Data Structure and Algorithms

Session-5

Dr. Subhra Rani Patra
SCOPE, VIT Chennai

Creation of Circular Single Linked List:

111

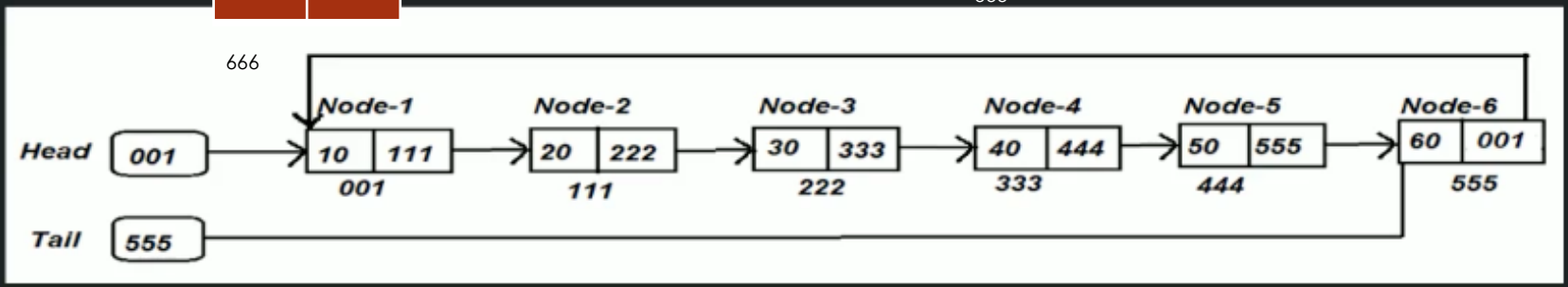
```
CreateSingleLinkedList (nodeValue):

    create a blank node

    node.value = nodeValue;

    node.next = node

    head = node;

    tail = node;
```

Insertion in Circular Single Linked List:

666

666

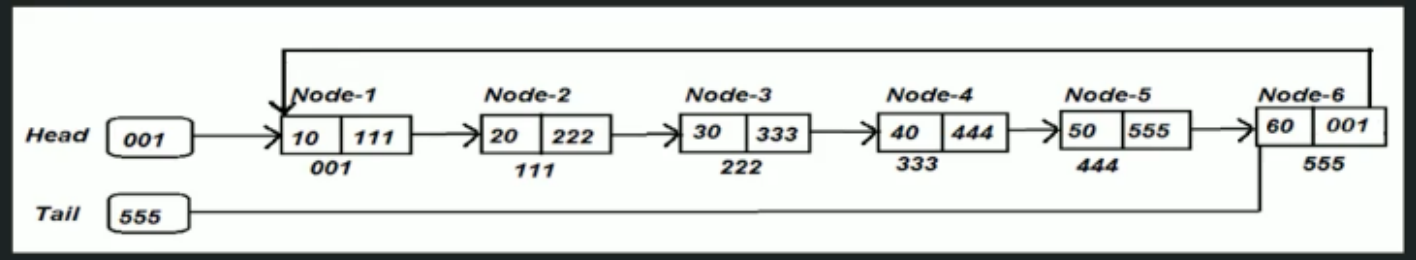
666



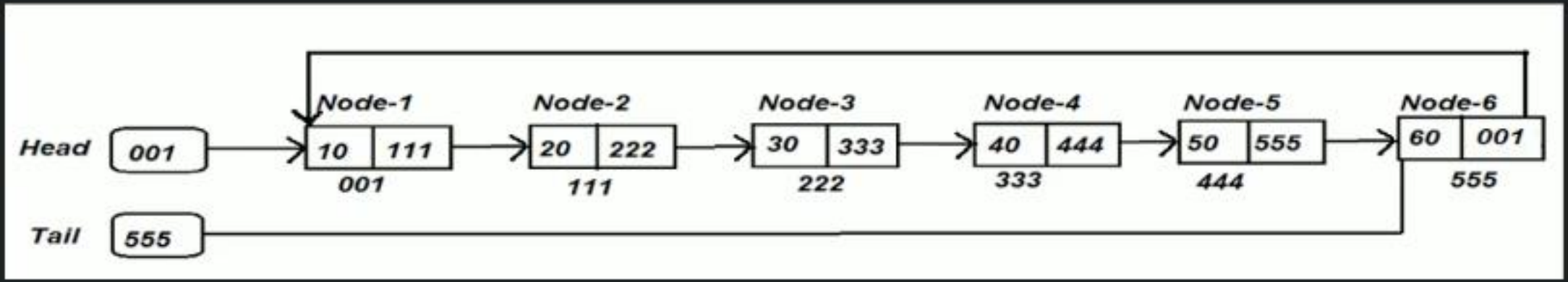
- ✓ There can be 3 cases:
  - ✓ Insert at start of Linked List
  - ✓ Insert at a specified Location in Linked List
  - ✓ Insert at end of Linked List

Insertion in Circular Single 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;

      head = node; tail.next = head;

 else if (location equals last) //insert at last position

      node.next = head;

      tail.next = node

      tail = node   //to keep track of last node

  else //insert at specified location

    loop: tmpNode = 0 to location-1  //loop till we reach specified node and end the loop

    node.next = tmpNode.next

    tmpNode.next = node
```

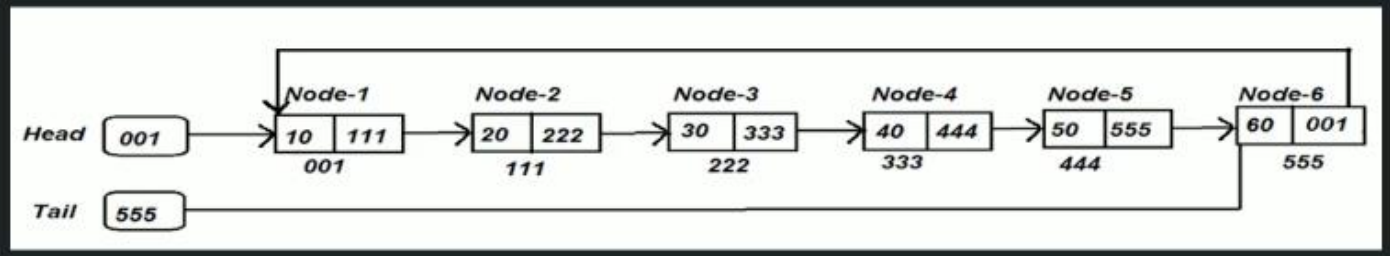


Traversal of Circular Single Linked List:



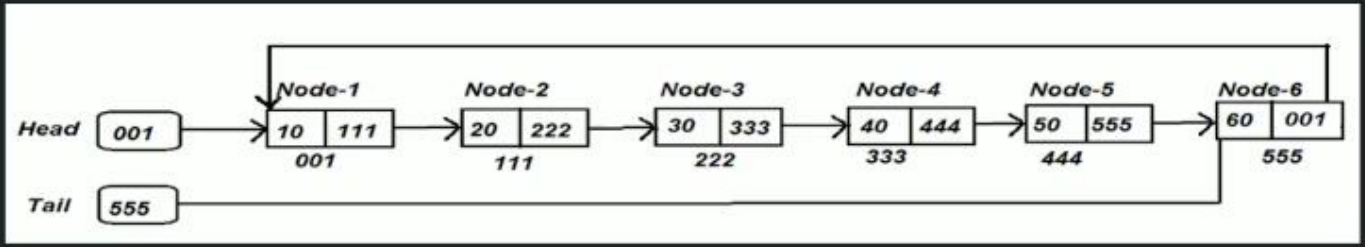

