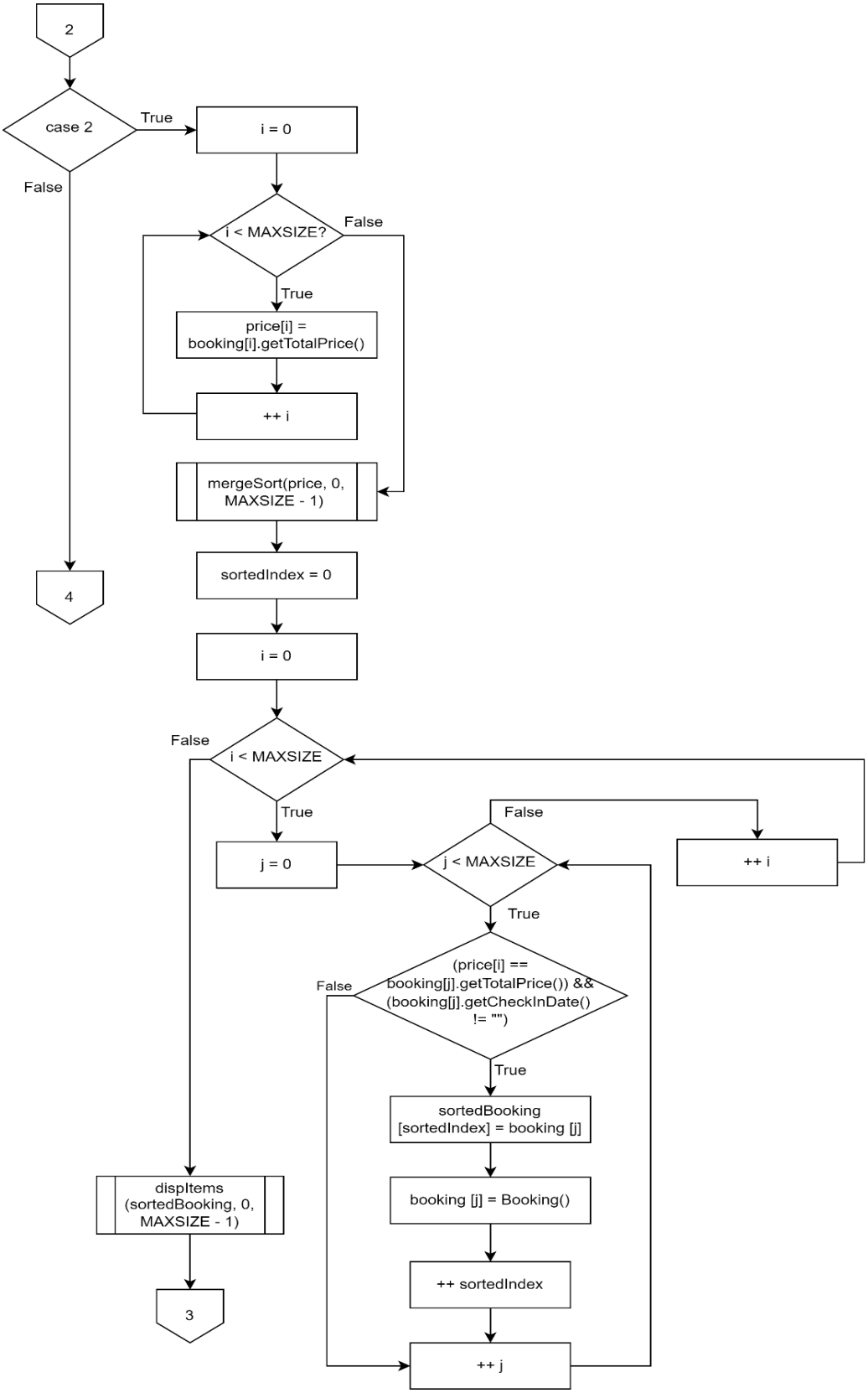


***Figure 5.3: Flowchart of bookingMenu Function***

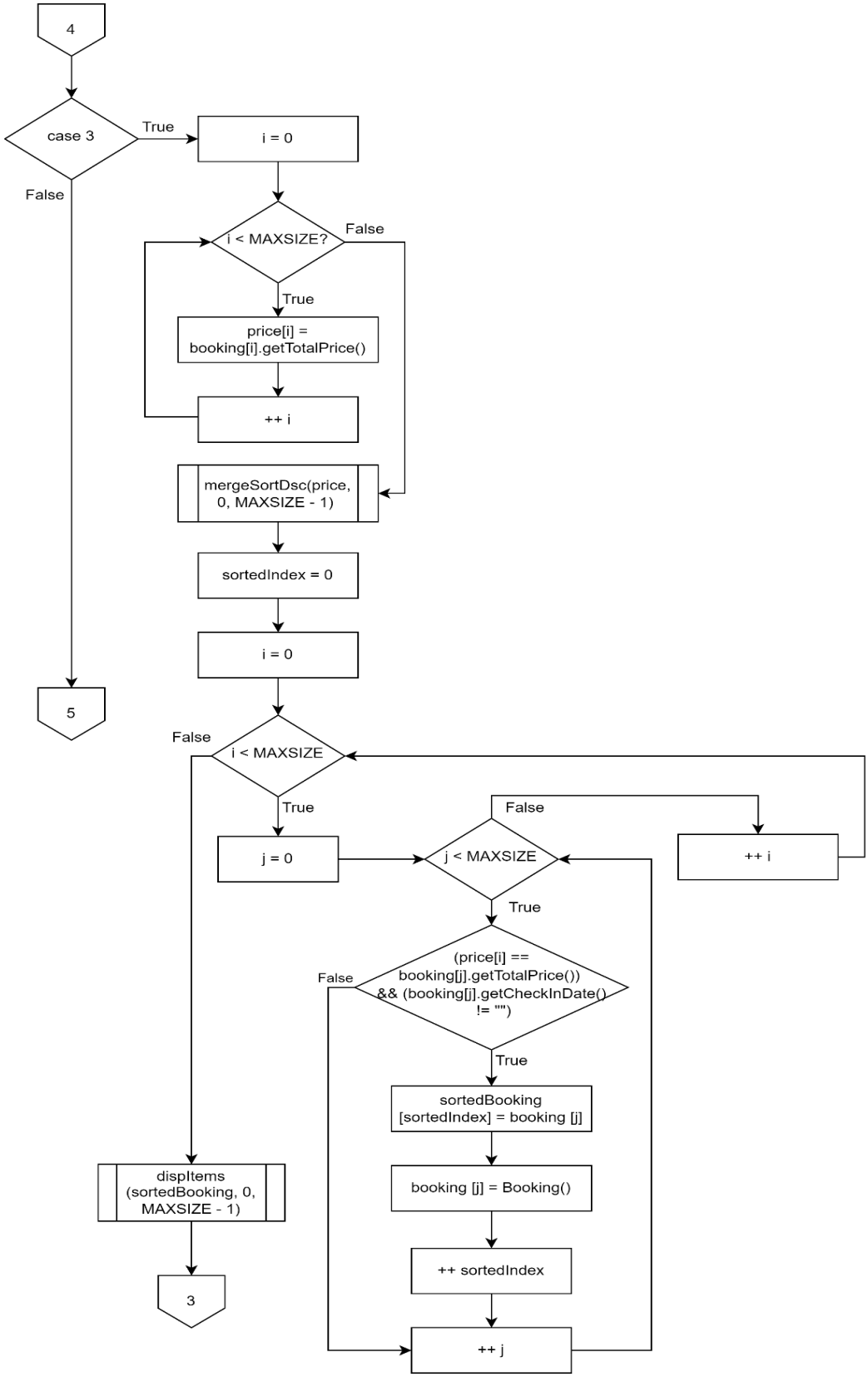


***Figure 5.4: Flowchart of bookingMenu Function***

