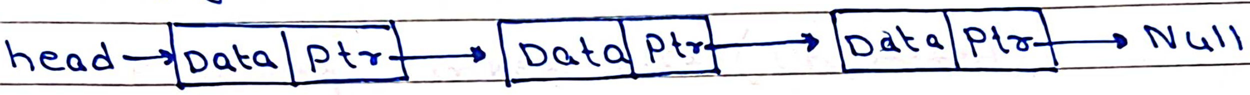
Linked List :

A Linked list is an ordered collection of finite homogenous data elements ~~as~~ called nodes where linear order is maintained by means of links or pointers. Each node is divided into two parts :

- * first part contains the information
- * second part called the link field or pointer field, contains the address of the next node in the list.

Singly Linked List :



* header linked list is the list which starts with head node

* Grounded linked list is the Linked list where last node points to Null

Traversing Linked List Algorithms:

```
1.) Set ptr = start
2.) Repeat steps 3 & 4 while ptr ≠ Null
3.)        ~~Apply~~ ~~p~~ Print ~~in~~(ptr.info)
4.)            ptr = ptr.next
                  // ptr now points to next node
5.) Exit
```

~~Traversing~~ Searching Linked List (sorted)

```
1.) set ptr=head
2.) Repeats steps 3 ~~& 4~~ while ptr ≠ Null
3.) if   item = ptr.info   then
              set loc=ptr & exit

    else
              set ptr=ptr.next
4.) if loc==Null
          print ("Search Unsuccessful)
```

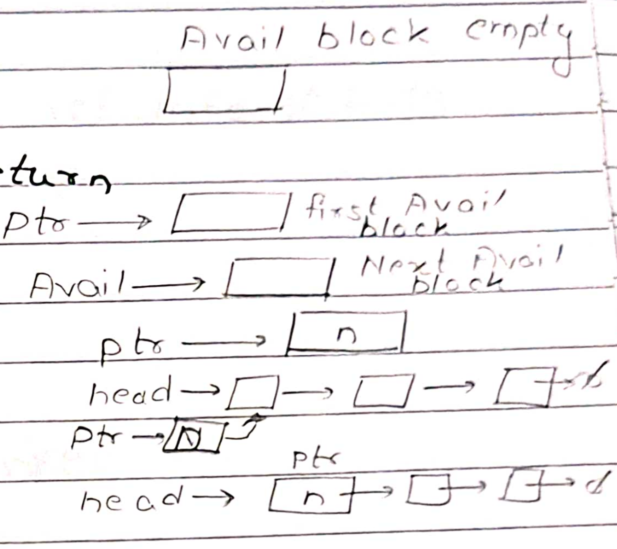
Searching Linked List (Unsorted)

```
1.) Set ptr= head
2.) Repeat steps 3 while ptr ≠ Null
3.) if item > ptr.info then:
              set ptr = ptr.next // pointing to next no
    else if item= ptr.info then
              set loc= ptr & exit // search successful
    else
            set loc= Null & exit

4.) if Loc = = Null
            print ("Search Unsuccessful
```

Insertion in Linked List :

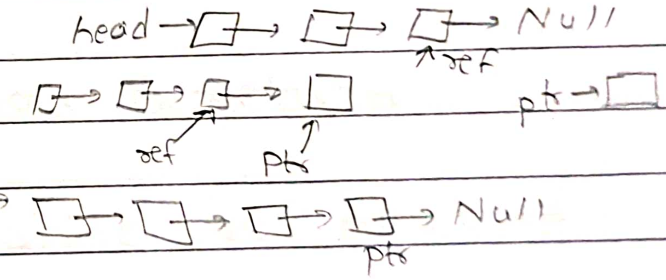
Insertion at the beginning :

```
1.) If Avail == Null
        print overflow & return
2.) Ptr = Avail
    Avail = Avail.Link
    Write ptr.info = n
3.) ptr.next = head
    head = ptr
```



Insertion at the end :

```
1.) If Avail == Null
        print overflow & return
2.) Ptr = Avail
    Avail = Avail.link
    write ptr.info = n
3.) ref = head
4.) Repeat step 5 while ref.next == Null
5.) ref = ref.next
6.) ref.next = ptr
    ptr.next = Null
```



Insertion at given node :

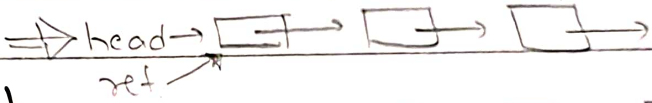
```
1.) If Avail == Null
        print overflow & return

2.) Ptr = Avail
```

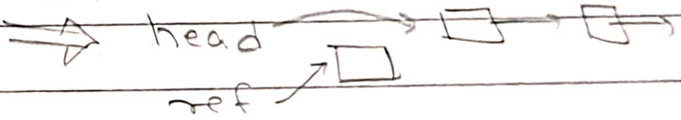
```
Avail= Avail. Link
write Ptr.info=n

3.) Ref= first
4.) Repeat step 5 while ref.info ≠ Data
5.) ref= ref.next
6.) ptr.next = ref.next
    ref.next = ptr
```

Deletion in a Singly Linked List

Deletion from the beginning:

```
1.) If first==Null
        print Underflow & return
2.) ref = head
    head= ref.next
4.) ref.next= Avail
    Avail= ref
```



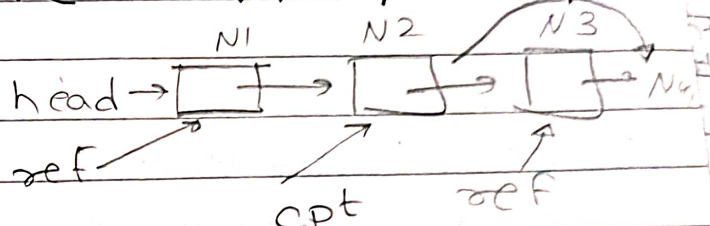


Deletion from the end:

```
1.) If first==Null
        print Underflow & return
2.) ref = head
3.) Repeat step 4 while ref.next ≠ N
4.)     Ptr = ref
        ~~ptr~~ ref=ref.next
5.) ptr.next= Null
6.) ref.next = Avail
    Avail = ref
7.) Stop
```

Deletion of given Info node:

```
1.) If first == Null
        print Underflow & return
2.) ref = ~~first~~ head
3.) Repeat step 5 while ref.info ≠ num
4.) cpt = ref
5.)         ref = ref. next
6.) cpt. next = ref. next
7) ref. next = Avail
8.) Avail = ref
```


N1
N2
N3
head
N4
ref
cpt
ref