Assignment - 7

Shrikrushna S Zirape
21286.

Problem Statement:-

The ticket booking system of Cinemax theater has to be implimented using C++ program. There are 10 rows & 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free seat of rows. Assume some random booking to start with. Use array to store pointers to each row on demand.

a) The list of available seat is to be displayed.
b) The seat are booked.
c) The booking can be canceled.

Objective:-
① To understand use of linked list in C++

Outcome:-
① To impliment ticket booking system using linked list in C++.
② To write menu driven program in C++.
③ To impliment user defined function in C++.

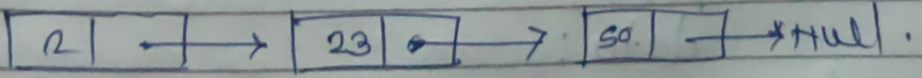
Software req:- and Hardware req!-

Operating system (64 bit) Fedora17 programming tool. latest open-source update of Ecllipse programming Framework.

Theory:-

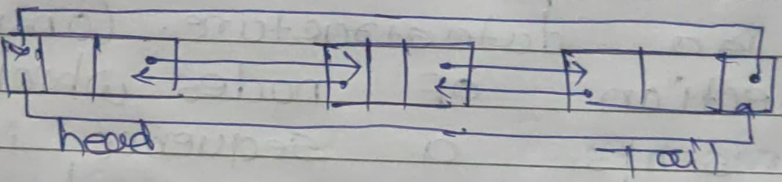
linked list:-

A linked list is a linear Collection of data elements. where order is not given by their physical placement is necessary. Instead each element. points to the next.
It is a datastructure. Consisting of Collection of nodes which together represents a sequence in it's most basic forms, each node Consist of data & ref i.e. link to the next node in the sequence. this structure allows for efficient insertion or removal of elements from any position in the sequence during Intreation.



Doubly Circular linked list!-

~~It~~ It is a type of linked list in which each node apart from storing its data. From has two links. The first link points to the previous node in the list & the second link points to the next node in the list. the first node of the list has its previous link pointing to last node. similerly the last node of the list has it is next node pointing to first node.



Algorithm!-

```
class. node {
    Public :
        next;
        prev;
        data;
}

class Doubly Circular linked list {
    public:
        node first node;
        node last node;
```

① Traversing a list:-

Forwards:-
```
node := list. firstnode.
while (node.next != list.lastnode)
<operate>
 node := node.next.
```

Backward.
```
node := list. lastnode
while (node. prev != list. firstnode)
    <operate>
 node := node. prev.
```

② Inserting a node:-

1) Algorithm insertion (list lst, node node, node, newnode)

```
1  Begin
2      newnode.prev := node
3      node.next    := node.next.
4      node.prev    := newnode.
5      node.next.   := newnode
6  End
```

2) Algorithm Remove (list lst, nodenode)
```
1      Begin
2          if node.next == node
3              list.lastnode := null
4          else.
5           node.next.prev := node.prev.
6           node.prev.next := node.next.
```

```
8    if node == list.lastnode
9         list.node := node.prev.
10       destroy node
11   end
```

Time Complexity :-

Traversing → O(n)
inserting → O(1)
Delete → O(1)

Test Case	Description.	Exep O/P	Actual O/P	Status
1	1. Display Seat 2. Book Seat 3. unbook Seat.	→ 2 row → 1 column → 1	Booked success. Fully	Booked success fully	pass
2		→ 3 row → 1 Column 1	Seat unbooked successfully	Seat unbooked successfully	pass

Conclusion :-
Successfully implimented the ticket booking system using linked list.