Movie Ticket Booking System



Sahithi M.R (205002076)

Rohith S (205002075)

# Description:

The Movie Ticket Booking System will provide a platform for customers to book tickets for films of their choice at theatres that screen the film. The system takes in the information of the user before booking a ticket and assigns an appropriate ticket ID. It will allow users to view the ticket availability of a film in a visual manner. It also gives the option to book multiple tickets at once with preferred seats as well as the option to cancel booked tickets.

# Modules:

New Booking:

This module is used to book a new batch of tickets for a film of choice.

Seat Availability:

This provides a visual representation of the cinema theatre which in turn gives the user an idea of the ticket availability and choice of seats according to their preference.

Cancel Tickets:

This module is used to cancel tickets that have been booked with the use of the appropriate ticket ID

# Data Structure Comparisons:

Reasons for choosing AVL Trees:

AVL tree enables the search operation to be completed in O(log n) time complexity, where log n is the height of the AVL tree.

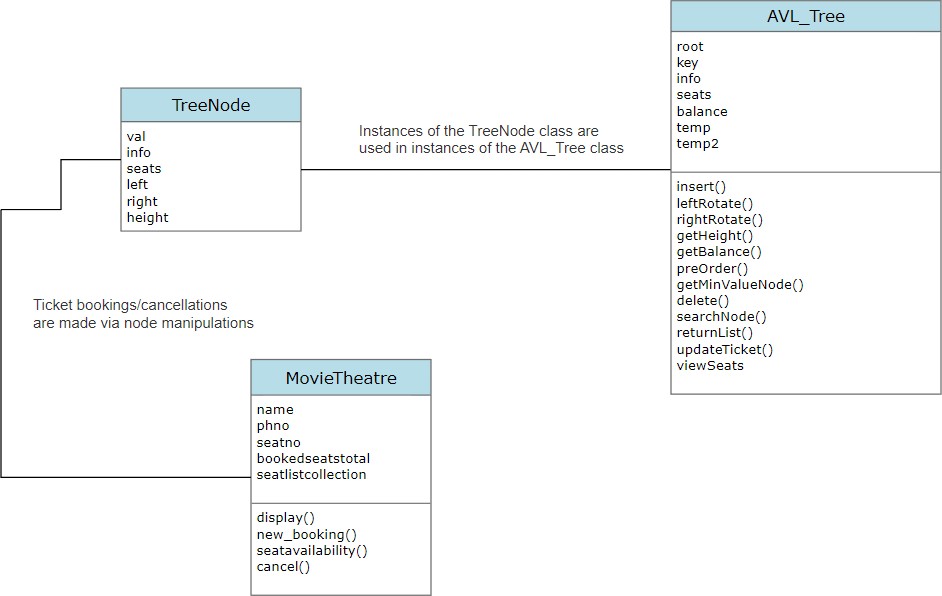
Linear data structures like stack, queue, linked lists are not used as the time complexity for searching is O(n).

A classical binary search tree also provides searching in O(log n) time complexity on an average but may become O(n) when the tree becomes skewed.

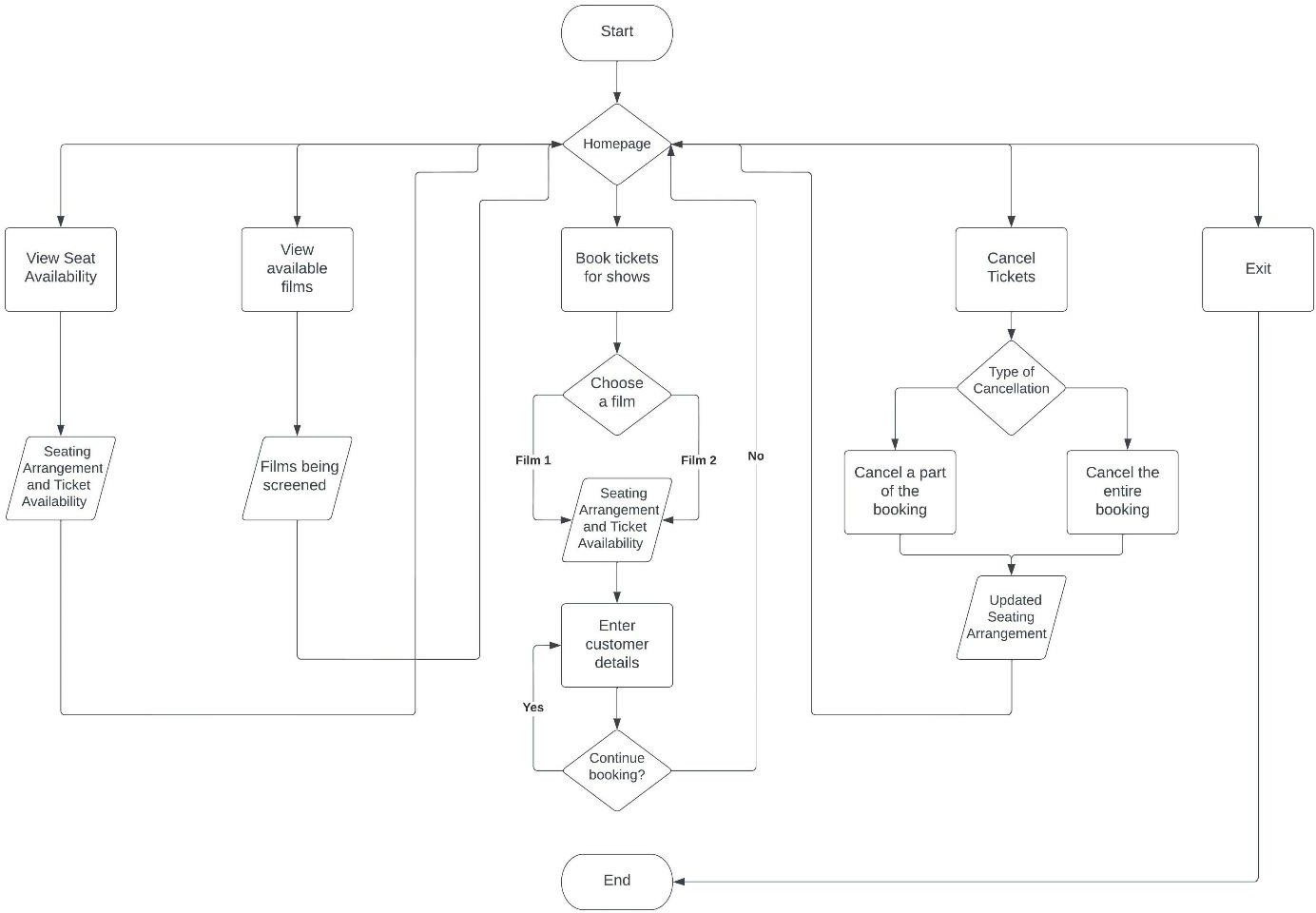
AVL trees are used in databases where faster retrievals are required.

