Data Structure and Algorithms

Session-3

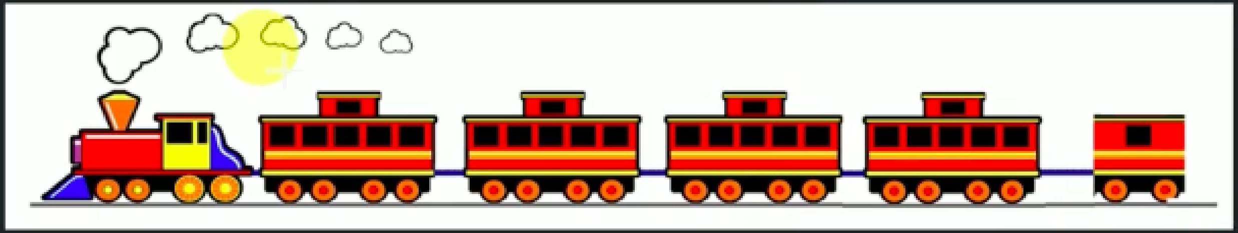
Dr. Subhra Rani Patra
SCOPE, VIT Chennai

Linked List

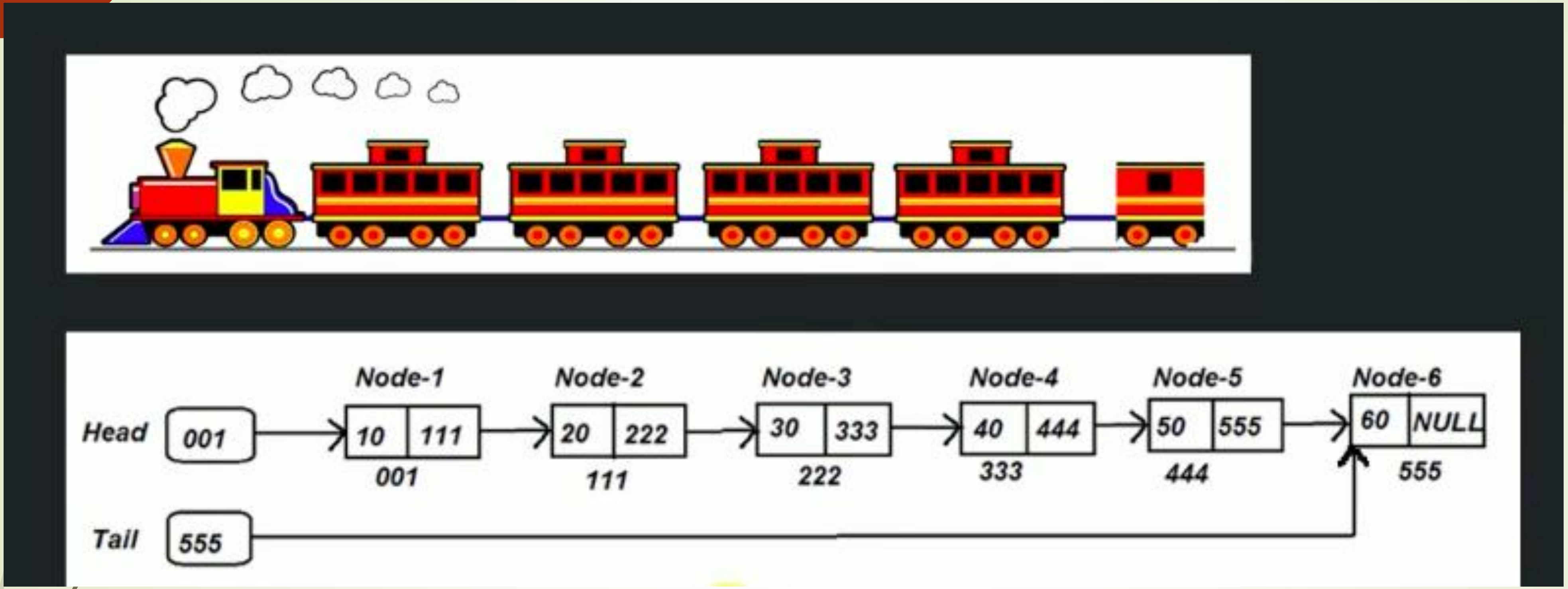
What is Linked List ?

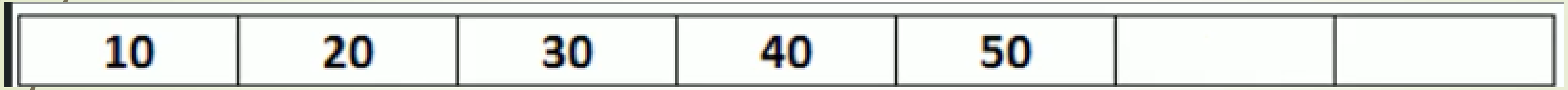
✓*A linked list is a linear data structure where each element is a separate object. Each element (node) of a list comprises of two items - the data and a reference to the next node. The most powerful feature of Linked List is that it is of variable size.*

