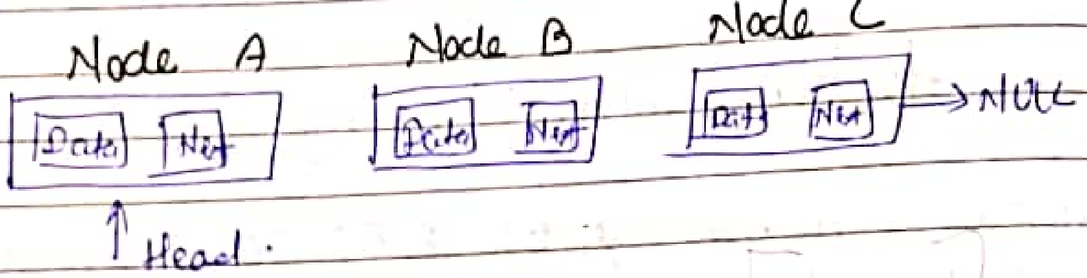
# Linked List

1) It is linear Data Structures [not a Contiguous
2> elements are linked using pointers
3> linked list consists of nodes where each node contains a data field and a reference (link) to the next node in the list.



Linked vs Arrays

① Advantages :-
  ① Dynamic size
  ② Ease of insertion / deletion.

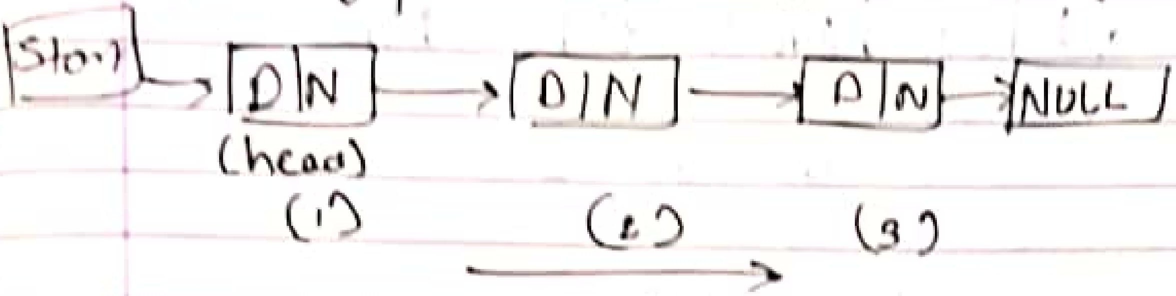
② Disadvantages :-
  ① Random access is not allowed. we have to access elements sequentially starting from the first node.
  ② Extra memory space for a pointer
  ③ Not cache friendly. [locality of reference not present]

Operations of linked list
① Traversing a LL
② Append a new node to the end of LL
③ Prepend a new node to the start of LL
④ Inserting a new node to a specific position in LL
⑤ Deleting node from the LL
⑥ Updating a node in the LL