Splay trees are more memory-efficient than AVL trees, because they do not need to store balance information in the nodes. However, AVL trees are more useful in multithreaded environments with lots of lookups, because lookups in an AVL tree can be done in parallel while they can't in splay trees.

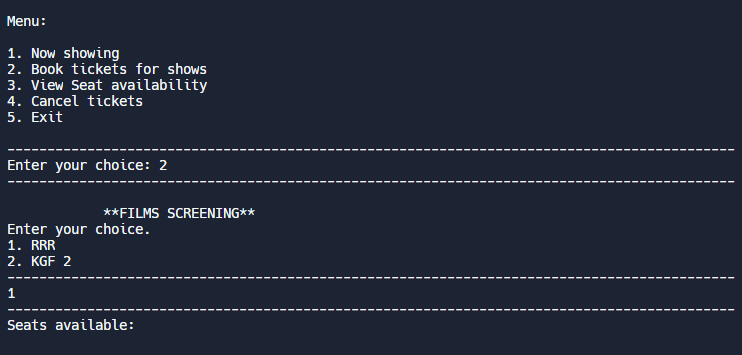
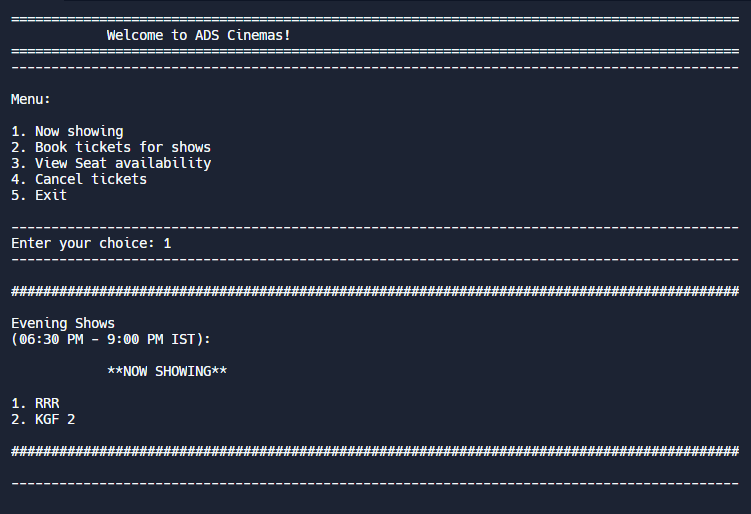
# Class Diagram:

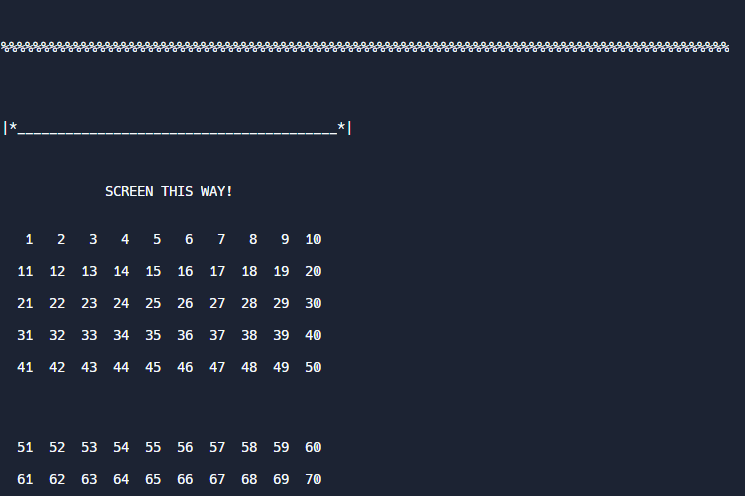


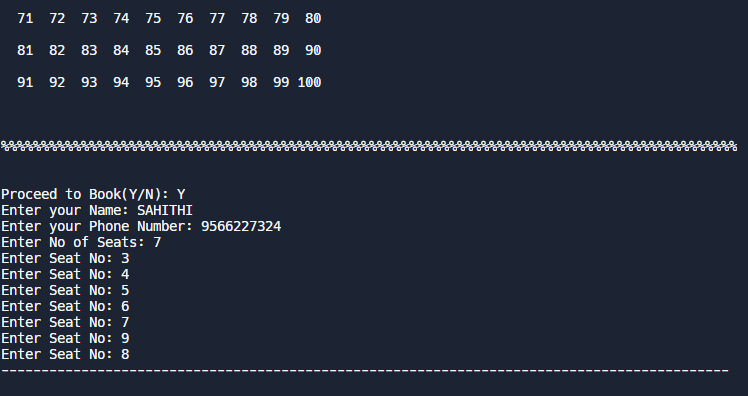
**Flow Diagram:**

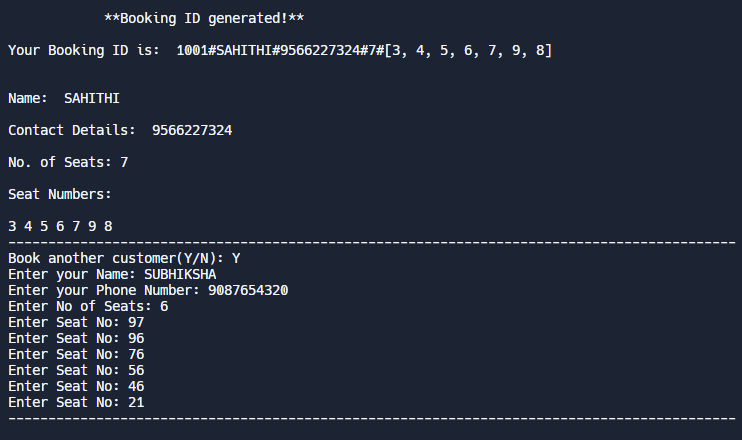


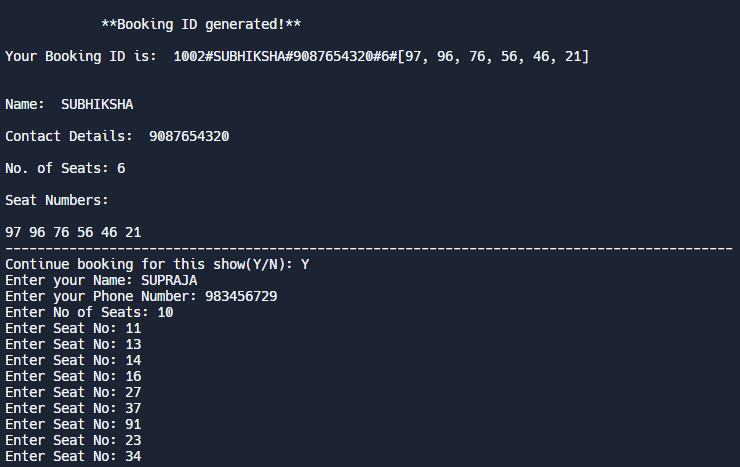
# Output:

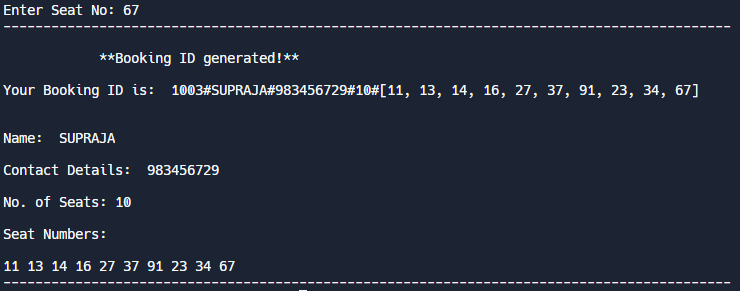


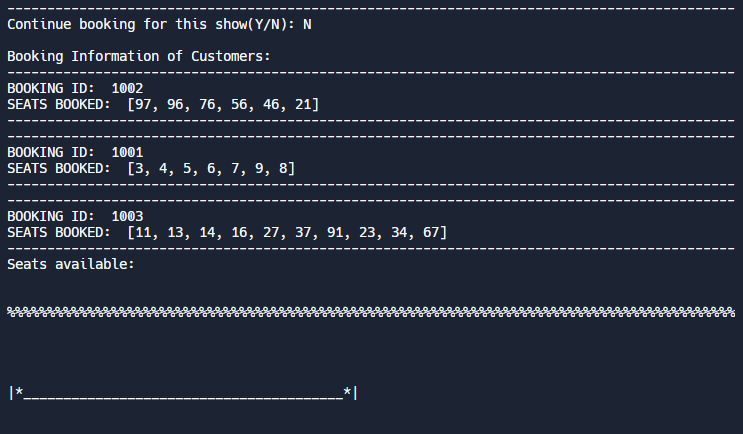


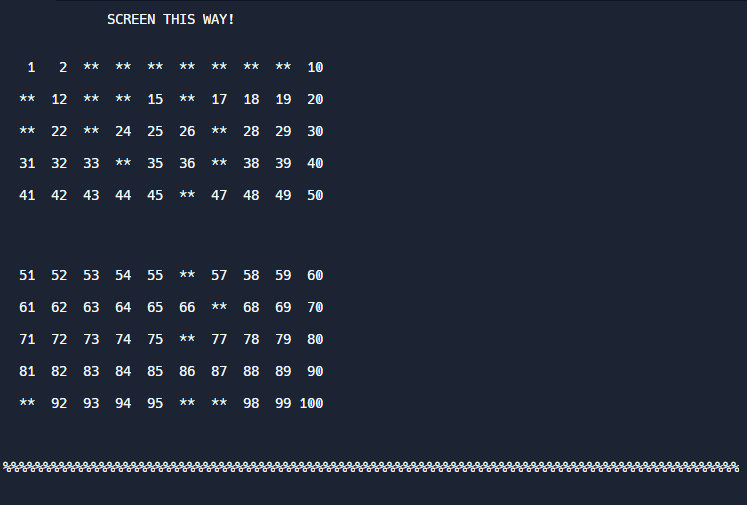


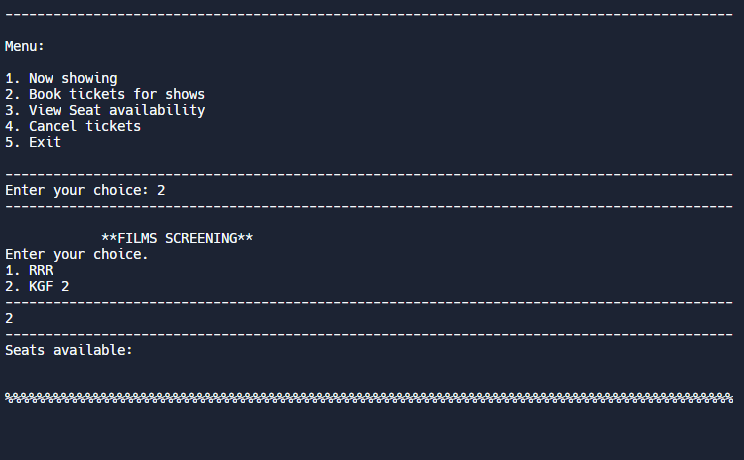


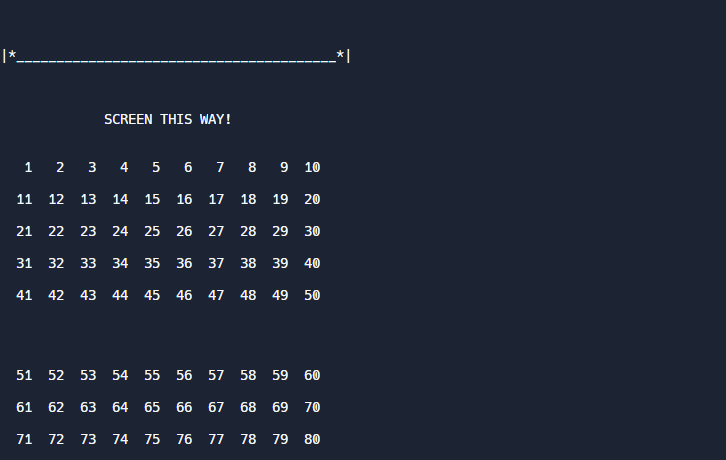


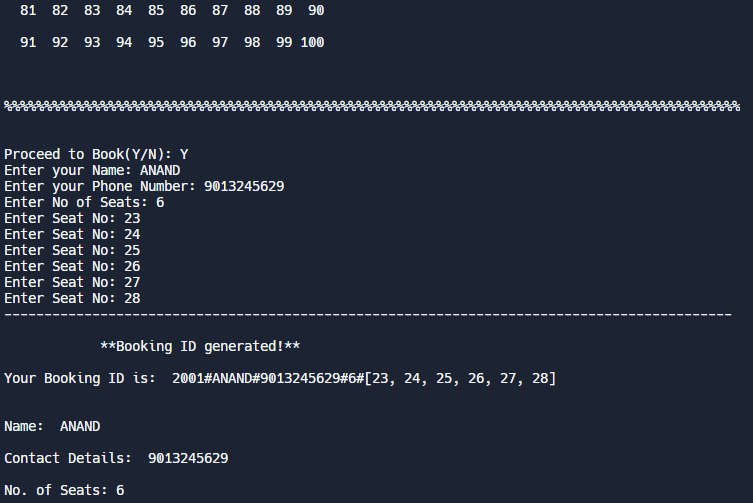


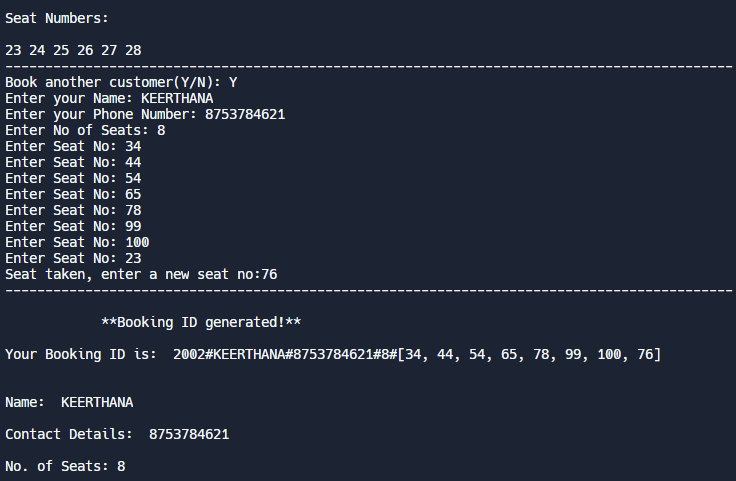


