Data Structures (15B11CI311)

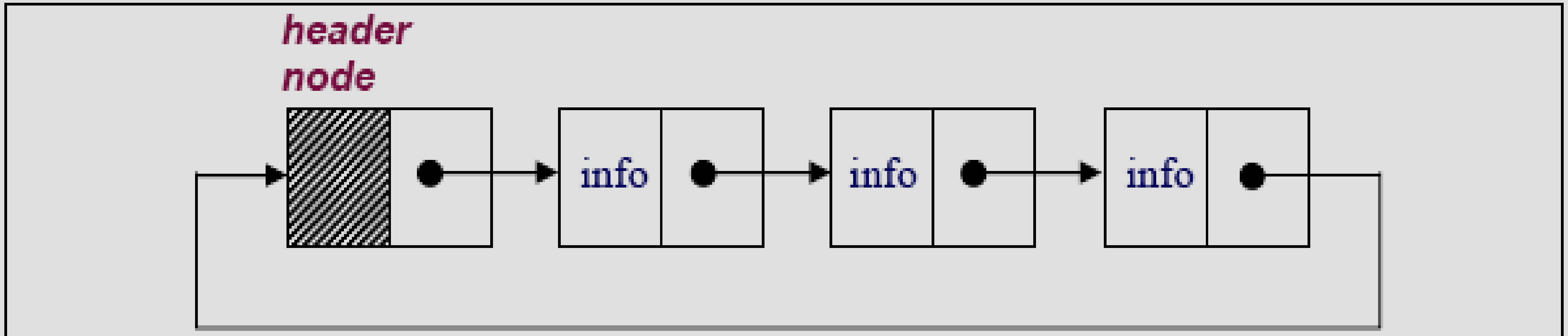
Odd Semester 2021

3rd Semester , Computer Science and Engineering

Jaypee Institute Of Information Technology (JIIT), Noida

Circular Linked Lists

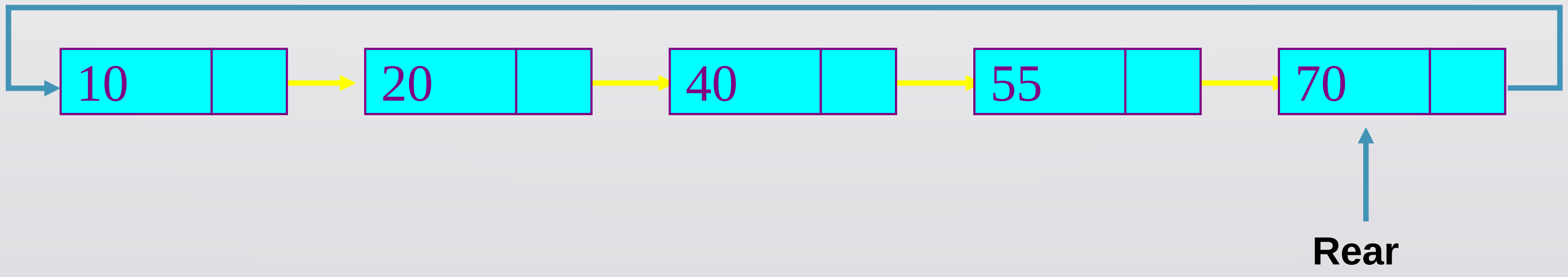
- Circular linked lists can be used to help the traverse the same list again and again if needed. A circular list is very similar to the linear list where in the circular list the pointer of the last node points not NULL but the first node.



a *circular* linked list with a *header node*

Circular Linked Lists Contd….

- Circular Linked List supports traversing from the end of the list to the beginning by making the last node point back to the head of the list.
- **Motivation**-Circular linked lists are useful for playing video and sound files in "looping" mode.
- A Rear pointer is often used instead of a Head pointer.



Circular Linked List Definition

```
#include <iostream>

using namespace std;

struct Node{

  int data;

  Node* next;

};

typedef Node* NodePtr;
```

