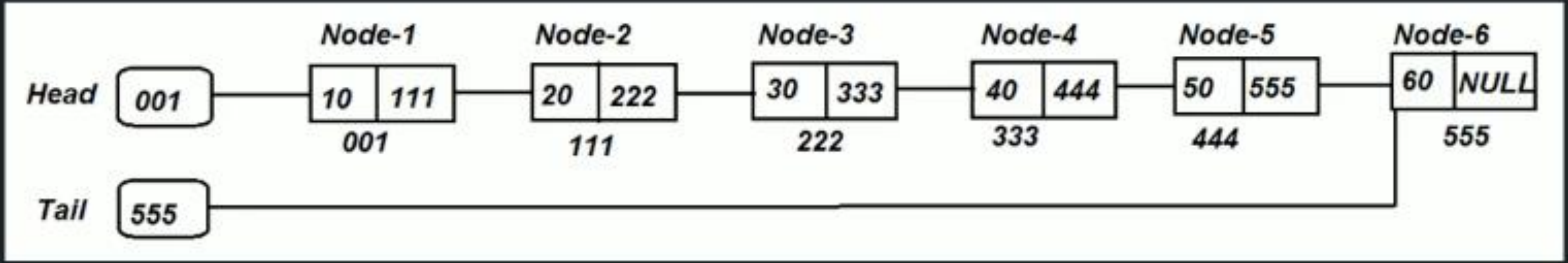
Data Structure and Algorithms

Session-4

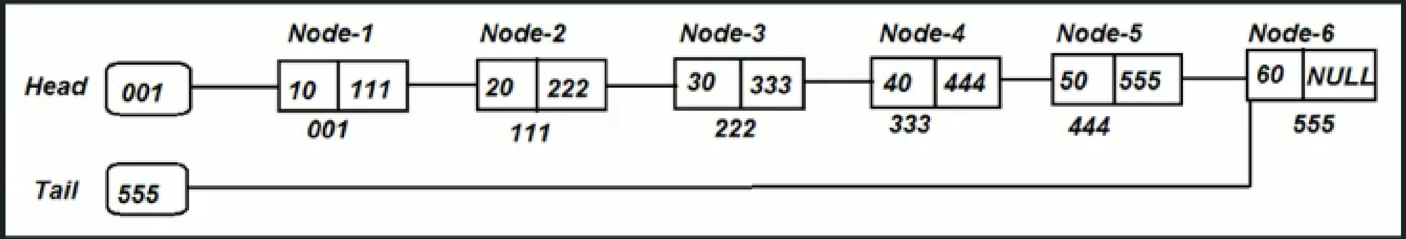
Dr. Subhra Rani Patra
SCOPE, VIT Chennai

Deletion of node from Single Linked List:



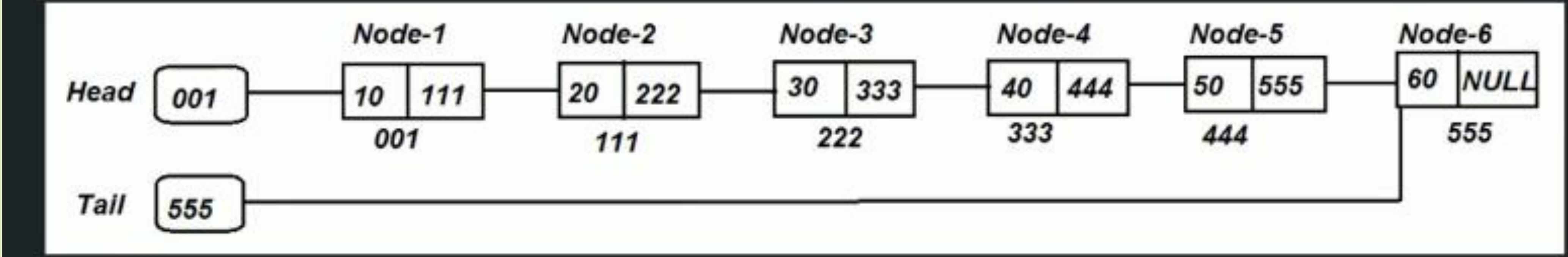
- ✓There can be 3 cases:
  - ✓ Delete first element
  - ✓ Delete last node
  - ✓ Delete any node apart from above 2

