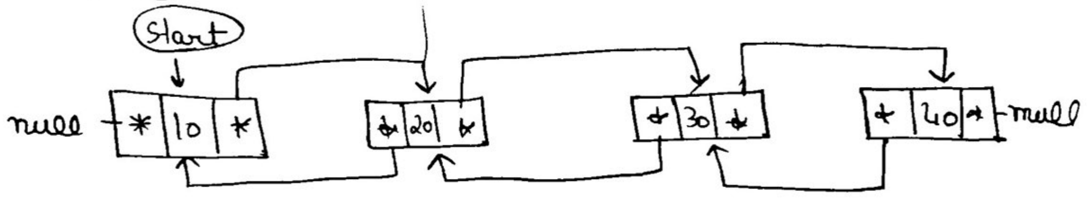
Doubly Linked List

Each node has two pointer Previous and next Pointer.
Previous pointer hold address of Previous node and next Pointer hold address of Next node

We Can traverse Doubly Linked List in forward or backward direction.

Previous pointer of first node Contain null and last Pointer of Last node Contain null.



Functions

add first()
add Last()
f disp()
b disp()
add at Index()
remove

Same $\boxed{15+1}$

doubly Linked List ADT
add first()

```
class node
{
    int data;
    node next, pre;
    node (int x)
    {
        data = x;
        next = null;
        pre = null;
    }
}

class DLink
{
        node start = null;

        void add last (int x)
        {
            node ptv = new node (x);
            if (start == null)
                start = ptv;
            else
            {
                node 7 = start;
                while (7.next ! = null)
                {
                    7 = 7.next;
                }

                7.next = Ptv;
                Ptv.pve = 7;
            }

        }
```

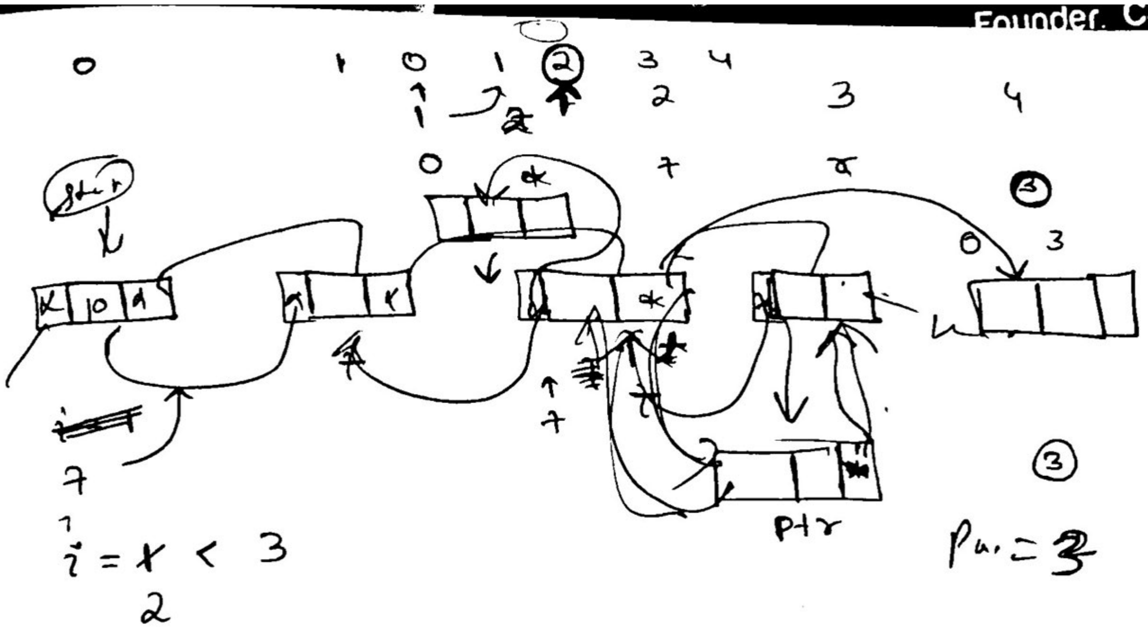
```
void fdush ()
{
   if (start == null)
   SOPL ("List is empty");

   else
   {
      nodo 7 = start;
      while (7! = null)
      {
         SOPL (7.data);
         7= 7. ~~es~~ next;
      }
   }
}
void bdush ()
{
      if (start == null)
      SOPL (" List is Empty");
      else
      {
         nodo 7 = start;
         while (7.next! = null)
          7= 7.next;

         while (7! = null)
         {
            SOPL (7 data);
            7 = 7.Pve;
         }
      }
}
```

```
int Count ()
{
      node t = start ;
      int c = 0;
      while (t! = null)
       {  c++;
          t= t. next ;
        }
       return c ;
   }
int Sum ()
{
       node t= Start ;
       int Sum =0 ;
       while (t! = null)
        {
            s = s + t.data;
            t= t. next ;
         }
        return Sum ;
 }
get first ( )
 {     if (start == null)
          ~~SOPL ("~~
          return;
       SOPL (start.data);
 }
```

```
void getLast ()
{
    if (start == null)
        return;
    node t = start;
    while (t.next != null)
      t = t.next;

    SOPL (t.data);
}

void addfirst (int x)
{
    node ptr = new node (x);
    if (start == null)
        start = ptr;
    else
     {    ptr.next = start;
          start.prev = ptr;
          start = ptr;
     }
 }
```

```
void  remove first ()
{   if (start == null)
      SOPL ("List is empty");
    else
     {     node  ptr = start ;
           Start = Start.next ;
           Start. pre = null ;
           ptr = null ;
     }
}

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

