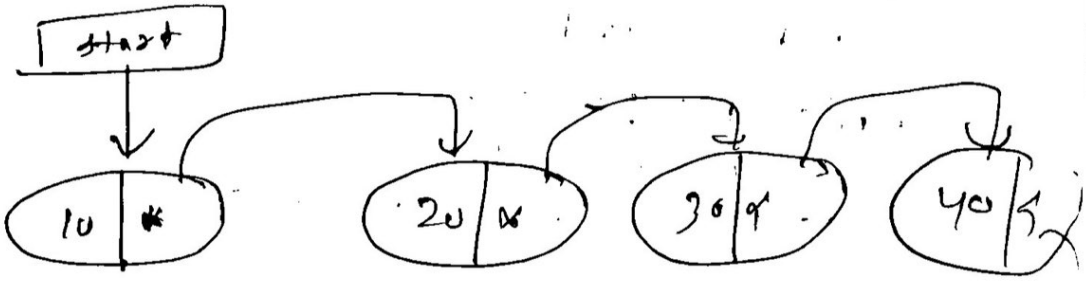
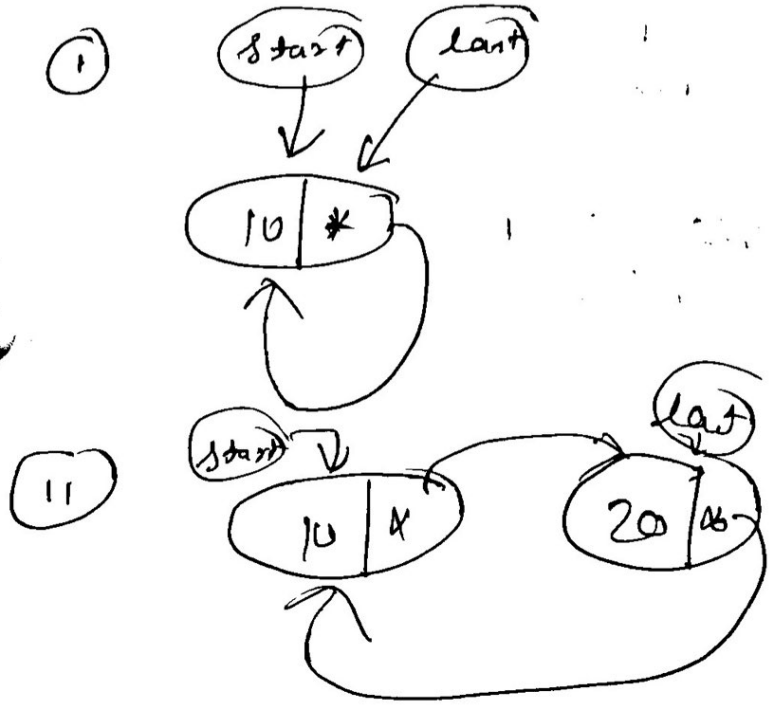
Circular Linked List :-

Same as Singly Linked list except pointer field of last node contain address of first node

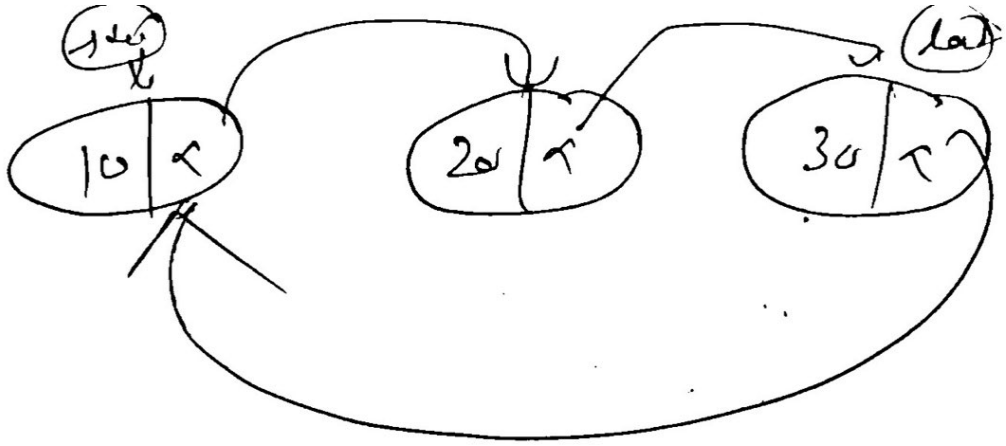
Singly Linked List.