Deletion of node from Single Linked List:



```
DeletionOfNode(head, Location):
    if (!existsLinkedList(head))
        return error //Linked List does not exists
    else if (location equals 0)  //we want to delete first node
        head = head.next;
        if this was the only element in list, then update tail = null;
    else if (location >= last)
        if (current node is only node in list) then, head = tail = null; return;
        loop till 2nd last node (tmpNode)
        tail = tmpNode; tmpNode.next = null;
    else // if any internal node needs to be deleted
        loop: tmpNode = start to location-1 //we need to traverse till we find the previous location
        tmpNode.next = tmpNode.next.next  //delete the required node
```

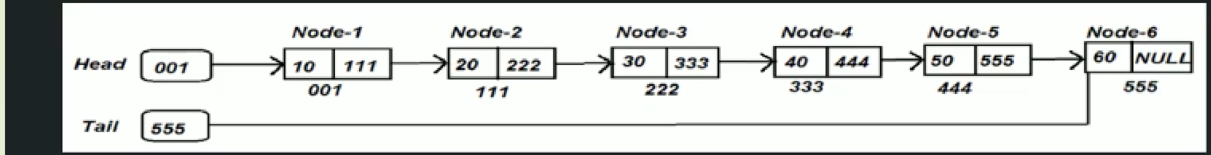
Deletion of entire Single Linked List:



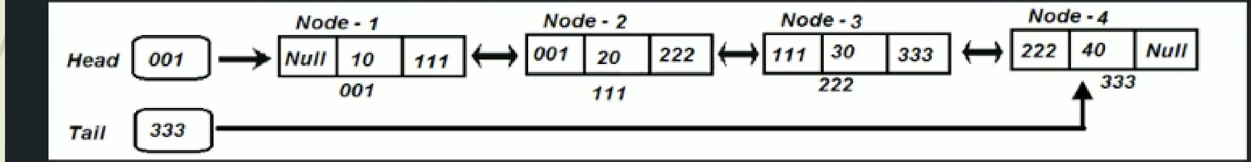
```
DeleteLinkedList(head, tail):

    head =null

    tail = null
```

Single vs Double Linked List:

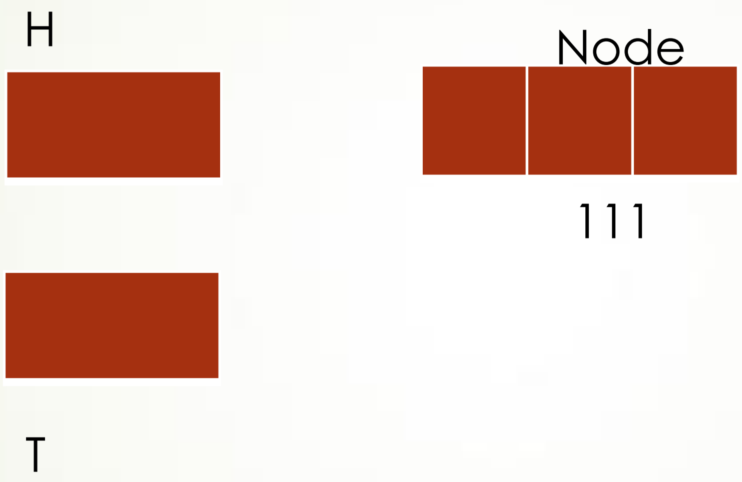
✓***Single linked List:*** *In a singly linked list each node in the list stores the data of the node and a reference to the next node in the list. It does not store any reference to the previous node.*

