DATA STRUCTURES AND APPLICATIONS

Vandana M L

Department of Computer Science and Engineering

DATA STRUCTURES AND APPLICATIONS

Doubly Linked List

Vandana M L

Department of Computer Science and Engineering

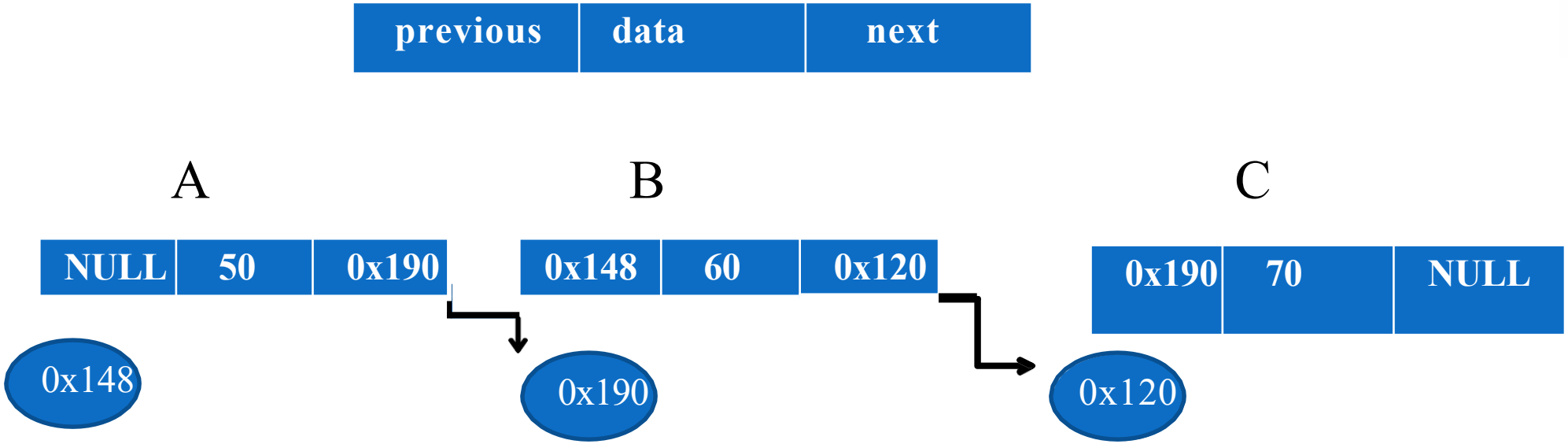
Doubly Linked List

Adoubly linked list contain three fields:

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

DATA STRUCTURES AND APPLICATIONS

Doubly Linked List :Node Structure



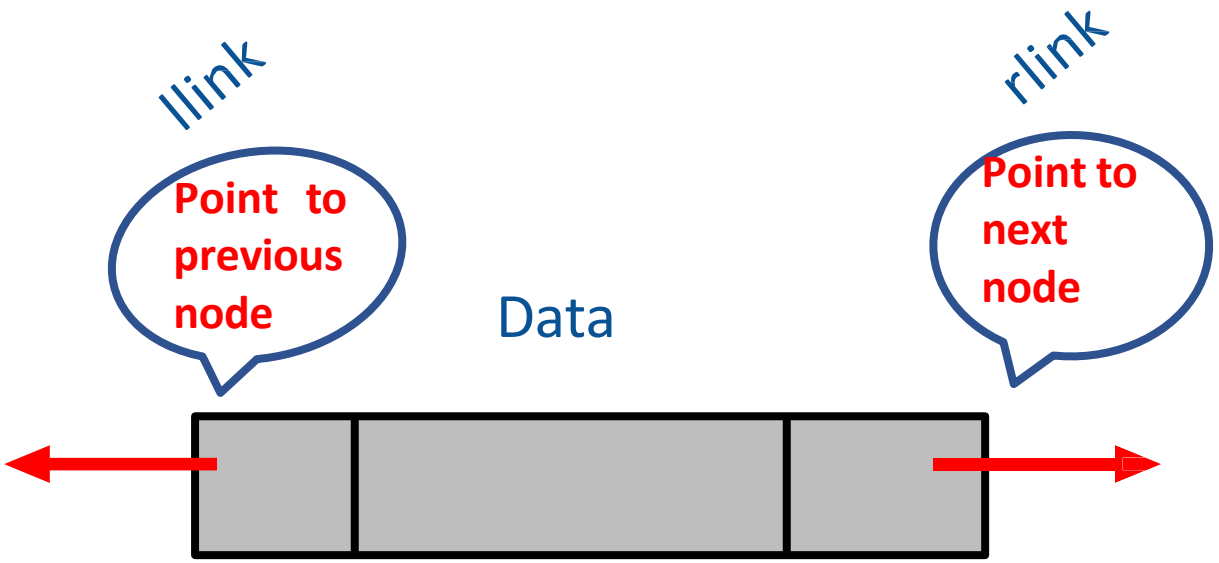
Doubly Linked List Vs Singly Linked List