```
TraverseLinkedList (head):

    if head == NULL, then return

    loop: head to tail

        print currentNode.Value
```

Searching a node in Circular Single Linked List:



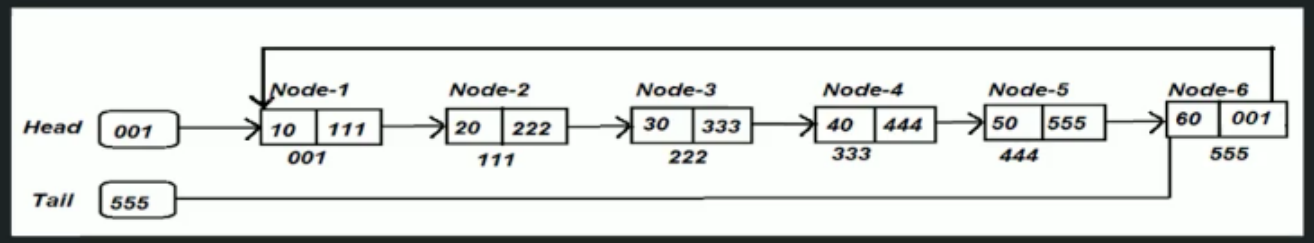

```
SearchNode(head, nodeValue):
    loop: tmpNode = start to tail
        if (tmpNode.value equals nodeValue)
            print tmpNode.Value //node value found
            return
    return //nodeValue not found
```

Deletion of node from Circular Single Linked List:



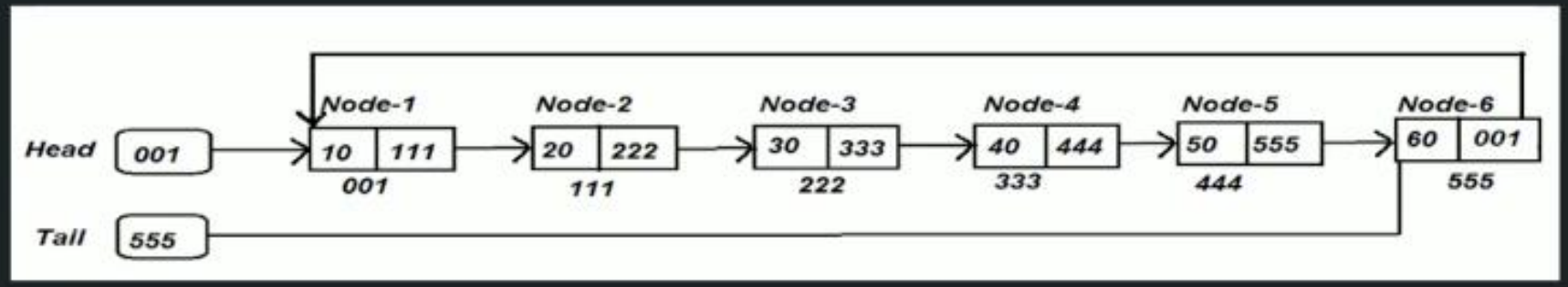
- ✓ There can be 3 cases:
  - ✓ Delete first node
  - ✓ Delete last node
  - ✓ Delete any node apart from above 2

Deletion of node from Circular 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 element

        head = head.next; tail.next = head

        if this was the only element in list, then update head = tail = node.next = null;

    else if (location >= last)

        if (current node is only node in list) then, head = tail = node.next = null; return;

        loop till 2nd last node (tmpNode)

        tail = tmpNode; tmpNode.next = head;

    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
```

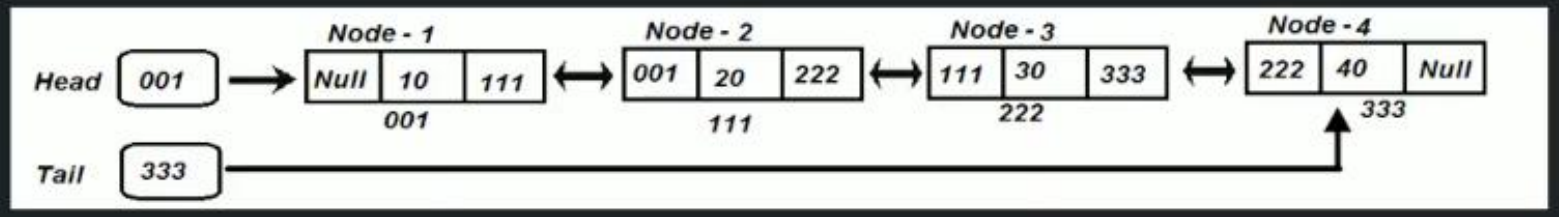
Deletion of entire Circular Single Linked List:



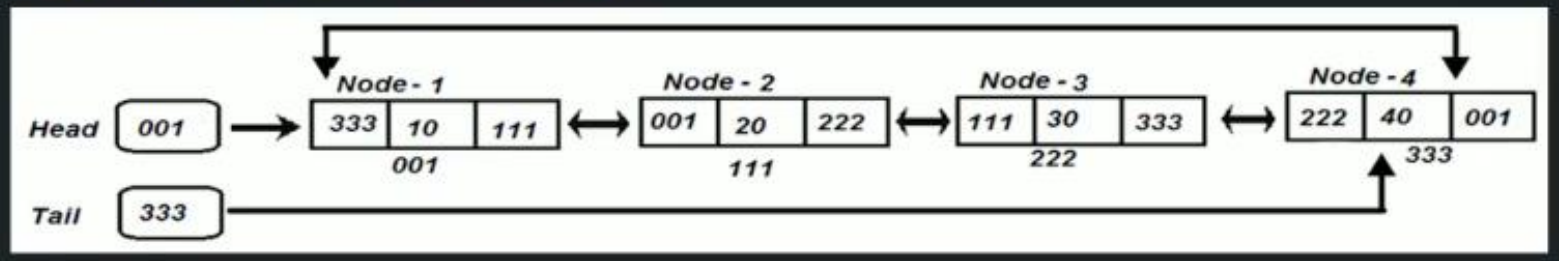

```
DeleteLinkedList(head, tail):

    head =NULL

    tail.next = NULL

    tail =NULL
```

Double Linked List vs Circular Double Linked List:

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



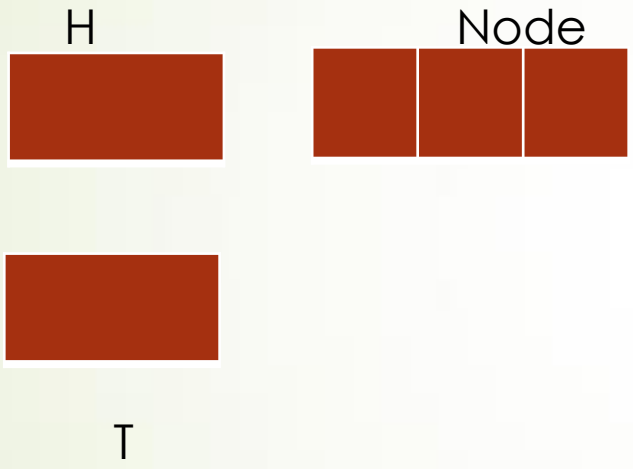
✓ **Circular Double Linked List:** *In the case of a circular doubly linked list, the only change that occurs is that the end of the given list is linked back to the front of the list and vice versa.*



Common operations of Circular Double Linked List:

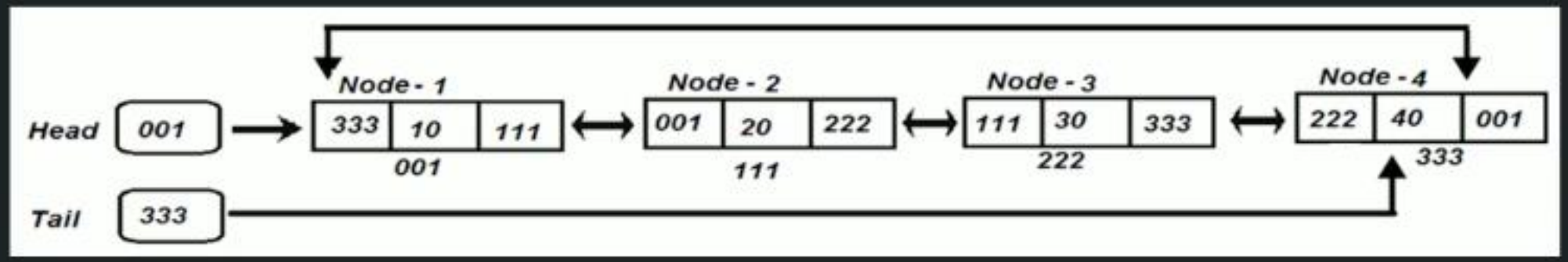
- ✓ Insertion of node in circular Double Linked List
- ✓ Traversal of circular Double Linked List
- ✓ Searching a value in circular Double Linked List
- ✓ Deletion of a node from a circular Double Linked List
- ✓ Deletion of circular Double Linked List