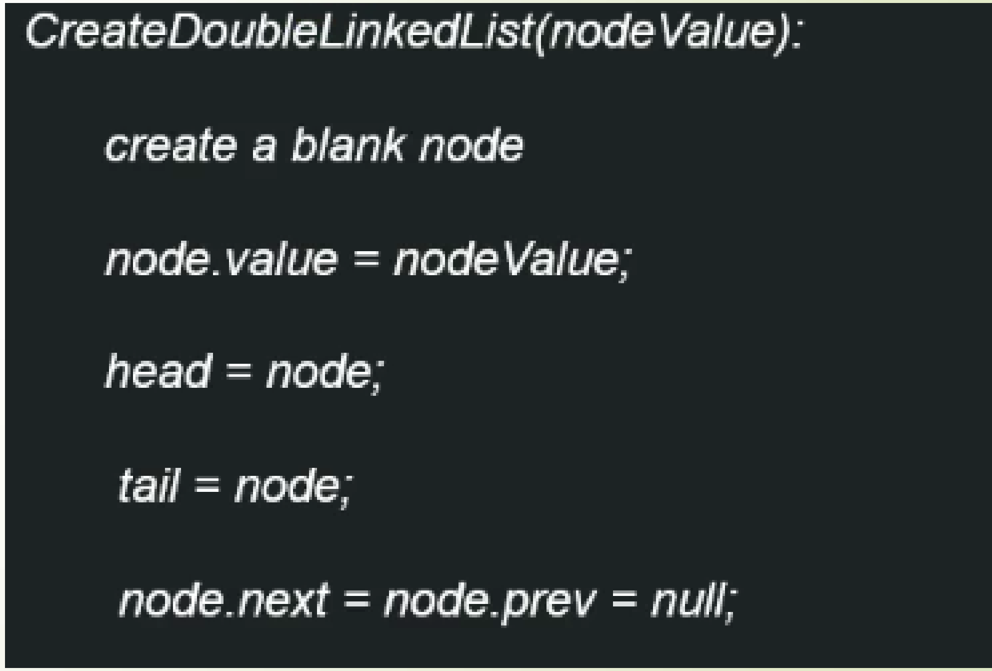


✓***Double Linked List:*** *In double linked list each node contains two references, that references to the previous and next node.*



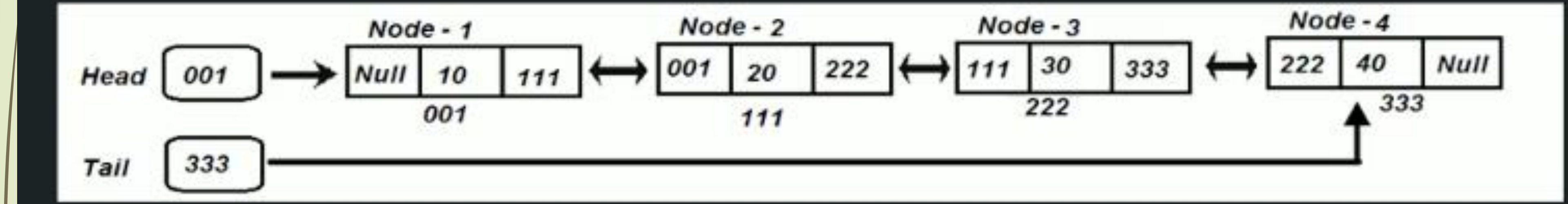
Creation of Double Linked List:





```
CreateDoubleLinkedList(nodeValue):

    create a blank node

    node.value = nodeValue;

    head = node;

     tail = node;

     node.next = node.prev = null;
```

Insertion in Double Linked List:



- ✓ There can be 3 cases:
  - ✓ Insert at start of Linked List
  - ✓ Insert at end of Linked List
  - ✓ Insert at any other place apart from above 2.

