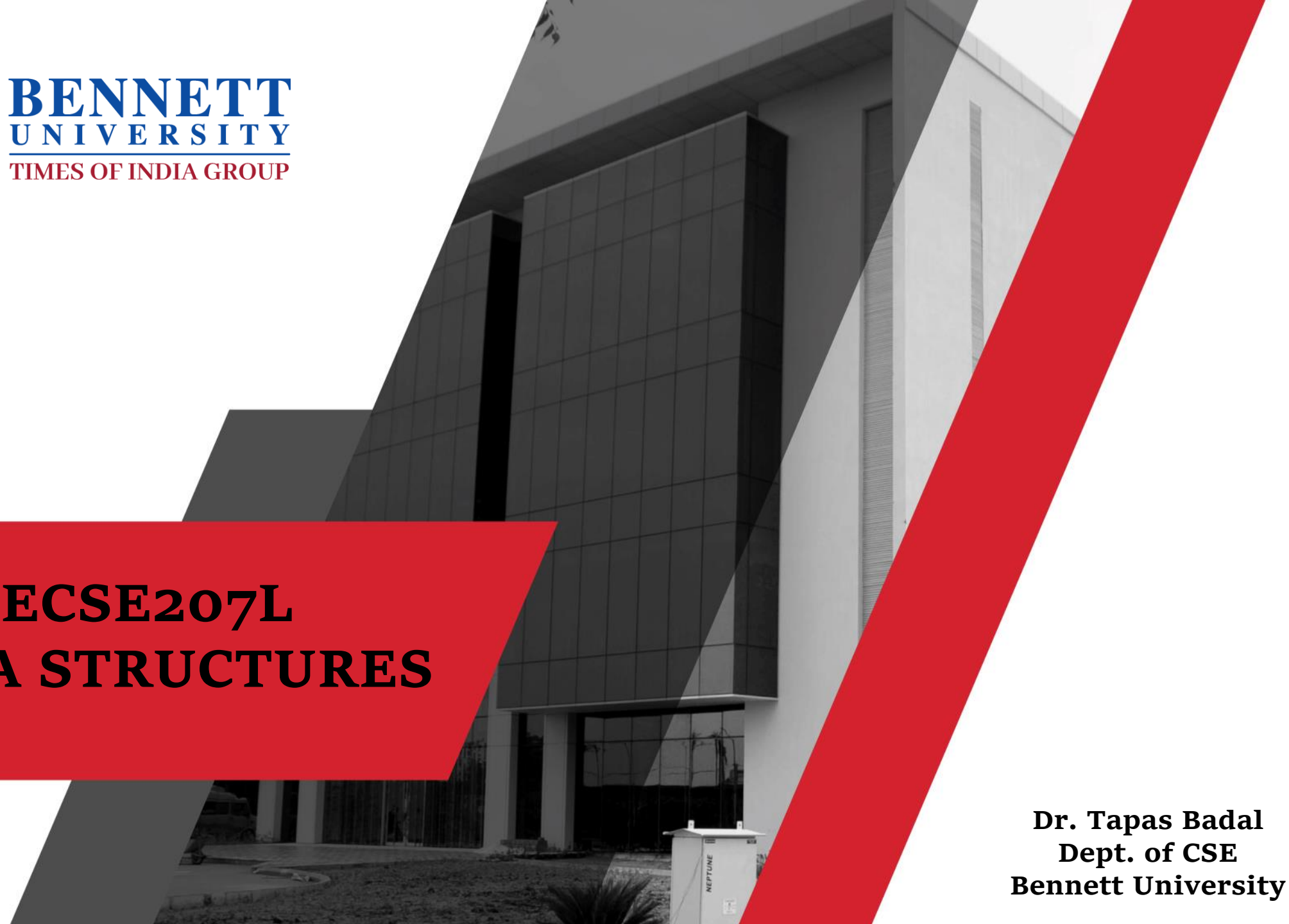
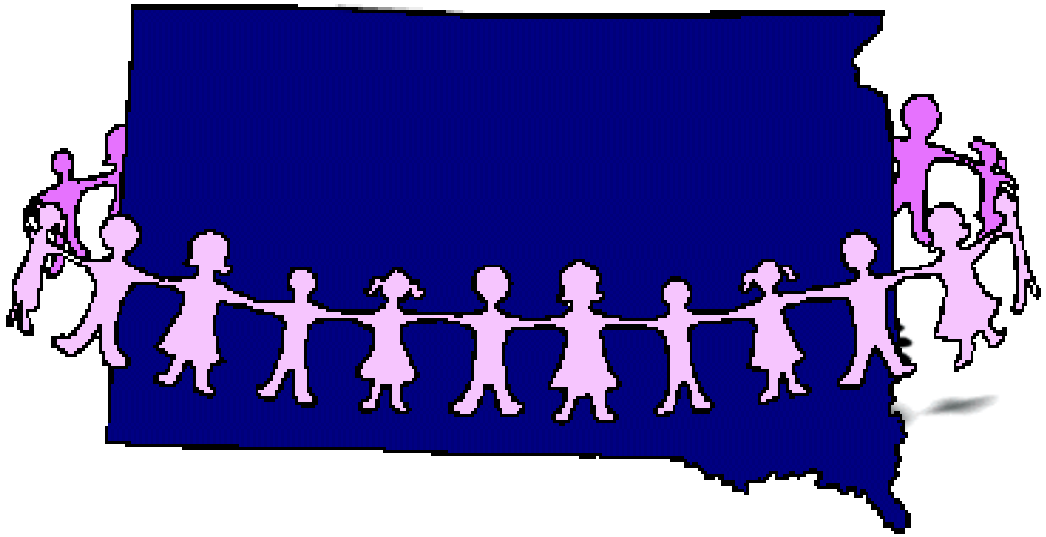
BENNETT UNIVERSITY
TIMES OF INDIA GROUP