✓*Example:*



Linked List Contd…

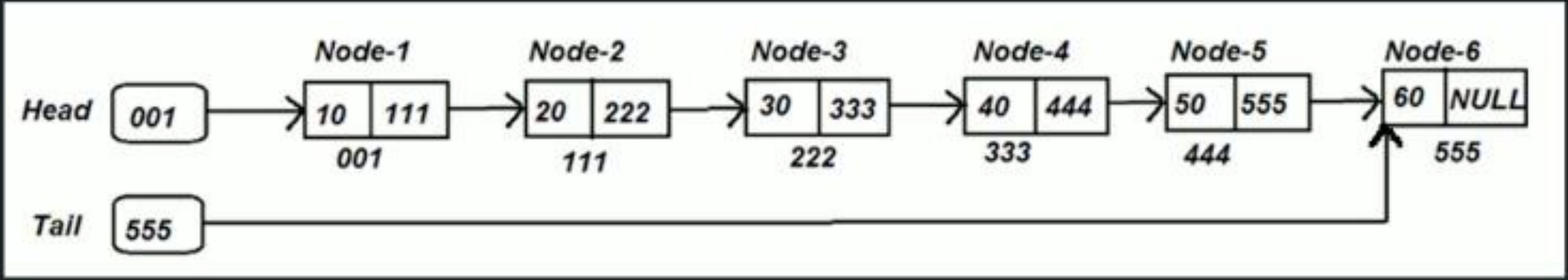




10	20	30	40	50		

Linked list vs Array
- Separate object
- Variable size
- Random access

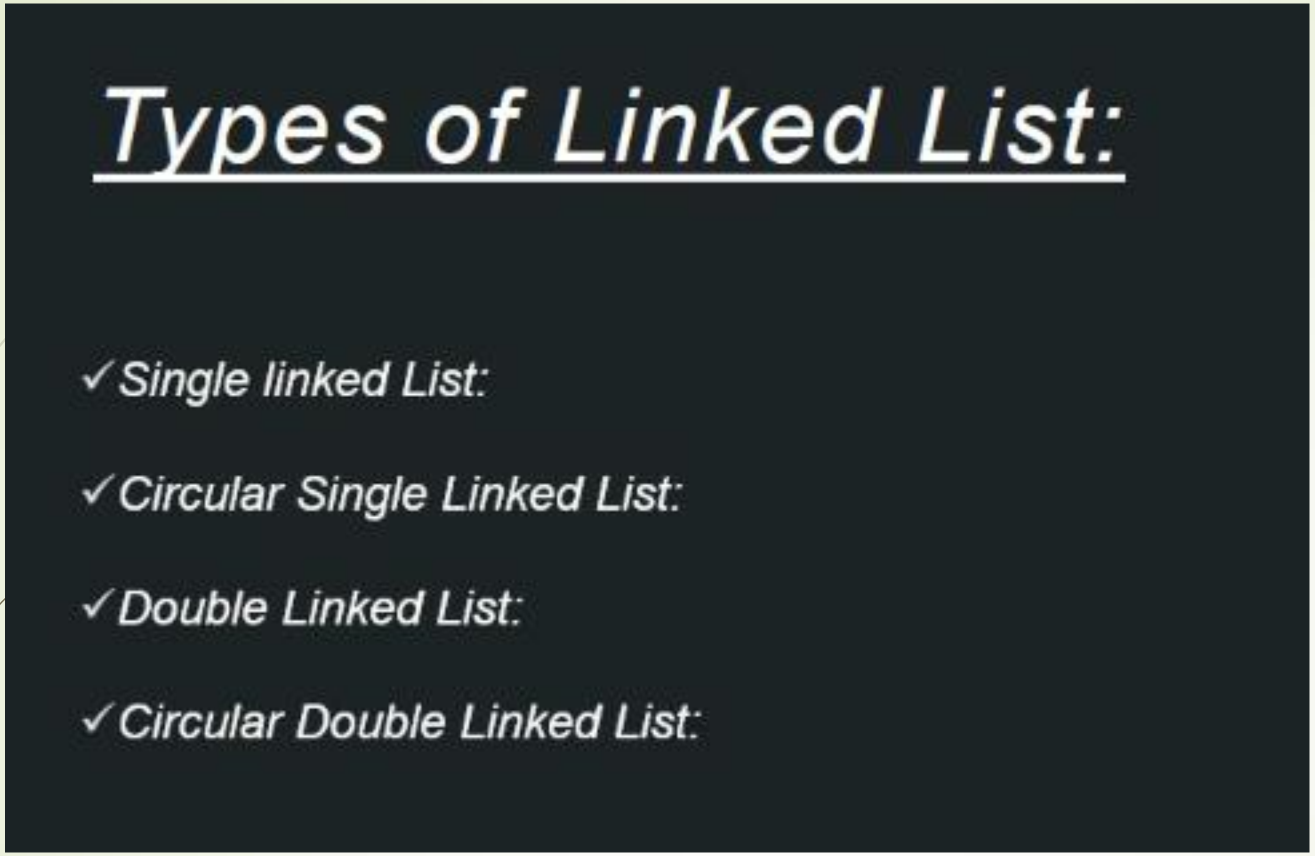
Components of Linked List:



Node: *Contains Data & Reference to next Node.*

Head: *Reference to first node in the list.*

Tail: *Reference to last node of the list.*

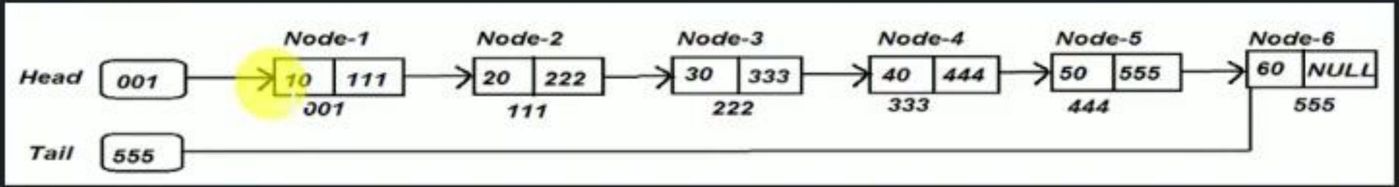


Types of Linked List:

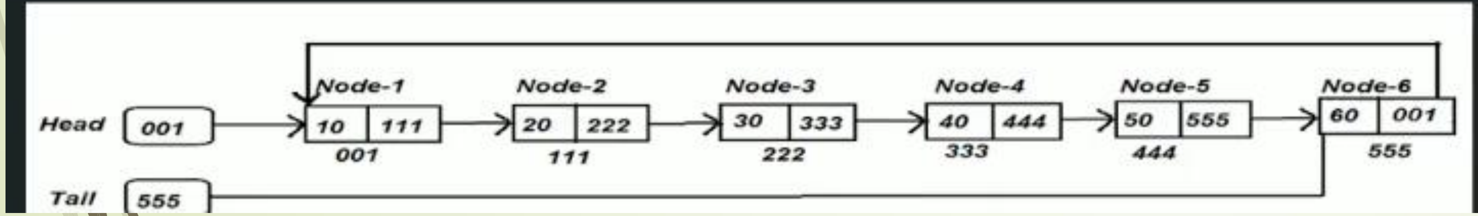
- ✓Single linked List:
- ✓Circular Single Linked List:
- ✓Double Linked List:
- ✓Circular Double Linked List:

Types of Linked List:

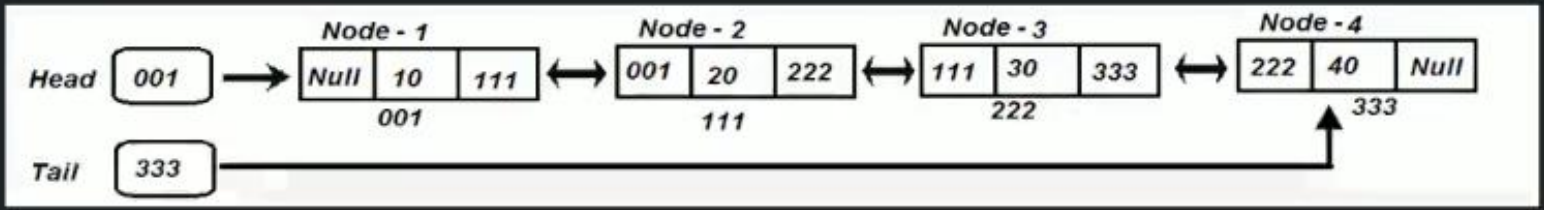
✓**Single linked List:** *In a singly linked list each node in the list stores the data of the node and a reference to the next node in the list. It does not store any reference to the previous node.*



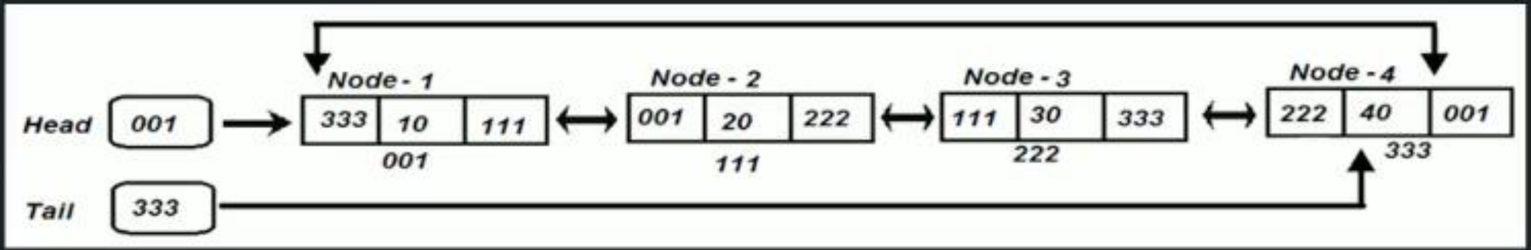
✓**Circular Single Linked List:** *In the case of a circular doubly linked list, the only change that occurs is that the end of the given list is linked back to the front.*

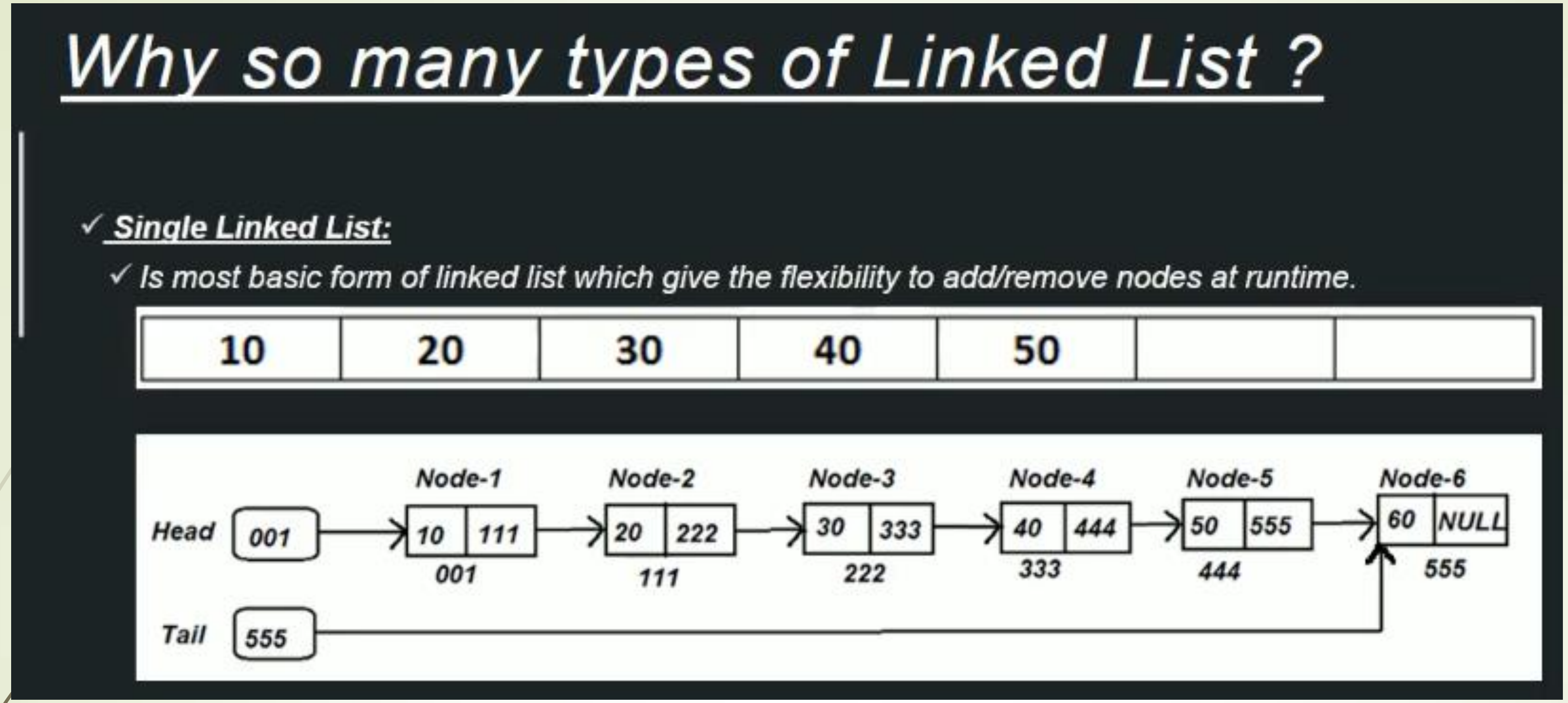


✓***Double Linked List:*** *In double linked list each node contains two references, that references to the previous and next node.*