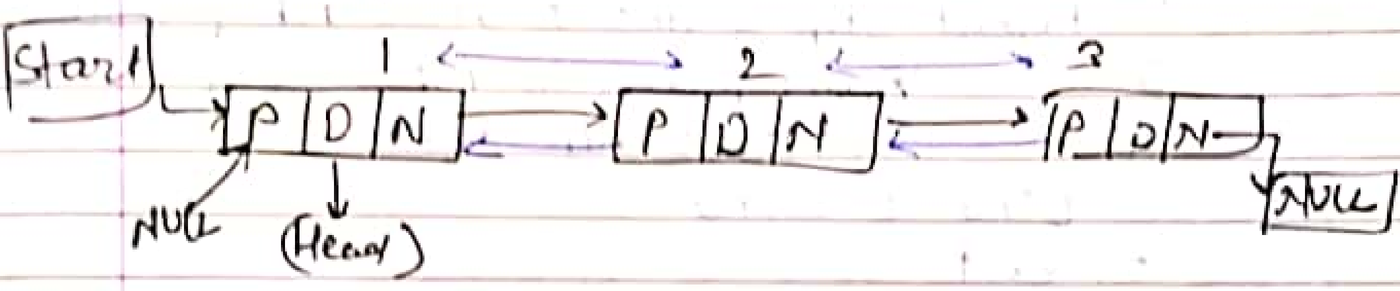
Scanned with CamScanner

# Types of LL

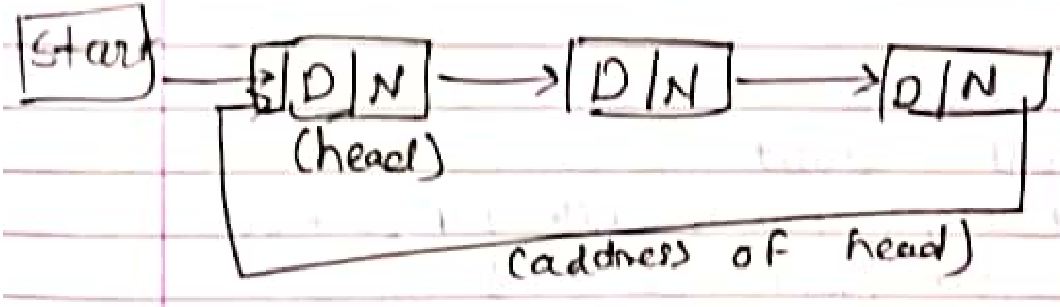
## ① Singly LL



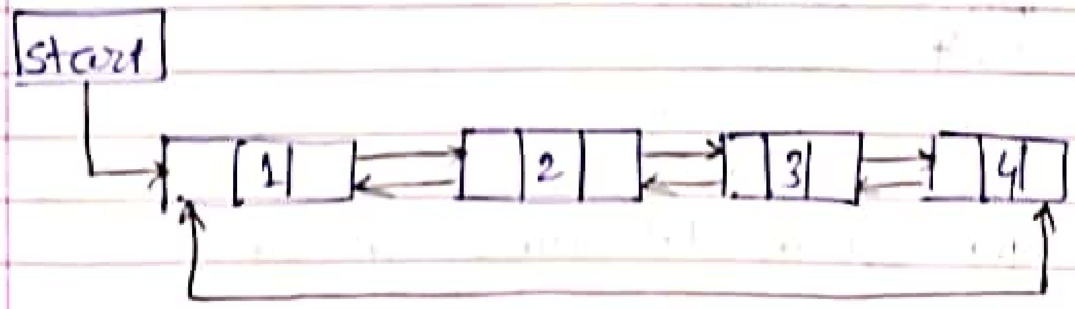
## ② Doubly linked list



## ③ Circular linked list



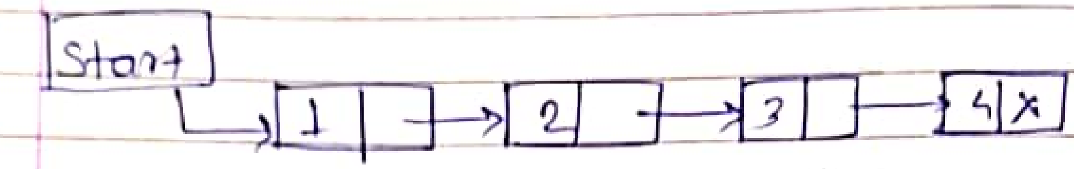
## * Circular Doubly linked lists



* Header linked list
* Multi linked list

Scanned with CamScanner

## *] Syngly linked list:-



x= Null or -1

## ① Traversing a linked list

(Algo. for traversing a linked list)

```
1: Set PTR = Start
2: Repeat step 3 and 4 while PTR != NULL
3:         Apply process to PTR → Data
4:         Set PTR = PTR → Next
   [end of Loop]
5: Exit
```

(Algo. for counting number of nodes in LL)

```
1: Set count = 0
2: Set PTR = start
3: Repeat steps 4 & 5 while PTR != NULL
4:         set count = count + 1
5:         Set PTR = PTR → Next
   [End of loop]
6: write count
7: Exit
```

How to create LL node in C:

```
struct node
{    int data;
     struct node * next;
};
```

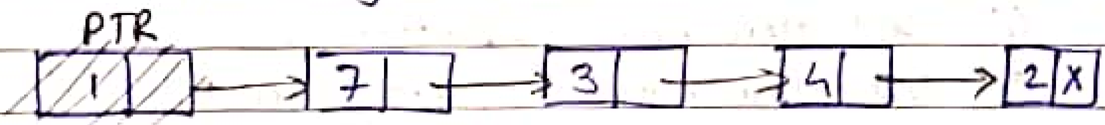
Scanned with CamScanner

## (2) Searching for a value in LL.

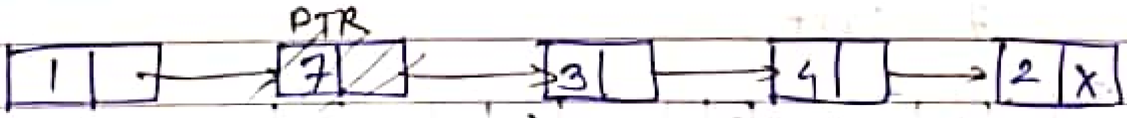
(Algo. to search a LL)

```
1: Set PTR = start
2: Repeat step 3 while PTR != NULL
3:     if val = PTR → Data
               set Pos = PTR
               Goto step 5
       else
               set PTR = PTR → Next
       [end of if]
   [End of loop]
4: Set Pos = NULL
5: exit
```