# ECSE207L
# DATA STRUCTURES

**Dr. Tapas Badal**
**Dept. of CSE**
**Bennett University**

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

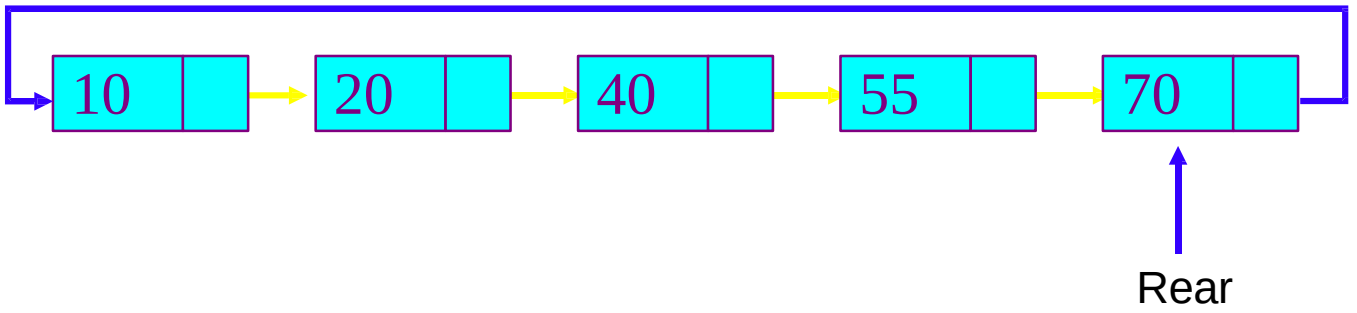


# Circular Linked List

# Circular Linked Lists

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

- A Circular Linked List is a special type of Linked List
- It supports traversing from the end of the list to the beginning by making the last node point back to the head of the list
- A Rear pointer is often used instead of a Head pointer



# Motivation



- Circular linked lists are useful for implementing queues.
- No need to keep separate front and rear references.
- Circular linked lists are useful for playing video and sound files in “looping” mode.
- Used in computers to go round and round through various applications and give a time slice to each of them.
- They are also a stepping stone to implementing graphs.

# Circular linked list structure

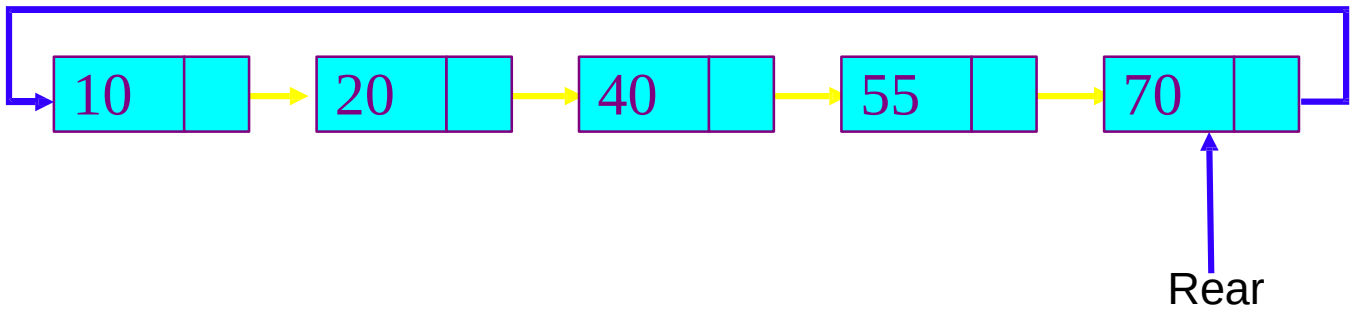


- Similar to a singly linked list except that last node (Rear) points to the first node.
- The convention is to call the list by the Rear node.
- Makes it easier, as front node is just next to the rear node.

