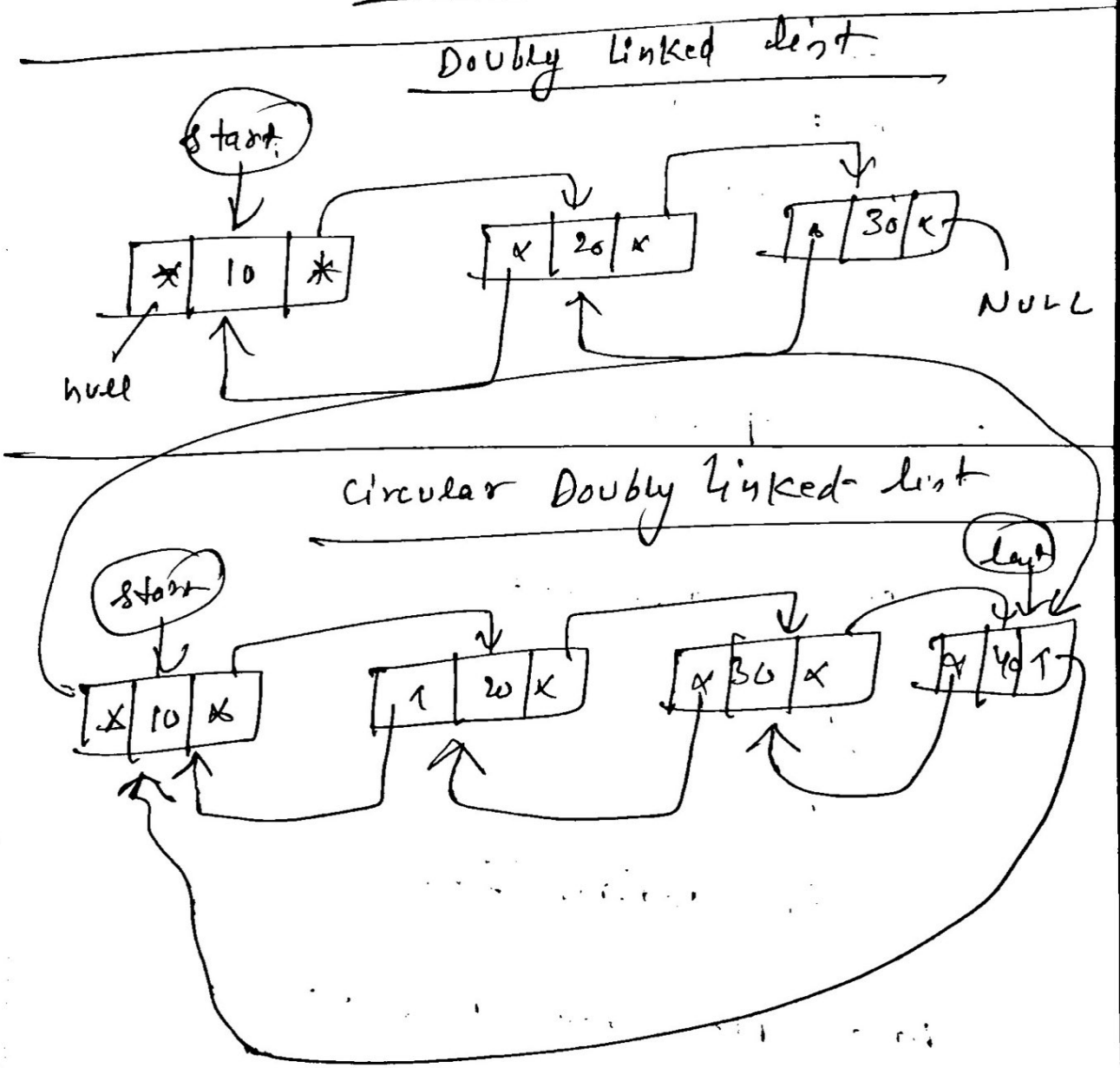
Circular Doubly Linked List

Same as Doubly Linked list except Previous Pointer of first Node Contain addres of last Node and Pointer field of last Node Contain addres of first Node



```
clan Node
{
    int data;
    Node Next, Pre;

    Node (int x)
    {
     data = x;
     Next = null;
     Pre = null;
    }
}

clan CDLink
{
    Node start, last;

    CDLink()
    {
    }

    void addlast (int x)
    {
Node Ptr = new Node (x);
```

```
if (start == null)
{
    start = ptr;
    last = ptr;
    last.next = start;
    start.pre = last;
}

else
{
    last.next = ptr;
    ptr.pre = last;
    last = ptr;
    last.next = start;
    start.pre = last;
}
}

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

```
Node t = start;
do
{
  Sop (t.data + " ");
  t = t.Next;
}
while (t != start);
}

void ldisp()
{
  if (last == null)
    return;
  Node t = last;
  do
  {
    Sop (t.data + " ");
    t = t.pre;
  }
  while (t != last);
}
```

```
void addfirst(int x)
{
Node Ptr = new Node(x);
if (start == null)
{
 start = Ptr;
 last = Ptr;
last.Next = start;
 start.Pre = last;
}
else
{
    Ptr.Next = start;
    start.Pre = Ptr;
    start = Ptr;
    last.Next = start;
    start.Pre = last;
}
}
```

addAtIndex()
getAtIndex()
removeAtIndex()
} Doubly

clear()
search()
sum()
big()
getFirst()
getLast()
} Circular

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

  Node t.

  else if (start == last)
    {
      start = null;
      last = null;
    }
  Else
    {   Node t = start;
        start = start.Next;
        last.Next = start;
        start.pre = last;
        t = null;
    }
}
```

```
class Main
{
public static void main (String K[])
{
CDLink obj = new CDLink();
    obj.addLast(10);
    obj.addLast(20);
    obj.addLast(30);
    obj.addLast(40);
    obj.addfirst(50);
    obj.fdisp();
    obj.bdisp();
}
}
```

```
void removeLast()
{
    if(start last == null)
        return;
    else if (start == last)
    { start = null;
      last = null;
    }
    else
    {
        ~~last~~
        ~~start = start~~
        node t = start;
        while (t.next != last)
            t = t.next;
        t.next = start;
        last = t;
        start.Pre = last;
    }
    ~~Node t = lastPre;~~
    last = last.pve;
    last.next = start;
    start.pve = last;
}
}
```