DATA STRUCTURES AND ITS APPLICATIONS

Vandana M L

Department of Computer Science and Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Circular Singly Linked List

Vandana M L

Department of Computer Science and Engineering

Circular Linked List

Circular linked list is a linked list where all nodes are connected to form a circle.

- Circular Singly Linked List
- Circular Doubly Linked List

  With additional head node
  Without additional head node

DATA STRUCTURES AND ITS APPLICATIONS

Circular Linked List Operations

Insert a node

- Insert at front
- Insert at end
- Insert at a position
- Ordered insertion

Delete node

- Delete front node
- Delete end node
- Delete a node from position
- Delete a node with a given value

Additional

- Display list
- Concatenate two list
- reverse a list

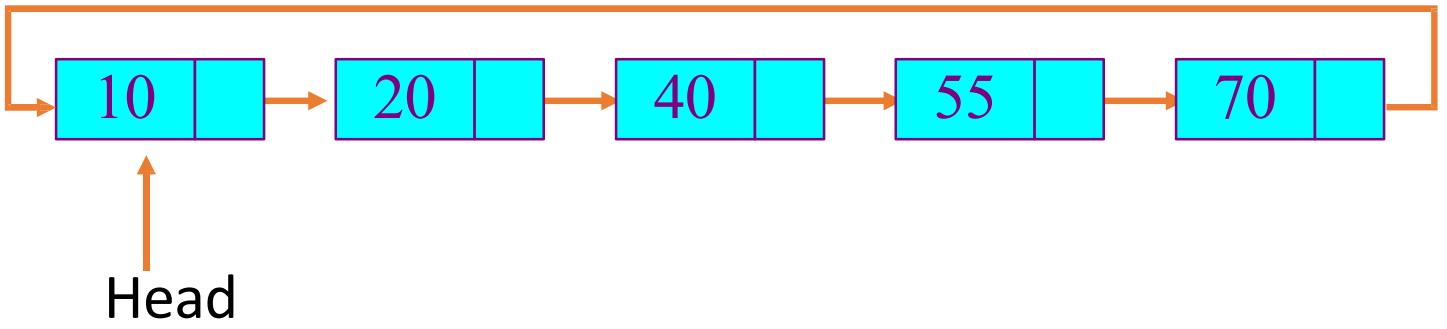
DATA STRUCTURES AND ITS APPLICATIONS

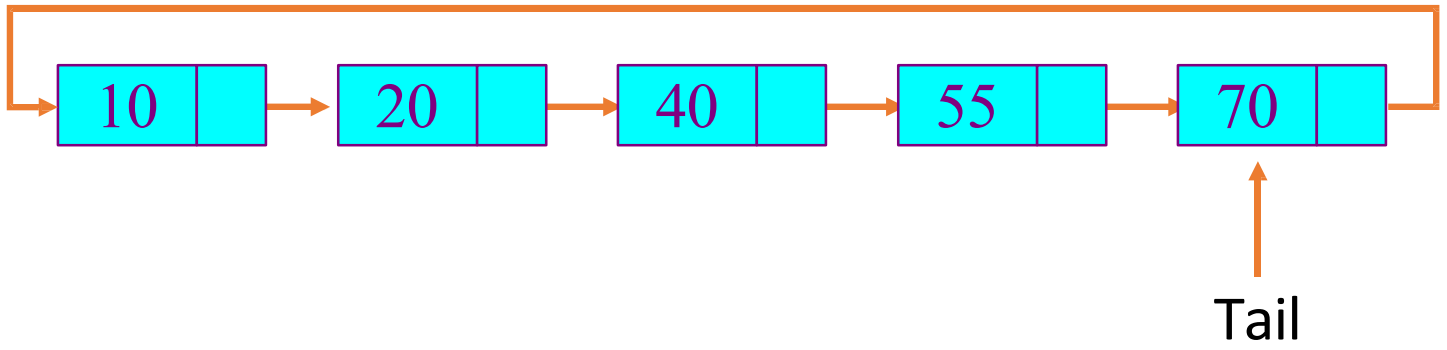
Circular Linked List: Applications

- Useful for implementation of queue, eliminates the need to maintain two pointers as in case of queue implementation using arrays
- Circular linked lists are useful for applications to repeatedly go around the list like playing video and sound files in “looping” mode
- Advanced data structures like Fibonacci Heap Implementation
- Plays a key role in linked implementation of graphs

