PROLOG ACADEMY

- **DATA STRUCTURE**

By- MASEERA ALI

- Book followed - Data structures by Seymour Lipschutz (Schaum Series)

LET’S START!

Circular Linked List

1. Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.

Advantages of Circular Linked Lists:

1) Any node can be a starting point. We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again.

2) Useful for implementation of queue.

3)It is convenient for the operating system to use a circular list so that when it reaches the end of the list it can cycle around to the front of the list.

Traversing in circular Linked list

```
void printList()
{
    temp = start;
        do
        {
            printf("%d ", temp->data);
            temp = temp->next;
        }
        while (temp != start)
}
```

Try These!

- Sorted insertion in LL
- Find the mid point of linked list in O(n)
- Tortoise and hare algorithm
- Representation of polynomial using linked list.

Header Linked Lists

- Header linked list is a linked list which always contains a special node called the Header Node, at the beginning of the list.
- It has two types:
  a) Grounded Header List
     Last Node Contains the NULL Pointer.
  b) Circular Header List
     Last Node Points Back to the Header Node.

Grounded HLL

1.A grounded header list is a header list where the last node contains the null pointer.

2.The term “**grounded**” comes from the fact that many texts use the electrical ground symbol to indicate the null pointer.

Circular HLL

A circular header linked list is a header list where the last node points back to the header node.

* START always points to the header node.

Start ->link = NULL means grounded header list is empty.
Start->link = START indicates that a circular header list is empty.

Traversing a Circular Header List

1. Set PTR := LINK[START]. [Initializes pointer PTR.]
2. Repeat Steps 3 and 4 while PTR != START.
3. Apply PROCESS to INFO[PTR].
4. Set PTR := LINK[PTR]. [PTR now points to the next node.]
   [End of Step 2 loop.]
5. Exit.

Doubly Linked list

* A two-way list is a linear collection of data elements, called nodes, where each node N is divided into three parts:
   1.Information field
   2.Forward Link which points to the next node
   3.Backward Link which points to the previous node

The starting address or the address of first node is stored in START / FIRST pointer .

Another pointer can be used to traverse list from end. This pointer is called END.

Insertion in the doubly LL

Bring a node from Avail and call it new

At beginning

```
new->right=start
start->left=new
new->left=NULL
new->info=Data
start=new
```

At End

```
temp=start
while(temp->right!=NULL)
{ temp=temp->right
 }
temp->right=new
 new->left=temp
 new->right=NULL
 new->info=Data
```

- At the middle
  new->left=temp
  new->right=temp->right
  temp->right=new
  new->right->left=new
  new->info=Data