DATA STRUCTURES AND ITS APPLICATIONS

Vandana M L
Department of Computer Science and Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Circular Doubly Linked List

Vandana M L

Department of Computer Science and Engineering

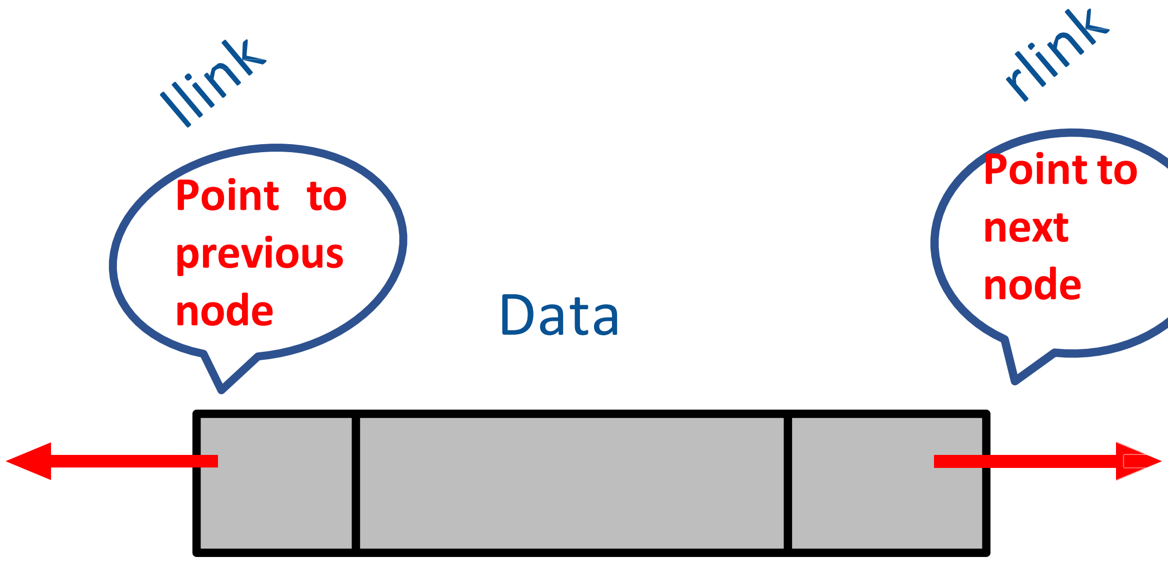
Node Structure Definition

A doubly linked list node contain three fields:

- Data
- link to the next node
- link to the previous node.

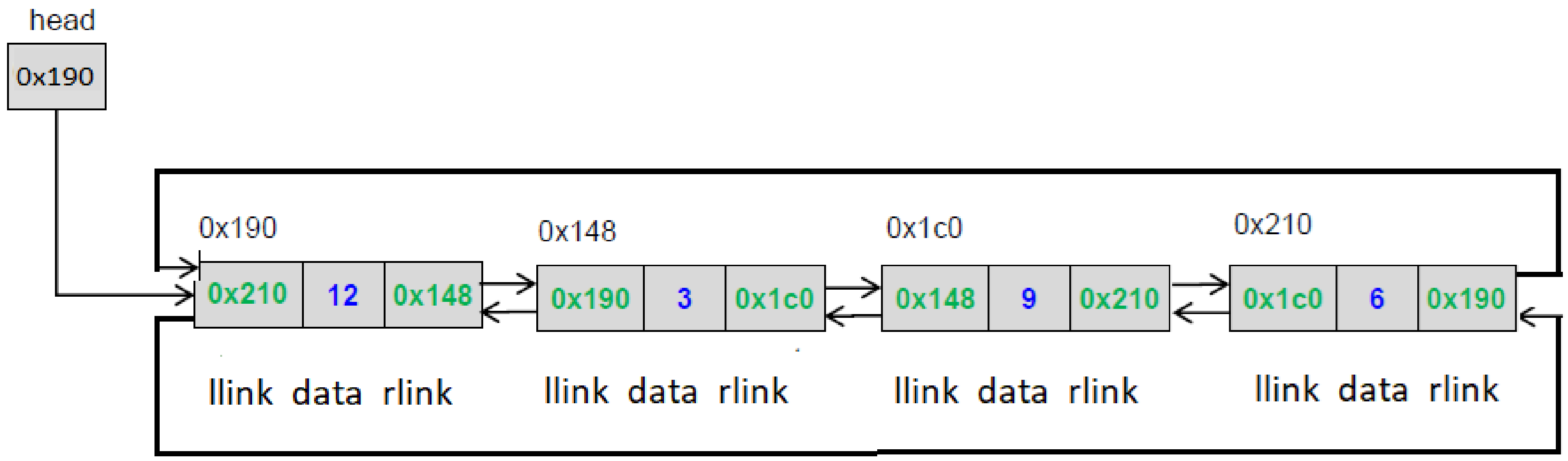
Node Structure Definition

```
struct node
{
   int data;
   struct node*llink;
   struct node*rlink;
};
```