Circular Singly 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 Tail pointer is often used instead of a Head pointer





Circular Singly Linked List Node Definition


```
#include <iostream>
using namespace std;

struct Node{
   int data;
   struct Node* next;
};
typedef struct node csll_node;
```

Insertion at the beginning

Insert at the front of linked list

- Create a node

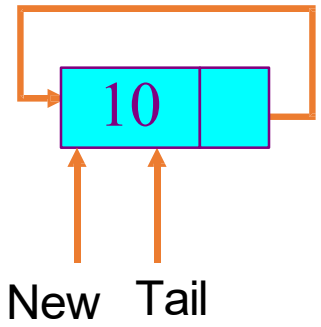
If the list is empty

- make the tail pointer point towards the new node

Else

- Change the new node link field to point to the first node
- Change the last node link to point to the new node

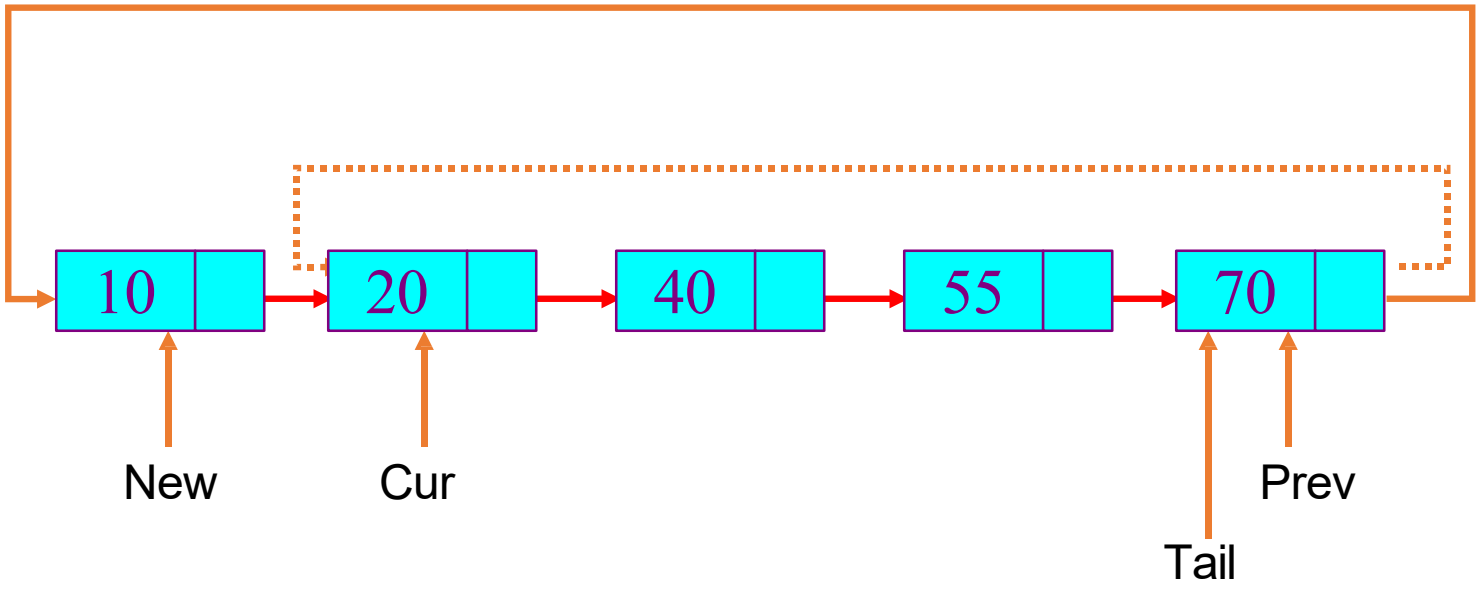
Insertion into an empty list



Insert to head of a Circular Linked List

New->next = Cur; ⟹ New->next = Tail->next;