Circular linked list



(3)




```
class Node
{
    int data;
    Node Next;
    Node(int x)
    {
        data = x;
        New = null;
    }
}
```

```
class CLink
{
    Node start;
    Node Last;

    void addLast (int n)
    {
Node ptr = new Node (n);
if(start == null)
    {
        start = ptr;
        Last = ptr;
        Last.New = start;
    }
else
{
    Last.New = ptr;
    last = ptr;
    last.New = start;
}
}
```

```
void disp()
{
  if (start == null)
   return;

  Node t = start;

  do
  {
    sop(t.data + " ");

    t = t.Next;
  }
  While (t != start);

}

}
```

class Main

{

p s v m (String K[])

{

CLinK Obj = new CLinK();

Obj.addLast(10);

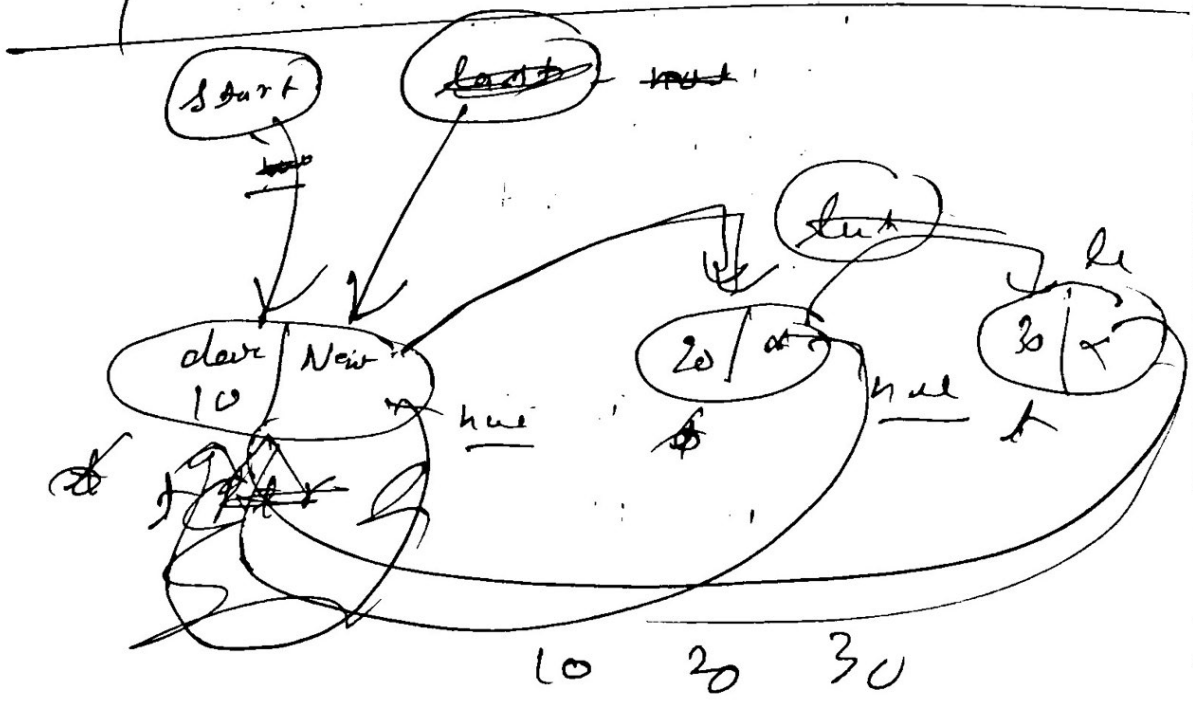
Obj.addLast(20);

Obj.addLast(30);