# Traverse the list

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

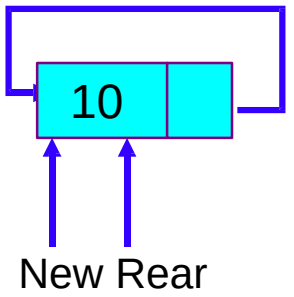
```
void print(NodePtr Rear){  NodePtr Cur;
if(Rear != NULL){
Cur = Rear.getlink();  do{
print node data;  Cur = Cur.getlink();
}while(Cur != Rear.getlink();

}
}
```



# Insert Node

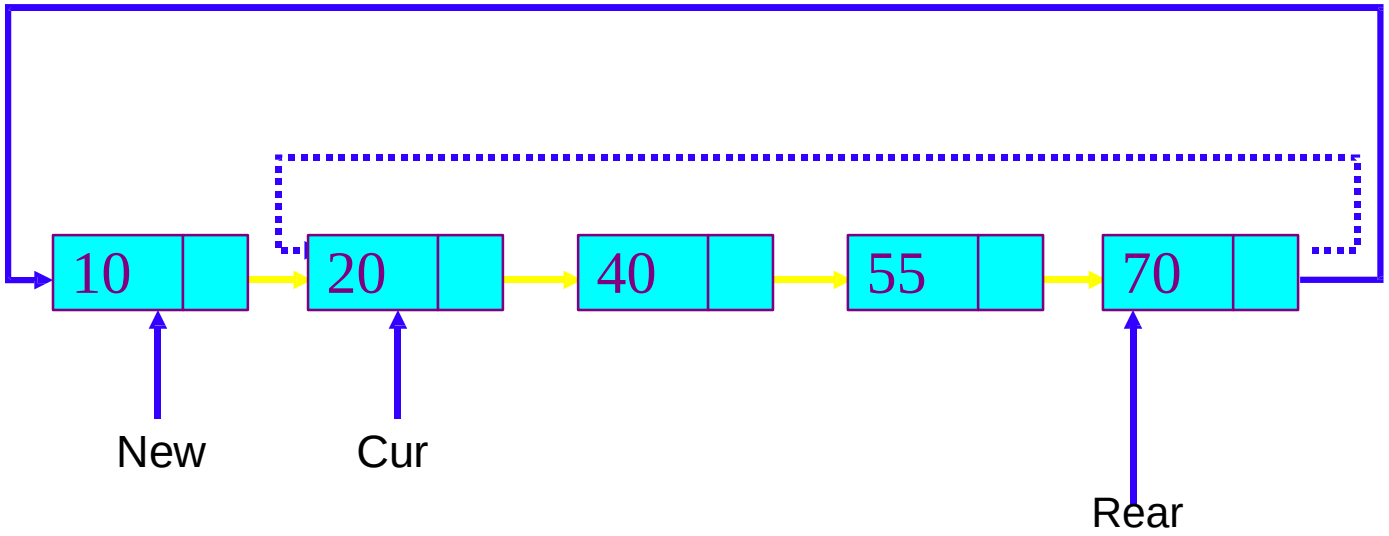
- Insert into an empty list

```
NotePtr New      =    Node;
   new
   New.setData(10);
    Rear = New;
    Rear.setlink(Rear);
```



- Insert at head of a Circular Linked List

```
New.setlink(Cur);

Rear.setlink(New);
```