```
void  aad at Index ( int  Pos, int x)
{
   if (Pos <0 || Pos > Count())
     SOPL (" Can not add at given
           index");
   else if (Pos ==0)
        add first (),
   else if (Pos == Count())
        add Last ();
   else
    {
      node ptr = ~~start~~ new node (x);
      node  t = start;
      for (int i=1 ; i < Pos ; i++)
          t = t. next ;

      ptr. next = t. next ;
      ptr. next. pre = ptr ;
      t. next = ptr ;
      ptr. pre = t ;
    }
}
```

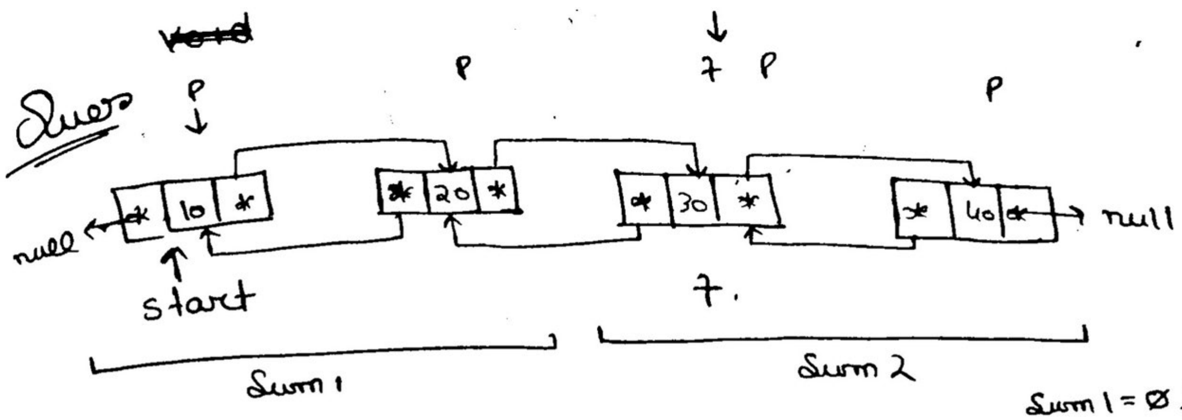


```
void removeatIndex (int x, int pos)
{
      if (pos <0 || pos>=count())
          SOPL ("cannot remove at index");
      else if (pos ==0)
          remove first();
      else if (pos == count()-1)
          remove Last ();

      else
       {      node t = start;
              node r = start.next;

              for (int i=1; i< pos; i++)
               {    t = r;
                    r = r.next;
               }

               t.next = r.next;
               r.next.pre = t.~~next~~;
               r = null;
       }
}
```

```
Class main
{
    P S V M (String k [])
    {
      DLnK obj = new DLink();
      obj.addLast(10);
      obj.add last(20);
      obj.add last(30);
      obj.add last(50);
      obj.disp();
      obj.bdisp();
    }
}
```

~~void~~

Ques



Sum 1 = ~~0~~ ~~10~~ 30

Sum 2 = ~~0~~ ~~30~~ 70

Sum difference = Sum 1 – Sum 2

```
void Sumdiff (node 7)
{   int Sum 1 = 0;
      node P7r = Start;
      while (P7r != 7)
      {  Sum1 += P7r.data;
         ~~P7r = 7.next~~ P7r = P7r.next;
      }
    int Sum 2 = 0;
    P7r = 7;
    while (P7r != null)
    {     Sum2 += P7r.data;
          P7r = P7r.next;
    }

    int Sumdiff = Sum1 - Sum2;
    SOPL (Sumdiff);
}
```