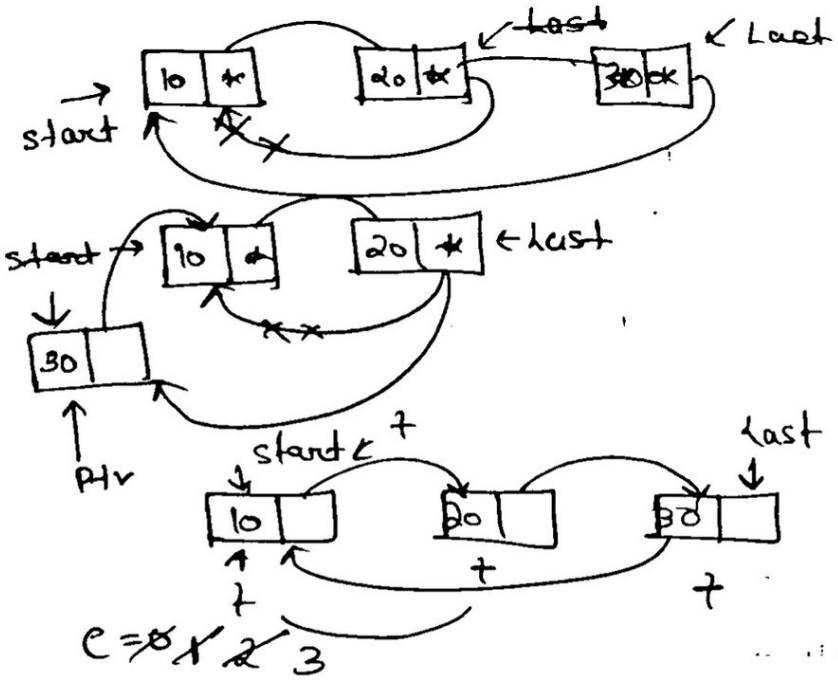
Obj.disp();

}

}

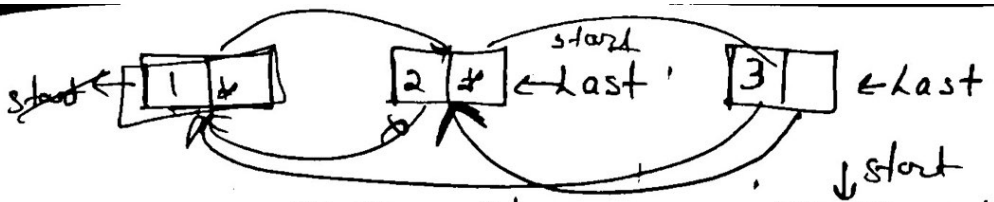


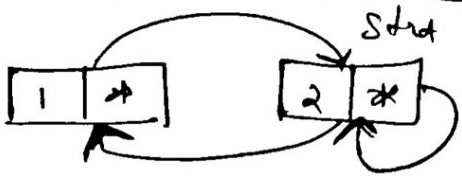
```
void addfirst (int x)
{
    ~~if (start == null)~~
    node ptv = new node (x);
    if (start == null)
    {
        start = ptv;
        last = ptv;
        last.next = start;
    }
    else
    {
        ptv.next = start;
        last.next = ptv;
        start = ptv;
    }
}

int Count ()
{
    ~~int c=0;~~
    ~~if (start == null)~~ if (start == null)
    ~~return;~~ return c;
    ~~else int c=0;~~
    node t = start;
    do
    {   c++;
        t=t.next;
    }
    while (t != start)
    return c;
}
```

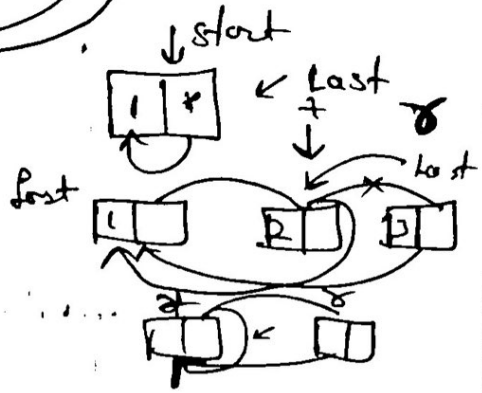



```
int Sum()
{ int s=0;
   if (Start == null)
      return 0;
   node t= Start;
   int c=0;

   do
    {  s = s + t.data;
       t = t.next;
    }
    while (t != Start)
    return s;
}

void getFirst()
{   if (Start == null)
       return;
    SOPL (Start.data);
}
```

```
void getlast()
{
    if(start == null)
    return;
    SOPL(last.data);
}

void removefirst()
{
    if (start == null)
    return;
    SOPL("List is Empty");
    else if (start == last)
    { start = null;
      last = null;
    }
    else
    {
        start
        last.next = start.next;
        start = start.next;
    }
}
```