✓***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.*



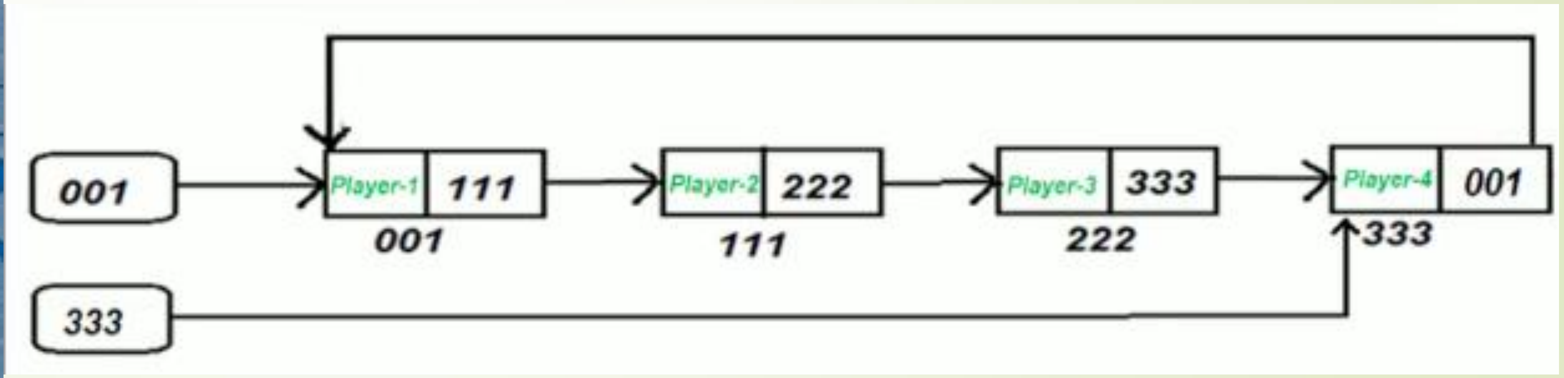


Why so many types of Linked List ?

✓ **Single Linked List:**

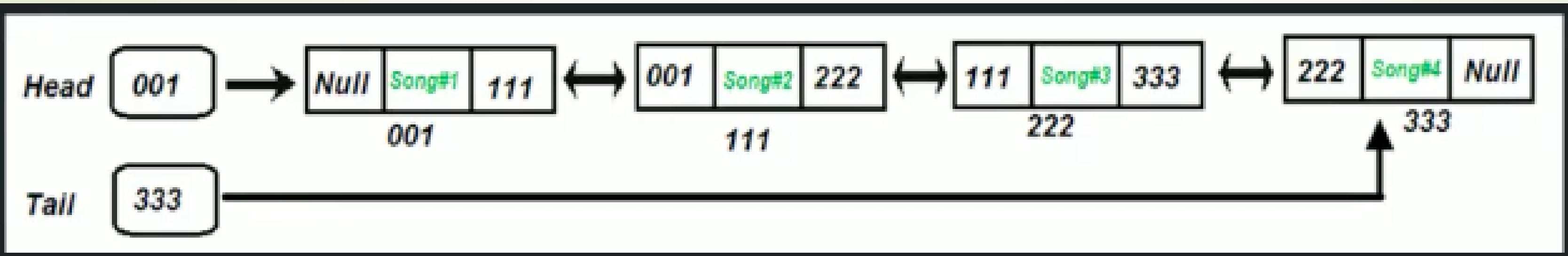
✓ Is most basic form of linked list which give the flexibility to add/remove nodes at runtime.

10	20	30	40	50		



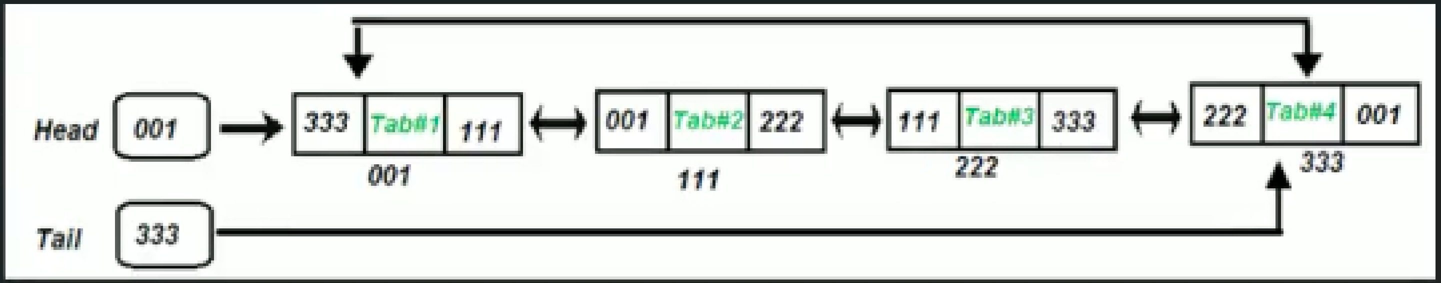
✓ ***Double Linked List:***

- ✓ *When we want to move in both direction depending on requirement.*
- ✓ *Example:* *Music player which has next and prev buttons.*