- Advantages:
  - Can be traversed in either direction (may be essential for some programs)
  - Some operations, such asdeletion and inserting before a node, become easier
- Disadvantages:
  - Requires more space
  - List manipulations are slower (because more links must be changed)
  - Greater chance of having bugs (because more links must be manipulated)

DATA STRUCTURES AND APPLICATIONS

Doubly Linked List Node definition

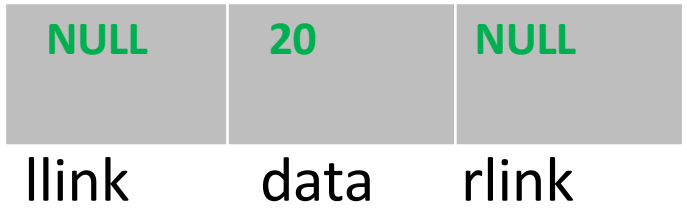
```
struct node
{
   int data;
   node*llink;
   node*rlink;
};
```



Creating a node

- Allocate memory for the node dynamically

- If the memory is allocated successfully
  set the data part to user defined value
  set the llink (address of previous node) and rlink (address of next node) part 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

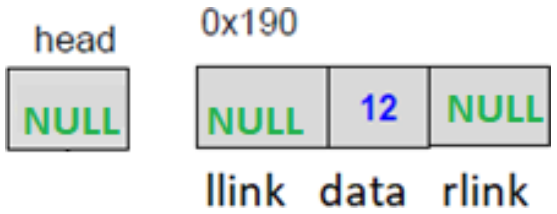
If the linked list empty(case 1)
Head/Start pointer
else (case2)

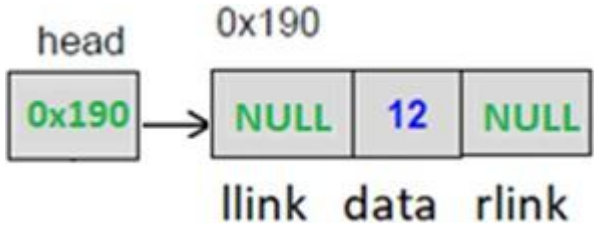
- Head/Start pointer
- New front's llink and rlink
- Old front's llink

DATA STRUCTURES AND APPLICATIONS

Doubly Linked List Implementation

Insertion at the beginning (Case1)