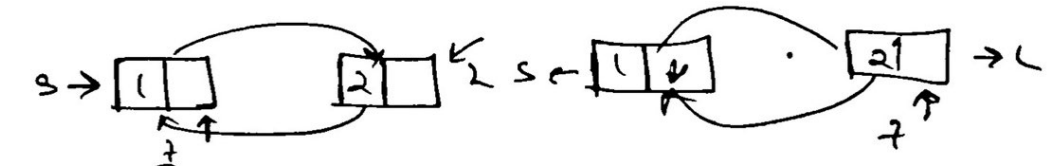





```
void removelast
{
    if (start == null)
      SOPL ("List is Empty")
    elseif (start == last)
     {  start = null;
        last = null;
     }
    else
     {
        node t = start;
        node r = start.next;
        while (r.next != start)
        {
            t = t.next r;
            r = r.next;
        }

        t.next = start;
        last = t;
     }
}
```

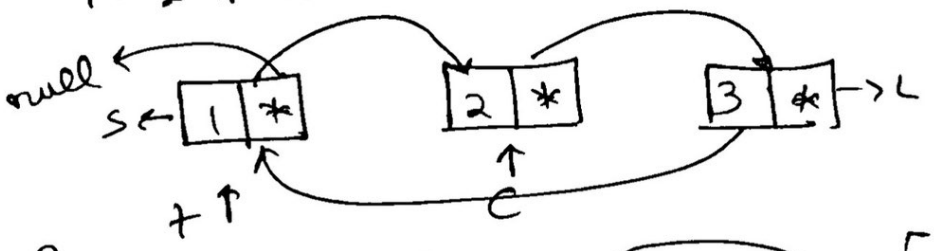
```
node t = start;
while (t.next != last)
  t = t.next;
t.next = start;
last = t;
```

```
int Search (int item)
{
    int P = -1;
    int i = 0;
    node *t = start;
    ~~while (t!=i~~
    do
    {   if (t->data = item)
        {   P = i;
            break;
        }
        i++;
        t = t->next;
    } while (t != start)
    return P;
}
void ~~reverse~~ ()
{
```

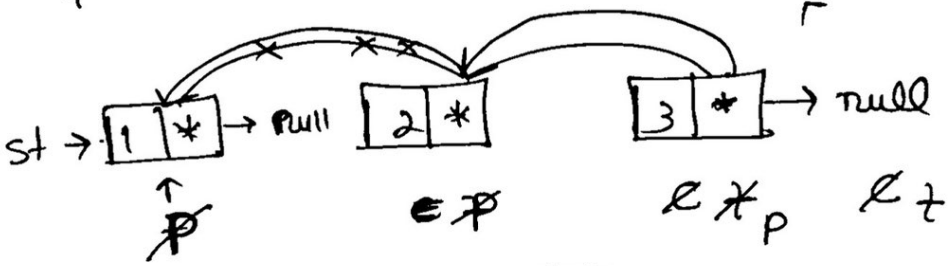


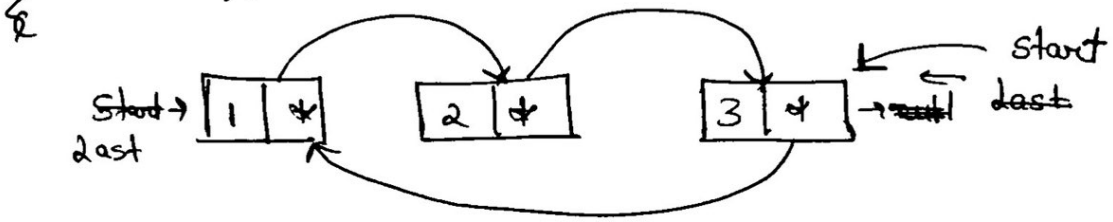
p = ~~-1~~ (1)

i = ~~0~~ 1 2



p





- remove at Index()
- get at Index()
- add at Index()