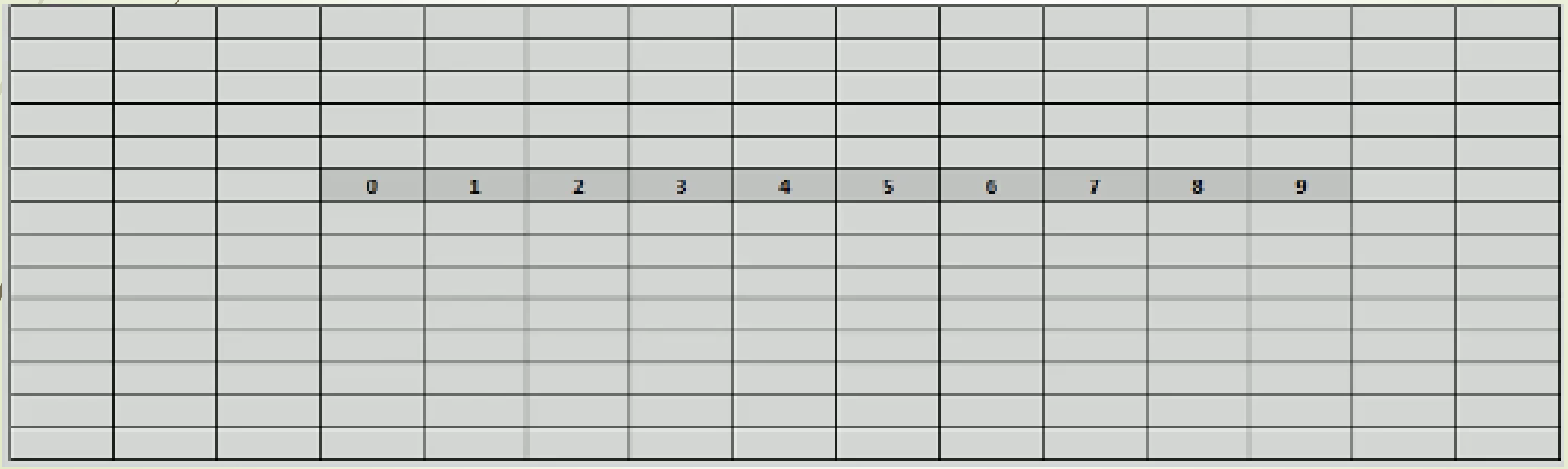


✓ ***Circular Double Linked List:***

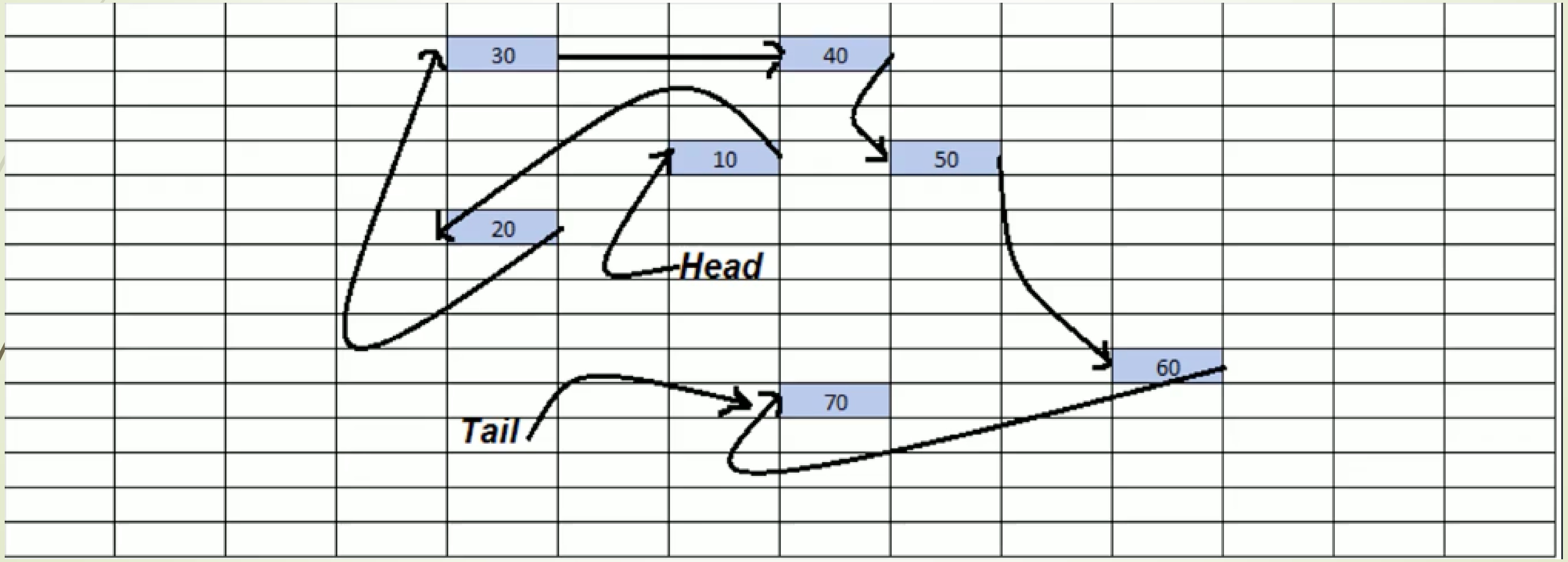
- ✓ *When we want to loop through the list indefinitely until the list exists. We also want to move both foreword and backward.*
- ✓ *Example:* *"Alt+Tab" button in Windows.*