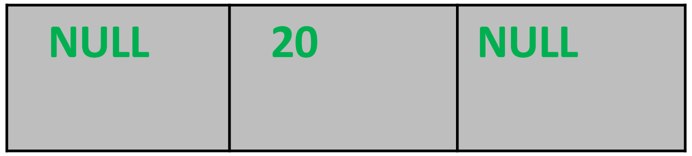
DATA STRUCTURES AND ITS APPLICATIONS

Circular Doubly Linked List: Example



Creating a node

- Allocate memory for the node dynamically
- If the memory is allocated successfully
  - set the data part
  - set the llink and rlink to NULL



Inserting a node

There are 3 cases

- Insertion at the beginning
- Insertion at the end
- Insertion at a given position

Insertion at the beginning

What all will change

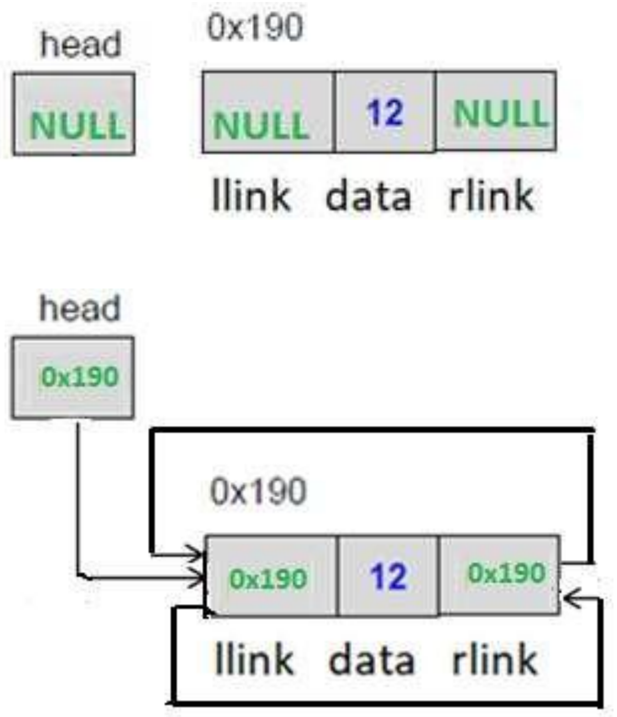
Case 1: linked list empty

- Head pointer

Case 2: linked list is not empty

- Head pointer
- New front node’s rlink and llink
- Old front node’s llink
- Last node’s rlink

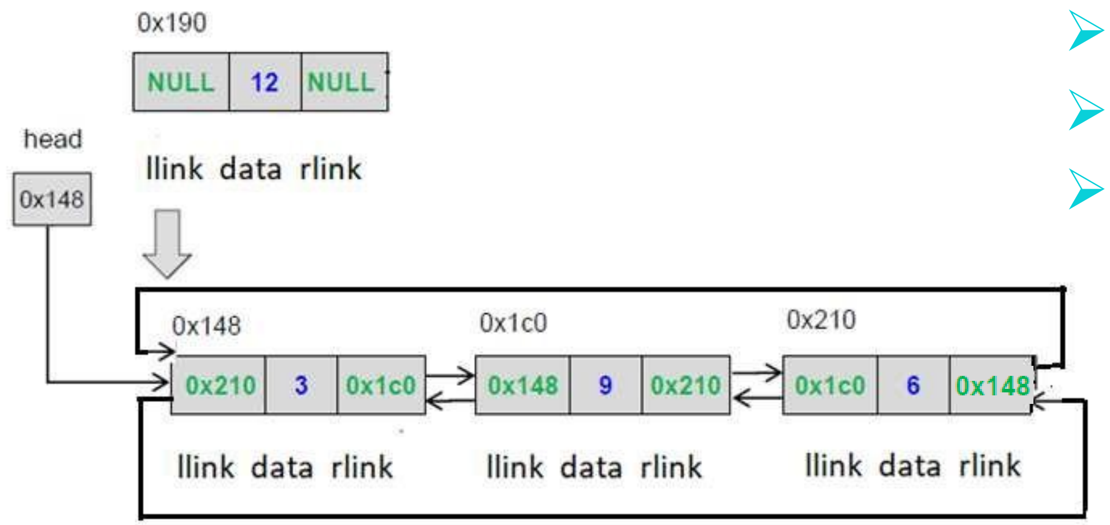
Insertion at the beginning (Case1)