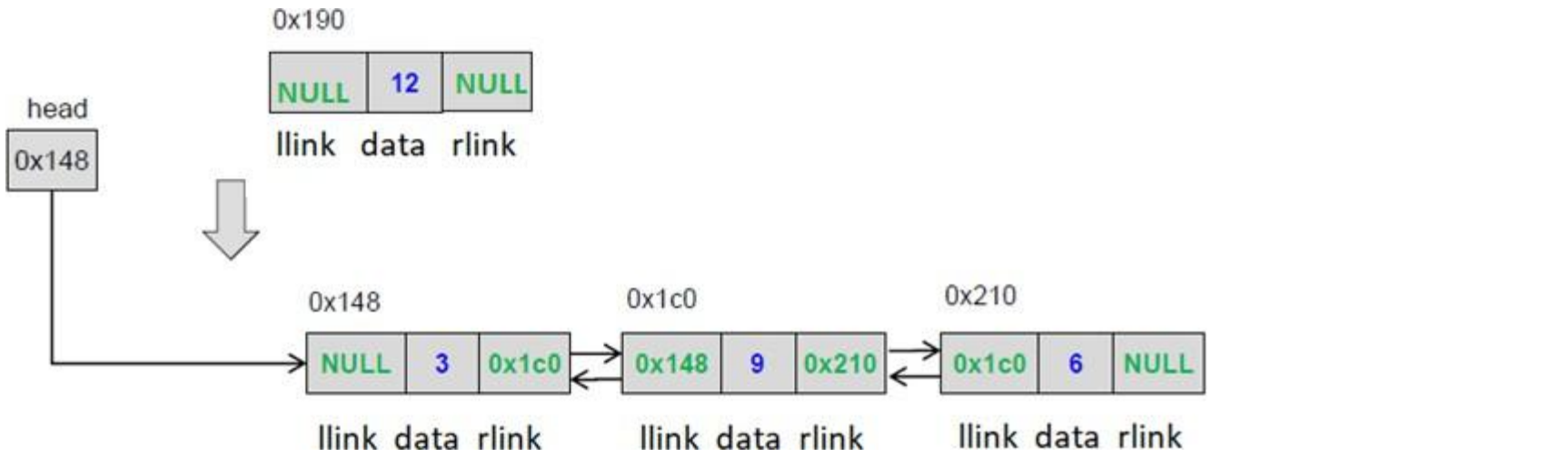


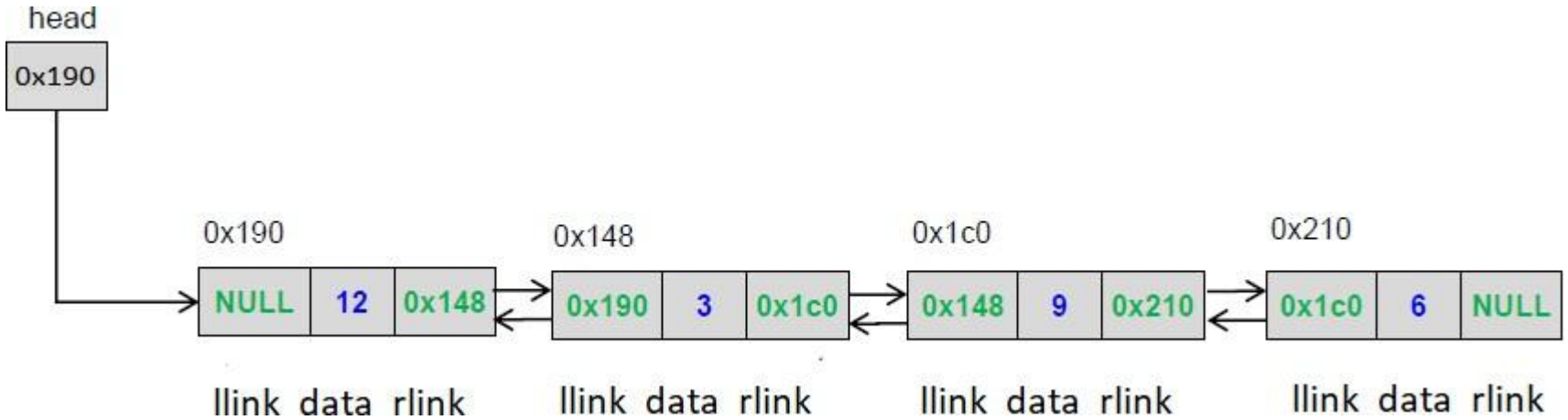


DATA STRUCTURES AND APPLICATIONS

Doubly Linked List Implementation

Insertion at the beginning(Case 2)





Insertion at the end

What all will change

If the linked list empty(same as case 1 of insert at front)
Head/Start pointer(case 2)
else