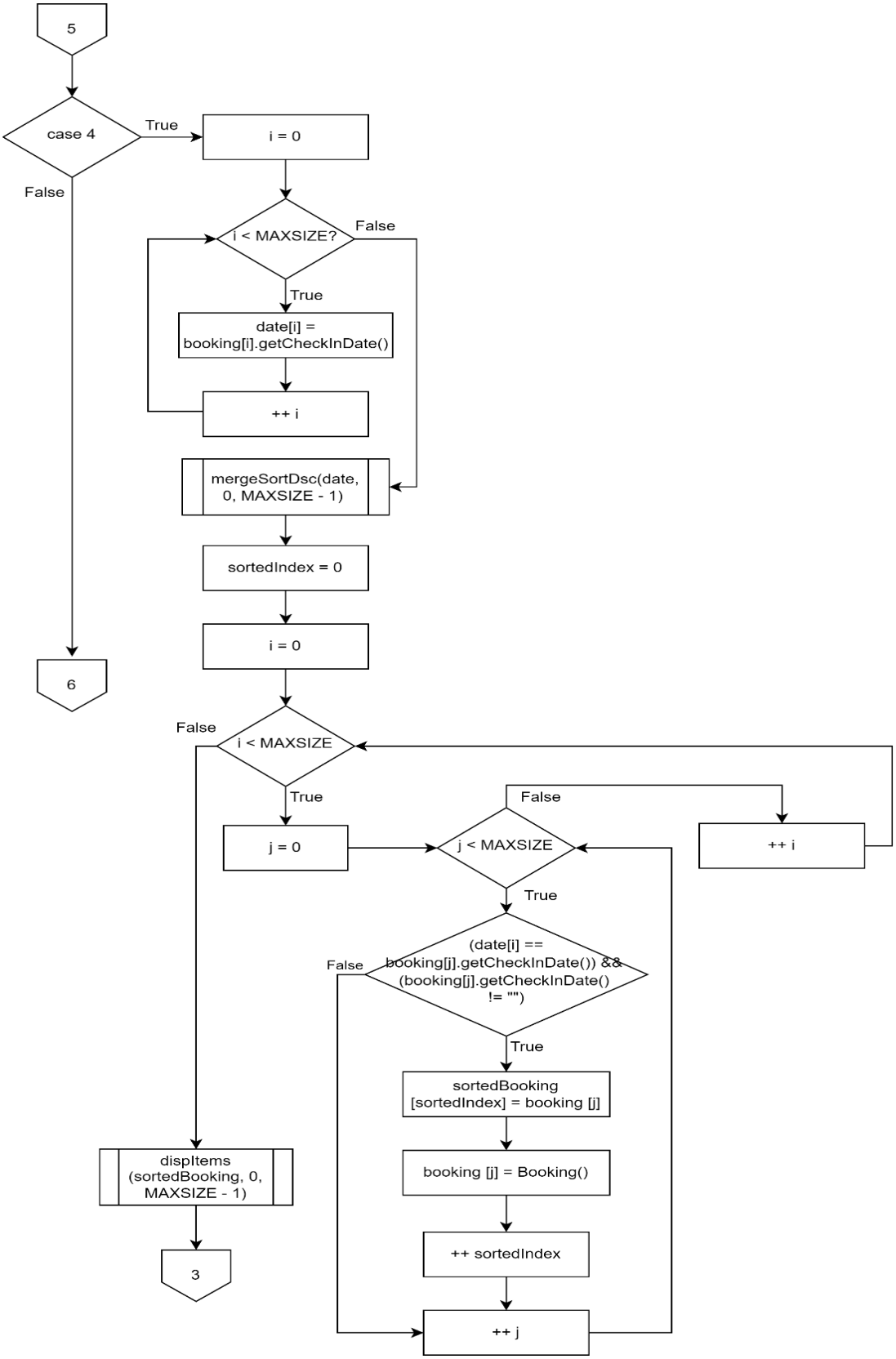


***Figure 5.5: Flowchart of bookingMenu Function***

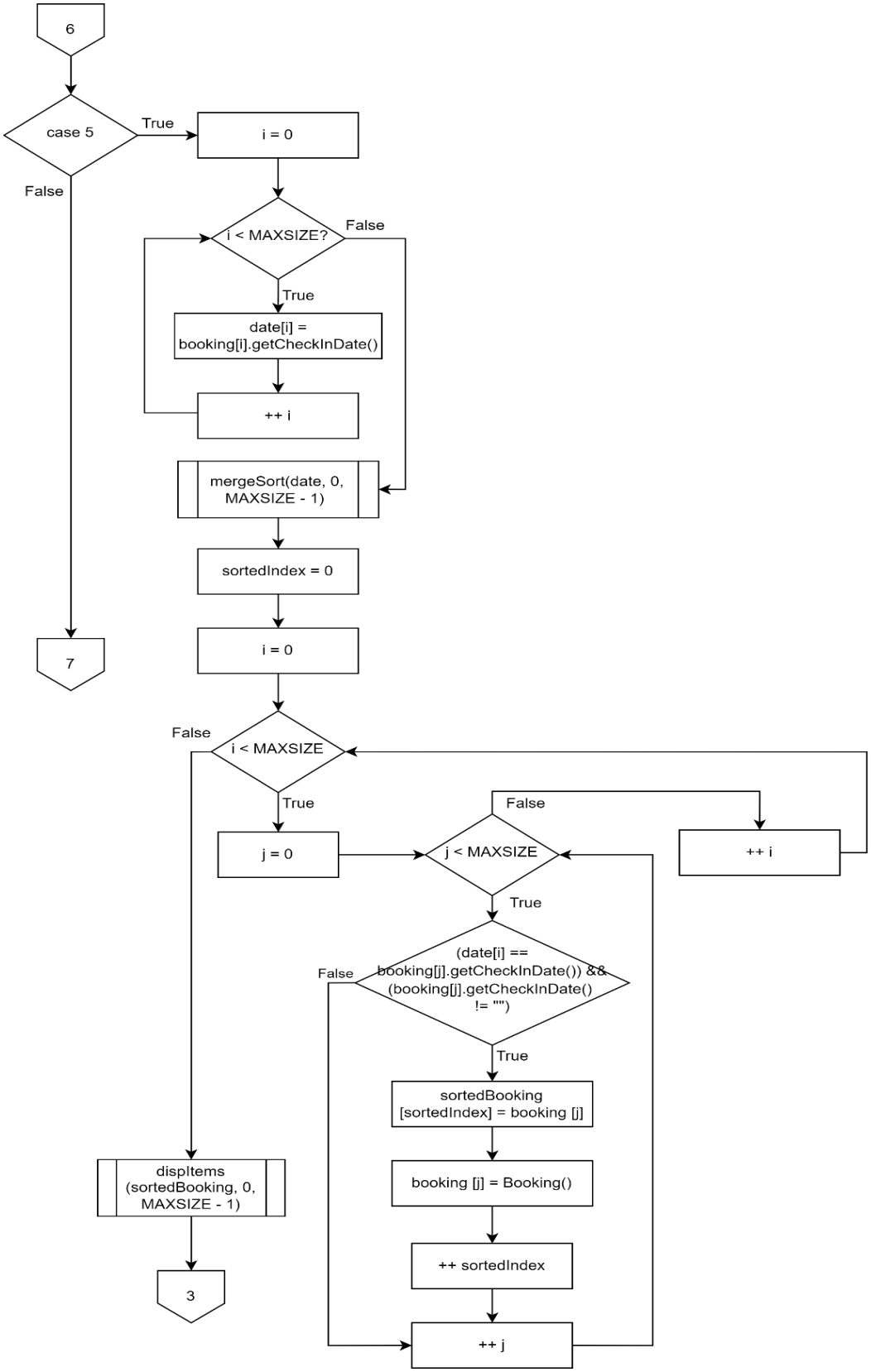


***Figure 5.6: Flowchart of bookingMenu Function***

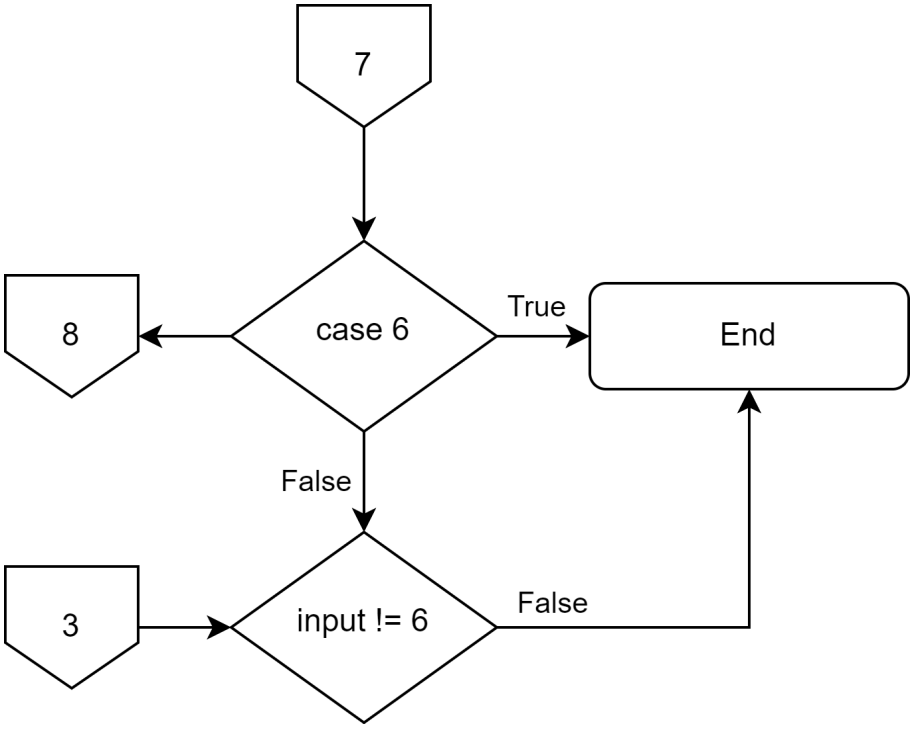


***Figure 5.7: Flowchart of bookingMenu Function***

***Link for whole flowchart diagram:***

*https://drive.google.com/file/d/1McNnrxXnNydsAVY5XBeCiwW9bcNT-sPO/view?usp=drive_link*