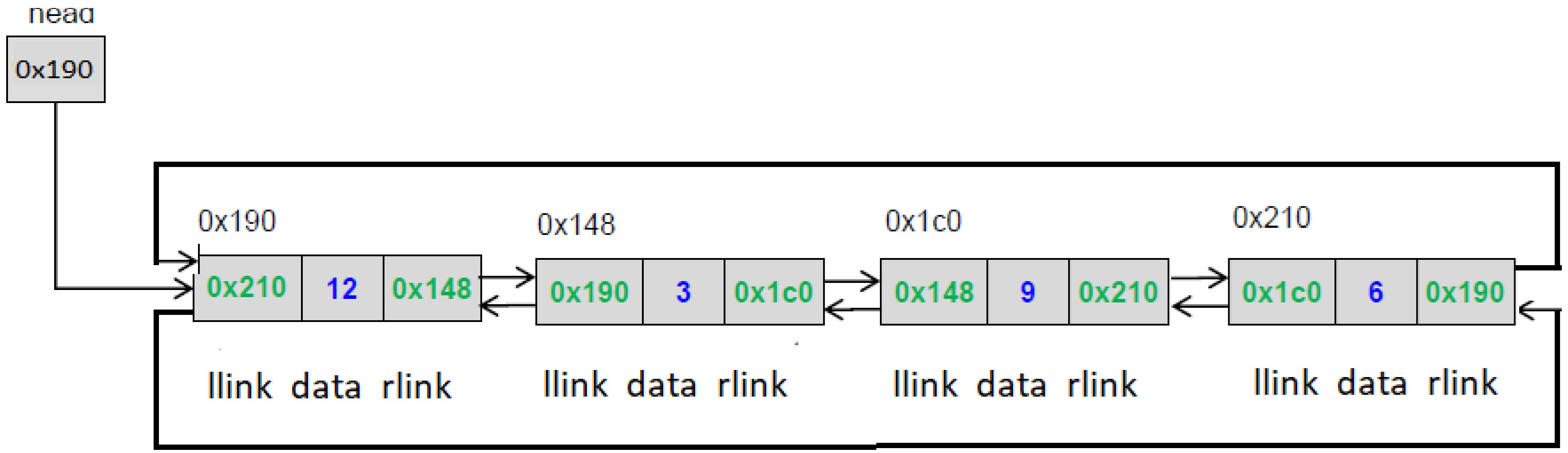


Circular Doubly Linked List Operations

Insertion at the beginning(Case 2)

- Head pointer
- New front node's rlink and lli...
- Old front node's llink
- Last node's rlink