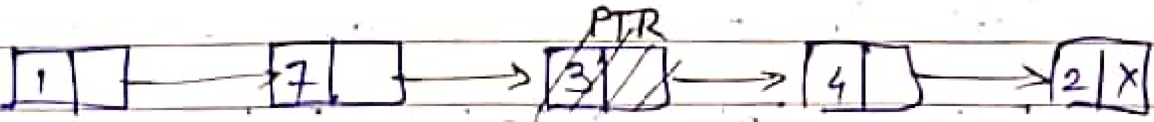
e.g. we are searching for val = 4 from below LL.



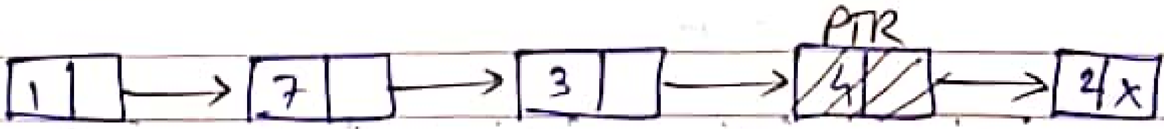
(1) PTR → Data = 1. Hence move to next



(2) PTR → Data = 7. Hence move to next



(3) PTR → Data = 3. Hence move to next



(4) PTR → Data = 4

set Pos = PTR.

↳ Now storing address of val.

return Pos

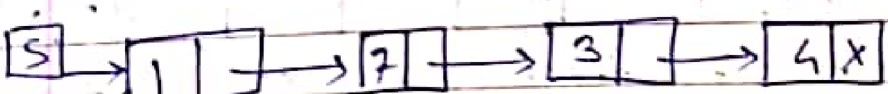
## ③ Inserting a New node in a linked list.

case 1: The new node is inserted at the begining.
case 2: The new node is inserted at the end.
case 3: The new node is inserted after a given node.
case 4: The new node is inserted before a given node.

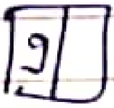
### case 1: Inserting node at the begining:

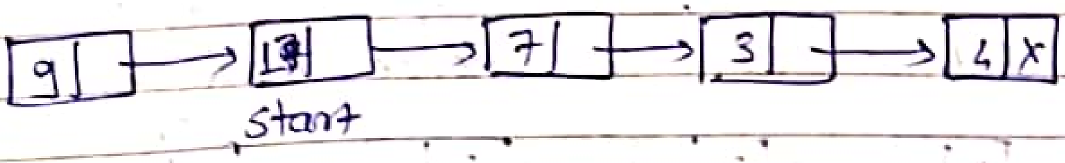
```
1: if Avail = NULL
        Write overflow
        Go to Step 7
     [End of if]
2: Set New-Node = Avail
3: set Avail = Avail -> Next
4: Set New-node -> Data = val
5: set New-node -> Next = start
6: set Start = New-Node
7: Exit.
```

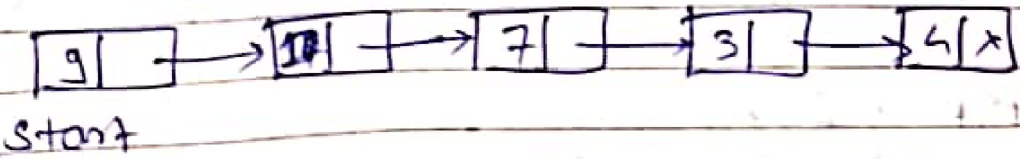
e.g.



Lets insert node '9' at the start.