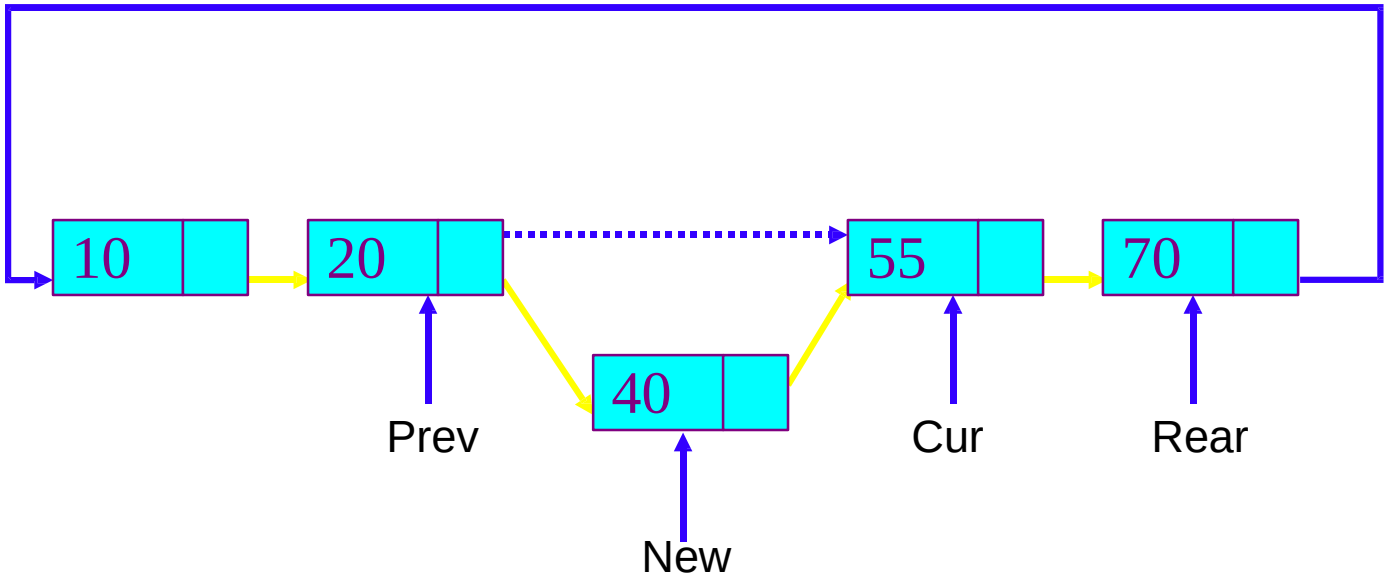
# Insert in middle of a circular list



- Insert in middle of a Circular Linked List between Pre and Cur

**New.setlink(Cur);**

**Prev.setlink(New);**

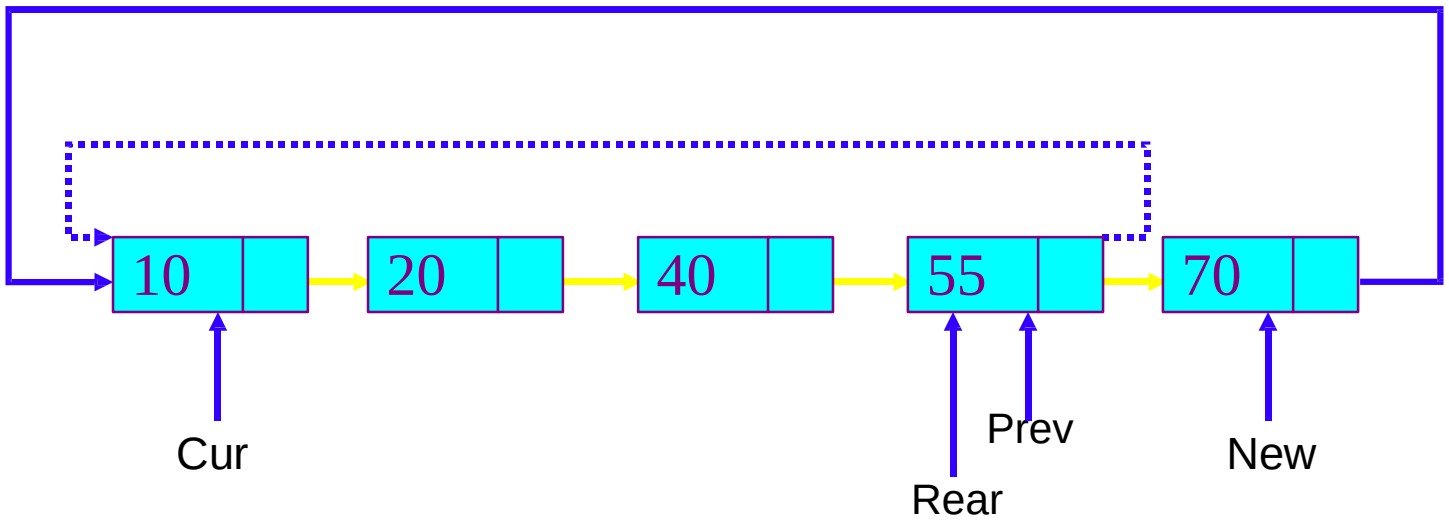


# Insert at end



- Insert at end of a Circular Linked List

```
New.setlink(Cur);   Prev.setlink(New);

   Rear = New;
```

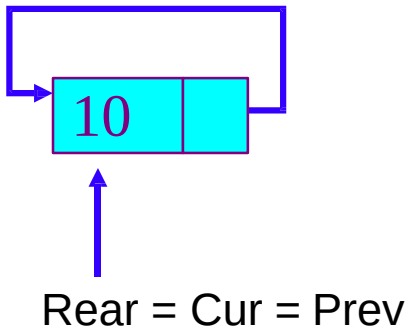


# Delete single remaining node



- Delete a node from a single-node Circular Linked List

```
Rear = NULL;
delete Cur;
```

