My implementation uses three data structures one for storing museums (Mu) one for storing visitors' data (book) who have made bookings for the visits to the museums or at least tried to and the third for storing reservations (res) for each of the visitors. Mu and book store the museums and visitors in a BST with their names as key field for ordering. BSTs are used to allow their retrieval in logarithmic time. While res stores reservations in a linked list because it allows to add a new reservation dynamically and because, for the given program, all reservations need to be accessed at once. They can also be stored in a BST with the number of day as ordering field without much modification if required in the future.

DATA STRUCTURES
Mu contains:
the name of the current museum
the number of time slots (N)
the N time slots as a string (because the time slots have a predefined format)
the number of seats available on each day for each time slot as an integer (dynamically allocated matrix)
and a pointer to the next left and right museums in the BST

book contains:
the name of the visitor
a pointer to their list of reservations
and a pointer to the next left and right visitors in the BST

res contains:
a pointer to the museum the visit is booked for
the number of people visiting for current visit
the day for which the reservation is made
the slot index for which the reservation is made
and a pointer to the remaining list of reservations by the given visitor

FUNCTIONS 12
main ()
it reads the file, goes into an infinite loop printing a menu until exit is specified, calls the reserve or print function according to the response to the menu, frees allocated memory after the loop and returns

readfile (filename)
opens the file using a my safefopen function, reads the time slots for each of the museums one by one, initializes seats available on all days of the year for each time slot to the seats available as mentioned in the file, adds the read data to the mu BST using the mu_binary_add function and finally closes the file and returns.

safefopen (filename, mode)
tries to open the file and checks if error occurred, returns the pointer if no, prints a message and quits the program if yes

mymalloc (size)
same as safefopen just with memory allocation

mu_binary_add (museumsbst, toadd)
adds the pointer toadd to museumsbst to update the Mu BST

reserve ()
asks for details of the reservation, checks for some errors, creates the visitors bst if not already created or looks for the visitor in the bst using or creates a node for them using pointerto and then tries to add the details of the reservation using addres which check for errors, updates museum data and makes the reservation

pointerto (visitorBSTroot, visitorname, create)
looks in the given BST for visitor's name and returns the pointer if found, or creates a new node in the bst at an appropriate location if create is set to TRUE

mpointerto (museumBSTroot,museum_name)
searches for museum with given museum name and returns the pointer to it if found otherwise NULL

addres (visitorbook,museumname,day,people,slot)
tries to decrement the seats for the given time slot checks for errors if none then adds to the reservation list using list_add

museum_decrease (mp,slot,day,people)
looks if museum exists then looks for the slot using slot_lookup, checks if enough seats are available and updates them. Return the status of the success or errors or index of the slot if no errors.

slot_lookup (museum,slot)
uses linear search to search for the slot in museum. And returns the index of the slot. Assuming the slots last at least half an hour this search will take at most 48 lookups so this function is O(1).

list_add (visitor_pointer, museum_pointer, day, people, slot)
adds the data to the new reservation and appends it to the head of the reservation list of the visitor.

printres ()
asks for the name of the visitor for whom to print the reservations, looks in the visitor BST using pointerto and then prints all the reservations for that visitor using printall. For visitors who has no reservations but did try at least once to make a reservation nothing is printed. For non existing visitors error is shown

printall (visitor_reservation_pointer)
uses recursion to print the list of reservations

free* (*)
free the data structures recursively

the changes have been commented in the code itself. There are no major changes. Just a few changes to make the code function correctly which were a little difficult to think about in examination conditions and with hands

the code compiles and has been checked for memory leaks

the data file is named "file.txt"
the test file is named "test.txt"