Circular Linked List Operations

- `Traversing the list`

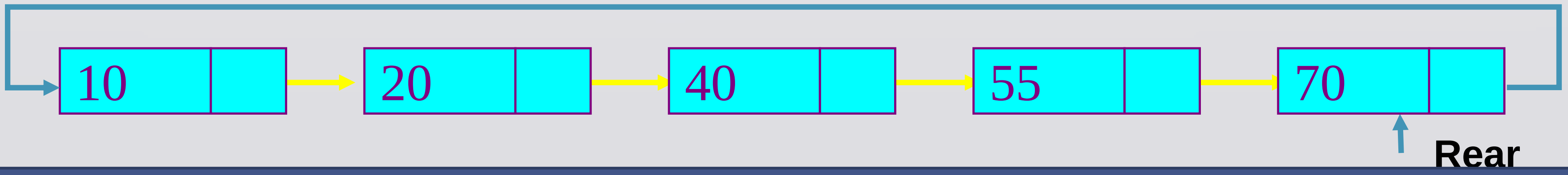
-
  ```
  insertNode(NodePtr& Rear, int item)
  //add new node to ordered circular linked list
  ```

-
  ```
  deleteNode(NodePtr& Rear, int item)
  //remove a node from circular linked list
  ```

-
  ```
  print(NodePtr Rear)
  //print the Circular Linked List once
  ```

Traverse the list

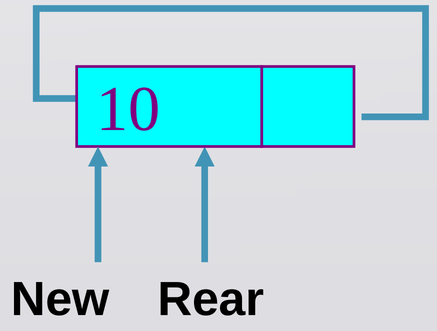
```
void print(NodePtr Rear){
  NodePtr Cur;
  if(Rear != NULL){
        Cur = Rear->next;
        do{
                cout << Cur->data << " ";
                Cur = Cur->next;
        }while(Cur != Rear->next);
        cout << endl;
  }
}
```



Insert Node

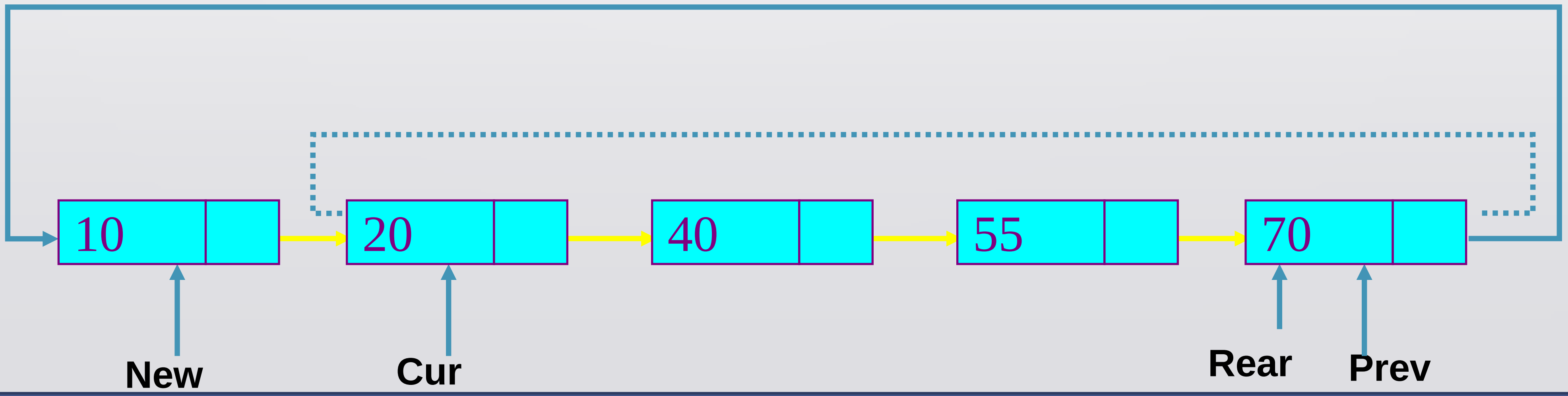
- Insert into an empty list

```
NotePtr New = new Node;
New->data = 10;

Rear = New;
Rear->next = Rear;
```