Insertion in Double Linked List:

Node

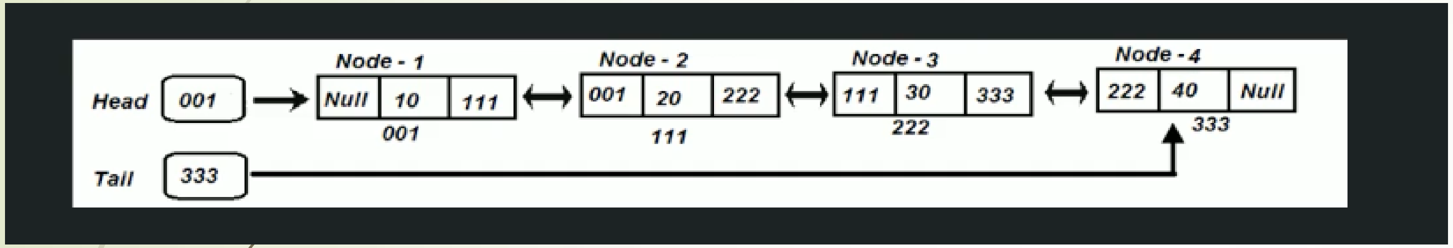
555

Node

555

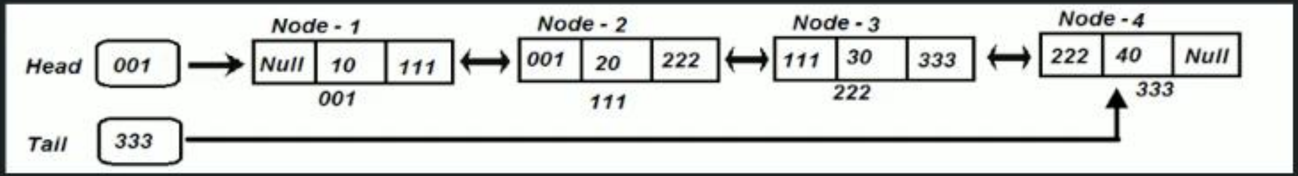
Node

555

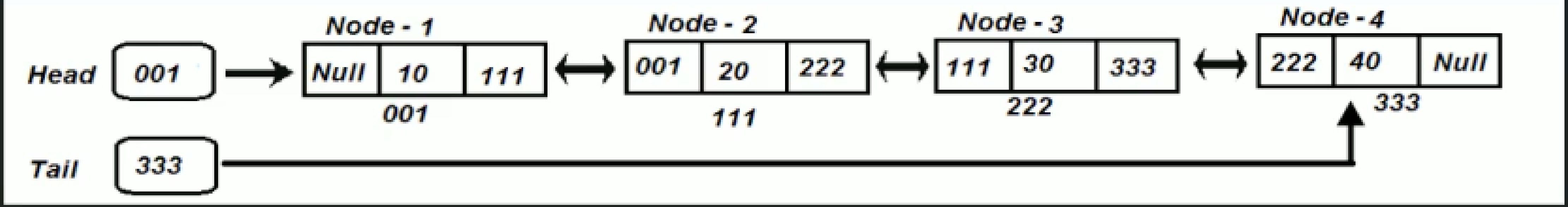


Insertion in Double Linked List:

```
InsertInLinkedList(head, nodeValue, location):
    create a blank node
    node.value = nodeValue;
    if (!existsLinkedList(head))
        return error //Linked List does not exists
    else if (location equals 0) //insert at first position
        node.next = head;
        node.prev = null;
        hea Prev node
        head = node;
    else if (location equals last) //insert at last position
        node.next = null;
        node.prev = tail
        tail.next = node
        tail = node
    else //insert at specified location
        loop: tmpNode = 0 to location-1
        node.next = tmpNode.next; node.prev = tmpNode;
        tmpNode.next = node; node.next.prev = node;
```

