Assignment no	**7**
Aim	**7 A)** The ticket booking system of Cinemax theatre has to be implemented using C++ program. There are 10 rows and 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Head pointer) to each row. On demand a) The list of available seats is to be displayed b) The seats are to be booked c) The booking can be cancelled. **OR** **7 B)** Write C++ program for storing binary number using doubly linked lists. Write functions- a) To compute 1's and 2's complement b) Add two binary numbers
Objective	1. To understand concept of OOP. 2. To understand DLL, CDLL data structure and its class structures DLL in C++. 3. To understand primitive operations of DLL, CDLL.
Outcome	1. To implement DLL, CDLL and its primitive operations in C++ 2. To use DLL and DCLL data structure in applications.
OS/Programming tools used	(64-Bit) 64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit) Eclipse with Python plugin

7 A) The ticket booking system of Cinemax theatre has to be implemented using C++ program. There are 10 rows and 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Head pointer) to each row. On demand a) The list of available seats is to be displayed b) The seats are to be booked c) The booking can be cancelled.

Doubly Linked List

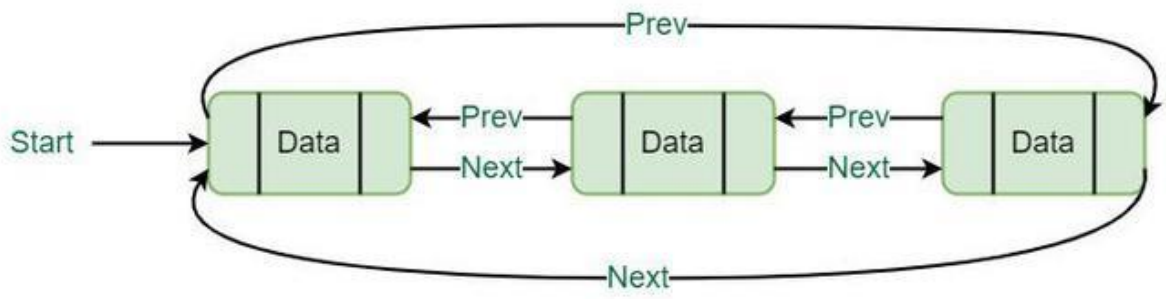
Doubly Linked List is a variation of Linked list in which navigation is possible in both ways either forward or backward easily as compared to Single Linked List. Following are important terms to understand the concepts of doubly Linked List:

Link − Each Link of a linked list can store a data called an element.

Next − Each Link of a linked list contain a link to next link called Next.

Prev − Each Link of a linked list contain a link to previous link called Prev.

Circular LinkedList − A LinkedList contains the connection link to the first Link called First and to the last link called Last.



Algorithm:

I. Book seat

1. Start

2. Create header array to store starting address of 10 rows which represented by circular doubly linked list.

3. Take input form the user to book the seat.

4. Transverse to particular row (CDLL) using corresponding header node and use seat number to reach to particular node in the CDLL.

5. Check the data in selected node whether the seat is already booked

5. Then the return seat already booked.

6. Else change the data to “X” (booked).

7. Display the all 10 rows to see the status

8. Stop.

II. Cancelling Booking

1. Start.

2. Take the input of the seat numbers to be cancelled.

3. Transverse to particular row and the node we are looking for

4. Check the data if it is “__” (not booked) return the seat was not yet booked.

5. Else change the data to “__”.

6. Display the updated rows.

7. Stop.

III. Printing the book status.

1. Start
2. Point a temporary pointer to the start of the row using corresponding header node.
3. Iterate through the list until the next pointer of this node is not equal to the header node.
4. Print the data for the given nodes.
5. Stop.

Pseudocode:

1. Booking the seat

```
Algorithm bookseat (row, seatno){
Temp:= header[row]
Count :=0
While(count<seatno)
   Temp:=temp->next
   Count:=count+1
If (temp->data == “X”}
Throw exception “Already Booked”,
}
Else {
Temp->data=’X’ //mark as booked
}
}
```

2. Cancel the seat booking

```
Algorithm bookseat (row, seatno){
Temp:= header[row]
Count :=0
While(count<seatno)
   Temp:=temp->next
   Count:=count+1
```

```
If (temp->data == " __"){
throw exception "seat not yet booked"
}
else {
temp->data='__
}
```

Conclusion: We got to know and understand doubly circular linked list and implemented it for booking by accessing and editing particular node.

