Google Developer Student Club
Meerut Institute of Engineering and Technology

DSA BOOTCAMP

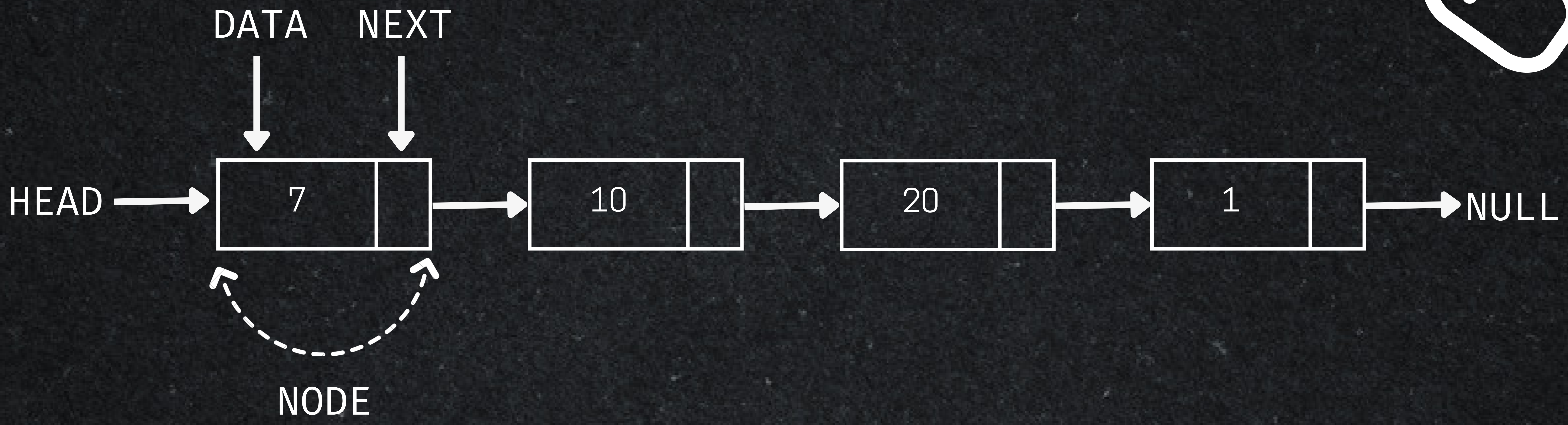
SESSION 4

19FEB2023

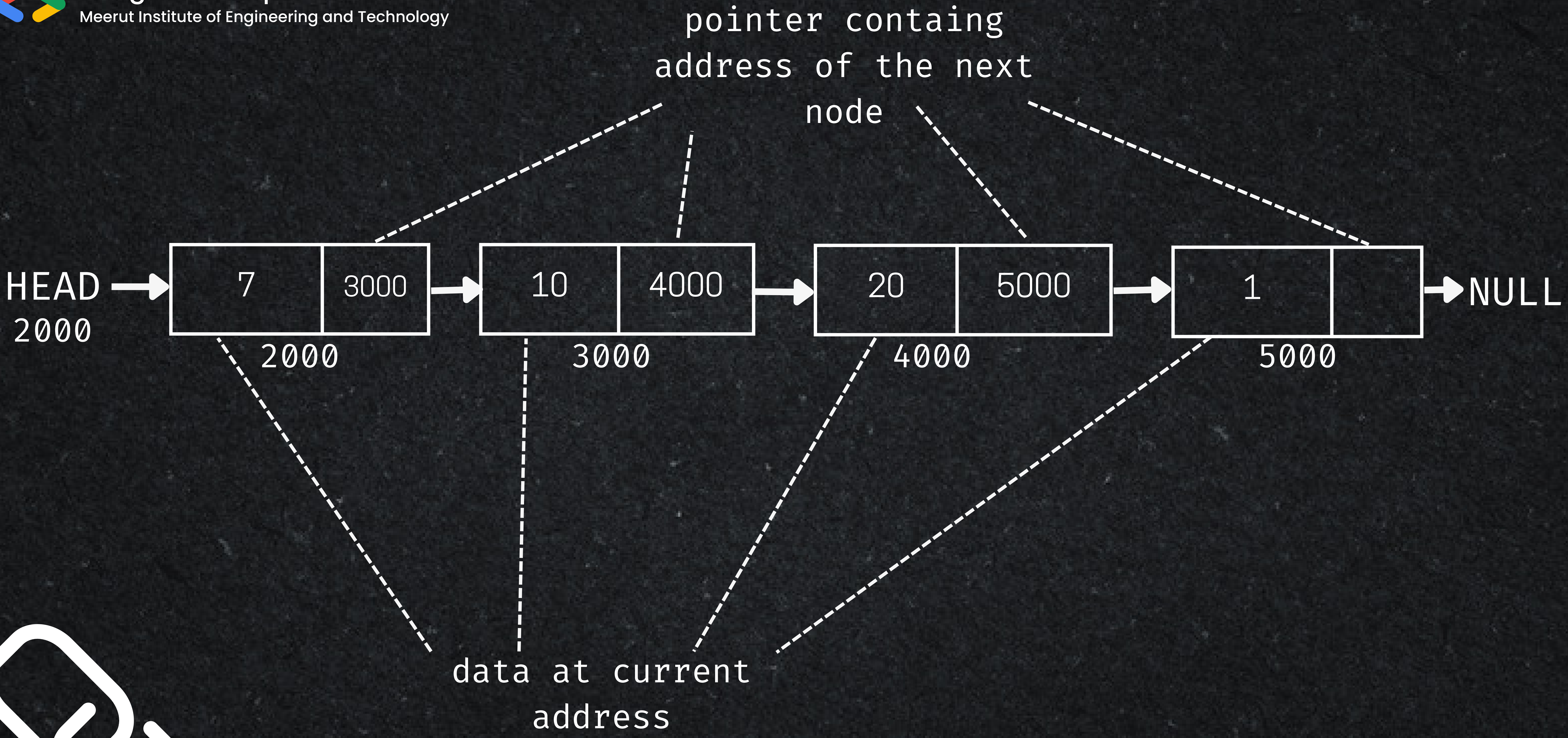
TODAY'S TOPIC:

Linked List

- Linked list is a Linear data structure.
- Linked list is defined as collection of objects called "NODES" that are randomly stored in the memory.



- Nodes make up linked list.
- A node contains two fields i.e. data stored at that particular address and the pointer which contains the address of the next node in the memory
- The last node of the list contains pointer to the null


pointer containg address of the next node
HEAD
2000
7
3000
2000
10
4000
3000
20
5000
4000
1
5000
NULL
data at current address

Array VS Linked Lists

Array	Linked Lists
Fixed Size	Dynamic size
Insertion and deletion are Inefficient	Insertion and deletion are efficient
Random Access	No Random Access
No memory waste if the array is full or almost full; otherwise may result in much memory waste	Since memory is allocated dynamically (acc. to our need), there is no waste of memory
Sequential access is faster(Reason: element in contiguous memory location)	Sequential access is slow(Reason: element not in contiguous memory location)

Initializing a linked list in C++

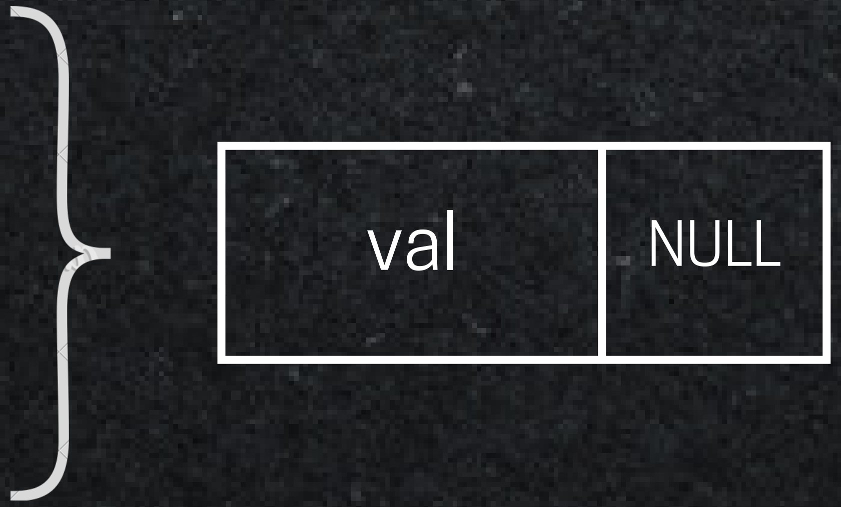
```
// Create A class node

class node{
     public:
     int data;
     node* next;

     // Node Class Constructor

     node(int val){
          data=val;
          next=NULL;

     }
};
```