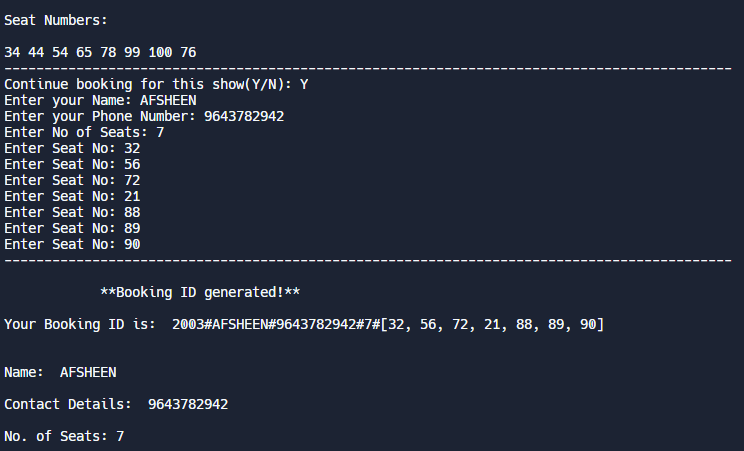


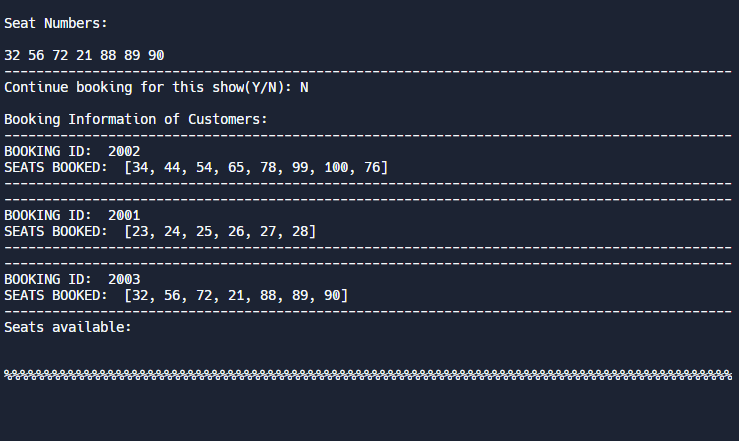


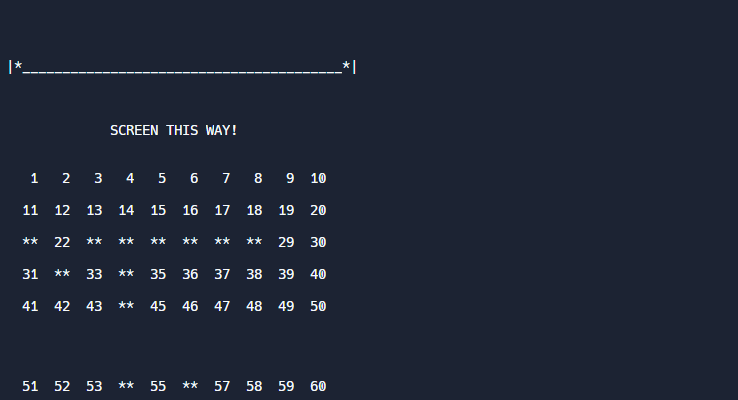


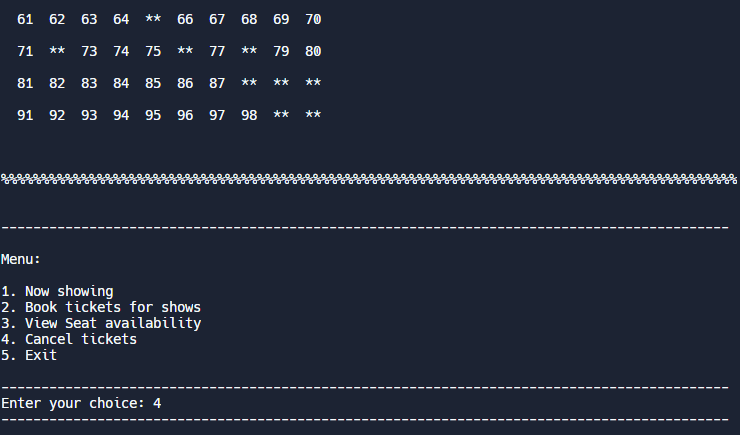


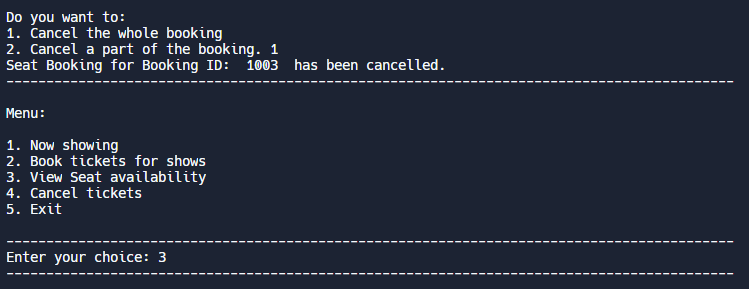
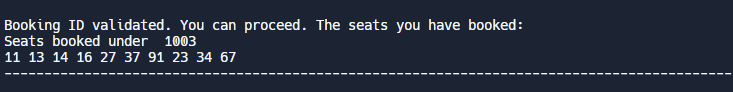
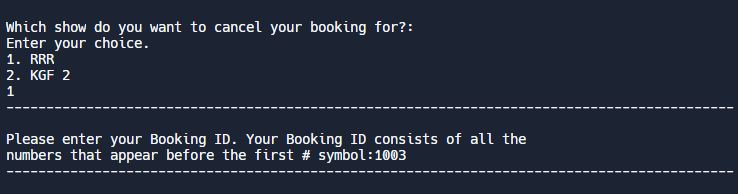


