Doubly Linked Lists:

If we consider a singly linked list, we cannot travers such a list backward, nor can a node be deleted from a singly linked list, given only a pointer to that node.

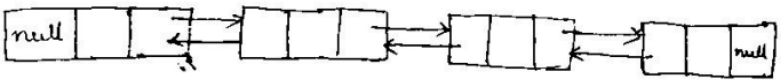
In order to overcome these drawbacks, we have use doubly linked list data structure.

We may consider the nodes on a doubly linked list to consist of 3 fields: an info field that consists the information stored in the node and left and right fields that contain pointers to the nodes on either side.

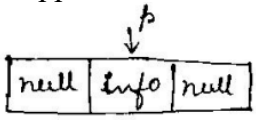
The dynamic implementation for such a node is:

```
Struct node
  {
        int info;
        Struct node *left, *right;
  };
Typedef Struct node *  NODEPTR;
```

A linear doubly linked list is shown below:



If suppose there is only one node in the doubly linked list then, the right & left link are null.

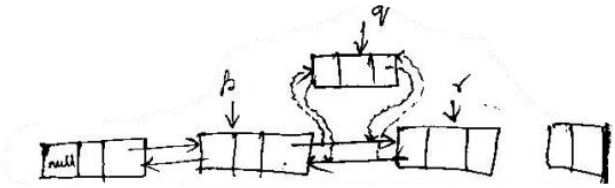


p→left = NULL and p→right = NULL.

C function to delete a node p in a Doubly linked list:

```
void delete(NODEPTR p)
{
NODEPTR l, r;
int x;
if(p==NULL)
   printf("Invalid deletion");
else
  {
    x=p→info;
    l=p→left;
    r=p→right;
    l→right=r;
    r→left=l;
    free(p);
    printf("The deleted element  = %d", x);
  }
}
```

C function to insert a node to the right of node p in a Doubly Linked list:




```
void insert_right(NODEPTR p, int x)
{
NODEPTR q, r;
if(p==NULL)
   printf("Invalid Insertion");
else
   {
    q=getnode();
    q→info=x;
    r=p→right;
    r→left=q;
    q→left=p;
    q→right=r;
    p→right=q;
   }
}
```

C function to insert a node to the left of node p in a Doubly Linked list:

```
 void insert_left(NODEPTR p, int x)
{
NODEPTR q, l;
if(p==NULL)
   printf("Invalid Insertion");
else
   {
    q=getnode();
    q→info=x;
    l=p→left;
    l→right=q;
    q→left=l;
    q→right=p;
    p→left=q;
   }
}
```

Advantages and Disadvantages of using Doubly linked Lists over Singly linked list:

Advantages:

1. One cannot traverse a singly linked list backward, ie, from right to left whereas a doubly linked list can be traversed in either direction ie, we can use right link to traverse forward direction and left link to traverse backward direction.
2. In the case of singly linked list, we cannot directly delete a node given only a pointer to that node. We have to keep tail pointer and we can only perform delete after operation. In the case of doubly linked list, we can directly delete a node given only a pointer to that node.

Disadvantages:

1. When space efficiency is a consideration, a program may not be able to afford the overhead of two pointer for each element of a list.