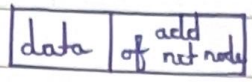
__/__/__

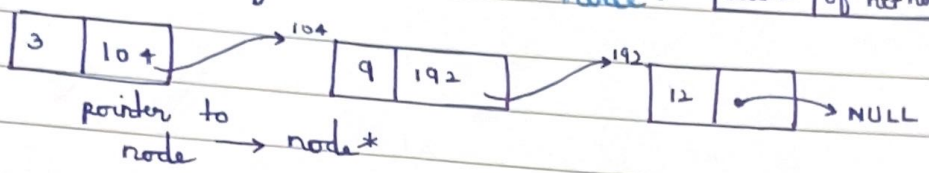
Linked List

- ✓ non - continous memory location
- ✓ dynamically grow or shrink
- ✓ no memory waste
- runtime insert / del

def :- coll of node

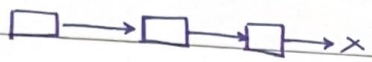
node :- 

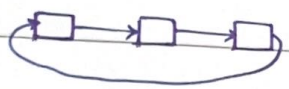
eg:)

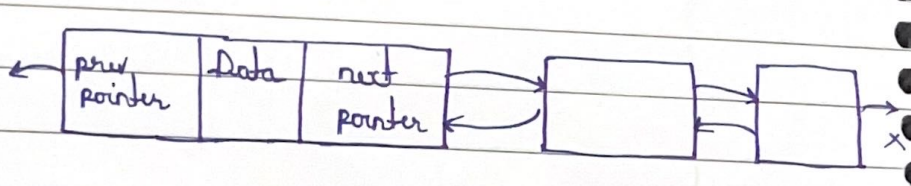


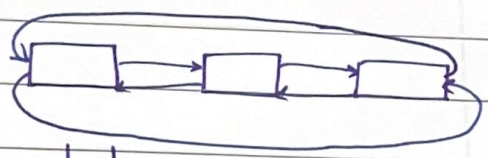
creation :-
```
class node {
    int data;
    node * next; }
```
→ singly L·L

→ Types :-

① singly L·L :- 

② circular singly L·L :- 

③ Doubly L·L :- 

④ Circular Doubly L·L :- 

no head n tail

* LL is hindi

print :-

```
void print (node * head ) {
node * temp = head;
while ( temp != head ) {
cout << " temp -> data << " ";
temp = temp -> next; } }
```

→ not continuous

- single descendant so linear, not multiple so no non-linear

insert a new node :-

1) create a node
2) new node -> next = head
3) head = new node

non-empty

```
void insert ( node* & head , int data ) {
node * newnode = new node ( data );
newnode -> next = head;
head = newnode;
```

empty :- tail = newnode

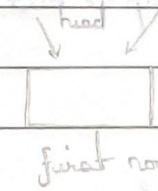
```
main { insert } ( head , 20) }
```

tail :-

1) create a node
2) tail -> new = newnode
3) tail = newnode

★ whenever v initialize head and tail with null make sure dat when first node comes initialize it and head n tail both. (empty L-L)

head = NULL
tail = NULL
} diag se handle if else lagake or bt



add dis in both head n tail func :-

```
if ( head == NULL ){
  node * newnode = new node (data);
  head = newnode;
  tail = newnode;
  return; }
```

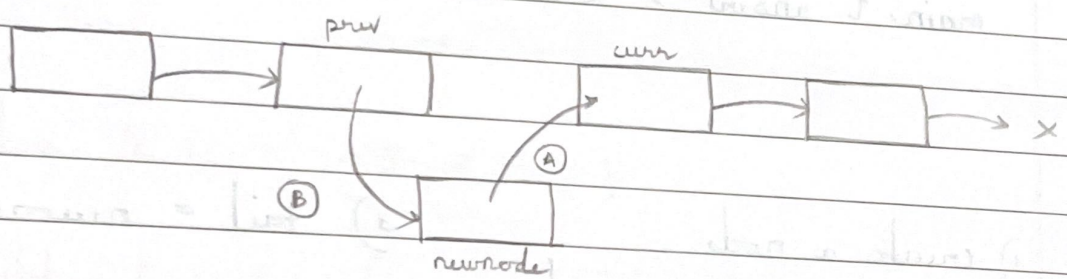
} LL empty

middle :- or atposition :-

1) position - 3rd position
2) value :- 5 ke baad

1) LL empty case
2) non empty case

A) find dat position
B) create a node
C) newnode → next = current
D) prev -> next = newnode




```
int findlength (node* & head){
 int len = 0;
 node * temp = head;
 while ( temp != NULL )
  {
   temp = temp -> next;
   len ++;
  }
   return len;
```