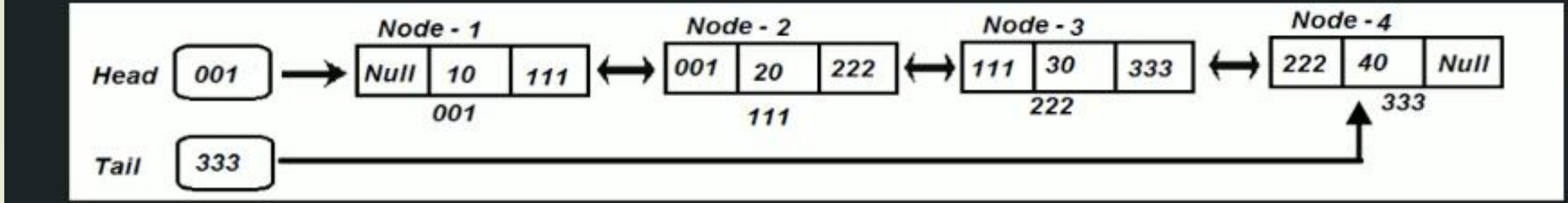


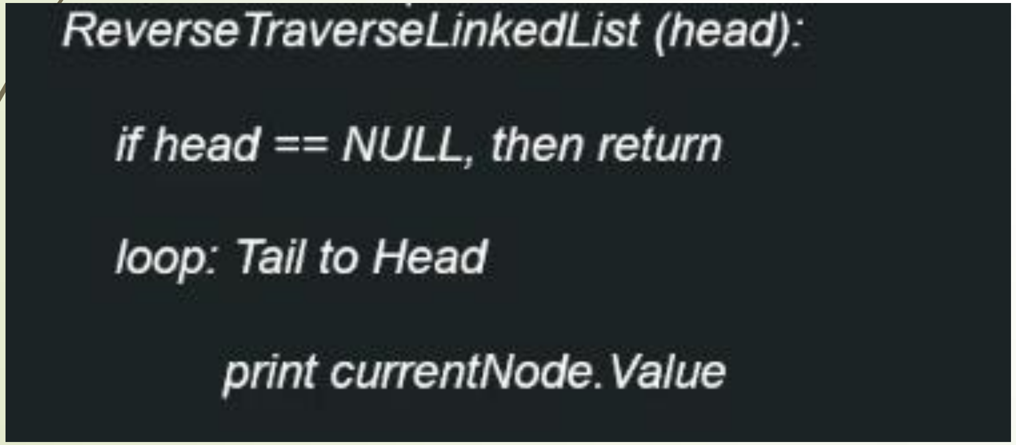
Traversal of Double Linked List:



```
TraverseLinkedList ():
    if head == NULL, then return
    loop: head to tail
        print currentNode.Value
```

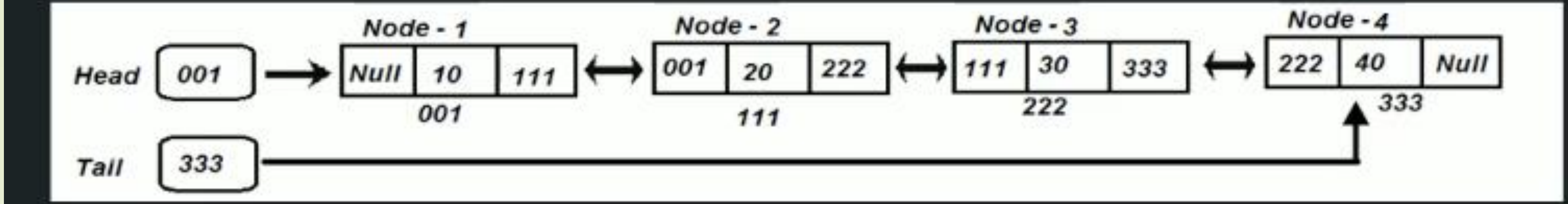
Reverse Traversal of Double Linked List:





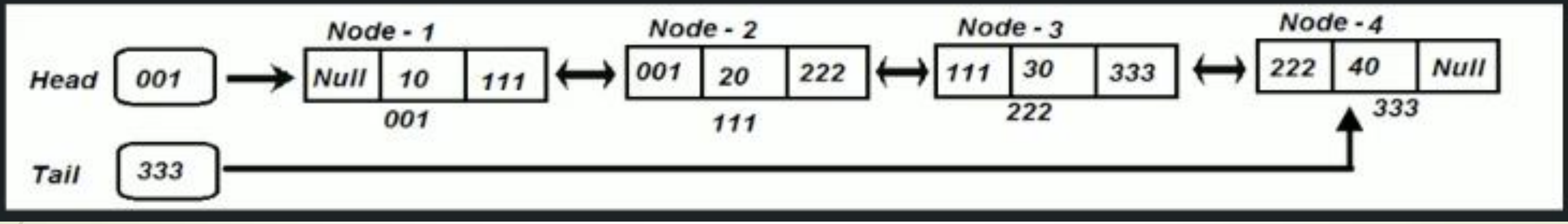
```
ReverseTraverseLinkedList (head):

    if head == NULL, then return

    loop: Tail to Head

            print currentNode.Value
```