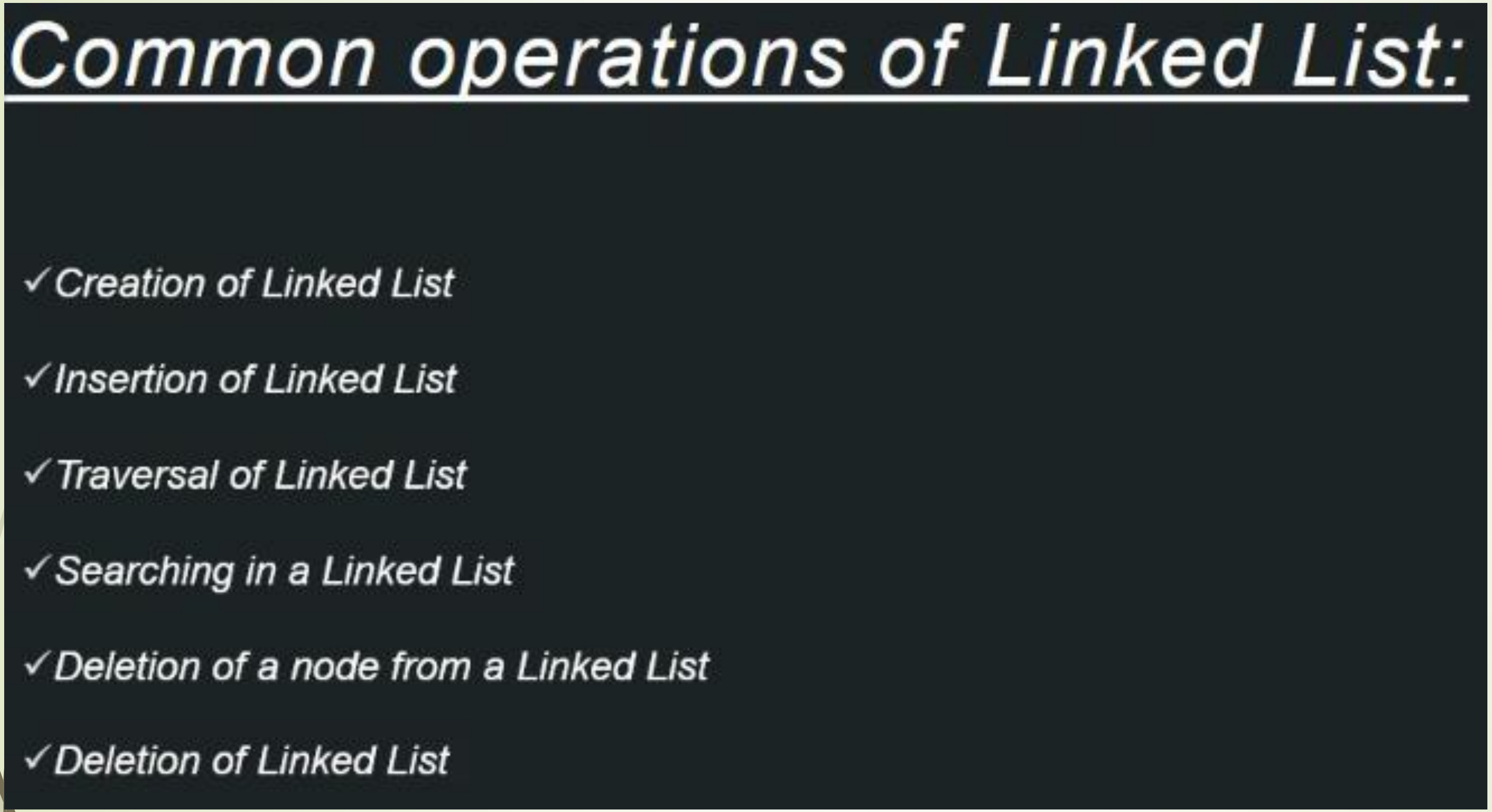


How is an Array represented in Memory ?



How is Linked List represented in Memory ?

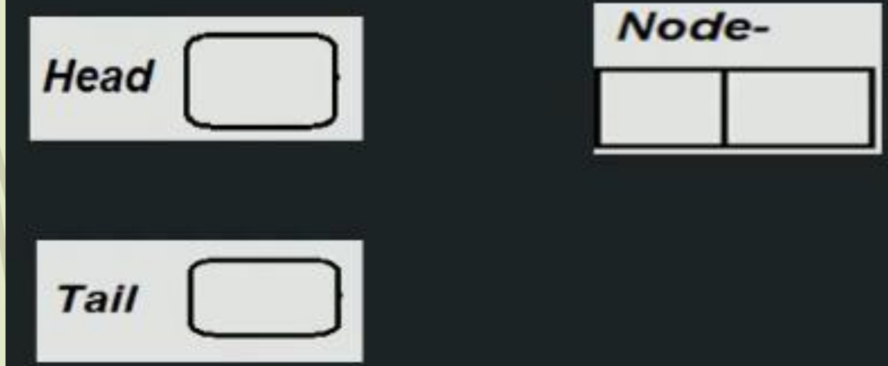




Common operations of Linked List:

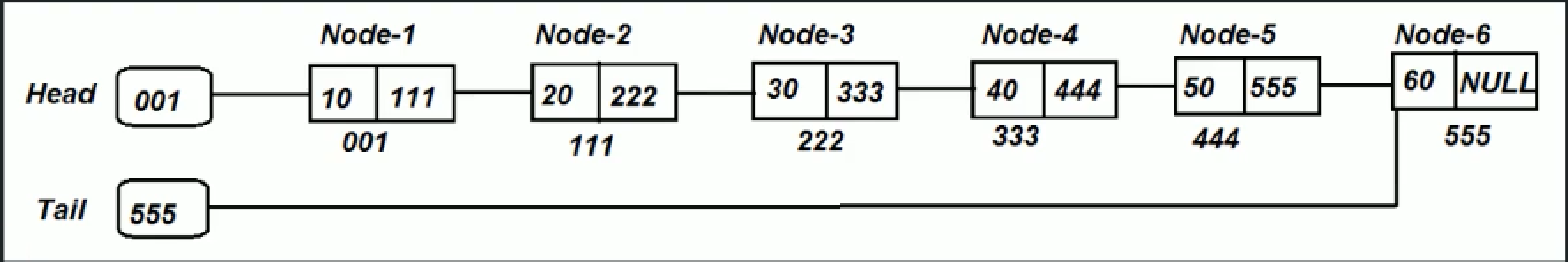
- ✓Creation of Linked List
- ✓Insertion of Linked List
- ✓Traversal of Linked List
- ✓Searching in a Linked List
- ✓Deletion of a node from a Linked List
- ✓Deletion of Linked List