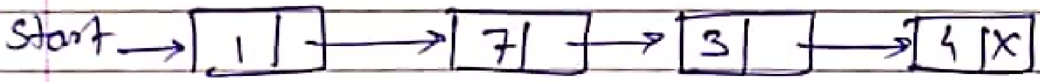




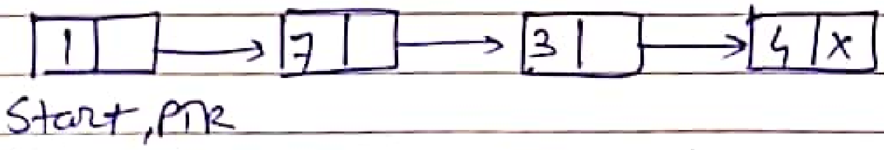
Case 2: Inserting node at the end of LL:

```
1: if Avail = NULL
           write overflow
           goto step 10
     [end of if]
2: Set New_Node = Avail
3: set Avail = Avail → Next
4: Set New_node → Data = Val
5: set New_Node → Next = NULL
6: set PTR = Start
7: Repeat step 8 while PTR → Next != NULL
8:      Set PTR = PTR → Next
     [End of loop]
9: Set PTR → Next = New_Node
10: Exit
```

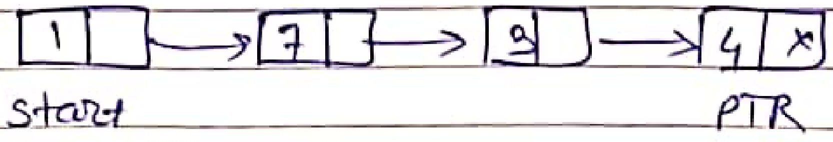
e.g. lets insert node '9' at the end



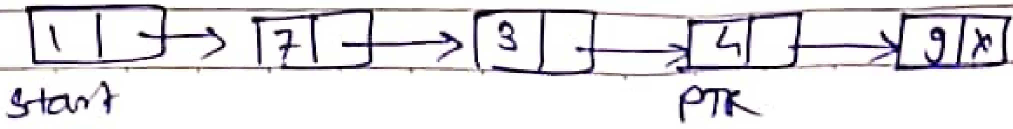
① let PTR point to start



then increment PTR till reach last Node



Now insert at the next of PTR



Scanned with CamScanner

Case 3: Inserting a node after given node of LL.

```
1: If Avail = NULL
        write overflow
        goto ~~set~~ Step 12
   [End of if]
2: Set New-node = Avail
3: set Avail = Avail → Next
4: Set New-Node → Data = VAL
5: Set PTR = start
6: set PREPTR = PTR
7: Repeat steps 8 & 9 while PREPTR → Data != NUM
8:     Set PREPTR = PTR
9:     set PTR = PTR → Next
   [End of Loop]
10: PREPTR → Next = New-node
11: Set New-Node → Next = PTR
12: Exit
```

Scanned with CamScanner

case4: Inserting a node Before given node of LL.

```
1: If Avail = NULL
        Write overflow
        Go to step 12
   [End of if]
2: Set New_Node = Avail
3: Set Avail = Avail -> Next
4: Set New_Node -> Data = val
5. Set PTR = Start
6: Set PREPTR = PTR
7: Repeat Steps 8 & 9 while PTR -> Data != NUM
8:      set PREPTR = PTR
9:      Set PTR = PTR -> Next
   [End of LOOP]
10: PREPTR -> Next = New_Node
11: Set New_Node -> Next = PTR
12. Exit
```

Scanned with CamScanner

④ Deleting a node from linked list
case1: The first node is deleted
case2: The last node is deleted
case3: The node after a given node is deleted.

Case 1:

```
1: If Start = NULL
       write underflow
       goto step 5
   [End of if]
2: Set PTR = Start
3: set start = start → Next
4: free PTR
5. exit
```

Case 2: Deleting Last Node

```
1: If start = NULL
       write underflow
       goto step 8
   [End of if]
2: Set PTR = Start
3: Repeat steps 4 & 5 while PTR → Next != NULL
4:     Set PREPTR = PTR
5:     set PTR = PTR → Next
   [End of loop]
6: Set PREPTR → Next = NULL
7: FREE PTR
8: Exit
```

Scanned with CamScanner

case 3: Deleting the node after given node.

```
1: if start = NULL
        Write underflow
        Goto step 10
   [End of if]
2: set PTR = Start
3: set PREPTR = PTR
4: Repeat steps 5 & 6 while PREPTR→Data != NUM
5:     set PREPTR = PTR
6:     set PTR = PTR →Next
   [End of loop]
7: Set TEMP = PTR
8: set PREPTR→Next = PTR→ NEXT
9: free Temp
10: exit
```

Scanned with CamScanner