Creation of Circular Double Linked List:




```
CreateCircularDoubleLinkedList(nodeValue):

    create a blank node

    node.value = nodeValue;

    head = node;

    tail = node;

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

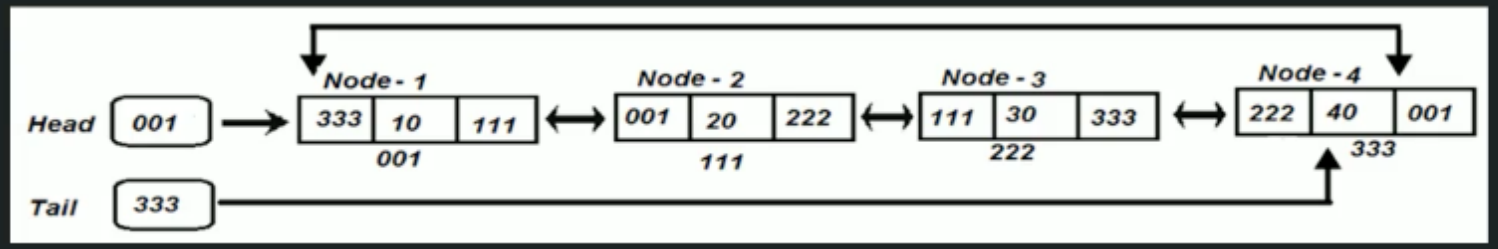
Insertion in Circular Double Linked List:



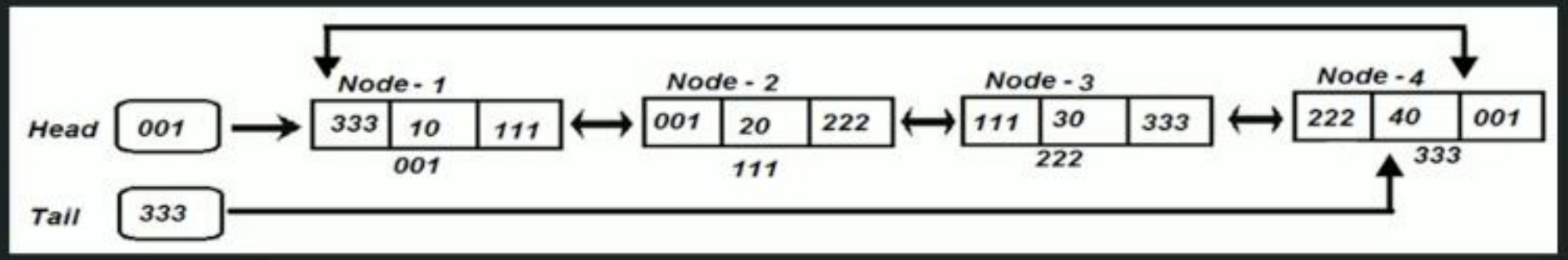
- ✓ There can be 3 cases:
  - ✓ Insert at start of Linked List
  - ✓ Insert at end of Linked List
  - ✓ Insert at any other place apart from above 2.

Insertion in Circular 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 = tail;

        head.prev= node

        head = node; tail.next = node;

    else if (location equals last) //insert at last position

        node.next = head;  node.prev = tail

        head.prev = node

        last.next = node

        tail = node   //to keep track of last node

    else //insert at specified location

        loop: tmpNode = 0 to location-1  //loop till we reach specified node

        node.next = tmpNode.next; node.prev = tmpNode;

        tmpNode.next = node; node.next.prev = node;
```