Insertion at the end

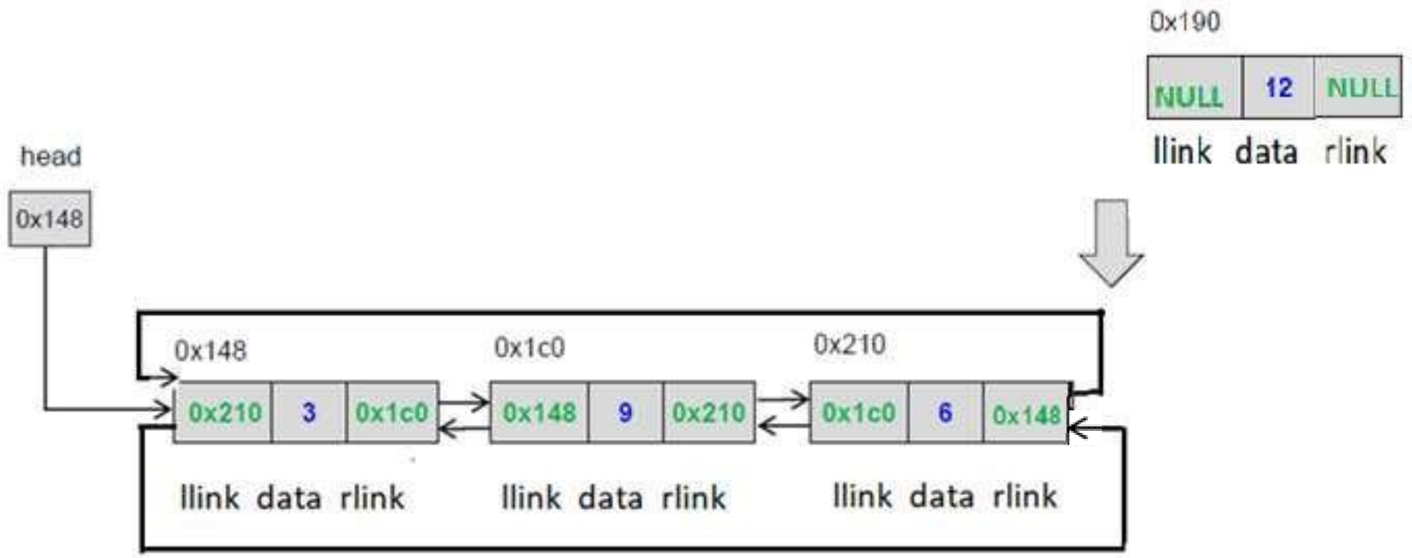
What all will change

Case 1: linked list empty
- Head pointer

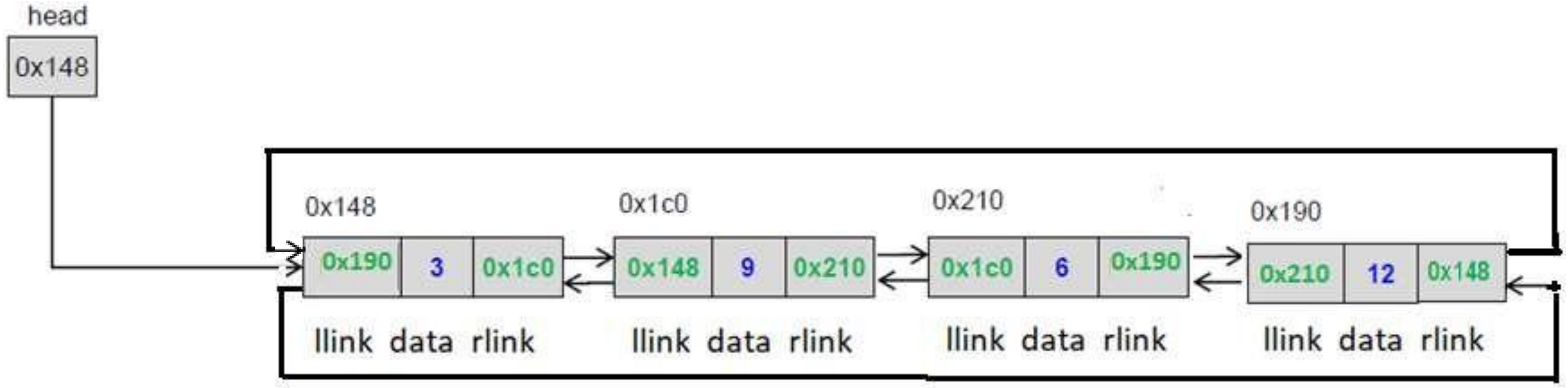
Case 2: linked list is not empty else
- Last node’s rlink
- First node llink
- New node’s llink and rlink

Circular Doubly Linked List Operations

Insertion at the end



: node's rlink
t node llink
v node's llink and rlink

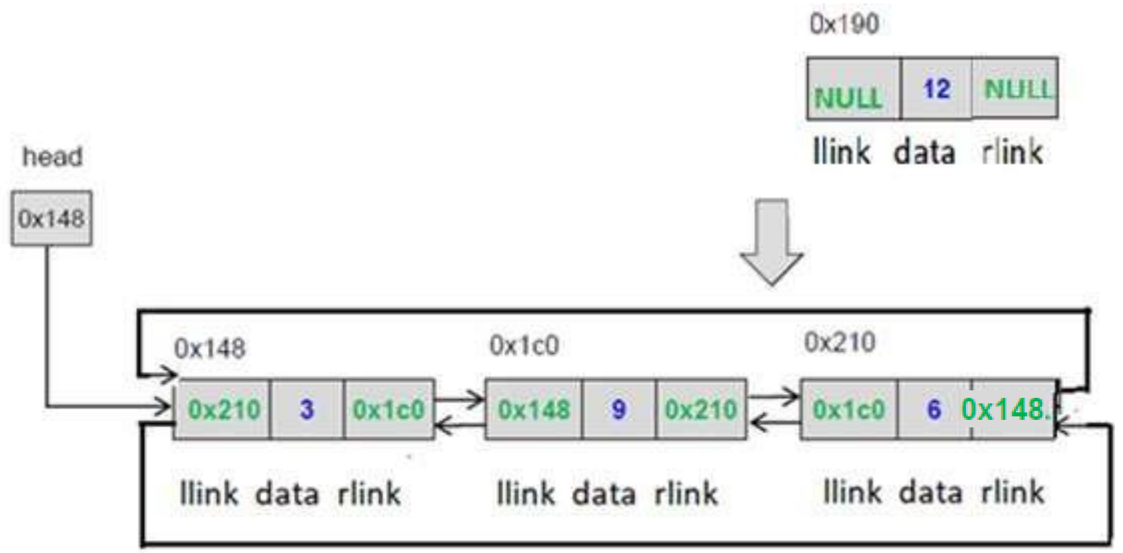


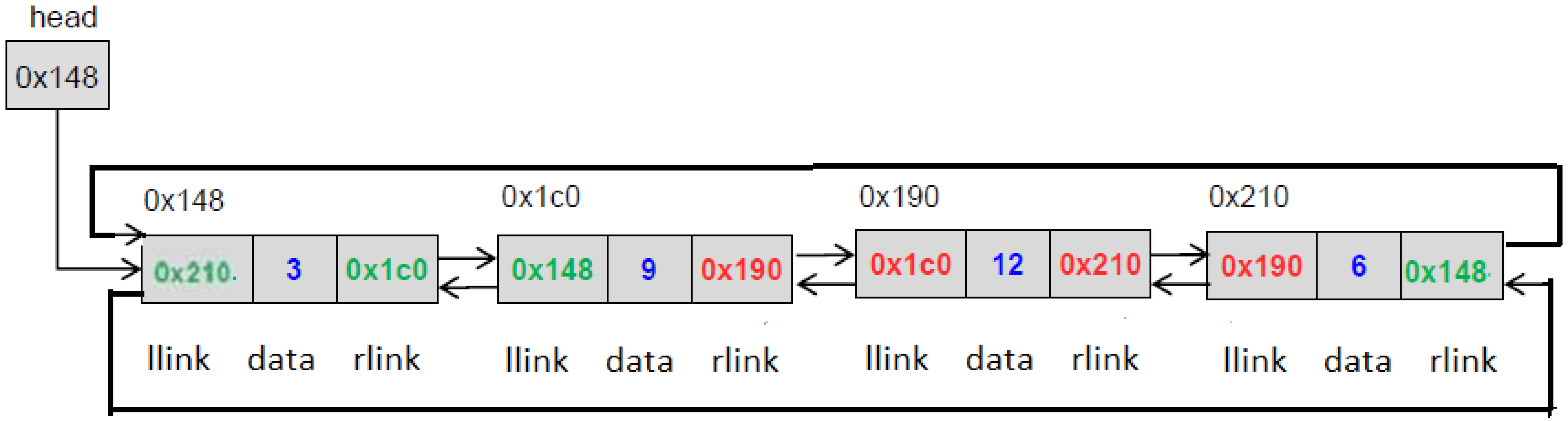
Insertion at the given position

```
➢ Create a node
If the list is empty
    ➢ make the start pointer  point towards the new node;
Else
    if it is first position
    ➢ Insert at front
    else
    ➢ Traverse the linked list to reach given position
    ➢ Keep track of the previous node
     If it is valid position
        intermediate position
        ➢    Change link fields of current previous and intermediate node
        last position
            ➢ insert at end
```