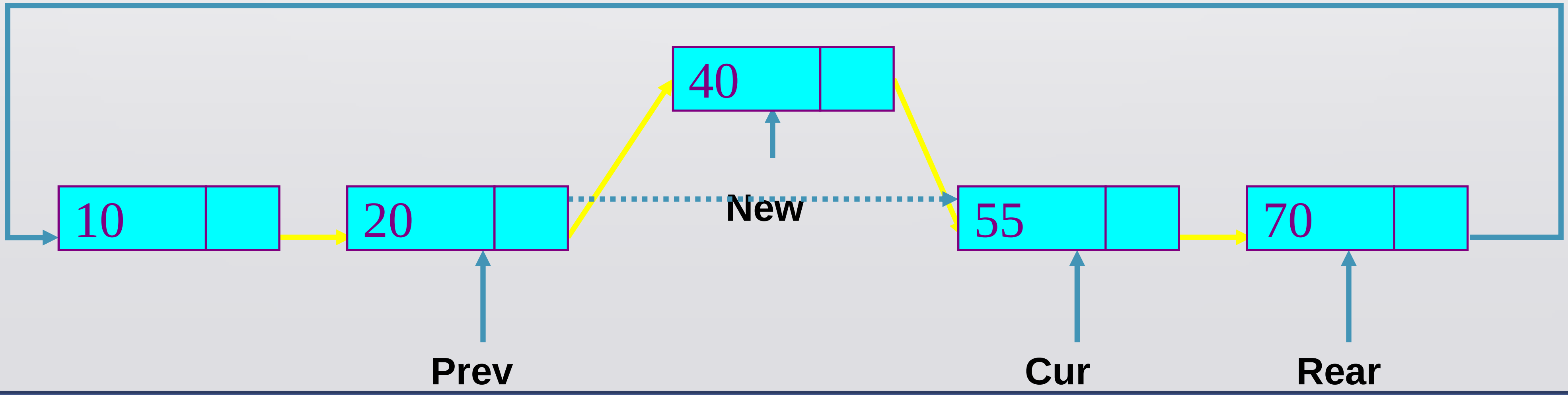
- Insert to head of a Circular Linked List

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



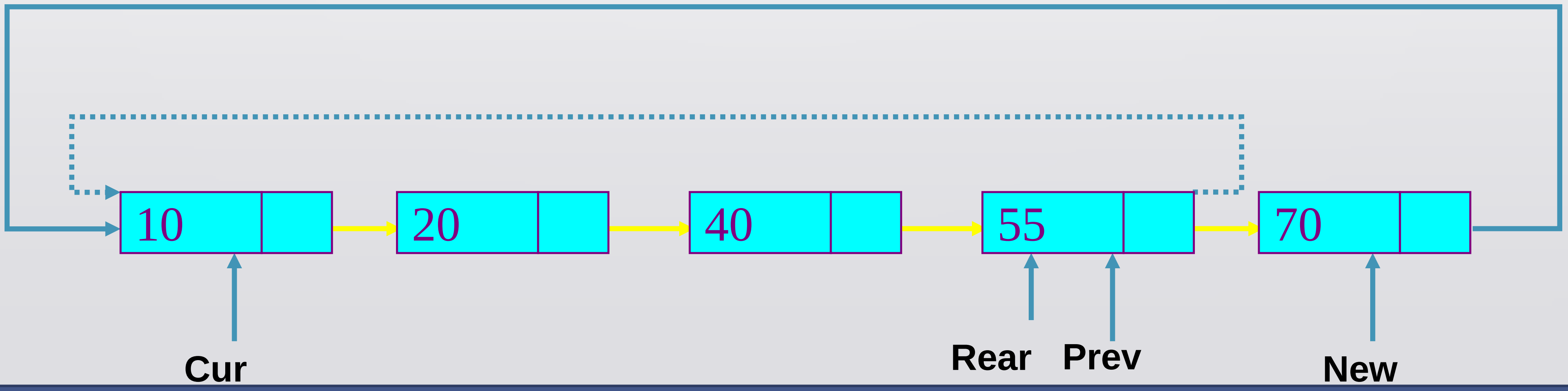
- Insert to middle of a Circular Linked List between `Pre` and `Cur`

```
New->next = Cur;

Prev->next = New;
```



- Insert to end of a Circular Linked List

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



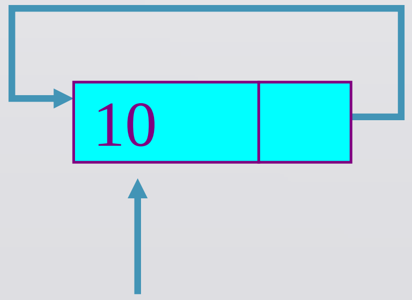
Example

```
void insertNode(NodePtr& Rear, int item){
  NodePtr  New, Cur, Prev;
  New = new Node;
  New->data = item;
  if(Rear == NULL){ // insert into empty
  list
        Rear = New;
        Rear->next = Rear;
        return;
  }
  Prev = Rear;
  Cur = Rear->next;
```

```
do{         // find Prev and Cur
        if(item <= Cur->data)
            break;
        Prev = Cur;
        Cur = Cur->next;
    }while(Cur != Rear->next);
    New->next = Cur; // revise
pointers
    Prev->next = New;
    if(item > Rear->data)
    //revise Rear pointer if adding to
end
        Rear = New;
}
```