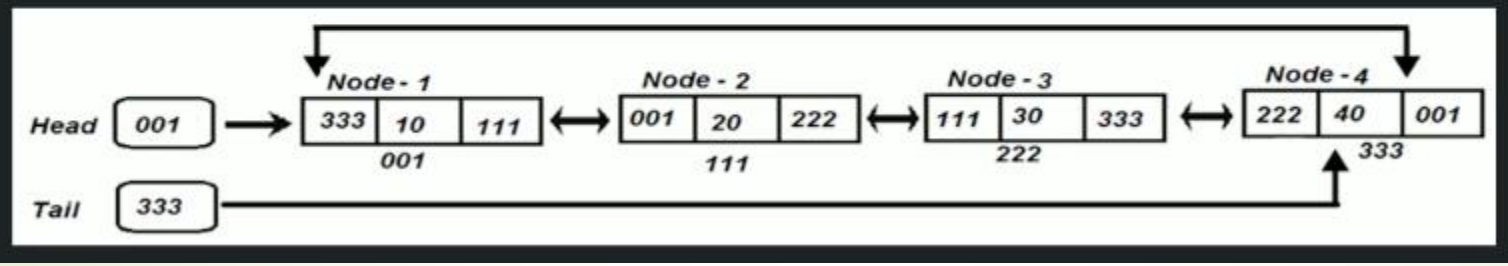


Traversal of Circular Double Linked List:



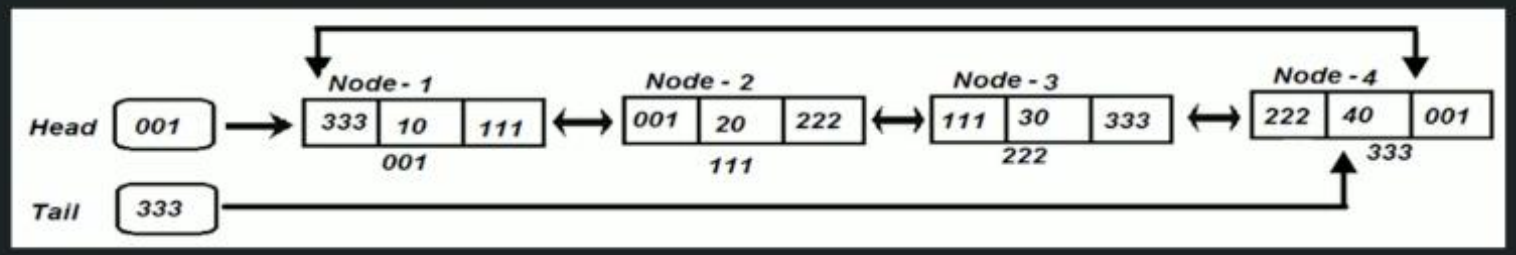

```
TraverseLinkedList ():

    if head == NULL, then return

    loop: head to tail

        print currentNode.Value
```

Reverse Traversal of Circular Double Linked List:



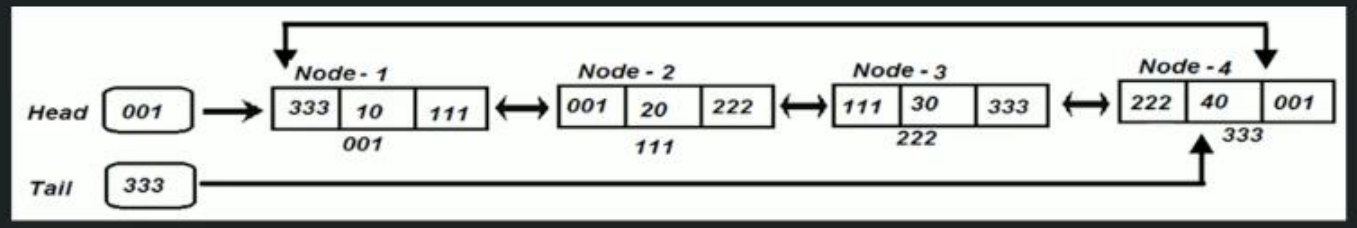

```
ReverseTraverseLinkedList ():

    if head == NULL, then return

    loop: tail to head

        print currentNode.Value
```