Circular Doubly Linked List Operations

Insertion at the given position





Deleting a node

There are 3 cases

- Deleting first node
- Deleting last node
- Deleting a node at a given position

Deleting a node

There are 3 cases

- Deleting first node
- Deleting last node
- Deleting a node at a given position

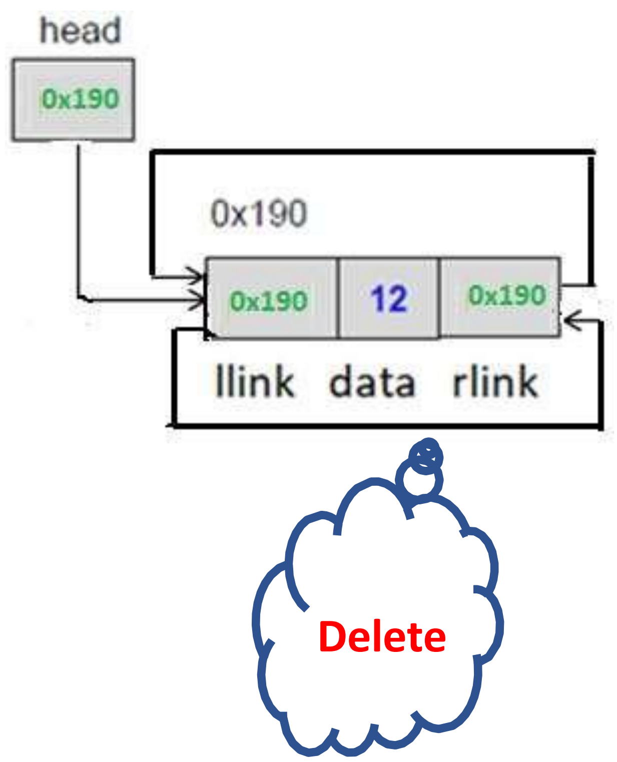
Deleting first node

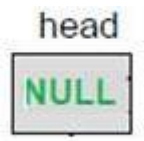
What will change??

- Case I : Empty Linked List
- Case II : Linked list with a single node
  first node gets freed up
  head points to NULL
- Case III : Linked List with more than one node
  Second node llink
  last node rlink
  first node gets freed off
  head pointer points to second node