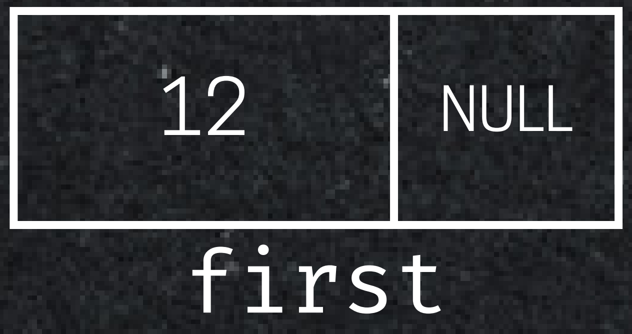
How you can access data fields in a node?

node_name->data_field

example:

```
node* first= new node(12)

first->data;//12
first->next;//NULL
```



Implementation of linked list in Java

```
class Node{

    int data;
    Node next;

    Node(int data){
        this.data = data;

    }
};
void main(){
 Node n1 = new Node(10);
 Node n2 = new Node(20);
 Node n3 = new Node(30);
 Node head = n1;
 head.next = n2;
 n2.next = n3;
 n3.next = null;
}
```

Illustration: Insertion

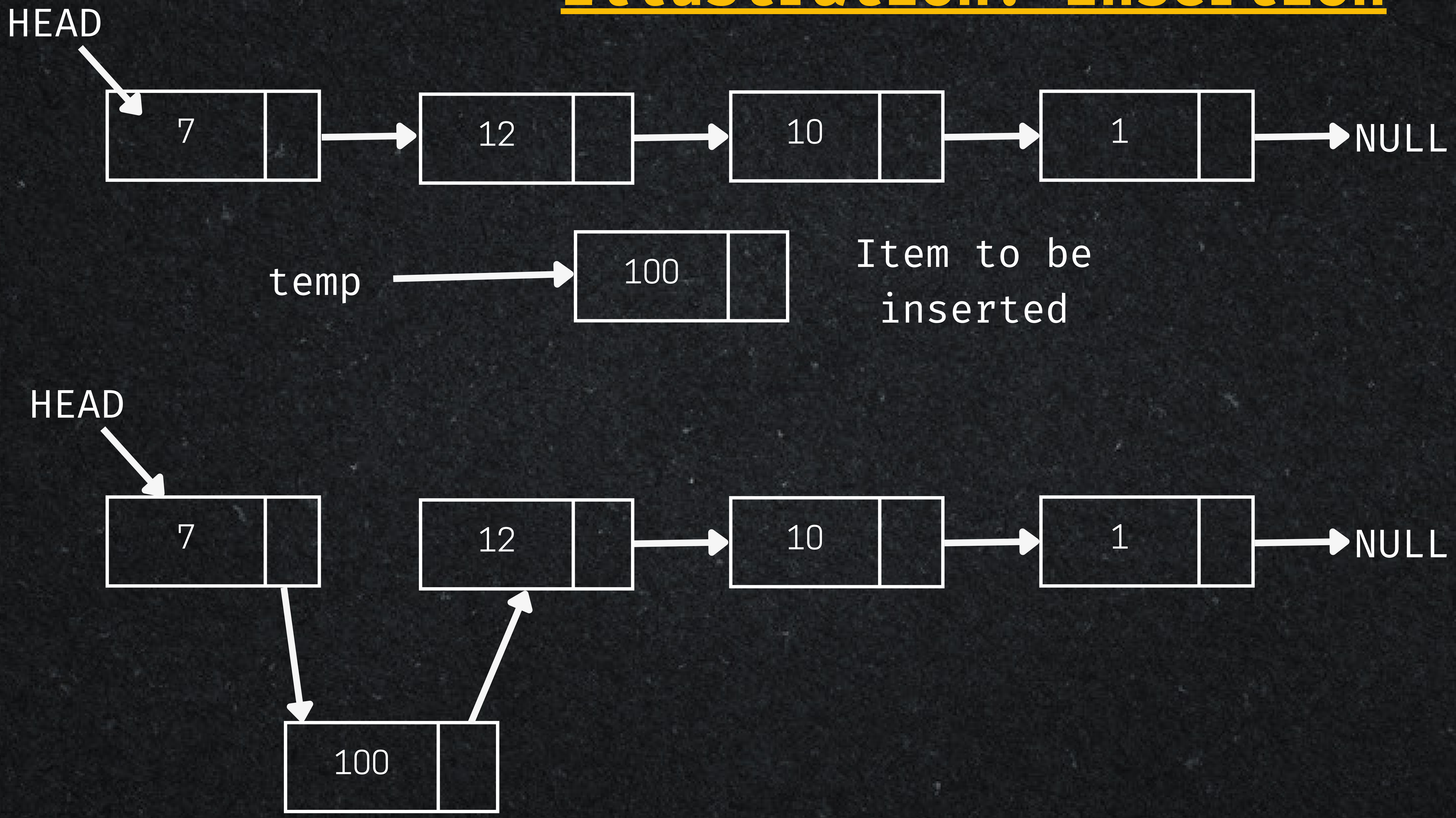
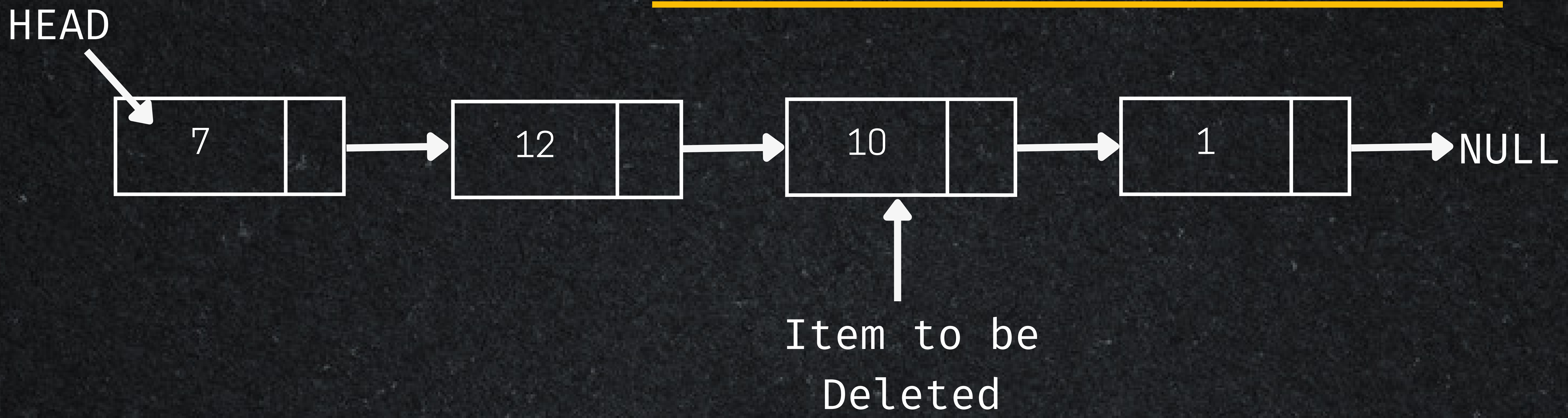
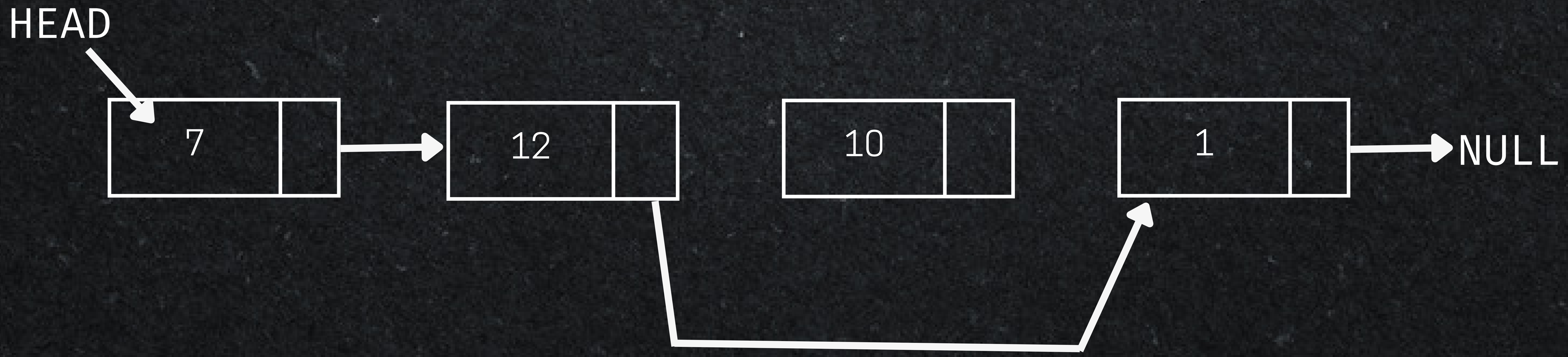


