Lab Exercise 4
Travel Desk

You are included as a member of the organising team of Ignus 2023. You are in charge of the travel desk. You decide to make a software to allow participants/Attendees to reach IIT Jodhpur campus from various locations in the city using cabs and buses arranged by your team.

As an administrator, you can perform the following operation:

ADDTRIP [Vehicle Number] [Seating Capacity] [Pick-up Location] [Drop-Location] [Departure Time]

Description: With this operation, a vehicle identified by its vehicle number and with a designated seating capacity will be made available for service between a pickup location and a drop location with a specified departure time.

The software will read such operations from an input file and invoke appropriate functions to perform the task.

You need to offer the visitors (users) a search functionality with the following operation:

SHOWTRIPS MBM IITJ 10:00 13:00

SHOWTRIPS IITJ Poata 18:00 23:00

A user can specify a pick-up location and drop-location and a chosen time at which he/she needs to board. The software answers the query with a list of available vehicles that depart after the chosen time and before a chosen time.
All times are specified in a 24 hour clock. The *after* and *before* times should be interpreted as *on or after* and *on or before*.

This operation invokes the query function on the appropriate ***location*** object. The returned value of the query is a list of *trip* objects.
The returned *trip* objects are used to print the details such as
Vehicle Number, Available Seats, Pick-up Location, Drop-Location, after time, before time

A user can book a vehicle by specifying

BOOK Vehicle Number IITJ Paota 18:00

Software Organization

Define a class ***vehicle*** with details, such as, vehicle number and carrying capacity and list of ***trips*** assigned to that vehicle.

Define a class ***trip***, with elements such as — reference to a reference to the ***vehicle*** (object) used for this trip, pick-up location, drop location, departure time and number of seats booked on the trip.

Define a class ***Location*** with a data field *name* and a list of ***Service*** objects associated with it.

A ***Service*** object is uniquely identified by the pick and drop location pair. A distinct service object is created for every distinct drop location. All the ***Service*** objects associated with a ***Location*** object will have the same pick-up location.
A ***Service*** object collects together all the ***Trip*** objects that have the given location as the pick-up point. A transport ***Service*** object has a Binary Search Tree (BST) with nodes that refer to ***Trip*** objects. Each tree node refers to a ***trip*** object. The departure time given by the ***Trip*** object is used as a key for the node.
Note that there will be multiple ***Service*** objects associated with a ***Location***.

Define a class called ***TravelDesk*** which will have data fields such as — list of Vehicle objects and list of Location objects. This class will also have member functions that support the operations specified in the interaction file. For example, it will have member functions such as addTrip(……), showTrips(….) and book(…..) with the required arguments. These functions will invoke appropriate functions in the appropriate ***Vehicle*** object or ***Location*** object.

The addTrip(…..) method will check if the vehicle number mentioned in the trip is already there in the vehicle list. If not, a new vehicle object will be created and added to the vehicle list. It then creates a trip object and (i) adds it to the vehicle object, (i) adds it to the Location object associated with the pick-up location of the trip and (ii) adds it to the Service object (corresponding to the pickup-drop location pair) in the location object. The service object creates a node for this trip and performs the **node insert operation into the BST** using the departure time as the key.

To process an operation such as SHOWTRIPS IITJ Poata 18:00 23:00, the showTrip(….) function of the Travel Desk class will be called. This function will then identify the ***Location*** object for the location IITJ and select the transport ***Service*** object that corresponds to the destination Poata. The BST in the Transport ***Service*** object is then checked for nodes that depict trips with departure times in the time range specified by the user. You can do this by first identifying the node (if it is there) with the indicated *begin* time and then repeatedly call the successor function till the returned node has a departure time that is after the *end time*. The showTrip(…) function returns the list of trip objects relevant to the query.

A BOOK request will be processed by the book(….) function of the ***TravelDesk*** class. It will also It will be associated with an increase in the number of booked seats for the corresponding trip object. If the number of booked seats reaches the vehicle's carrying capacity, the corresponding **node will be deleted from the BST** with that transport ***Service***. The book(…) function returns the trip object that has been modified to increase the number of booked seats.

The interaction between the administrator/user and the software happens through a list of operations (commands) which are given in a file called interaction.txt.
The administrator interacts with the software using the ADDTRIP command. The user interacts with the software using the SHOWTRIPS and BOOK commands.

Your code should read these commands and take appropriate actions.

Error Conditions:
The BOOK function will have no effect in case the command specifies an incorrect vehicle number or incorrect departure time. In case the same vehicle is used for multiple trips that correspond to the same departure time from multiple pick-up locations, allow that (though impossible in the real world). In case the specified time range in the query is incorrect, for example, a user specifies 21:00 15:30 for *after* time and *before* time, then your code should consider the earlier time (15:30) as the *after* time and the later time (21:00) as the *before* time. Any command that does not begin with keywords such as ADDTRIP, SHOWTRIPS, BOOK, can be ignored.

A sample interaction.txt file has entries such as:
(You will be provided with another file to test your code).

```
ADDTRIP RJ19PB001  25  Sardarpura IITJ 15:00
ADDTRIP RJ19UB001  7  AIIMS IITJ 11:00
ADDTRIP RJ19PB002  40  AIIMS IITJ 14:00
ADDTRIP RJ19PB003  40  MBM IITJ 11:00
ADDTRIP RJ19PB004  40 MBM IITJ 12:00
ADDTRIP RJ19PB005  40 MBM IITJ 13:00
ADDTRIP RJ19PB006  40 MBM IITJ 14:00
ADDTRIP RJ19PB007  40 MBM IITJ 15:00
ADDTRIP RJ19PB008  40 MBM IITJ 16:00
ADDTRIP RJ19PB009  40 MBM IITJ 17:00
ADDTRIP RJ19PB023  40 Paota IITJ 11:00
ADDTRIP RJ19PB057  40 Paota IITJ 15:00
ADDTRIP RJ19PB028  40 Paota IITJ 16:00
ADDTRIP RJ19PB019  40 Paota IITJ 17:00
ADDTRIP RJ19PB034  40 Paota IITJ 12:00
ADDTRIP RJ19PB035  40 Paota IITJ 13:00
ADDTRIP RJ19PB026  40 Paota IITJ 14:00
ADDTRIP RJ19PB057  40 Paota IITJ 15:00
ADDTRIP RJ19PB028  40 Paota IITJ 16:00
ADDTRIP RJ19PB019  40 Paota IITJ 17:00
ADDTRIP RJ19PB004  50  Sardarpura IITJ 21:00

SHOWTRIPS MBM IITJ 13:00
BOOKTRIP MBM IITJ RJ19PB004 12:00
BOOKTRIP AIIMS IITJ RJ19UB001 11:00
BOOKTRIP AIIMS IITJ RJ19UB001 11:00
BOOKTRIP AIIMS IITJ RJ19UB001 11:00
BOOKTRIP AIIMS IITJ RJ19UB001 11:00
BOOKTRIP AIIMS IITJ RJ19UB001 11:00
SHOWTRIPS AIIMS IITJ 11:00
```