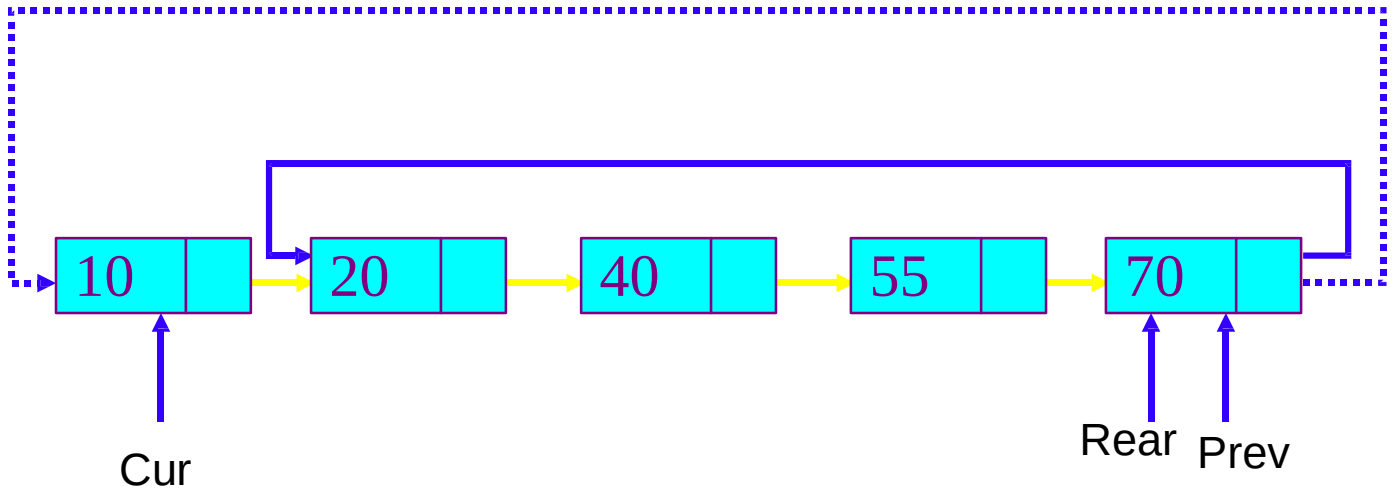


# Delete Node



- Delete the head node from a Circular Linked List

```
Prev.setlink( Cur.getlink());
```