Deleting first node

- Case II : Linked list with a single node

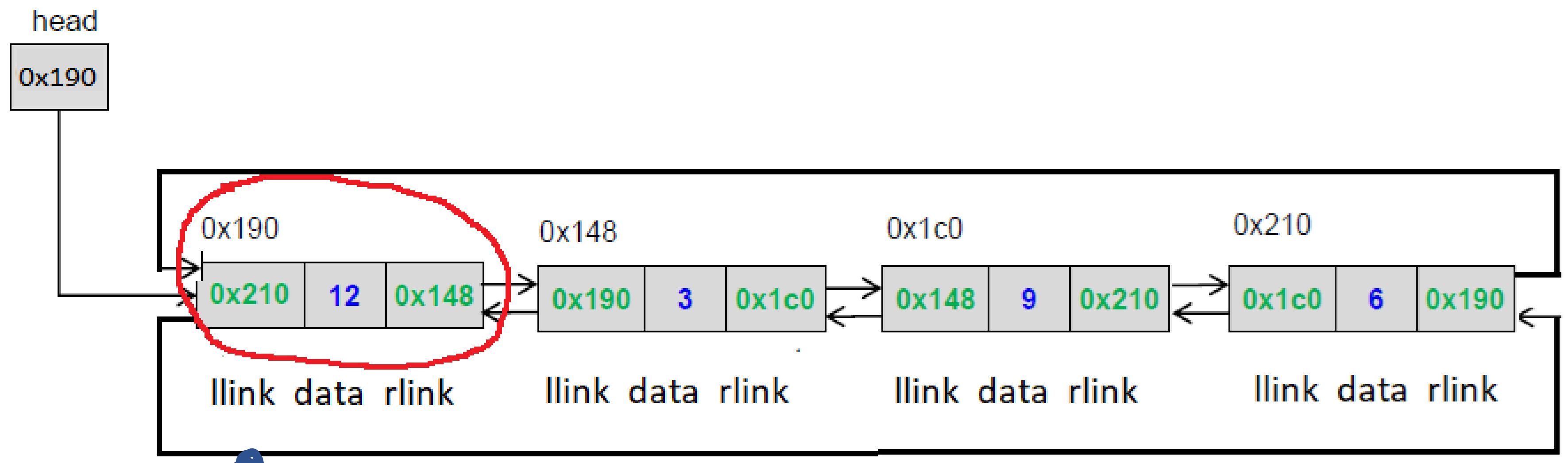


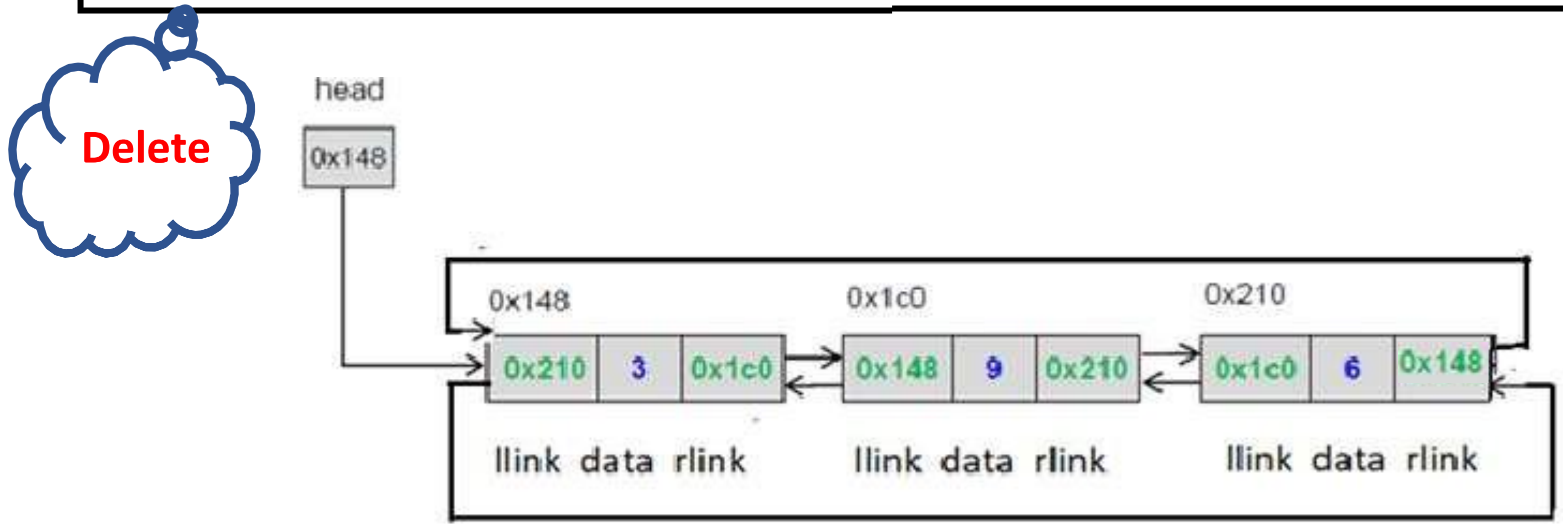


Circular Doubly Linked List Operations

Deleting first node

- Case III : Linked List with more than one node





Circular Doubly Linked List Operations

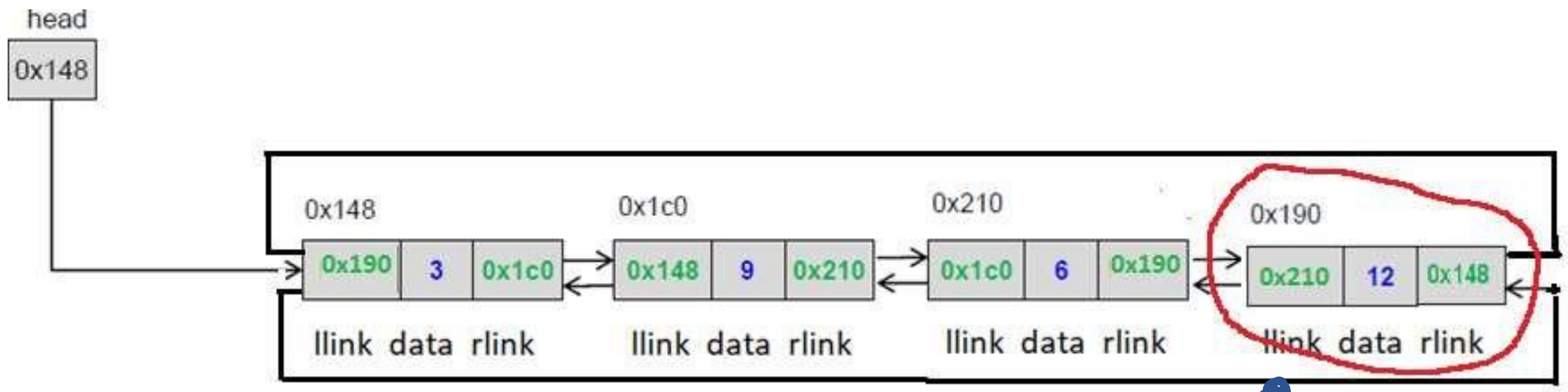
Deleting last node

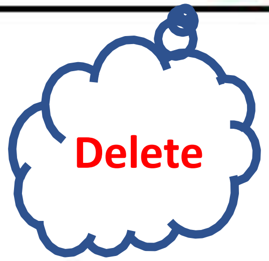
What will change??