7 B) Write C++ program for storing binary number using doubly linked lists. Write functions- a) To compute 1's and 2's complement b) Add two binary numbers

1's and 2's complement of a Binary Number

Given a Binary Number as a string, print its 1's and 2's complements.

1's complement of a binary number is another binary number obtained by toggling all bits in it, i.e., transforming the 0 bit to 1 and the 1 bit to 0.In the 1's complement format , the positive numbers remain unchanged . The negative numbers are obtained by taking the 1's complement of positive counterparts.

for example +9 will be represented as 00001001 in eight-bit notation and -9 will be represented as 11110110, which is the 1's complement of 00001001.

Examples:

1's complement of "0111" is "1000"

1's complement of "1100" is "0011"

2's complement of a binary number is 1, added to the 1's complement of the binary number. In the 2's complement representation of binary numbers, the MSB represents the sign with a '0' used for plus sign and a '1' used for a minus sign. the remaining bits are used for representing magnitude. Positive magnitudes are represented in the same way as in the case of sign-bit or 1's complement representation. Negative magnitudes are represented by the 2's complement of their positive counterparts.

Examples:

2's complement of "0111" is “1001"

2's complement of "1100" is “0100"

Another trick to finding two’s complement:

Step 1: Start from the Least Significant Bit and traverse left until you find a 1. Until you find 1, the bits stay the same

Step 2: Once you have found 1, let the 1 as it is, and now

Step 3: Flip all the bits left into the 1.

Algorithm:

A. Prepare Binary (n)

1. Start
2. Assign new node (no/02) to start and last.
3. Assign n/2 to n.
4. While n is greater than 0.
    a. Make node pointer new node = Node (No / 02)
    b. Declare start → prev = new node
    c. New node → next = start
    d. Start = start → prev.
    e. Assign n/2 to n.
5. Repeat step 4
6. End

B. Print Function

1. Start
2. Assign Start to pointer loop.
3. While temp is not equal to NULL.
    a. Print the data at temp.
    b. Assign temp → next to temp.
4. End

C. One’s Complement

1. Start
2. Declare a node printer ‘temp’ = start
3. While temp is not equal to NULL.
    a. If data at temp is ‘0’ change it to ‘1’.
    b. Else change it to ‘1’.
    c. Assign temp → next to temp.
4. Repeat step 3.
5. End.

D. Two’s Complement

1. Start
2. Call Ones Complement function

3. Initialize integer variable ‘carry’ equal to ‘1’.
4. Initialize node pointer ‘temp’ equal to lost.
5. While temp not equal to NULL
    a. If temp → data = 1 and carry = 1, change data to temp equal to zero and carry equal to ‘1’.
    b. Elseif temp → data = 0 and carry = 1 , change data temp equal to 1 and carry to 0.
    c. If carry = 0, break
6. Assign temp → prev to temp.
7. Repeat step 5.
8. End.

E. “+” operator overloading [Binary operator + (Binary b2)]

1. Start
2. Declare binary ‘sum’ object.
3. Initialize node pointer a to last and b to b2lat and initialize carry equal to “0”.
4. While a is not equal to NULL and b is not equal to NULL.
    a. Initialize additional variable as:
        (a → data) ^ (b → data) ^ (carry)
    b. Add this addition node to sum of carry own node at start (addition).
    c. Make changes in the carry in similar way.
    d. Move a to its previous node, do same for b.
5. If a is having more bites than b
    a. While (a not equal to NULL)
    b. Add (a → data) ^ Carry node in the sum.
6. If b is having max bit than ‘a’
   While (b not equal to NULL)
   Call sum node at start to add another node.
7. If carry is obtained, we will add carry using node at start carry.
8. Stop

F. Node at Start Function

1. Start.
2. Initialize node pointer ‘new node’ to new node (data).
3. If start is NULL, assign start and last equal to new node.
4. Else, connect the start of previous node with newly created node i.e. link the previous node and current new node.

CONCLUSION:

1. We learnt about user defined data types, doubly linked list.
2. We also learnt about operator overloading.