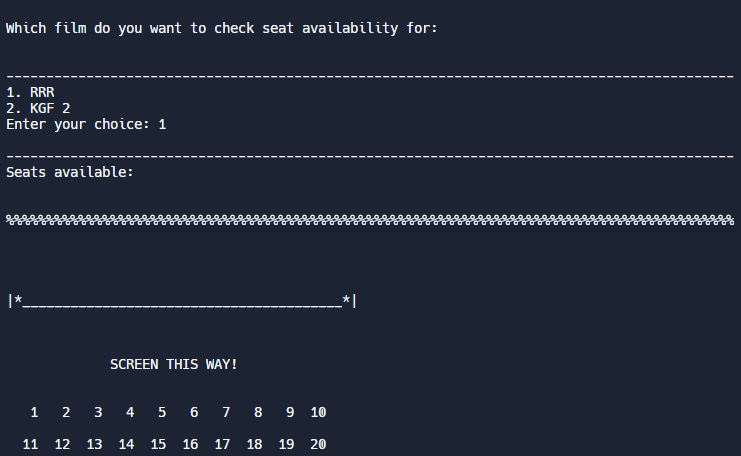


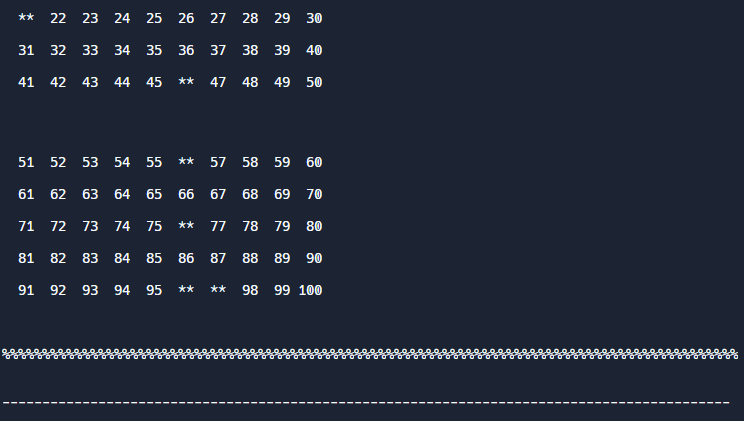


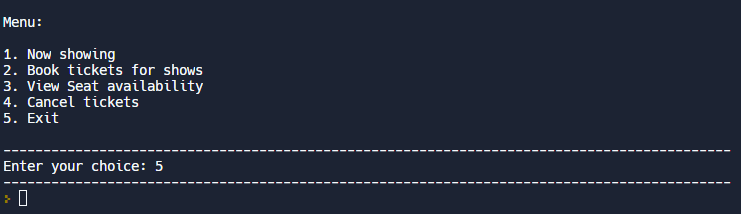


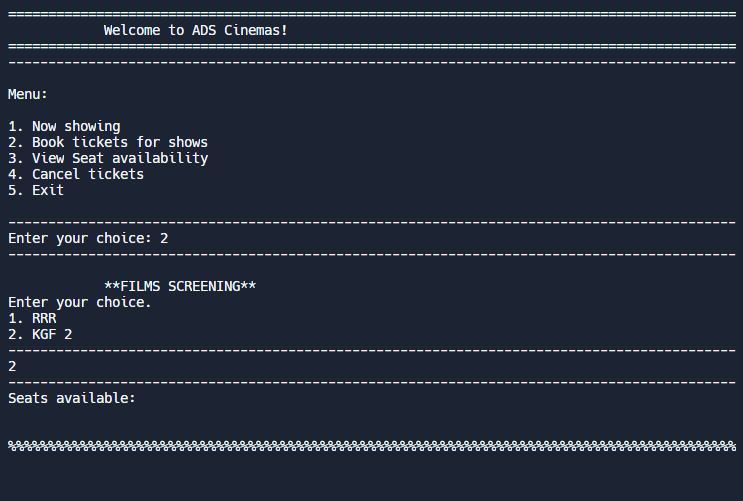


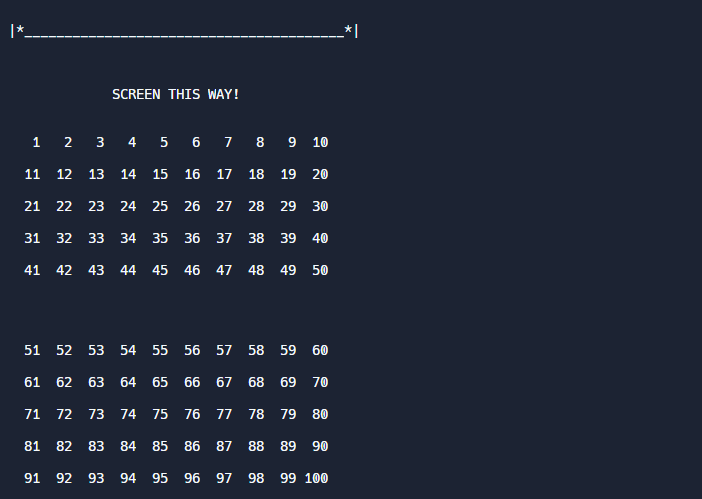


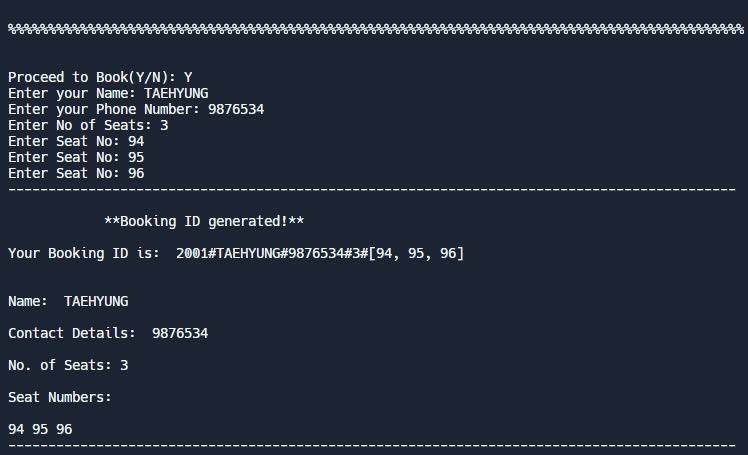


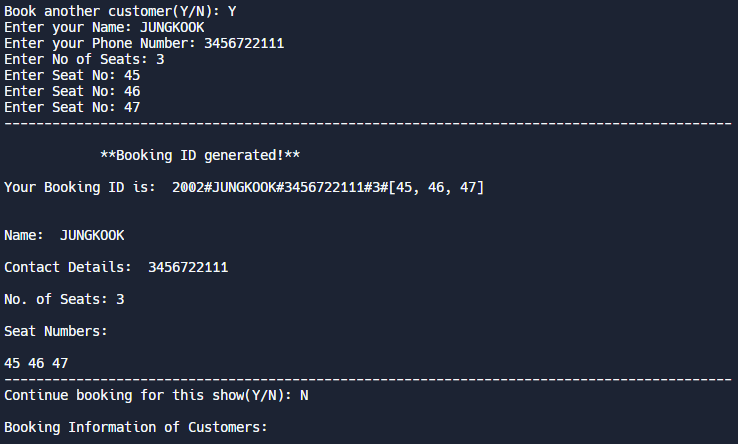


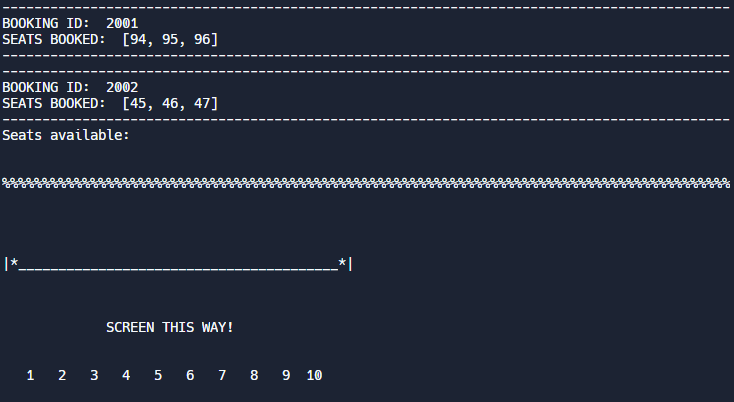


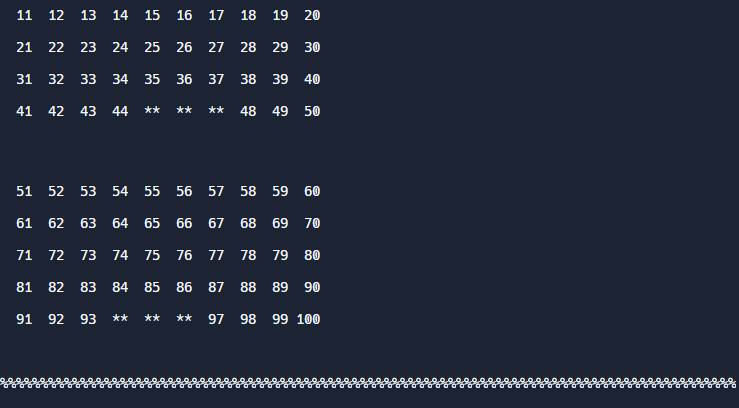


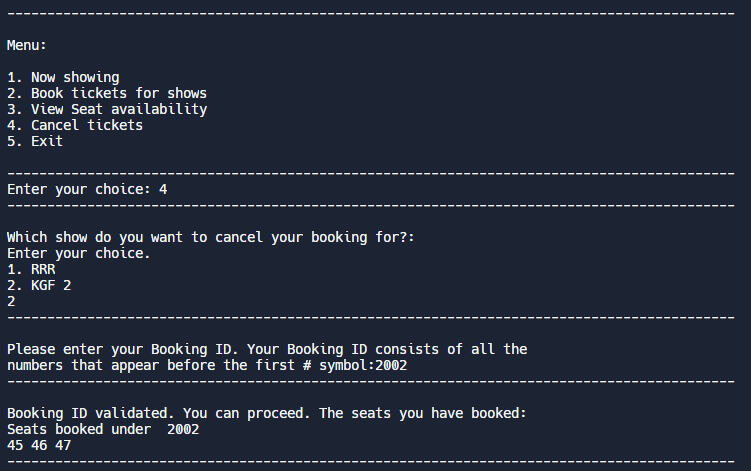