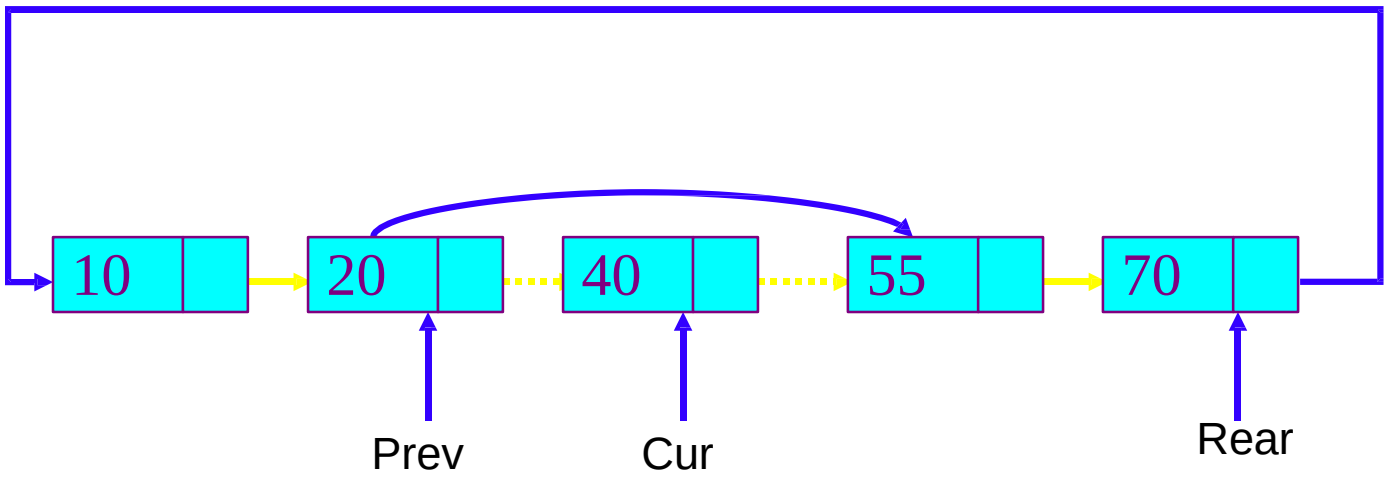


# Delete any node in middle of list



- Delete a middle node `Cur` from a Circular Linked List

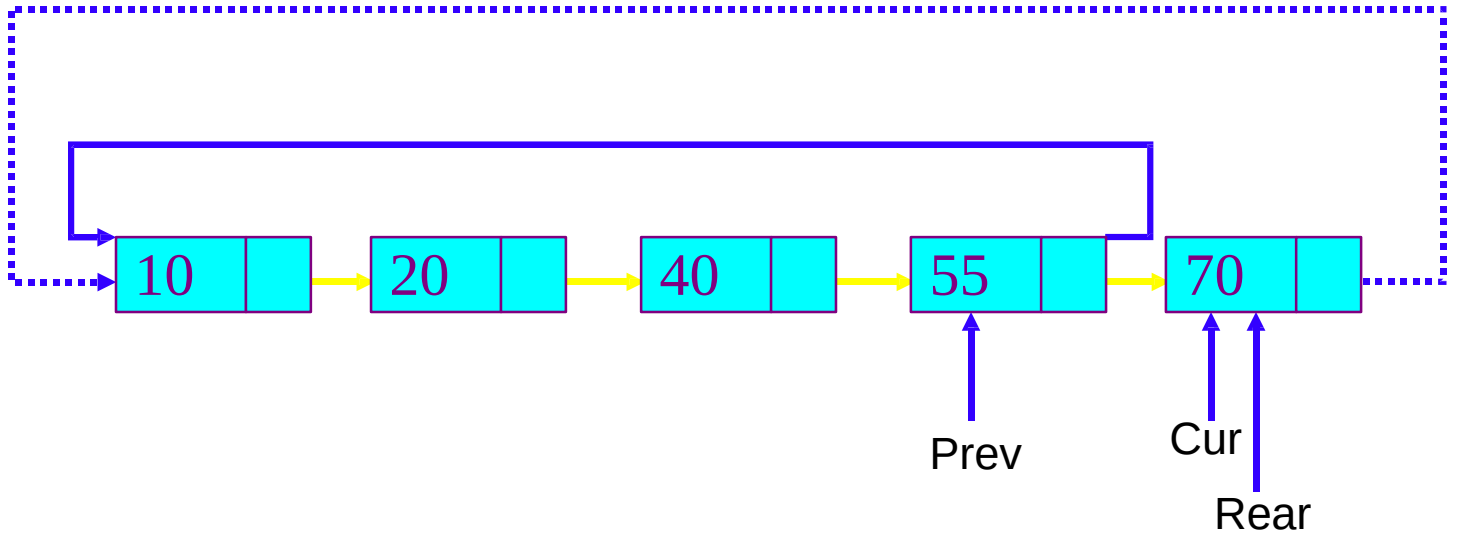
```
Prev.setlink(Cur.getlink());
```



# Delete end node in middle of list

- Delete the end node from a Circular Linked List

```
Prev->next = Cur->next;    // same as: Rear->next;
delete Cur;
Rear = Prev;
```



# Doubly Linked Lists

# Doubly Linked Lists

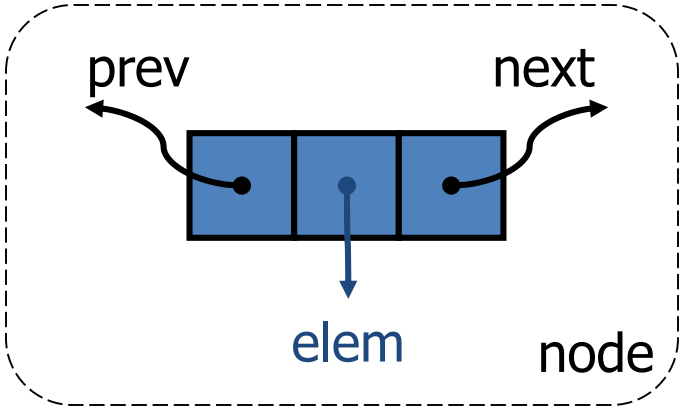
BENNETT UNIVERSITY
TIMES OF INDIA GROUP

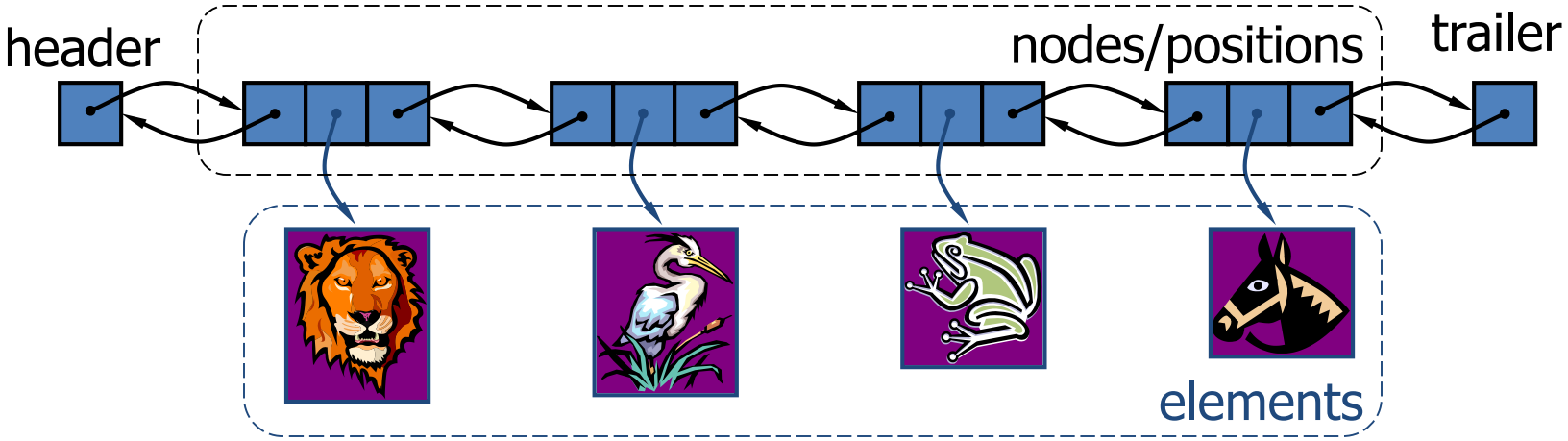
- Permits traversal of list in both directions
- Useful where navigation in both directions needed
- Used in memory management systems to keep track of allocated blocks and free blocks
- Used by browsers to navigate forwards and backwards
- Can be used to implement a graph
- Various applications use this to redo and undo functionality (games)