Prev->next = New; ⟹ Tail->next = New;

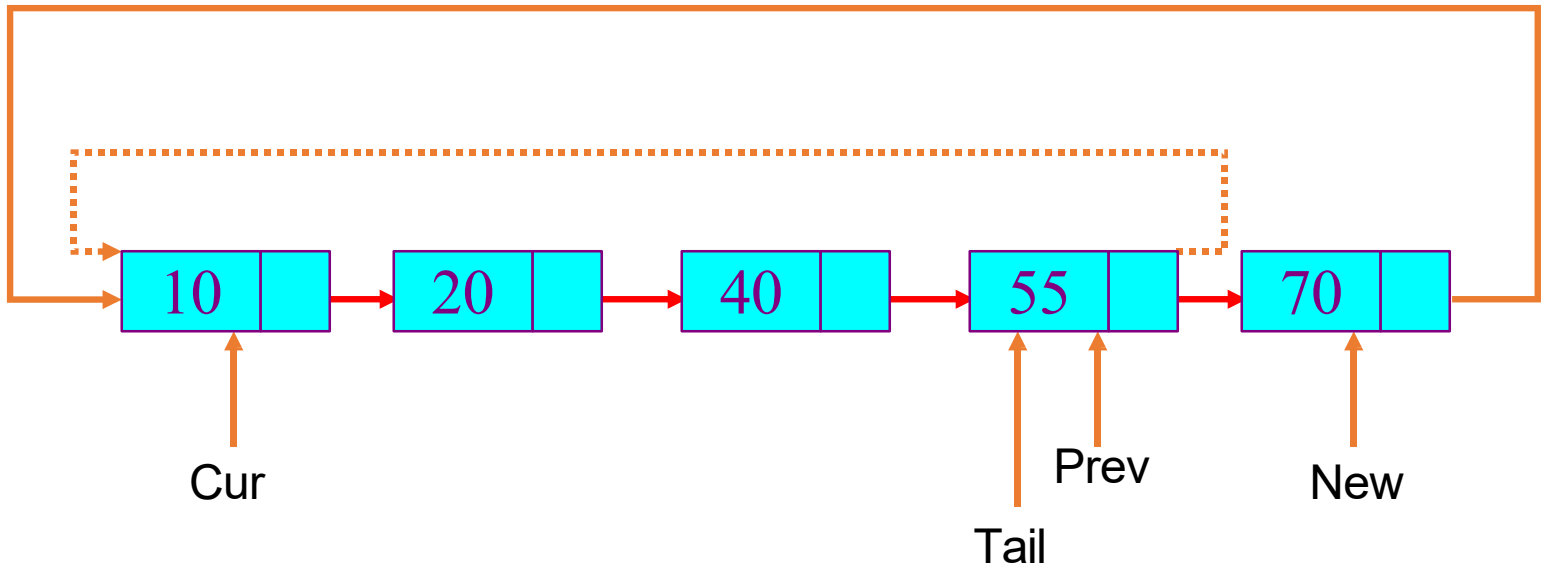


Insert to the end of a Circular Linked List

New->next = Cur; ⟶ New->next = Tail->next;

Prev->next = New; ⟶ Tail->next = New;