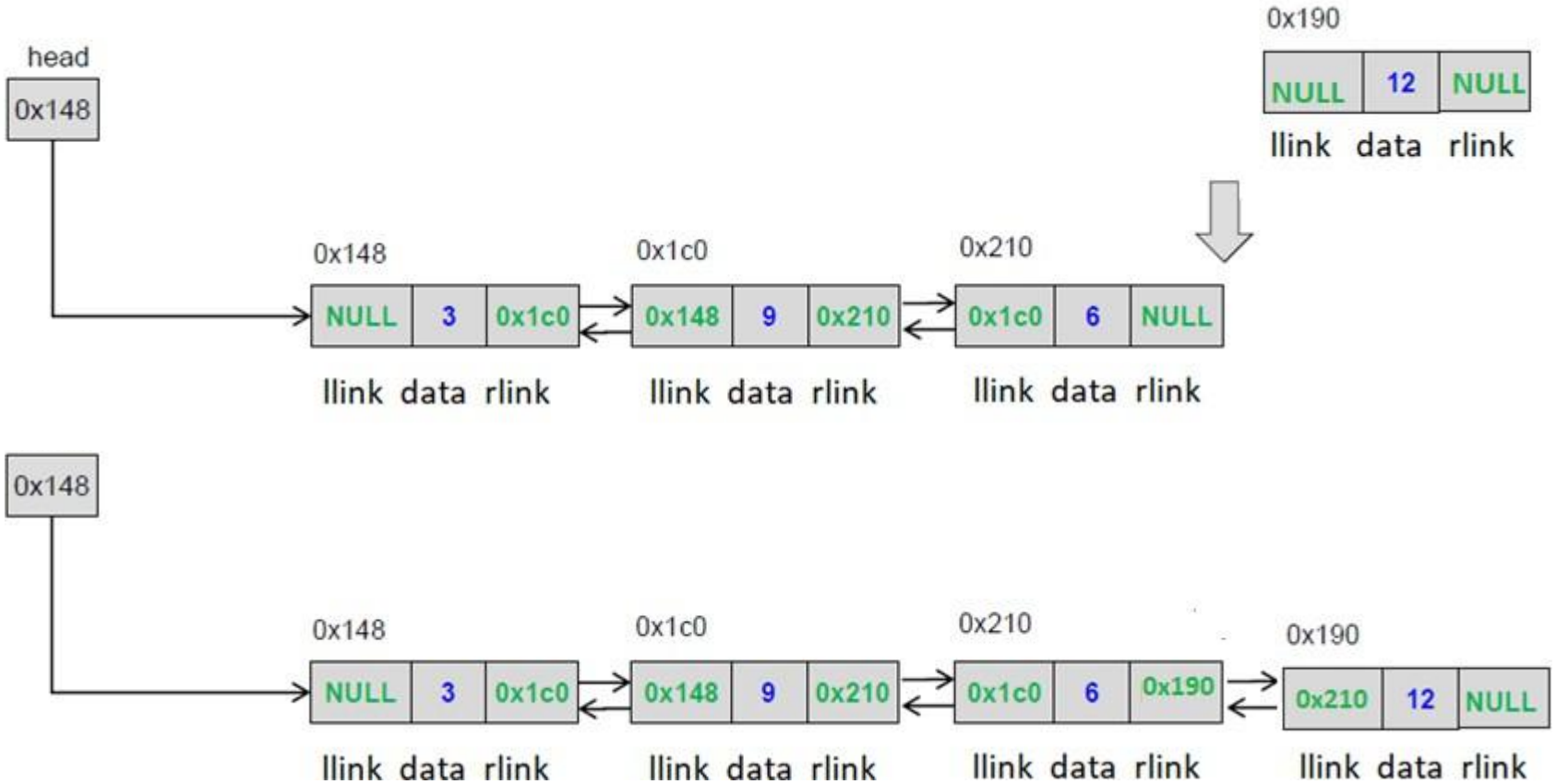
- Last node's rlink
- New node's llink

DATA STRUCTURES AND APPLICATIONS

Doubly Linked List Implementation

PES UNIVERSITY

Insertion at the end



Insertion at the given position

- Create a node

If the list is empty

- make the start pointer point towards the new node;

Else

- Traverse the linked list to reach given position
- Keep track of the previous node