Creation of Single Linked List:



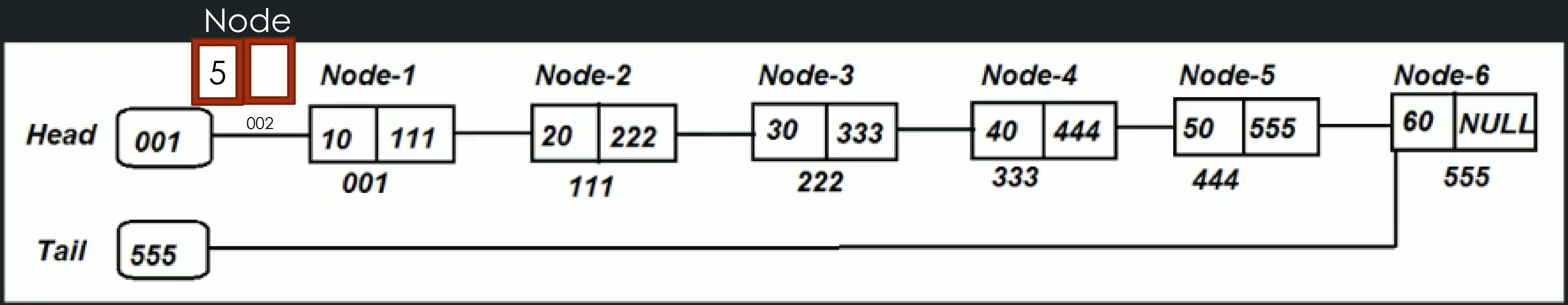
```
CreateSingleLinkedList(nodeValue):
    create a head, tail pointer and initialize with NULL
    create a blank node
    node.value = nodeValue;
    node.next = null;
    head = node;
    tail = node;
```

Insertion in Single Linked List:

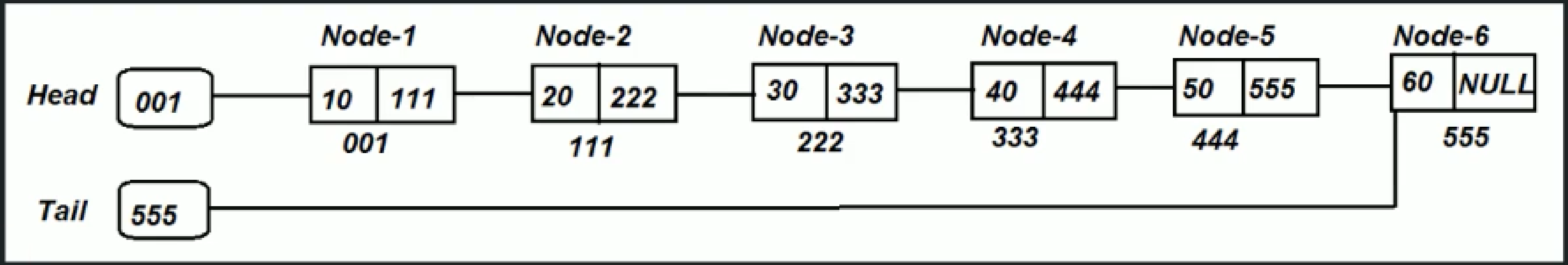


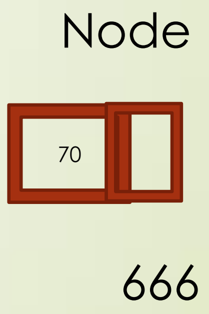
- ✓ There can be 3 cases:
  - ✓ Insert at start of Linked List
  - ✓ Insert at end of Linked List
  - ✓ Insert at a specified Location in Linked List

Insertion in Single Linked List:

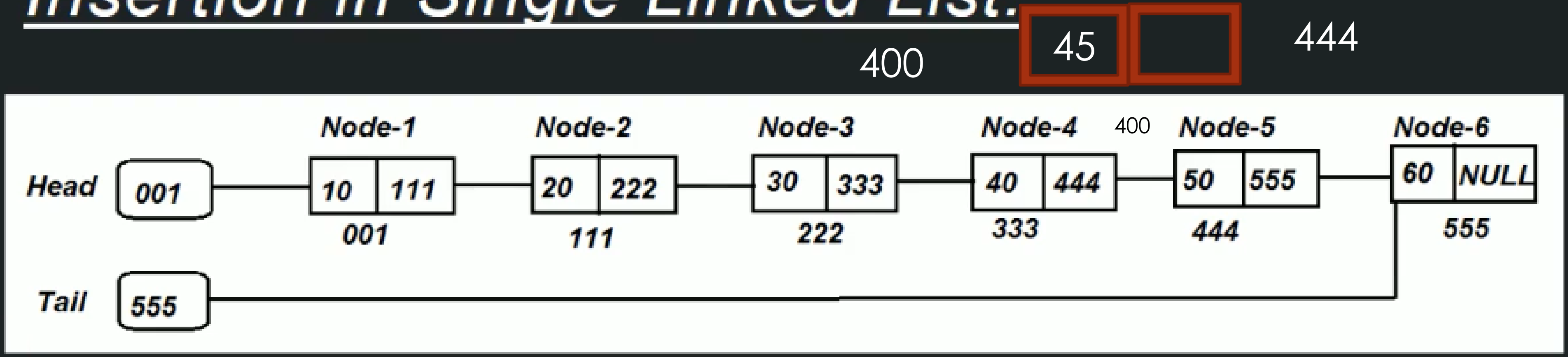


Insertion in Single Linked List:

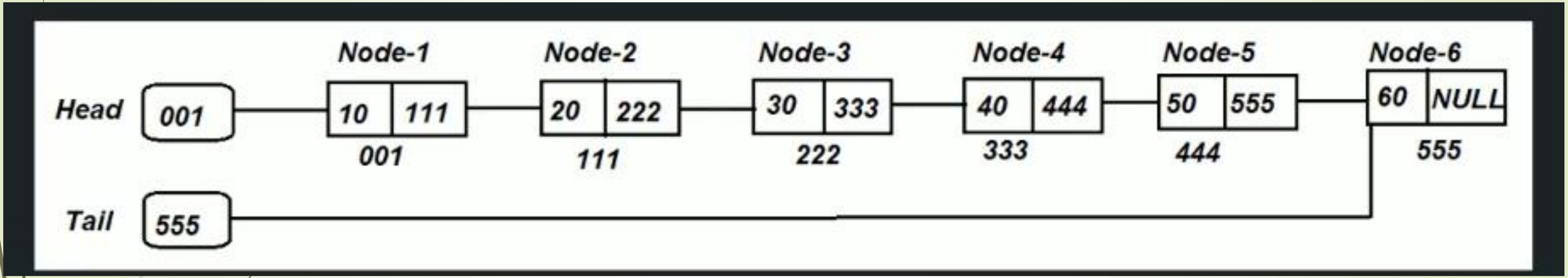


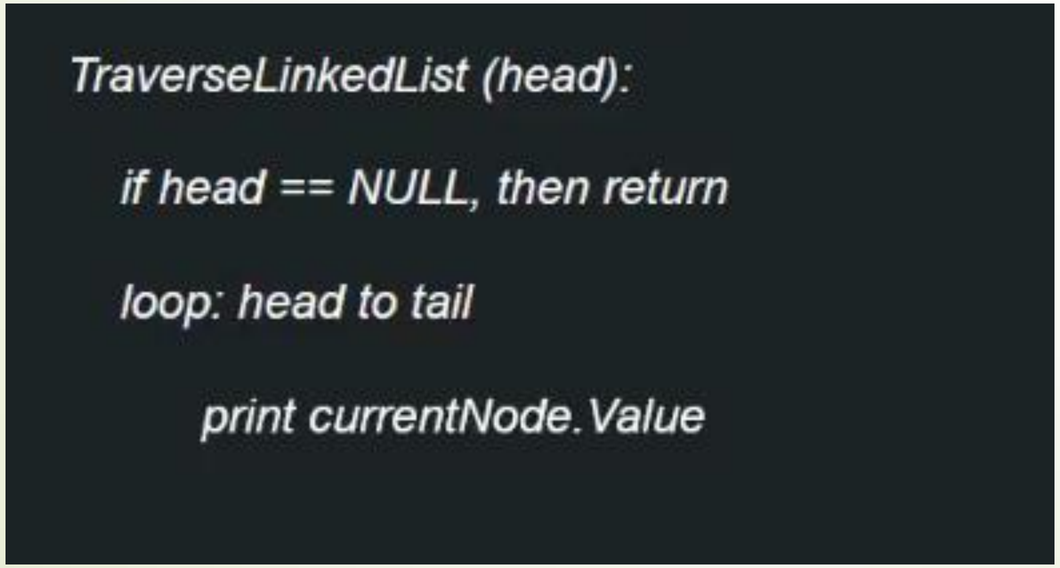


Insertion in Single Linked List:



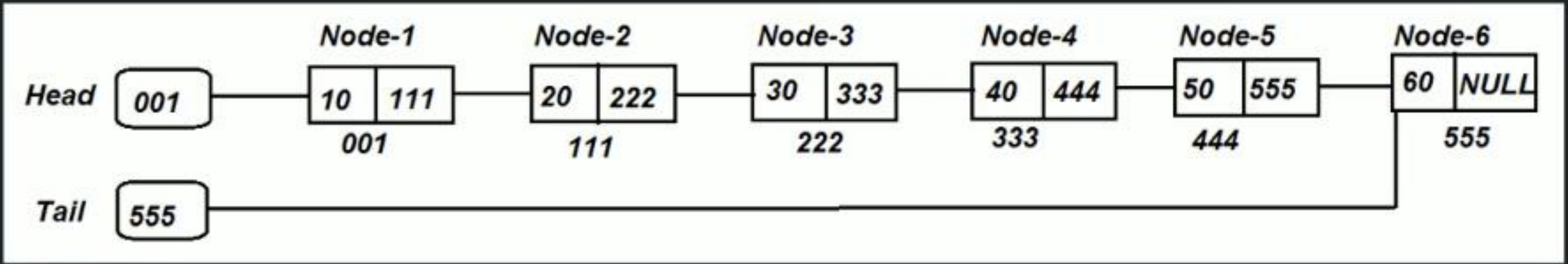
Traversal of a linked list





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

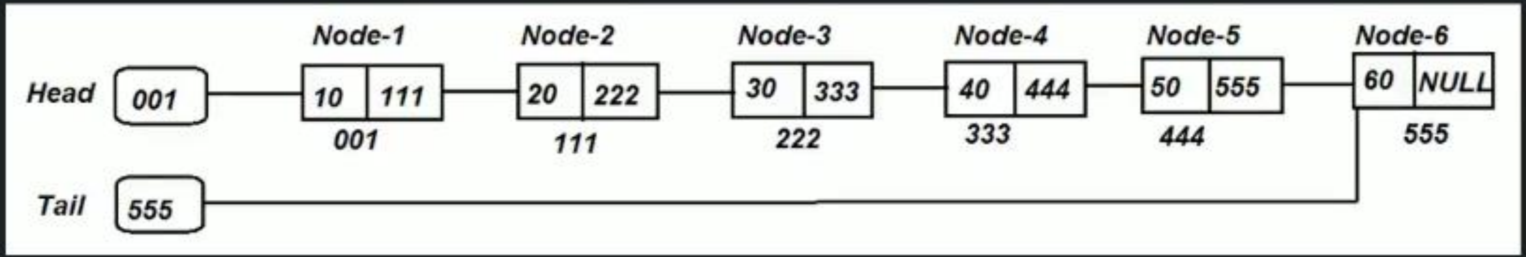
Searching a node in Single Linked List:



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

Insertion in 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;
    else if (location equals last) //insert at last position
         node.next = null;
         last.next = node
         last = 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
```



Thank you