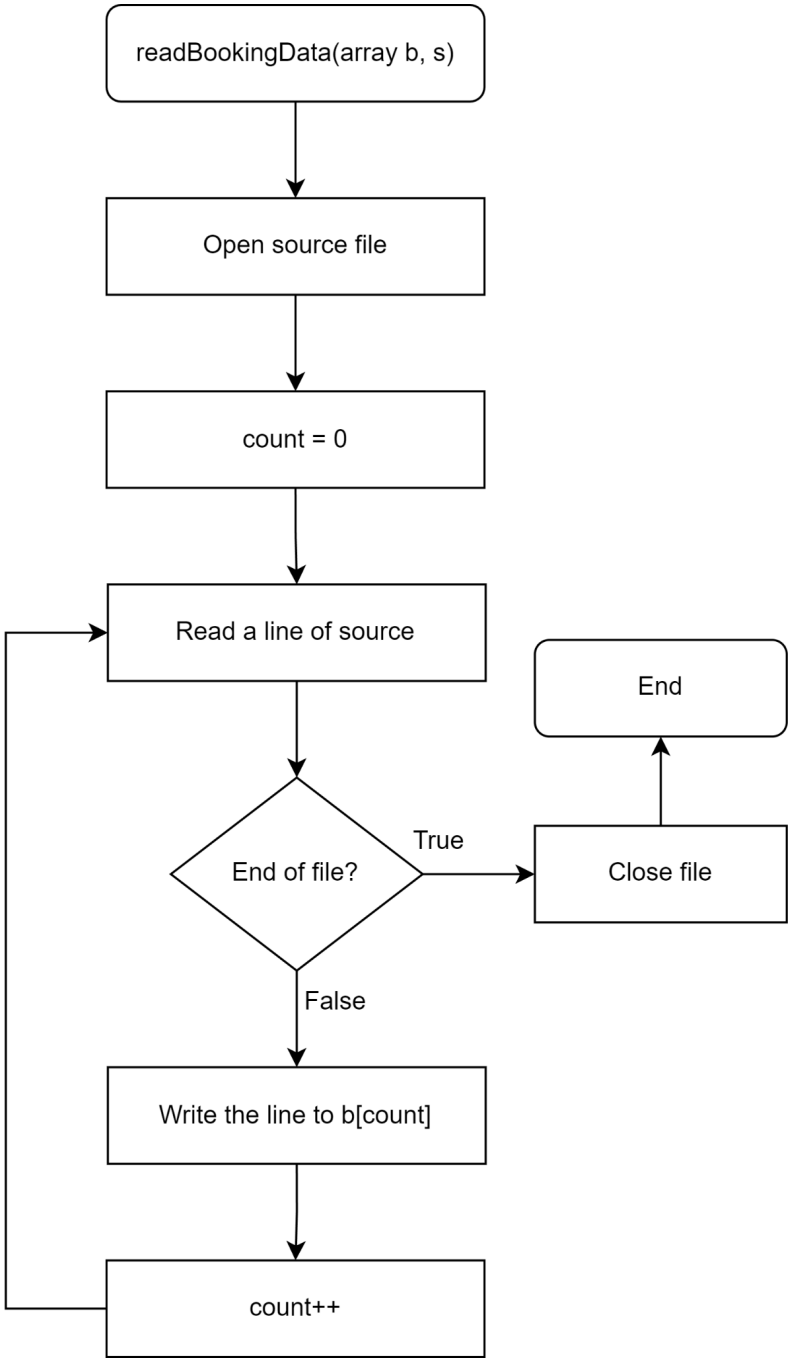


***Figure 6: Flowchart of readBookingData Function***

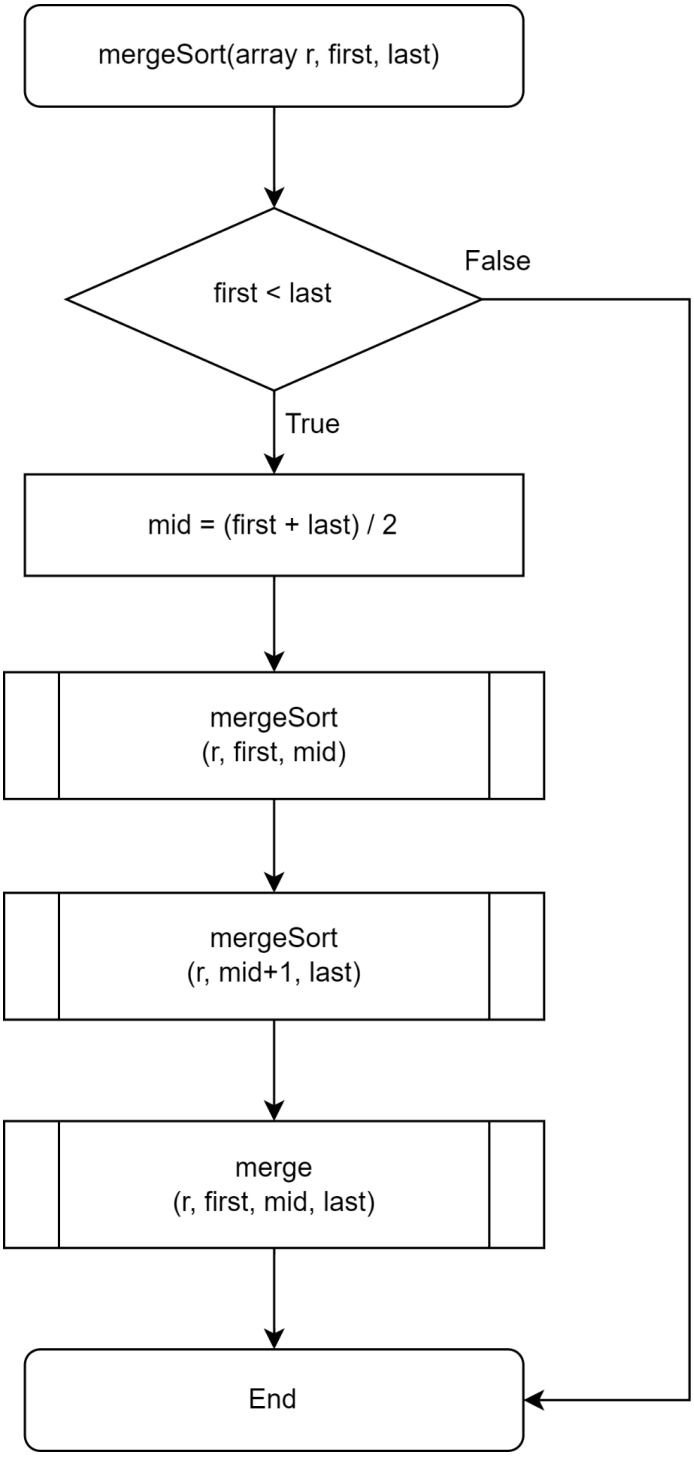


***Figure 7: Flowchart of mergeSort Function***

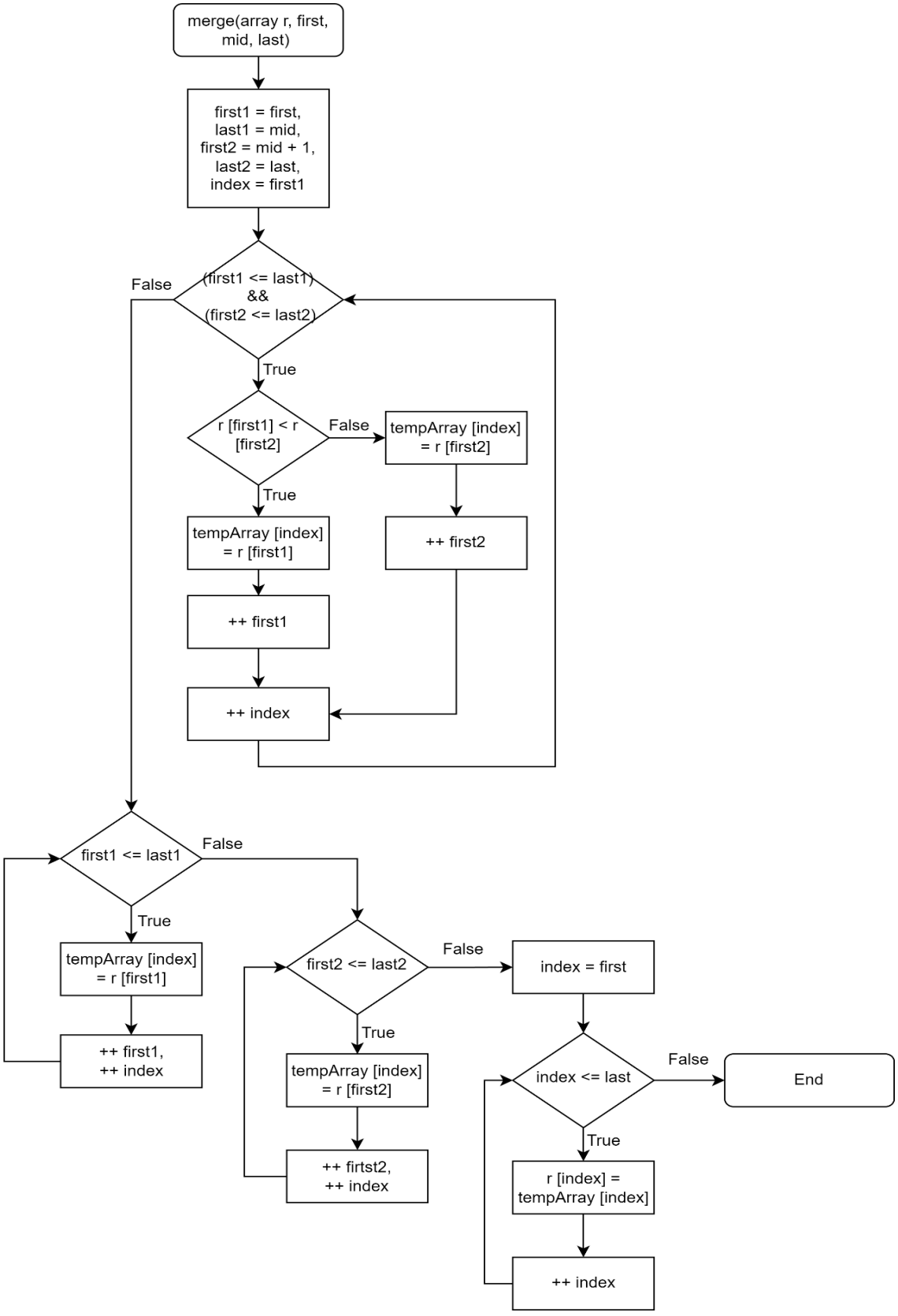


***Figure 8: Flowchart of merge Function***

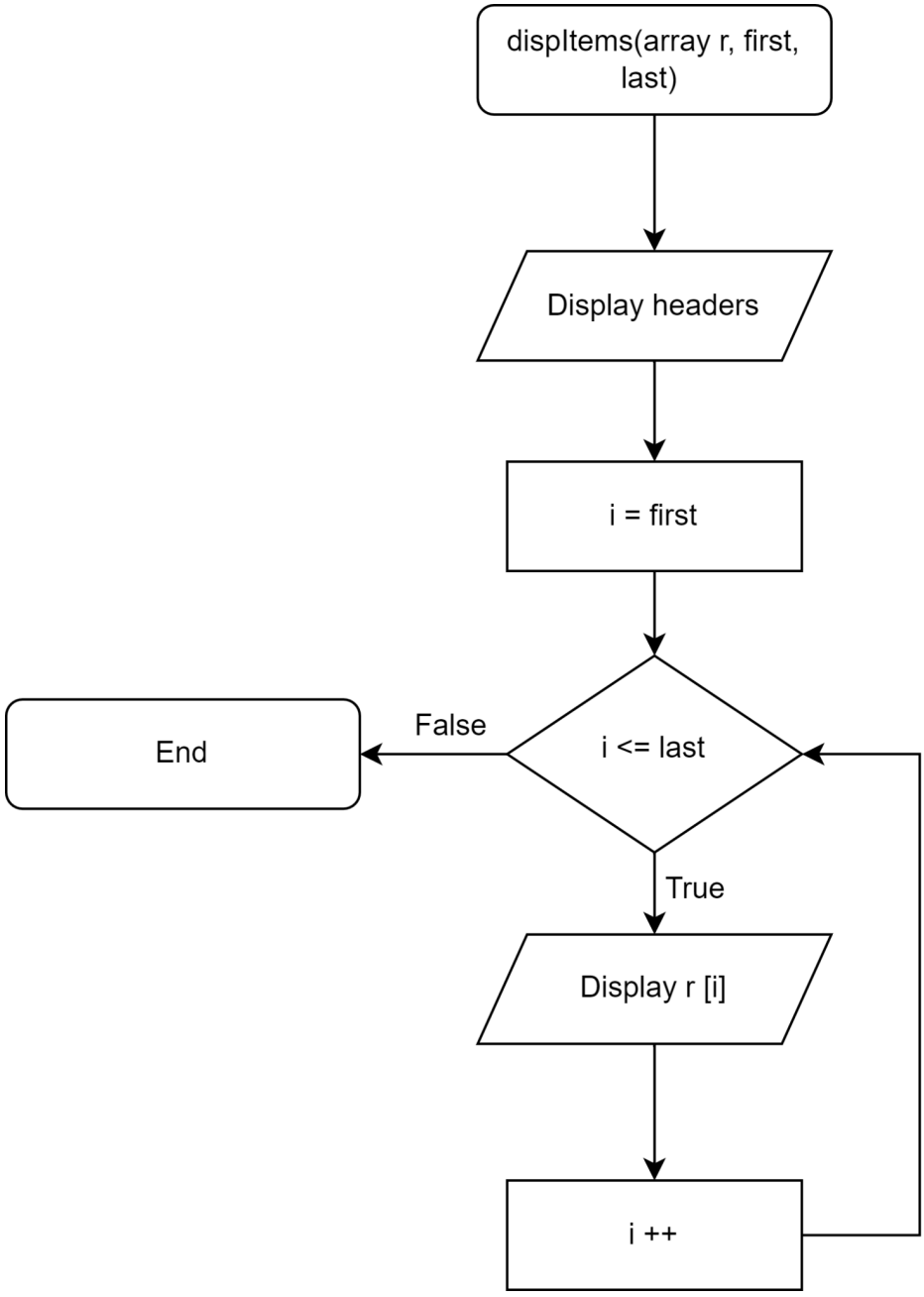


***Figure 9: Flowchart of dispItems Function***

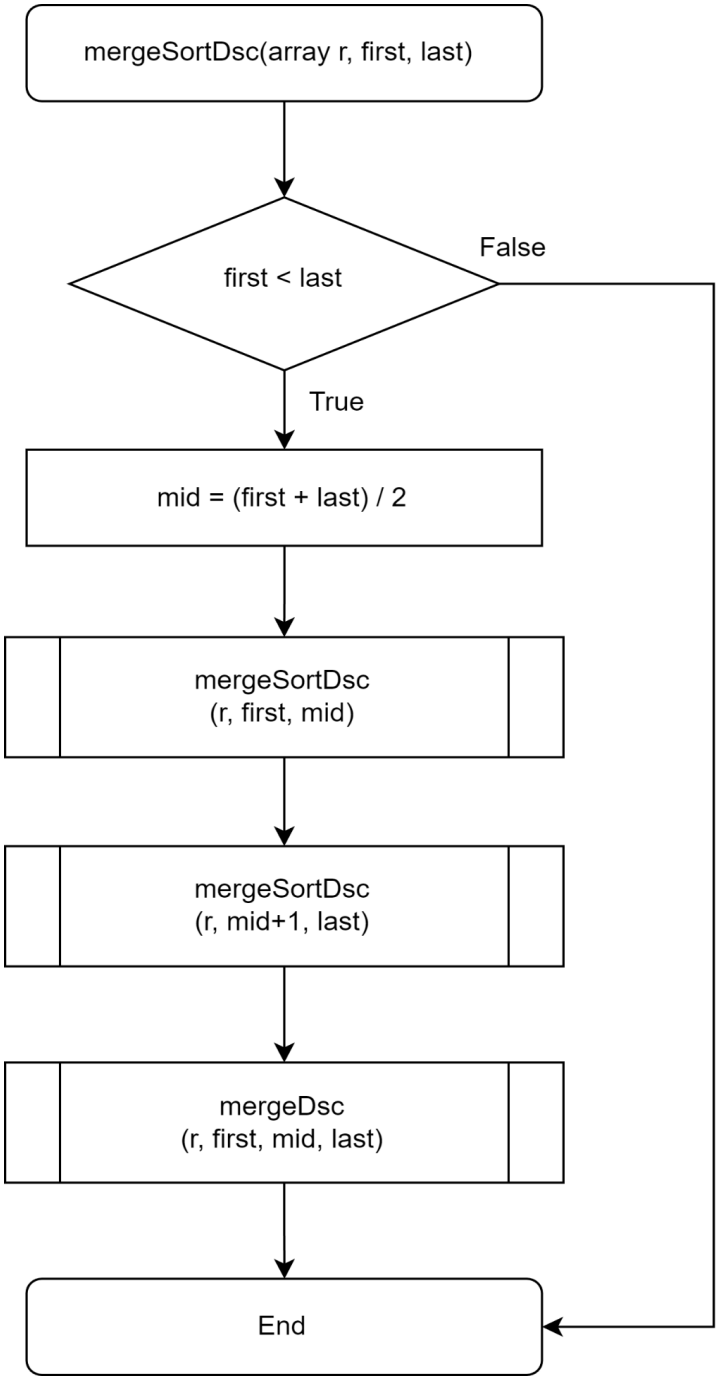


***Figure 10: Flowchart of mergeSortDsc Function***

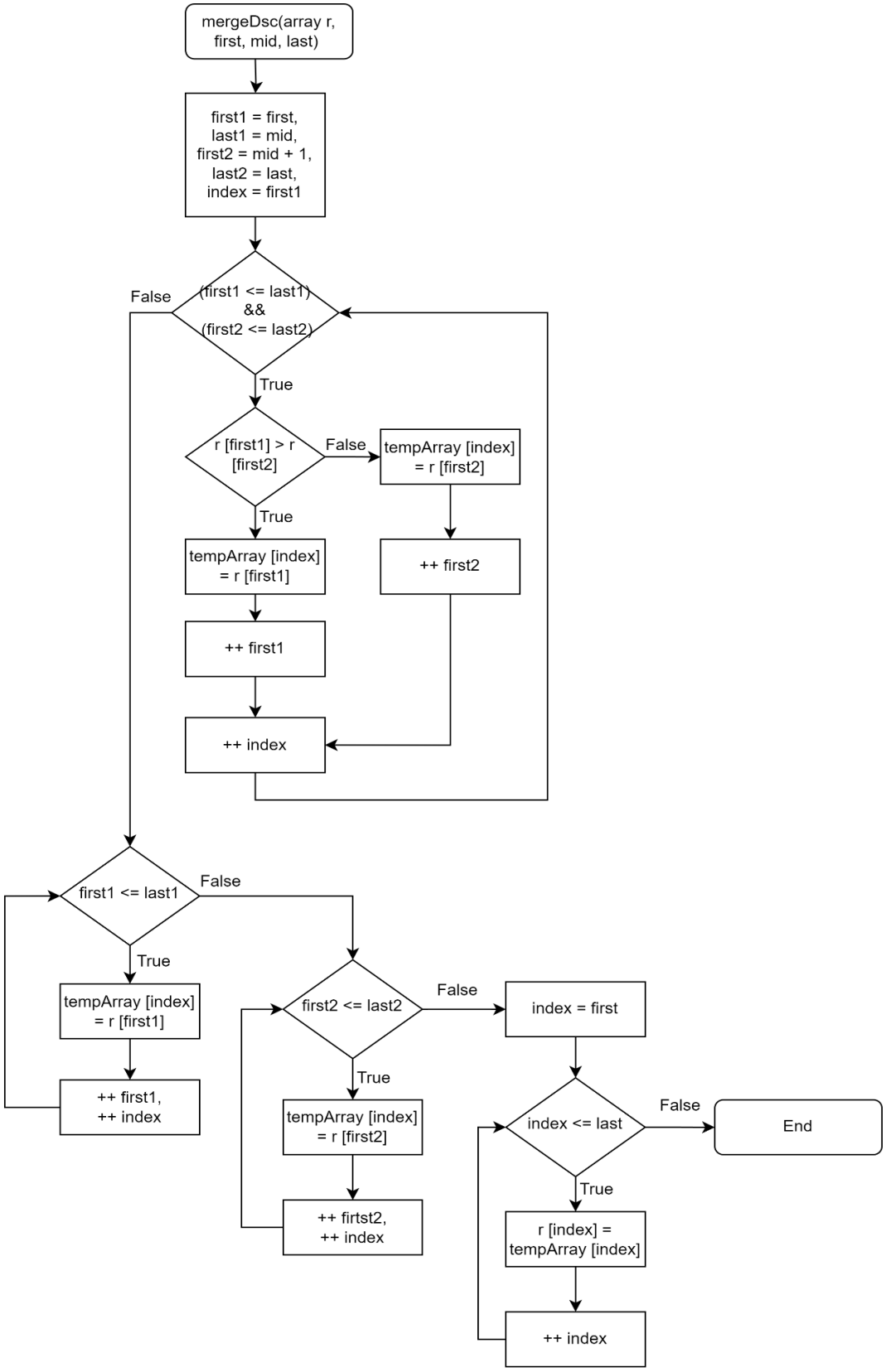


***Figure 11: Flowchart of mergeDsc Function***

## Description of how to implement data structure operations: Sorting and Searching

### 1. Sorting

In the Hotel Booking System, we have implemented Advanced Sort - Merge Sort which uses the Divide and Conquer Sorting Strategy in the