code :-

```
void insertatPosition (int data, int * position,
node * head, node * tail) {

L.L empty code

if (position == 0) {
    insertHead (head, tail, data);
    return;
  }

int len = findlength (head);
if (position >= len) {
    insertAtTail (head, tail, data);
    return; }

int i = 1;
node * prev = head;
while (i < position) {
 prev = prev -> next;
   i++; }

node * curr = prev -> next;

node * newNode = new Node (data);

newNode -> next = curr;

prev -> next = newNode;
```

→ Deletion :- A) 1) posi 2) value

B) 1) start 2) mid 3) end C) empty n non emp case handle

head / tail upd

head :-
1) head = head -> next
2) temp -> new = ~~next~~ NULL
3) del temp

tail :-
1) find prev
2) prev -> next = NULL
3) tail = prev
4) delete temp

mid :-
1) find prev & curr
2) prev -> next = curr -> next
3) curr -> next = NULL
4) del curr

vid 2

Doubly L.L :-

print and length T.C = O(n) S.C = O(1)

insert :-

head -

1) create a node
2) newnode → next = head
3) head → prev = newnode
4) head = newnode

tail -

1) create a node
2) ~~new~~ tail → next = newnode
3) newnode → prev = tail
4) tail = newnode

mid -

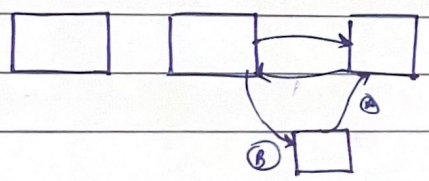
1) find prev n curr
2) create a node
3) prevnode → next = newnode
4) newnode → prev = prevnode
5) curr → prev = newnode
6) newnode → next = curr

(in dis doesnt)

(double page pichu down se connected hota but not single so v attach next first of newnode)

or

1) find prevnode
2) create new node
3) prevnode → next → prev = newnode
4) newnode → next = prev → next
5) prevNode → next = newnode
6) newnode → prev = prevnode



(in dis steporder matters)

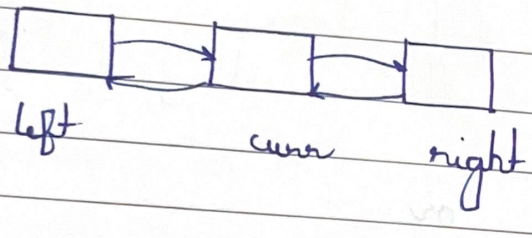
deletion :-

head :-
1) temp = head
2) head = head -> next
3) head -> prev = NULL
4) temp -> next = NULL
5) delete temp

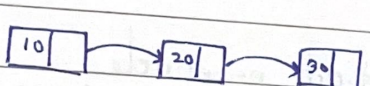
tail -
1) temp = tail
2) tail = tail -> prev
3) temp -> prev = NULL
4) tail -> next = NULL
5) delete temp

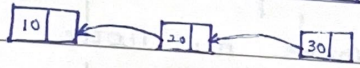
mid :-
~~1) find prev & curr~~

1) find left, curr, right
2) left -> next = right
3) right -> prev = left
4) curr -> prev = NULL
5) curr -> next = NULL
6) delete curr