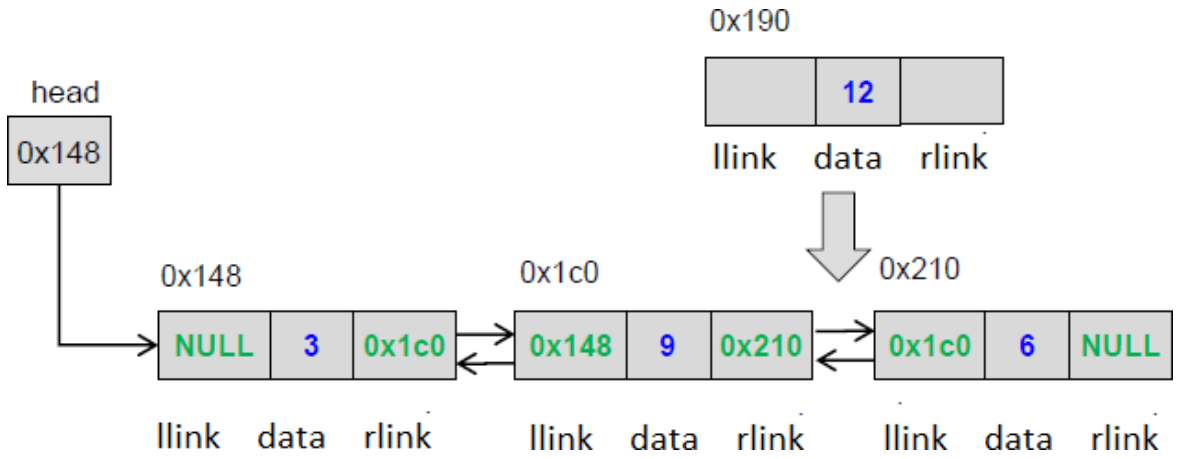
If it is an intermediate position

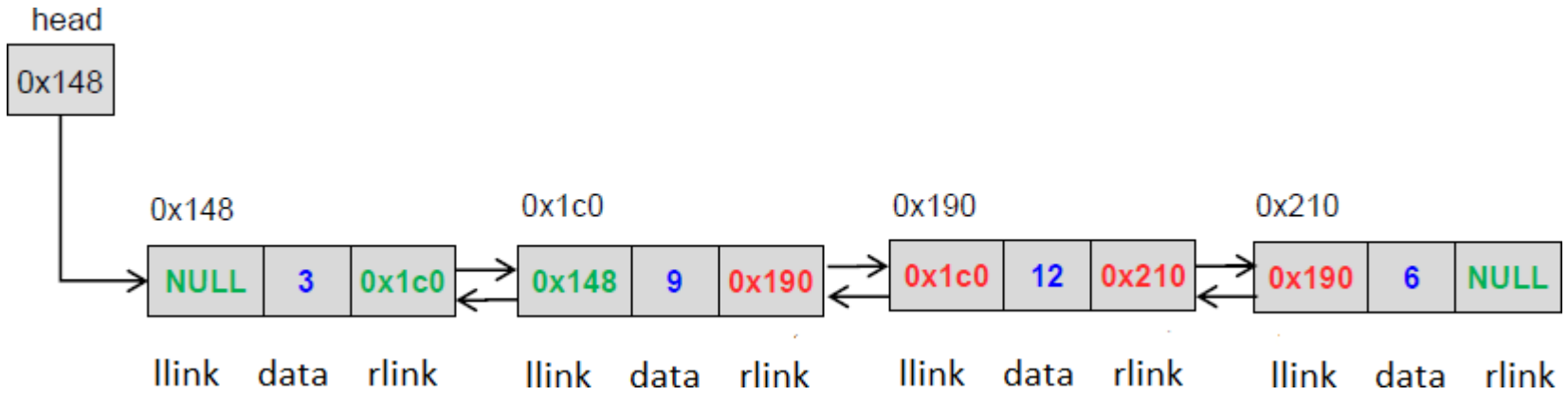
- Change previous node rlink to point to the newnode
- Newnode's llink to point to previous node and rlink to point to the next node
- Next node llink to point to the newnode

DATA STRUCTURES AND APPLICATIONS

Doubly Linked List Implementation

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

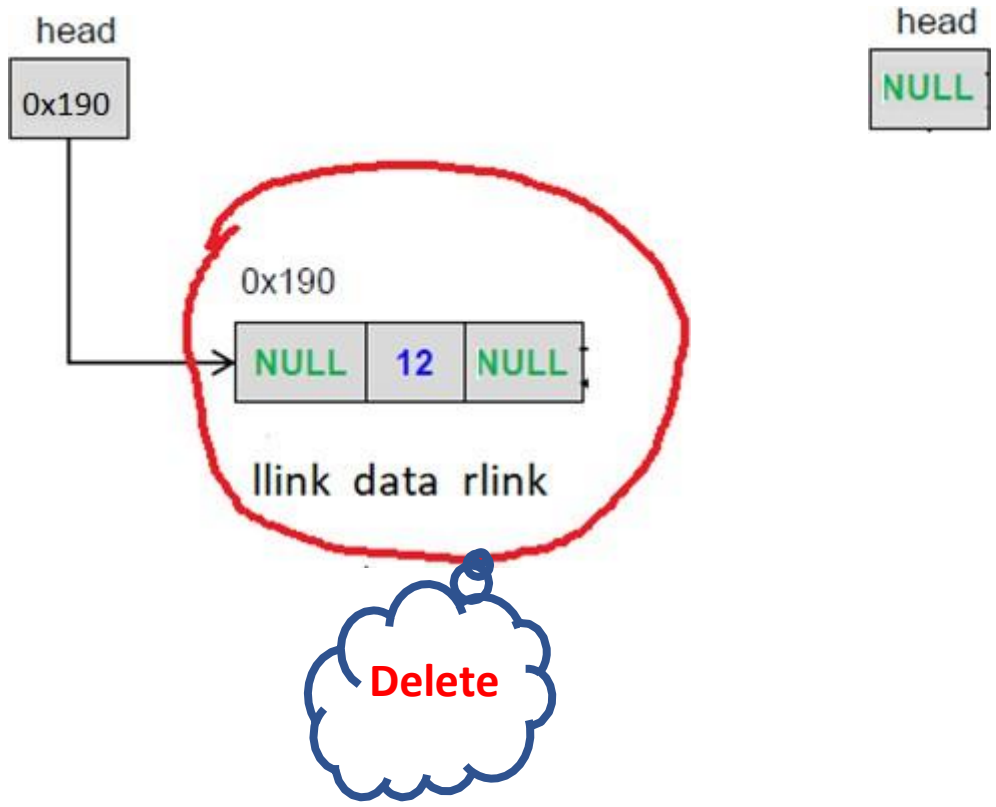
Deleting first node

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 gets changed to NULL
  first node gets freed off
  head points to second node