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

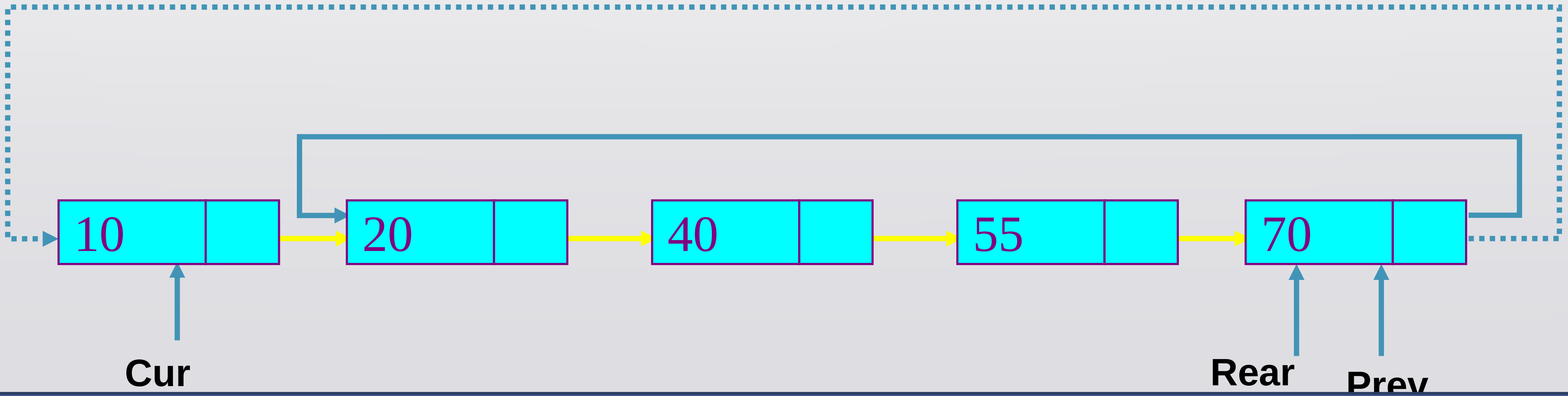
```
Rear = NULL;

delete Cur;
```



Rear = Cur = Prev

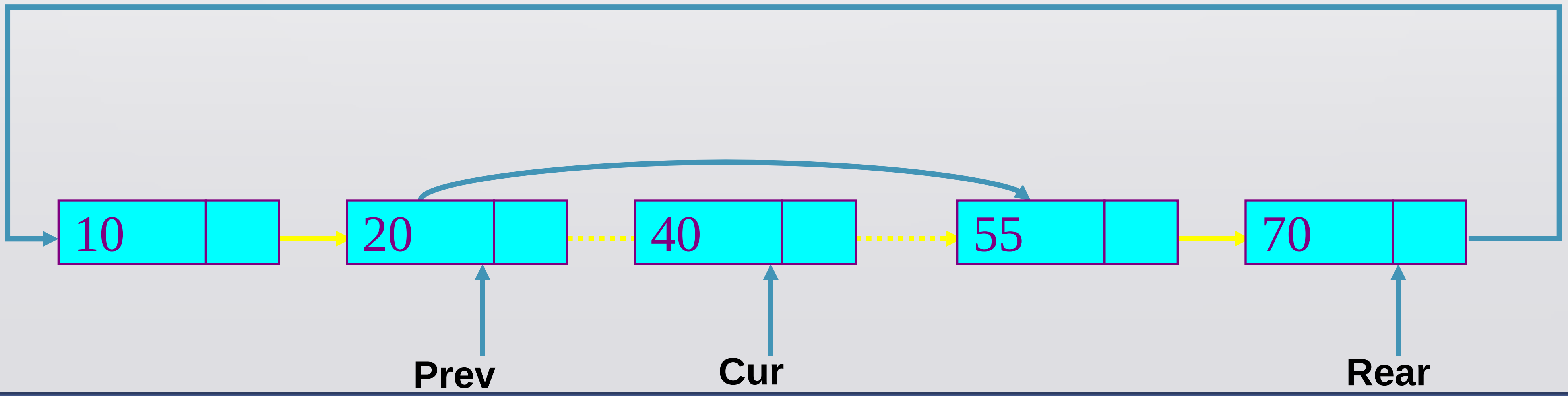
- **Delete the head node from a Circular Linked List**