mergeSort(), mergeSortDsc(), merge() and mergeDsc() functions. This data structure operation works by dividing an array received in the parameter into halves, sort each half and merge the sorted halves into one sorted array.

As an example, in our system, the mergeSort() function is used to divide the array received in parameter into halves by the second and third arguments indicating first and last index numbers of the array. This function is recursively called to divide the array received in the parameter list into small pieces until each of them has one item only. Then, by calling the merge() function, the small pieces are merged into larger yet sorted pieces repeatedly until the array is sorted. For the merge() function, it compares one item in the first half of the array with another item in the other half and moves the item with smaller value into a temporary array declared in the function and also the items remaining are moved to the temporary array. At the end of the function, the temporary array is copied into the original array. We also implement the mergeDsc() functions to sort the array in an descending order by modifying the algorithm, for instance, in merge() function, the condition r[first1] < r[first2] is changed to r[first1] >= r[first2] when implementing mergeDsc() function. These functions are used to sort the Room Type, Room Price (Low to High), Room Price (High to Low), Latest Check-In Date and Earliest Check-In Date of the customers. Function template technique in C++ is also implemented when declaring the sorting functions in order to be flexible to work with different data types.

We have chosen the sorting algorithm - Merge Sort out of five sorting algorithms we have learned after the consideration of the growth rates of time required by the Merge Sort has the Big O Notation of O(n log n) which is faster compared to $O(n^2)$.

**2. Searching**

In the Hotel Booking System, we have implemented Searching Technique - Sequential search in SequenceSearch() function. This data structure operation works by examining each element in the array sequentially and compares its value with the search key.

As an example, in our system, the SequenceSearch() will receive a search key which is the adminID entered by the admin, an array to be searched which is the admin information read from the admin file (database) and the size of the array in its parameter list as arguments. The variable index is set to -1 initially to indicate the record is not found. When comparing the adminID with the array which is accessing getadminID() function, if the adminID matches with any adminID from the database, the index is assigned the current array index and eventually returns the index. After returning to the main function, the result of SequenceSearch() is checked as if the value is equal to -1 (not found) or ! = -1 (found). This function acts as a verification and authentication for the admin identity.

We have chosen the Searching Technique - Sequential search as it is a basic function that is suitable to be implemented in a small size of list in which after considering the number of administrators in a hotel will not be a large population.