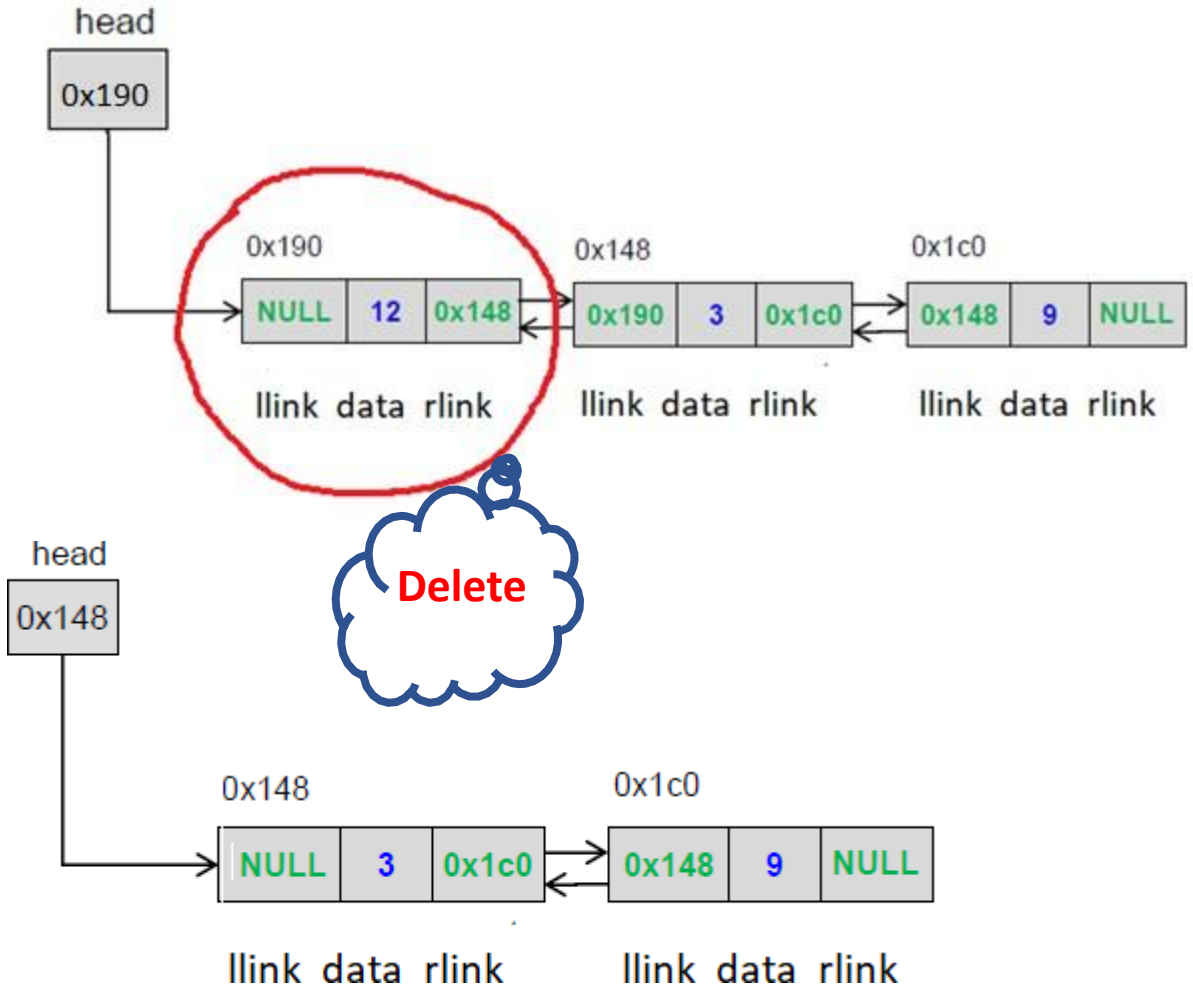
Deleting first node

- Case II : Linked list with a single node



Deleting first node

- Case III : Linked List with more than one node