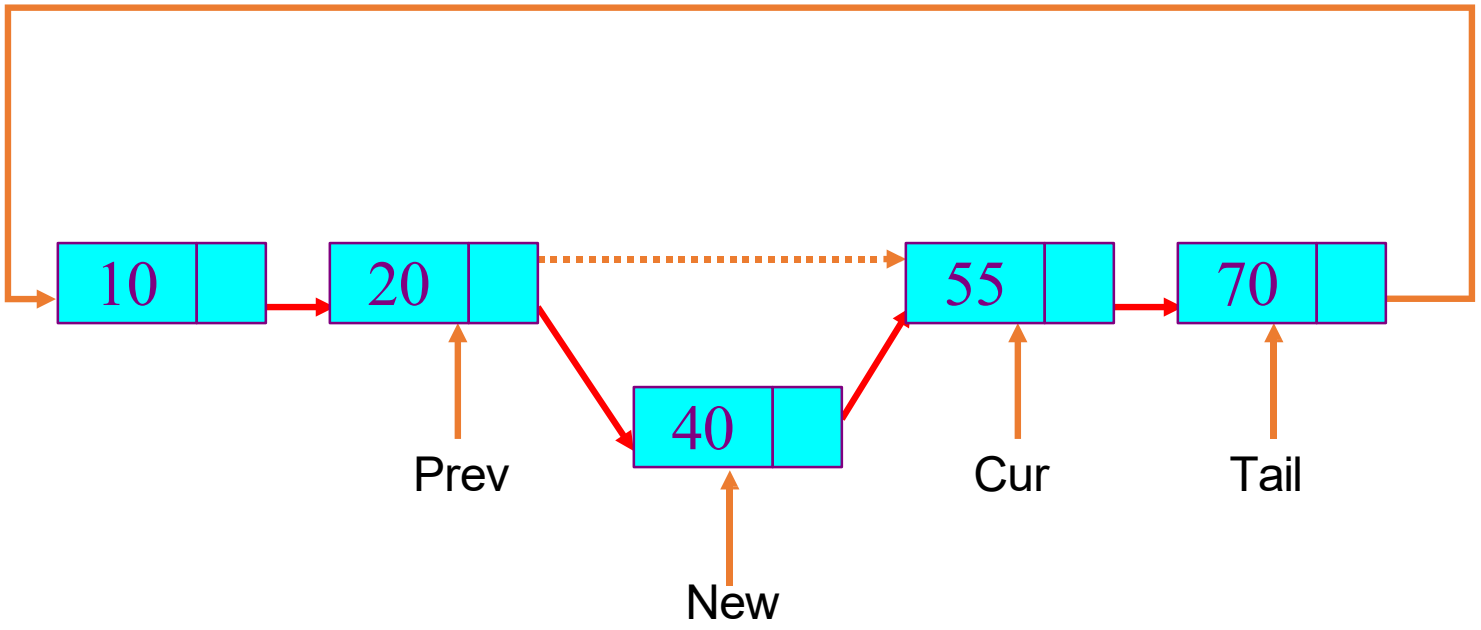
Tail = New;



Insert to the middle of Circular Linked List

New->next = Cur;

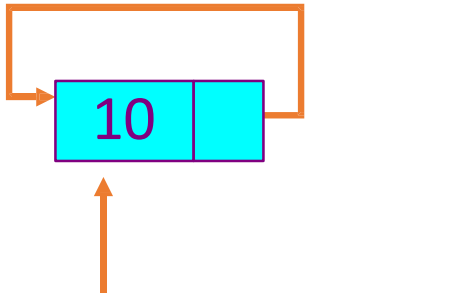
Prev->next = New;



Delete a node from a single-node Circular Linked List

Tail = NULL;