Searching a node in Double Linked List:



```
SearchNode(head, nodeValue):
  loop: tmpNode = head to tail
     if (tmpNode.value equals nodeValue)
         print tmpNode.Value //node value found
         return
   return //nodeValue not found
```

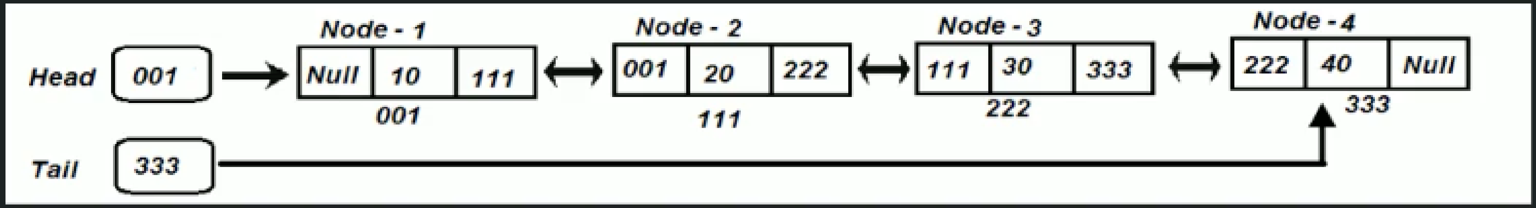
Deletion of node from Double Linked List:



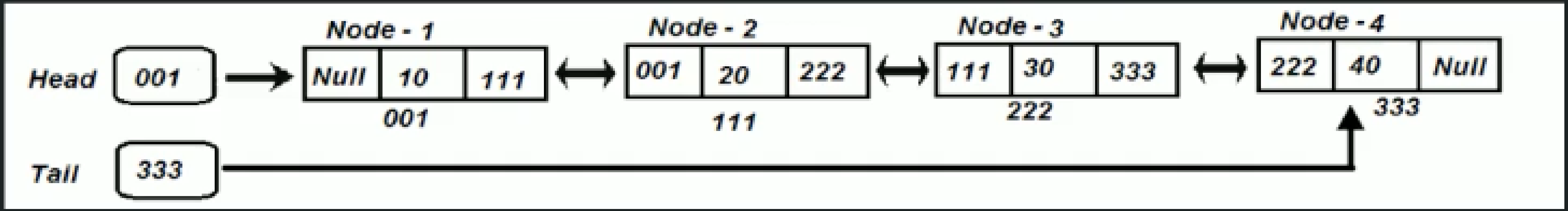
- ✓ There can be 3 cases:
  - ✓ Delete first node
  - ✓ Delete last node
- ✓ Delete any node apart from above 2

Deletion of node from Double Linked List:

```
DeletionOfNode(head, Location):
    if (!existsLinkedList(head))
        return error //Linked List does not exists
    else if (location equals 0)  //we want to delete first element
        if this was the only element in list, then update head = tail = null; return
        head = head.next; head.prev = null
    else if (location >= last)
        if this was the only element in list, then update head = tail = null; return
        tail = tail.prev; tail.next = null;
    else // if any internal node needs to be deleted
        loop: tmpNode = start to location-1 //we need to traverse till we find the previous location
        tmpNode.next = tmpNode.next.next  //link current node with next node
        tmpNode.next.prev = tmpNode //link  next node with current node
```



Deletion of entire Double Linked List:



```
DeleteLinkedList(head, tail):
  loop(tmp : head to tail)
    tmp.prev = null;
  head = tail = null
```

Thank you