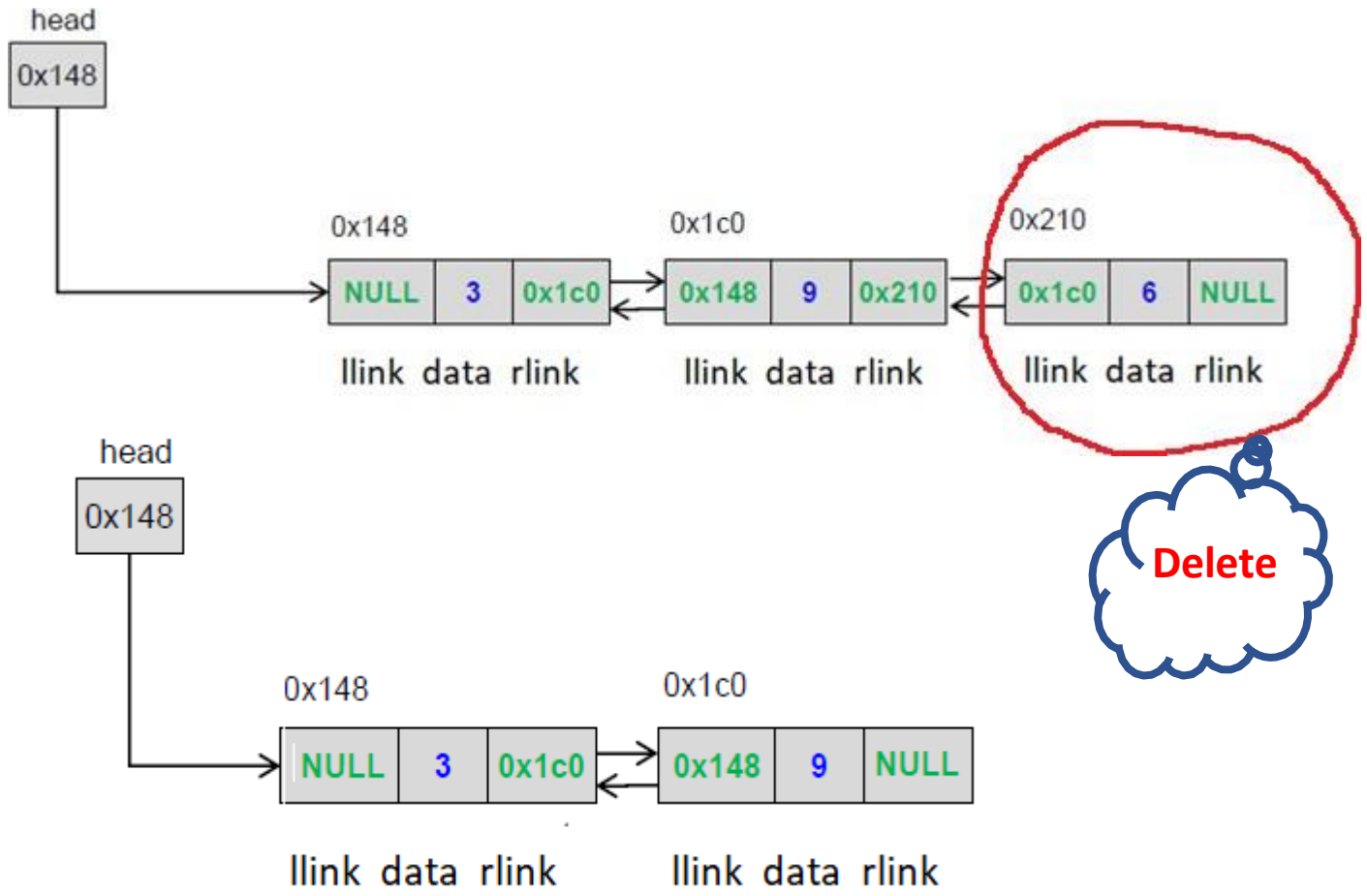
Deleting last node

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 point to NULL
  last node gets freed up