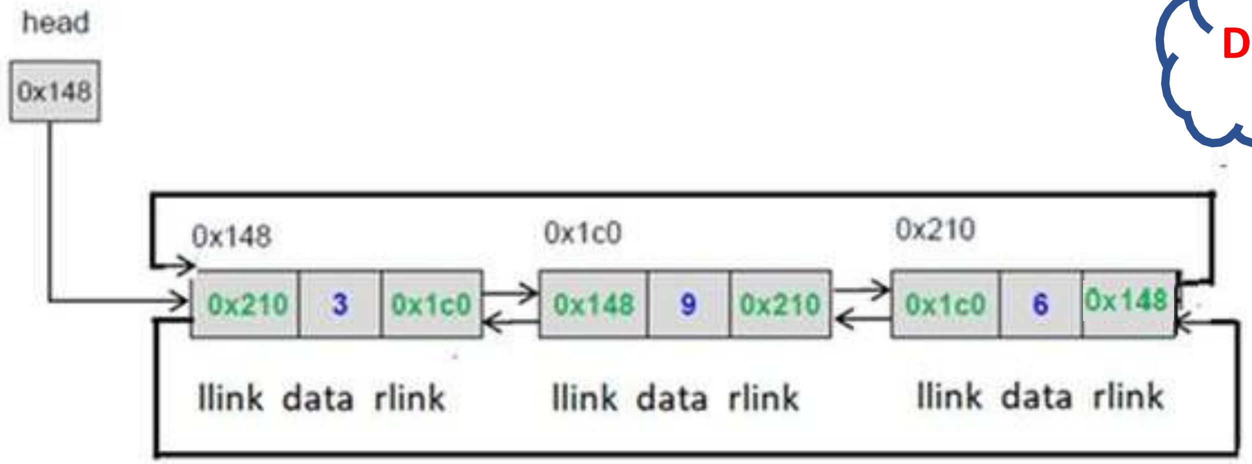
- Case I : Empty Linked List
- Case II : Linked list with a single node
  - first node gets freed up
  - head points to NULL
- Case III : Linked List with more than one node
  - Second last node rlink
  - first node llink
  - last node gets freed up