Searching a node in Circular Double Linked List:



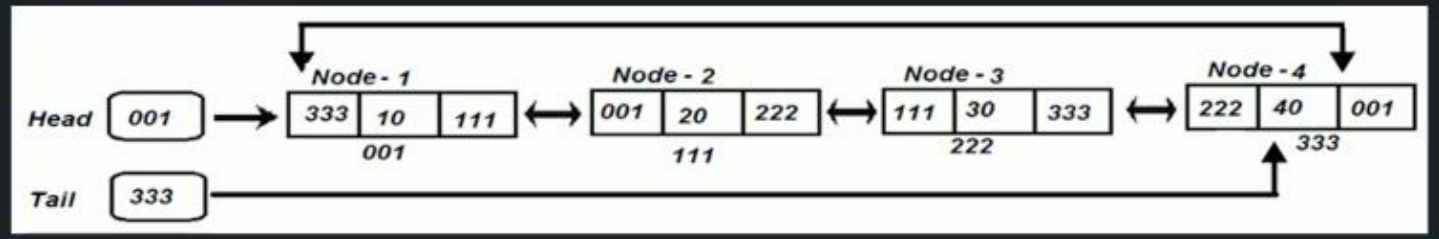

```
SearchNode(nodeValue):

    loop: tmpNode = start to tail

        if (tmpNode.value equals nodeValue)

            print tmpNode.Value //node value found

            return

    return //nodeValue not found
```

Deletion of node from Circular Double Linked List:



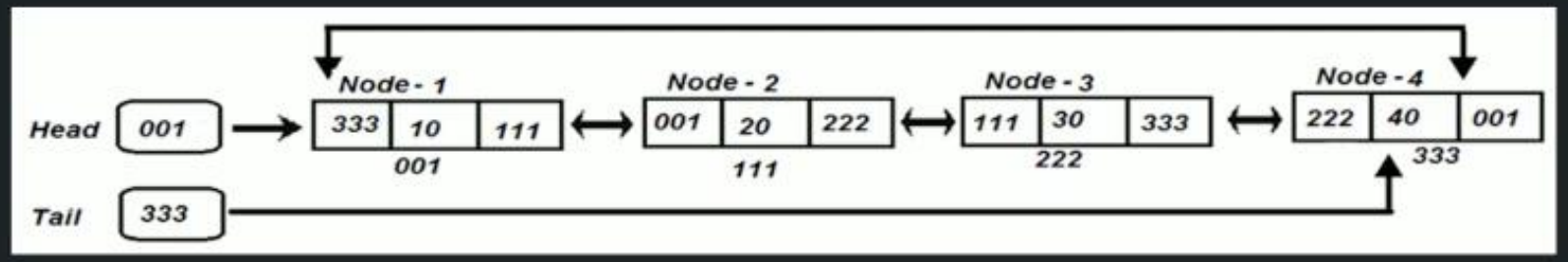
- ✓ There can be 3 cases:
  - ✓ Delete first node
  - ✓ Delete last node
  - ✓ Delete any node apart from above 2

Deletion of node from Circular 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.next = head.prev = head = tail = null; return

            head = head.next; head.prev = tail; tail.next = head;

    else if (location >= last)

            if this was the only element in list, then update head.next = head.prev = head = tail = null; return

            tail = tail.prev; tail.next = head; head.prev=tail

    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

            tmpNode.next.prev = tmpNode
```

Deletion of entire Circular Double Linked List:




```
DeleteLinkedList(head, tail):

    tail.next = null;

    loop(tmp : head to tail)

        tmp.prev = null;

    head = tail = null
```


Thank you