# Doubly Linked List



- A doubly linked list provides a natural implementation of the Node List ADT
- Nodes implement Position and store:
  - element
  - link to the previous node
  - link to the next node
- Special trailer and header nodes

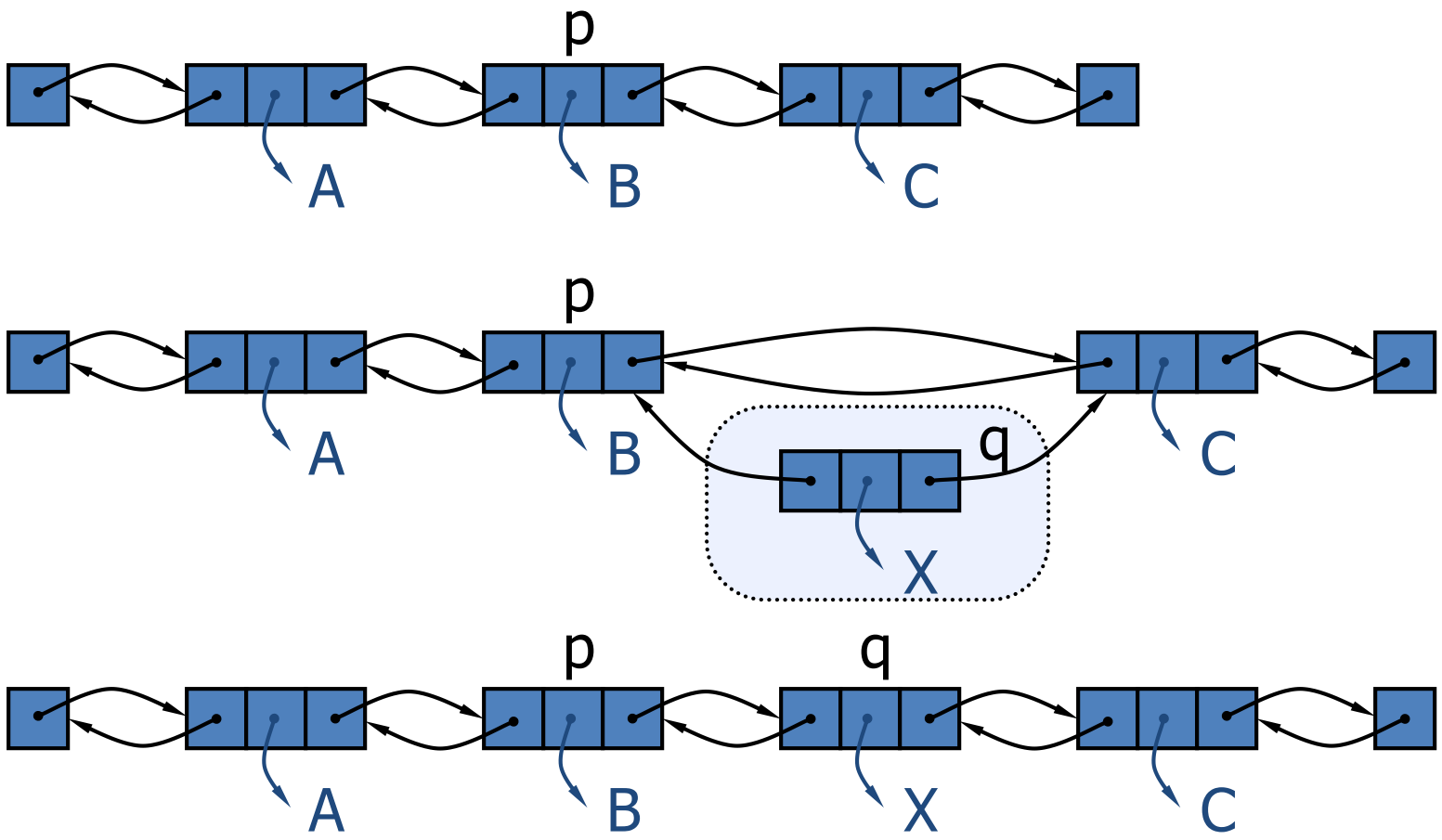


# Insertion



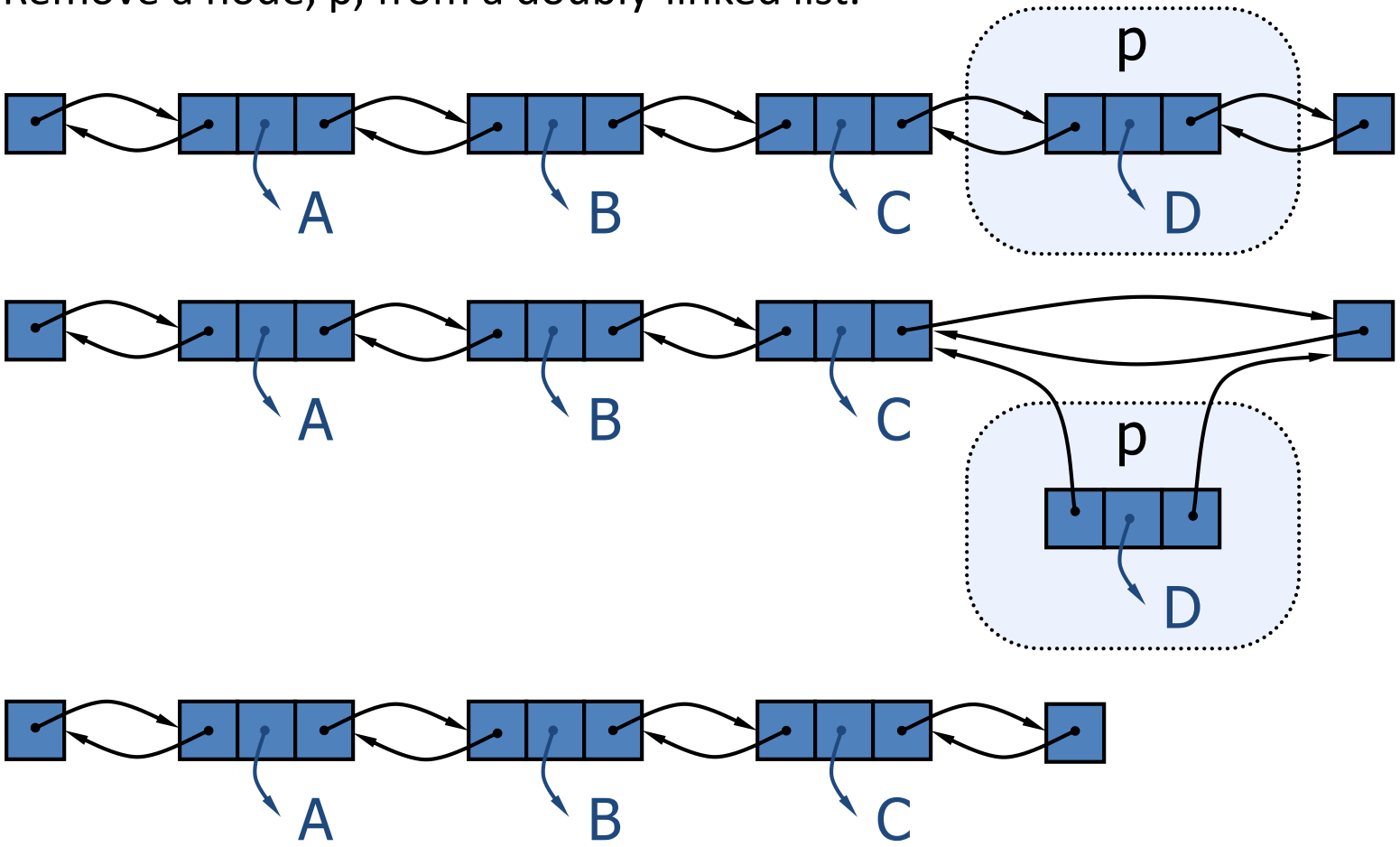
- Insert a new node, q, between p and its successor.



# Deletion



- Remove a node, p, from a doubly-linked list.



# Doubly Linked List structure



- Each node has
  - A data field
  - A left link
  - A right link
- Thus each node has two links, one pointing to previous node and one pointing to next node in the list.
- The first node is named Front
- The last node is named Rear
- The left link of Front is set to NULL
- The right  link of Rear is set to NULL

```
public class Node {
        public int data;
        public Node left;
         public Node right;


         Node(int n, Node p, Node q) {
                data = n;
                left = p;
                right = q;
         }
 };
```

# ADT of Doubly Linked List

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

- Print the list from first to last node
- Print the list backwards
- Insert a new element between cur and next nodes
- Delete the cur node
- Insert a new node in empty list
- Delete the first node
- Delete the last node

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

# THANKYOU

@csebennett

cse_bennett