Deleting last node

- Case II : Linked List with more than one node

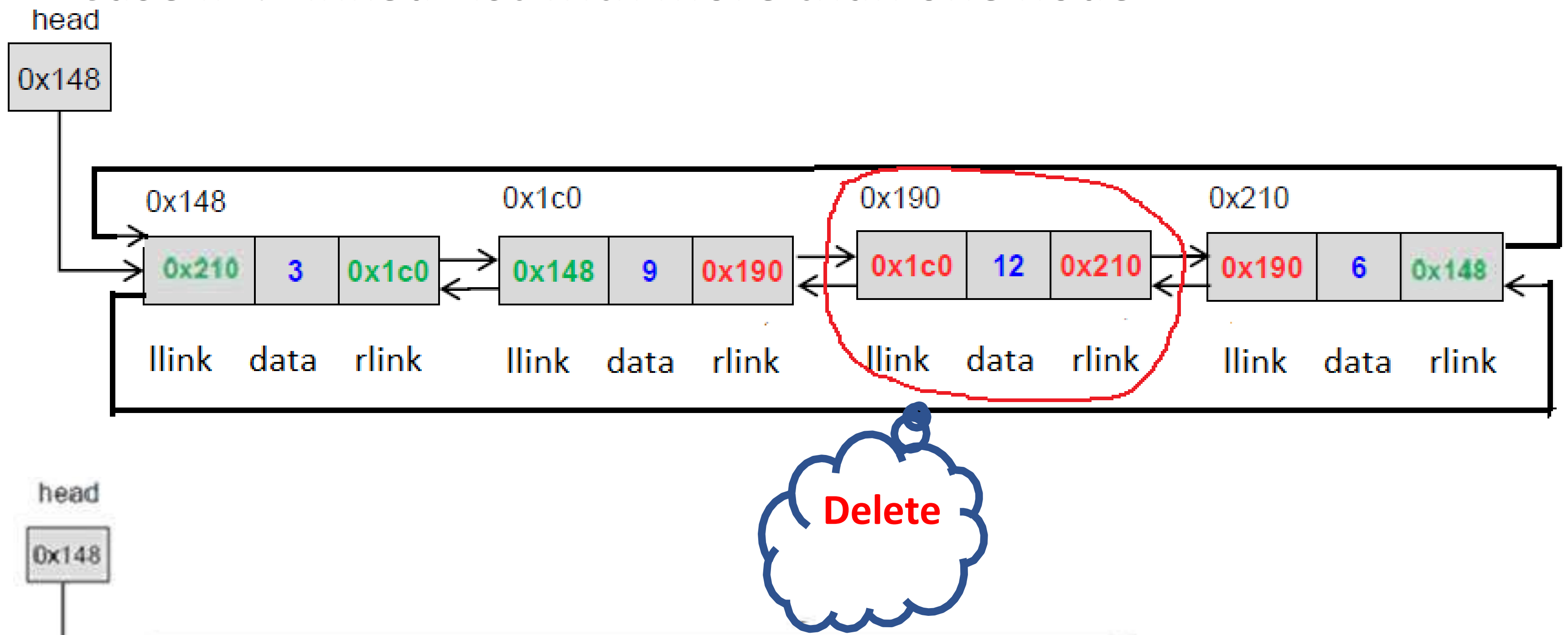


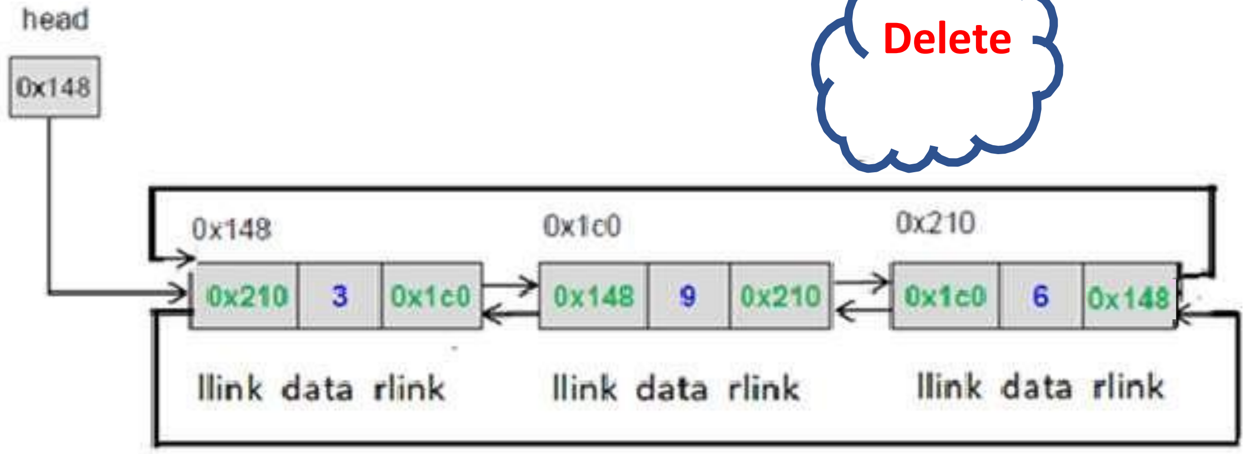




Deleting a node at intermediate position

- Case II : Linked List with more than one node





Circular doubly Linked List operations

Apply the concepts to implement following operations for a doubly circular linked list

- Reverse list using recursion
- Search given element in the list
- Find the largest value in the list

1. **Which statement correctly describes a Circular Doubly Linked List (CDLL)?**

a) The next pointer of the last node is NULL.

b) Both the next pointer of the last node and the prev pointer of the head point back to each other.

c) Only the head node's prev is NULL.

d) A CDLL cannot have more than two nodes.

1. **Which statement correctly describes a Circular Doubly Linked List (CDLL)?**

a) The next pointer of the last node is NULL.

b) Both the next pointer of the last node and the prev pointer of the head point back to each other.

c) Only the head node's prev is NULL.

d) A CDLL cannot have more than two nodes.

2. Which of the following loop structures is best for traversing a CDLL starting from head?

a) while (temp != NULL)

b) for (temp = head; temp != NULL; temp = temp->next)

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

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

2. Which of the following loop structures is best for traversing a CDLL starting from head?

a) while (temp != NULL)

b) for (temp = head; temp != NULL; temp = temp->next)

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

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

3. To insert a new node at the head of a CDLL, which sequence is correct?

a) Adjust head->next and head->prev only.

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

c) Set new->next = head->next and head->next = new.

d) Update only prev pointers, as next is automatically updated.

3. To insert a new node at the head of a CDLL, which sequence is correct?

a) Adjust head->next and head->prev only.

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

c) Set new->next = head->next and head->next = new.

d) Update only prev pointers, as next is automatically updated.

4. Which pointer updates are required to insert newNode at the end of a CDLL?

a) Update tail->next = newNode, newNode->prev = tail, newNode->next = NULL.

b) Update tail->next = newNode, newNode->prev = tail, newNode->next = head, and head->prev = newNode.

c) Update only head->prev = newNode.

d) Use recursion to find the last node and insert.

4. Which pointer updates are required to insert newNode at the end of a CDLL?

a) Update tail->next = newNode, newNode->prev = tail, newNode->next = NULL.

b) Update tail->next = newNode, newNode->prev = tail, newNode->next = head, and head->prev = newNode.

c) Update only head->prev = newNode.

d) Use recursion to find the last node and insert.

5. Which steps correctly delete the head node in a CDLL (with more than one node)?

a) head = head->next; head->prev = NULL; free(oldHead);

b) head->prev->next = head->next; head->next->prev = head->prev; head = head->next; free(oldHead);

c) head = NULL; free(head);

d) head->next = head; free(head);

5. Which steps correctly delete the head node in a CDLL (with more than one node)?

a) head = head->next; head->prev = NULL; free(oldHead);

b) head->prev->next = head->next; head->next->prev = head->prev; head = head->next; free(oldHead);

c) head = NULL; free(head);

d) head->next = head; free(head);

THANK YOU

Vandana M L

Department of Computer Science & Engineering

vandanamd@pes.edu

+91 7411716615