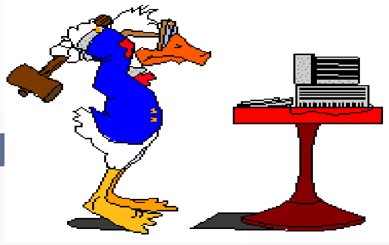
```
Prev->next = Cur->next; // same as: Rear->next = Cur->next

delete Cur;
```



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

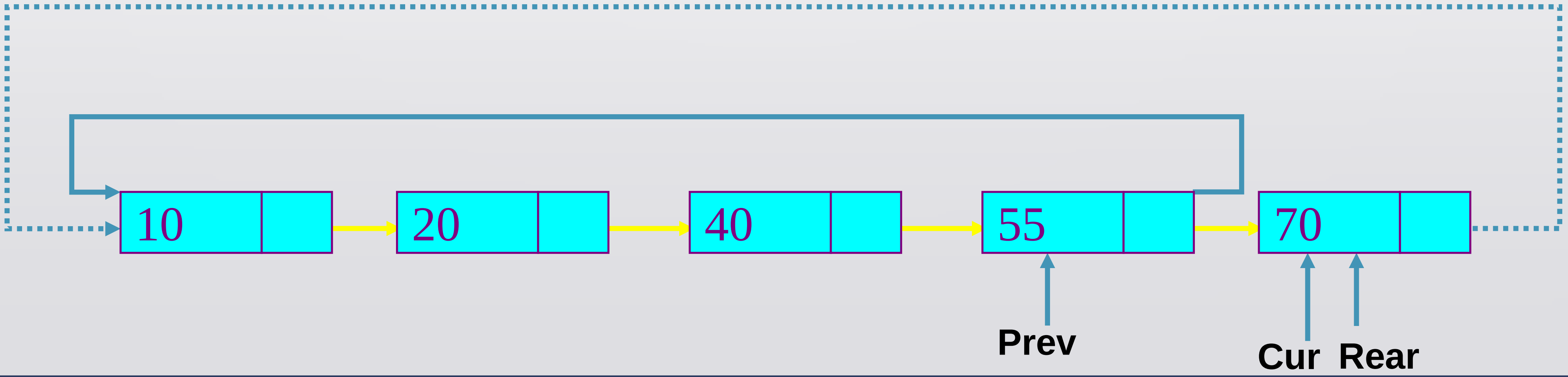
```
Prev->next = Cur->next;

delete Cur;
```





- **Delete the end node from a Circular Linked List**

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

delete Cur;

Rear = Prev;
```



Example-2

```
void deleteNode(NodePtr& Rear, int item){
  NodePtr Cur, Prev;
  if(Rear == NULL){
        cout << "Trying to delete empty list" << endl;
        return;
  }
  Prev = Rear;
  Cur = Rear->next;
  do{                          // find Prev and Cur
        if(item <= Cur->data)  break;
        Prev = Cur;
        Cur = Cur->next;
  }while(Cur != Rear->next);
  if(Cur->data != item){       // data does not exist
        cout << "Data Not Found" << endl;
        return;
  }
```

```
if(Cur == Prev){     // delete
single-node list
        Rear = NULL;
        delete Cur;
        return;
    }
    if(Cur == Rear)      // revise
Rear pointer if deleting end
        Rear = Prev;
    Prev->next = Cur->next; //
revise pointers
    delete Cur;
}
```

Example-2 Contd…

```
void main(){
 NodePtr Rear = NULL;

 insertNode(Rear, 3);
 insertNode(Rear, 1);
 insertNode(Rear, 7);
 insertNode(Rear, 5);
 insertNode(Rear, 8);
 print(Rear);
 deleteNode(Rear, 1);
 deleteNode(Rear, 3);
 deleteNode(Rear, 8);
 print(Rear);
 insertNode(Rear, 1);
 insertNode(Rear, 8);
 print(Rear);
}
```

Output is:

```
1 3 5 7 8
5 7
1 5 7 8
```

References

- https://www.hackerearth.com/practice/notes/stacks-and-queues/
- https://www.softwaretestinghelp.com/stacks-and-queues-in-stl/