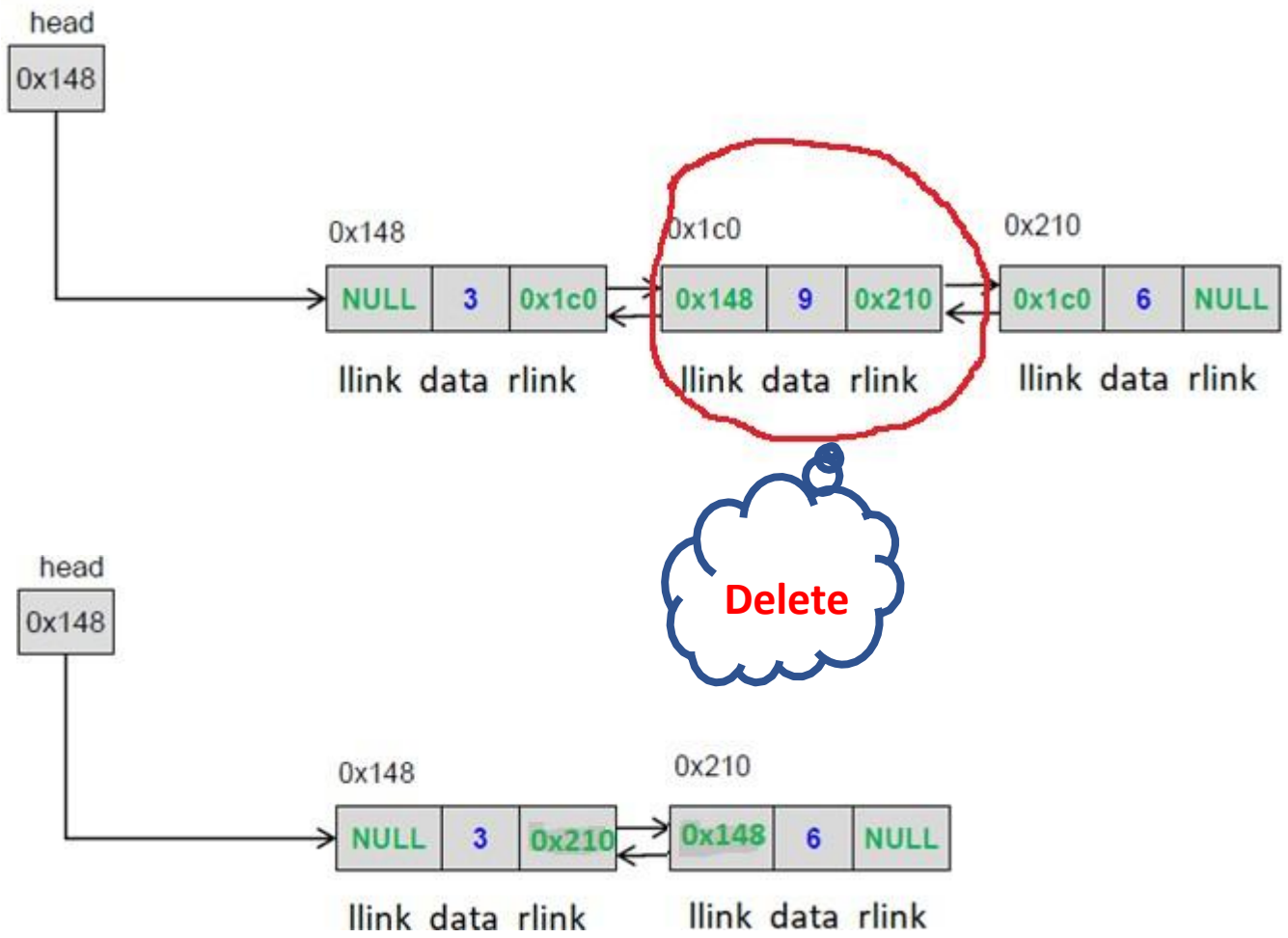
Deleting last node

- Case II : Linked List with more than one node



Deleting a node at intermediate position

- Case II : Linked List with more than one node



Doubly Linked List insert operation

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

- reverse a doubly linked list
- Find the node pairs with a given sum in a doubly linked list
- Insert a node after a node with a given value
- Remove duplicate nodes from a doubly linked list

1. Which of the following is a disadvantage of a DLL compared to an SLL?

a) Faster traversal in both directions.

b) Requires extra memory for storing the prev pointer in each node.

c) Easier insertion and deletion from the middle.

d) Can handle reverse traversal efficiently.

1. Which of the following is a disadvantage of a DLL compared to an SLL?

a) Faster traversal in both directions.

b) Requires extra memory for storing the prev pointer in each node.

c) Easier insertion and deletion from the middle.

d) Can handle reverse traversal efficiently.

2. Which sequence correctly inserts a new node newNode at the head of a DLL?

a) newNode->next = head; head = newNode;

b) newNode->next = head; head->prev = newNode; head = newNode;

c) head->next = newNode; newNode->prev = head; head = newNode;

d) newNode->prev = NULL; head = newNode;

2. Which sequence correctly inserts a new node newNode at the head of a DLL?

a) newNode->next = head; head = newNode;

b) newNode->next = head; head->prev = newNode; head = newNode;

c) head->next = newNode; newNode->prev = head; head = newNode;

d) newNode->prev = NULL; head = newNode;

3. To insert at the end of a DLL, which step is incorrect?

a) Traverse to the last node.

b) Set last->next = newNode.

c) Set newNode->prev = last.

d) Set newNode->next = head.

3. To insert at the end of a DLL, which step is incorrect?

a) Traverse to the last node.

b) Set last->next = newNode.

c) Set newNode->prev = last.

d) Set newNode->next = head.

4. To insert newNode after a node p in a DLL, which of the following steps is essential but often missed?

a) newNode->next = p->next;

b) p->next = newNode;

c) p->next->prev = newNode; (if p->next is not NULL)

d) All of the above.

4. To insert newNode after a node p in a DLL, which of the following steps is essential but often missed?

a) newNode->next = p->next;

b) p->next = newNode;

c) p->next->prev = newNode; (if p->next is not NULL)

d) All of the above.

5. When deleting a node target (not head/tail) in a DLL, which of the following is correct?

a) target->prev->next = target->next; target->next->prev = target->prev; free(target);

b) target->prev = target->next; target->next = target->prev; free(target);

c) target->next->prev = NULL; free(target);

d) target->prev = NULL; free(target);

5. When deleting a node target (not head/tail) in a DLL, which of the following is correct?

a) target->prev->next = target->next; target->next->prev = target->prev; free(target);

b) target->prev = target->next; target->next = target->prev; free(target);

c) target->next->prev = NULL; free(target);

d) target->prev = NULL; free(target);

Vandana M L

Department of Computer Science and Engineering

vandanamd@pes.edu