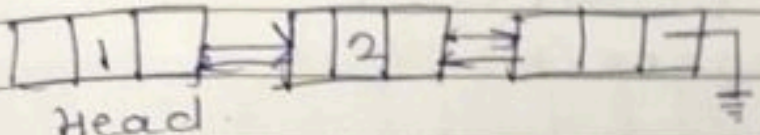
Assignment No:- 10

Explain DLL & all operations on it with example & pictorial representation.

→ In Doubly linked list a node represent next as well as previous location
- It has two address field
- It can be traversed in both direction.

eg.



struct

creation:

```
struct node
{
  int data;
  node *prev, *next;
};
node * nnode = new node
nnode->prev = nnode->next = NULL

if (head == NULL)
{ head = tail = nnode;
}
else
{ tail->next = nnode;
  nnode->prev = tail;
  tail = nnode;
}
```

display operation:
to display linked list we have to traverse linked list.

- to display ll forward

```
temp = head
while (temp != NULL)
{
  cout << temp -> data << " ";
  temp = temp -> next;
}
```

o/p : 1 2 3

- Reverse ;

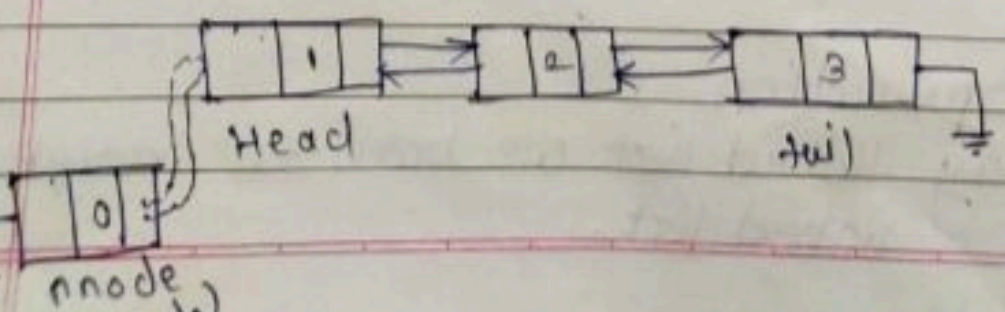
```
temp = tail;
while (temp != NULL)
{
  cout << temp -> data << " ";
  temp = temp -> prev;
}
```

o/p : 3 2 1

- Insertion :

In insertion we are inserching elements / node in linked list at particular pos^n

- Insertion of first pos^n



- to display it forward

```
temp = head
while (temp1 = NULL)
{
  Cout << temp -> data << " ";
  temp = temp -> next;
}
```

o/p : 1 2 3

- Reverse;

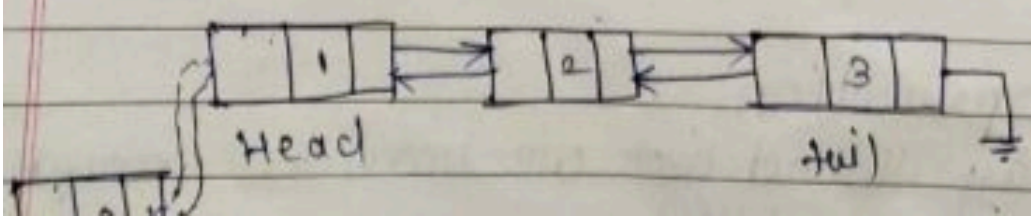
```
temp = tail;
while (temp; = NULL)
{
  Cout << temp -> data << " ";
  temp = temp -> Prev;
}
```

o/p : 3 2 1

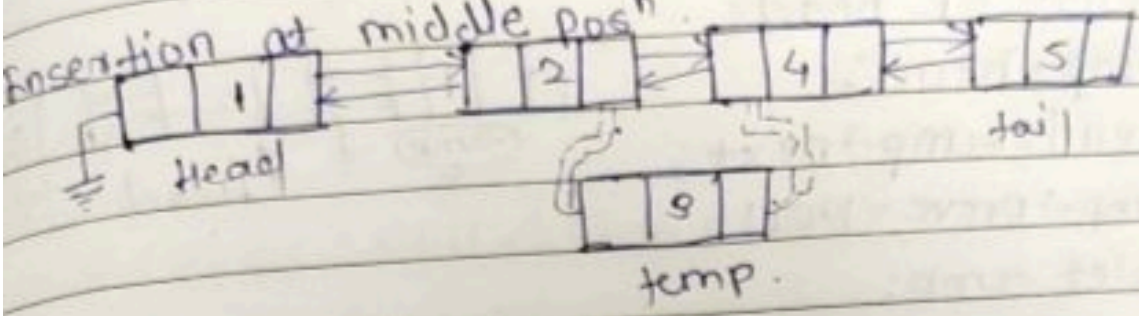
- Insertion:

In insertion we are inserching elements / node in linked list at particular pos^n

- Insertion of first pos^n



```
nnode = new node
Cin>>nnode -> data;
nnode ->next = nnode ->prev = NULL;
nnode->next = head
head->prev = nnode;
head = nnode
```

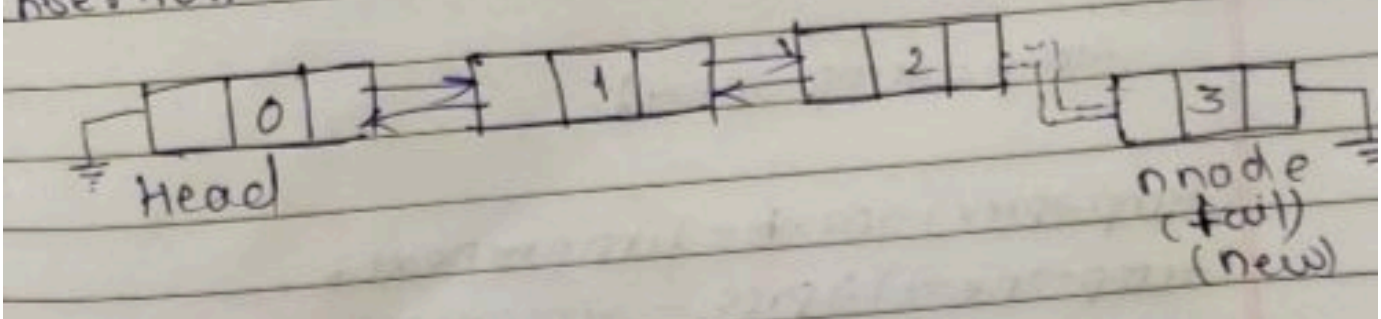



```
temp = head
while (temp->data != 3)

  temp = temp->next;

nnode->next = temp->next;
temp->next->prev = nnode;
temp->next = nnode
nnode->prev = temp;
```

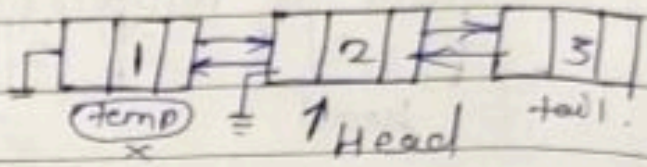
Insertion



```
nnode = new node;
Cin>> nnode→ data;
nnode→next = nnode→prev = NULL
tail→next = nnode;
nnode→prev = tail;
tail = nnode;
```

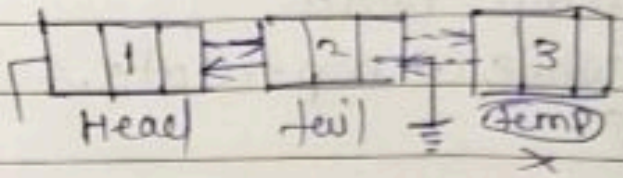
* Deletion in DLL

① Deletn of head:




```
temp = head;
head = temp→next;
temp→preve = NULL;
delet temp;
```

② Deletn of tail:-



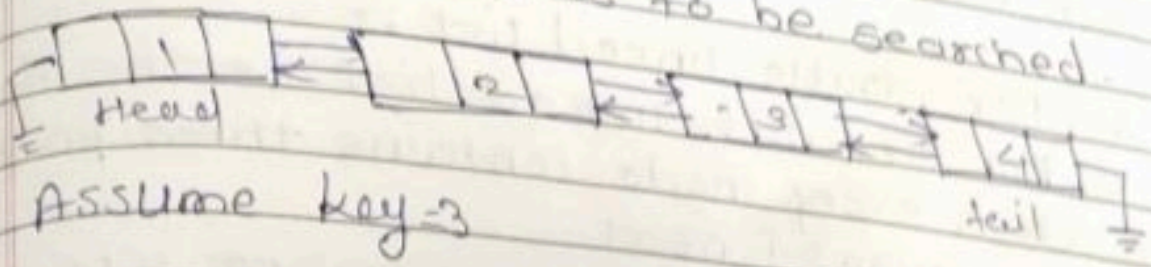

```
temp = tail;
tail = temp→prev;
tail→next = NULL;
delete temp;
```

③ Deletn of other node:

```
Delet node with data 'd'
temp = head;
while (temp→data != d)
{
    temp = temp→next;
}
(temp→prev)→next = temp→next;
(temp→next)→prev = temp→prev;
delet temp;
```

Searching:

key = data which is to be searched



Assume key = 3

```
temp = head;
found = 0;
while(temp != NULL && found == 0)
{
  if(temp->data == key)
  { found = 1;
  }
  else {
      temp = temp -> next;
}
if(found == 1)
cout<<" element found";
else
cout <<" element not found";
```