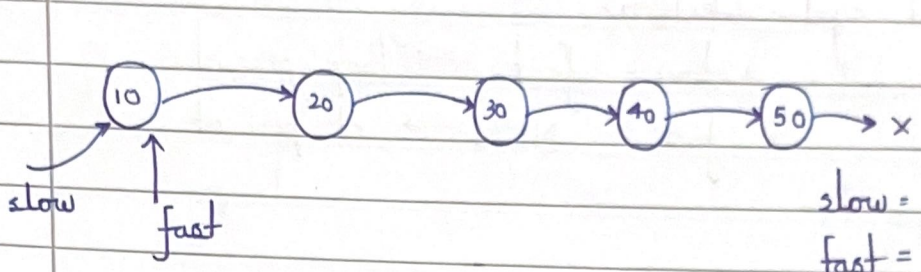
Q. Rev a L.L

i/p 

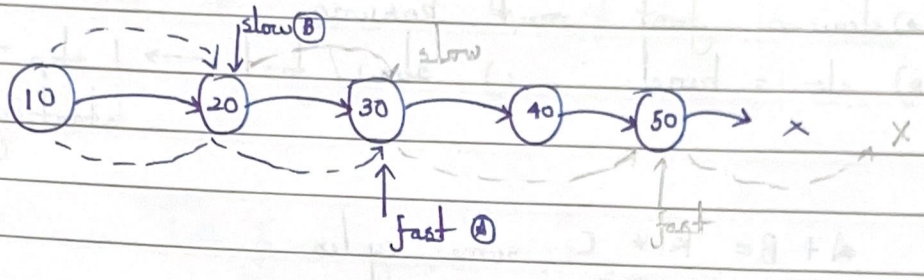
o/p 

vid 3 Q: Find middle of L.L 1) length 2) even or odd len
↓ n/2 ↓ n/2+1
3) mid

Q: slow - fast , tortoise approach

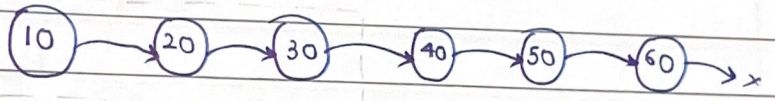
1) 

slow = 1 step
fast = 2 step

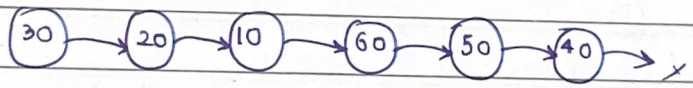
2) 

when v move fast 2 step den only v can move slow, so slow jahan last hoti dat will become mid

Q: K group → reverse LL
↓ head

i/p = 

K = 3

o/p = 

Detect and delete loop :-

- check loop present or not in LL
- starting pt of loop
- remove loop

1) Floyd cycle detection

slow == fast → loop present
fast == NULL → no present

2) a) slow n fast meet karavna
b) slow = head c) slow / fast → 1 step → meet → starting point

A + B = K * C some cycles

☆ LL qts r simple but in interview not able to solve coz lack of proof :-

★ NULL pe operation gives error

vid 4

q. check whether palidrome or not

1) find mid
2) reverse LL after mid node
3) start comparing both halves

T.C. ∈ O(1)

q. Remove dupli from LL

1) curr → data == curr → next → data

equal

curr → next = curr → next → next

not equal

curr = curr → next

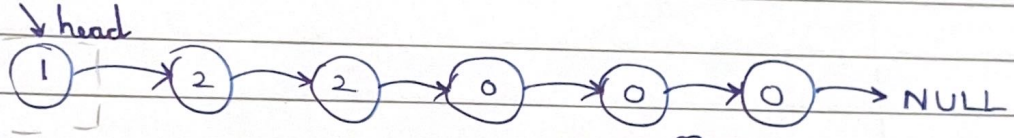
2) delete also

q. Remove dupli from unsorted L.L :-

q. Sort 0's 1's n 2's in L.L

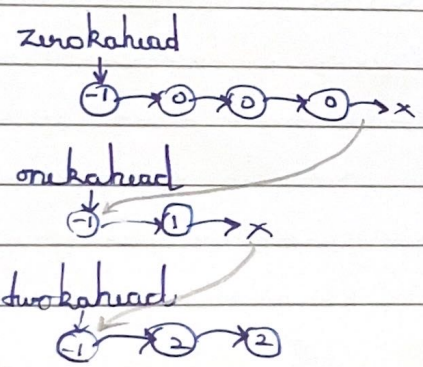
replacement :- 1) make one, zero two den count it
2) den put it in d LL in order

pyaari approach :-



(1)

A) temp = head
B) head = head → next
C) temp → next = NULL

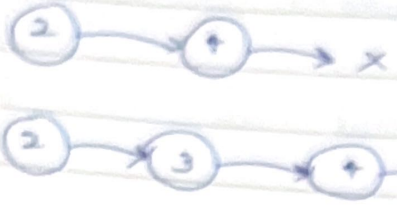


(2)

A) join
B) remove dummy nodes C) return head

same wid all node

8 add 2 no represented by L·L :-

I/P = 

2 + 3 = 5

O/P = 5

1) reverse both L·L
2) add
3) ans LL reverse