Illustration: Deletion





Types of Lists

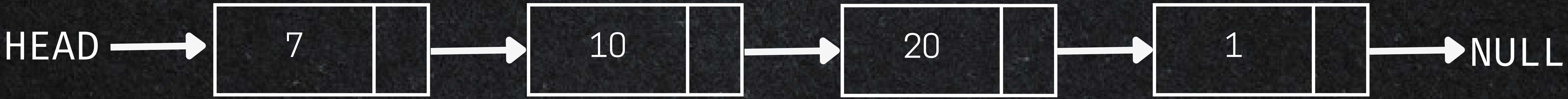
- Depending on the way in which the links are used to maintain adjacency, several different types of linked lists are possible

1. Singly Linked List

2.Circular Linked List

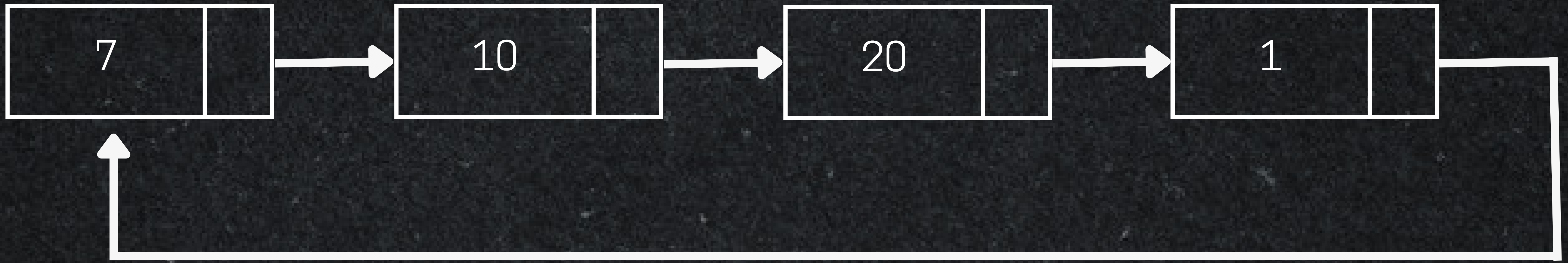
3.Doubly Linked List

Singly linked list (One - way list)



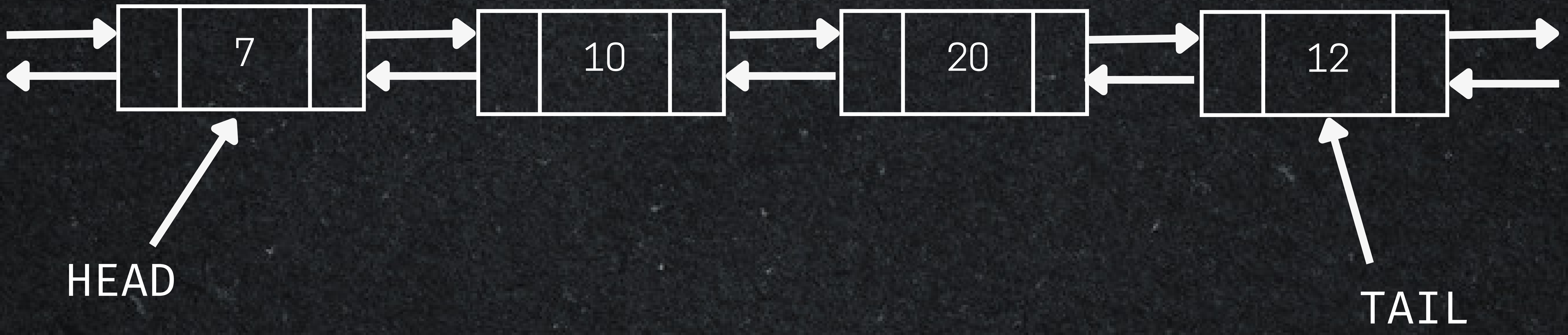
- A singly linked list is a linked list in which each node contains only one link field pointing to the next node.
- The element can be traversed only from left to right.

Circular linked list



- If we replace the NULL pointer in the last node of a list with address of its first node , such a list is called **Circular link list**.
- It does not have a first or last node.

Doubly linked list (Two - way list)



- Pointers exist between adjacent nodes in both directions.
- The list can be traversed either forward or backward.
- Usually two pointers are maintained to keep track of the list, head and tail.

Basic Operation on Linked List

- Creating a list
- Traversing the list
- Inserting an item in the list
- Deleting an item from the list
- Concatenating two lists into one

Thank You

@cpTeamGDSC_MIET