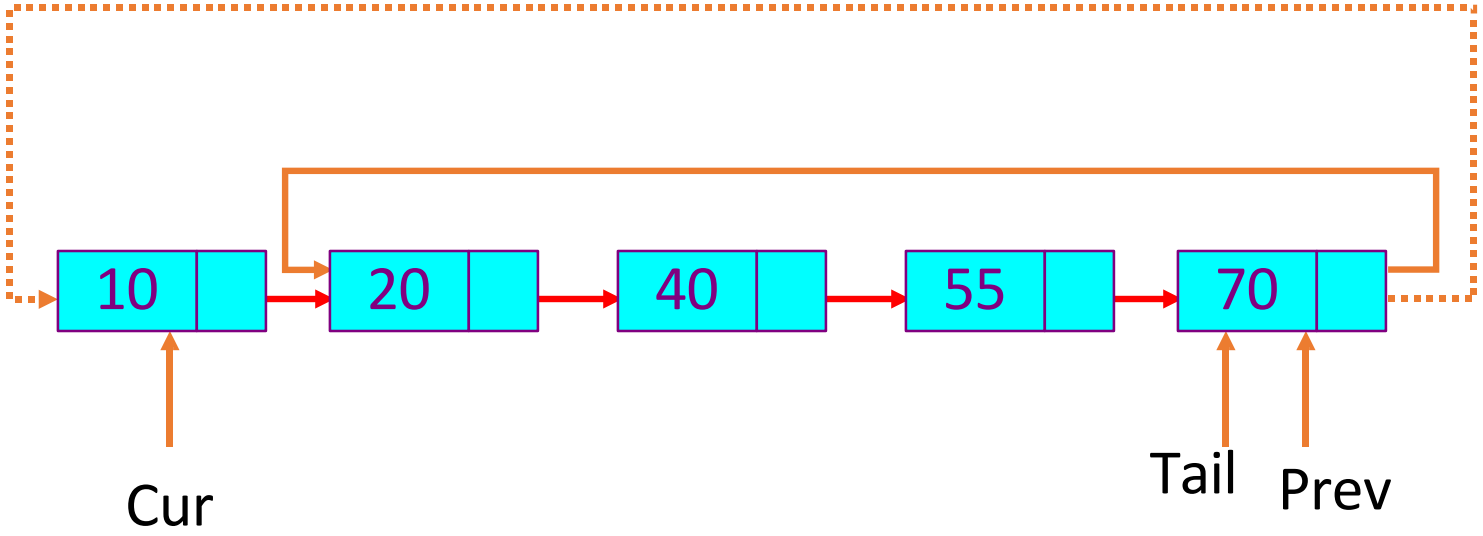
free(Cur);



Tail = Cur = Prev

Delete the head node from a Circular Linked List

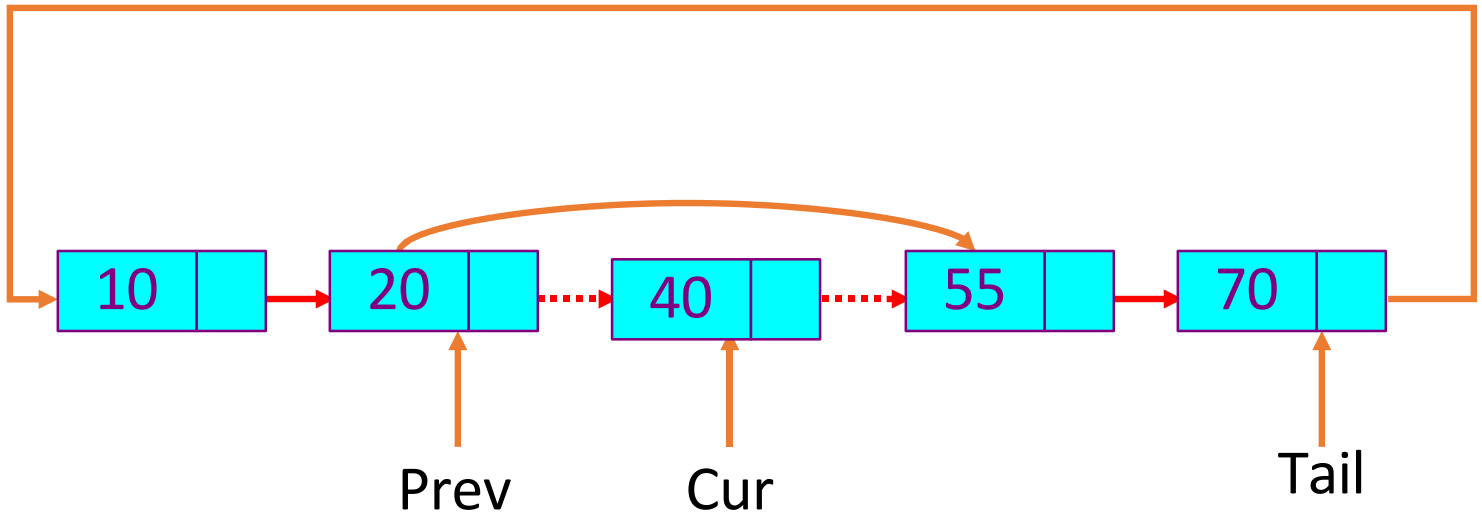
```
Prev->next = Cur->next;          // same as: Tail->next = Cur->next

free(cur);
```