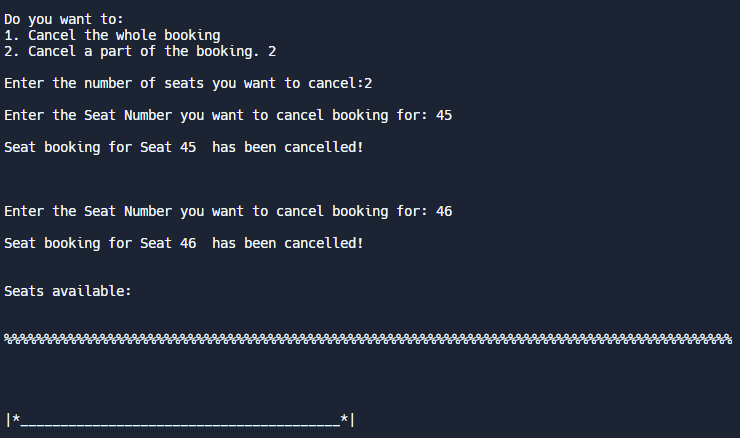


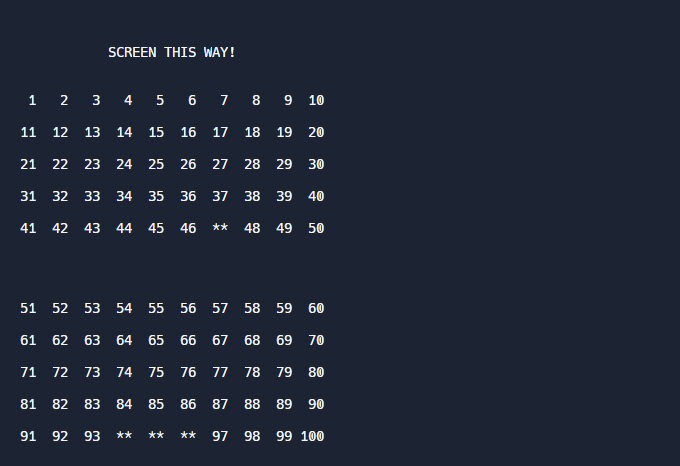


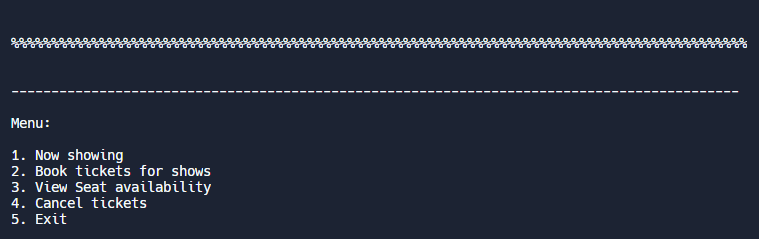


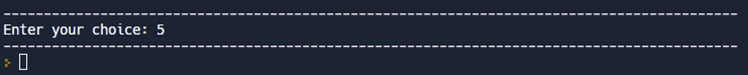












**Result:**

Thus the implementation of a ticket booking system using an AVL Tree has been completed successfully.

# Inference:

The AVL tree and other self-balancing search trees like Red Black are useful to get all basic operations done in O(log n) time. The AVL trees are more balanced compared to Red-Black Trees, but they may cause more rotations during insertion and deletion. If an application requires many frequent insertions and deletions, then Red Black trees should be preferred. But in the case of this particular application, the insertions and deletions are less frequent and search is the more frequent operation and hence AVL tree has been preferred over Red Black Tree.