Delete a middle node Cur from a Circular Linked List

Prev->next = Cur->next;

Free(Cur);



Circular Singly Linked List operations

Apply the concepts to implement following operations for a singly linked list

- insert a node after a given node(pointer)
- Insert a node after a node with a given value

1. In a CSLL, which is the correct condition for traversal starting from head?

a) while (temp != NULL)

b) while (temp->next != NULL)

c) do { ... } while (temp != head)

d) while (temp->next != head->next)

1. In a CSLL, which is the correct condition for traversal starting from head?

a) while (temp != NULL)

b) while (temp->next != NULL)

c) do { ... } while (temp != head)

d) while (temp->next != head->next)

2. Which of the following is true about the head pointer in a CSLL?

a) head always points to the last node.

b) head points to the first node, but last->next = head.

c) head->next = NULL.

d) head cannot be NULL in any case.

2. Which of the following is true about the head pointer in a CSLL?

a) head always points to the last node.

b) head points to the first node, but last->next = head.

c) head->next = NULL.

d) head cannot be NULL in any case.

3. To insert a node at the beginning of a CSLL, which of the following sequences is correct?

a) Add new node, set new->next = head, find last node, set last->next = new, then update head = new.

b) Set new->next = head, update head = new, done.

c) Set new->next = head->next, update head to new.

d) Set head->next = new, then new->next = head.

3. To insert a node at the beginning of a CSLL, which of the following sequences is correct?

a) Add new node, set new->next = head, find last node, set last->next = new, then update head = new.

b) Set new->next = head, update head = new, done.

c) Set new->next = head->next, update head to new.

d) Set head->next = new, then new->next = head.

4. For inserting a node at the end of a CSLL, which step is mandatory?

a) Update last->next = new and new->next = NULL.

b) Find last node (whose next = head), set new->next = head, and update last->next = new.

c) Set head = new if last node exists.

d) No need to link to head, as list is circular by default.

4. For inserting a node at the end of a CSLL, which step is mandatory?

a) Update last->next = new and new->next = NULL.

b) Find last node (whose next = head), set new->next = head, and update last->next = new.

c) Set head = new if last node exists.

d) No need to link to head, as list is circular by default.

5. If a CSLL contains only one node and we delete it, what must happen?

a) head = NULL.

b) head->next = head.

c) head->next = NULL.

d) Nothing, list remains unchanged.

5. If a CSLL contains only one node and we delete it, what must happen?

a) head = NULL.

b) head->next = head.

c) head->next = NULL.

d) Nothing, list remains unchanged.

THANK YOU

Vandana M L

Department of Computer Science & Engineering

vandanamd@pes.edu