Linked List - class 1

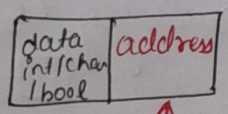
Date - 03/11/2023

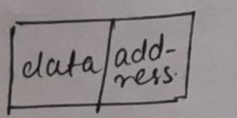
vector / Array → linear data structure
continuous memory allocation
Memory waste hota hai
insertion / shift → TC - O(n).

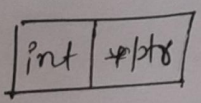
Ques Operating System ke andar storage management ke andar linked list ka usecase kha pe aata hai. ?

Linked List ⇒ * Linear Data structure
* Non-continuous memory allocation
* Collection of nodes

node →

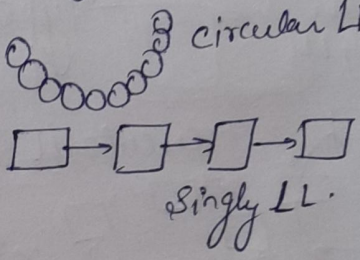






pointer

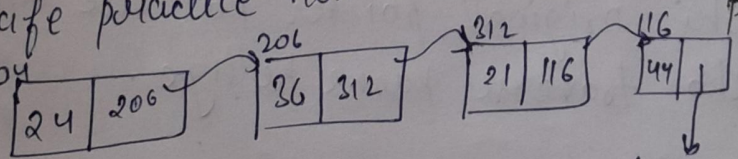
Pointer to an integer
int *ptr
pointer to a node
Node * ptr




```
class Node
{
    int data;
    Node * next;
}
```

Linked List → Magical words → LL is hindi

Safe practice hoti hai ki Null ko point karwado.



pointer declaration nhi karni hai
pointer ko initialise karni ~~NULL~~ se hai
bhale hi Null se initialise kar do.

* CHAI
(A) Paani garam
(B) TATA Tea insert
(C) Sugar daldo
(D) Doodh daalo
(E) thora sa Adrak / Elaichi

LL →

Not a concept of index

Linked List → concept of address → non-continuous → pointer ka concept aa jata hai

Linked List → wastes less than array

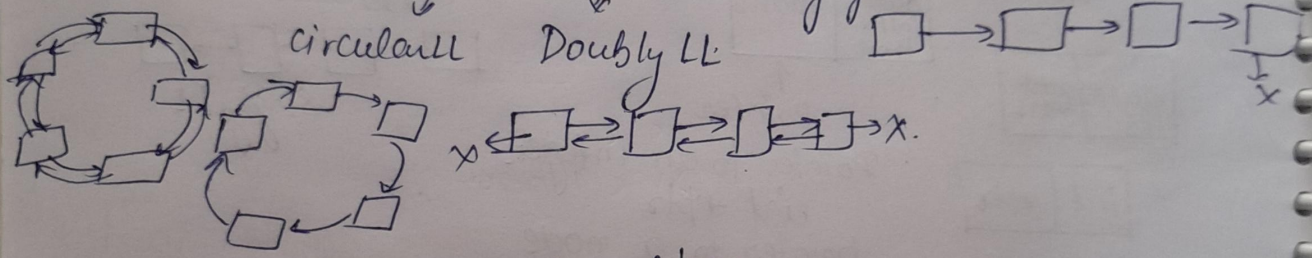
Linked List → runtime/dynamically → grow / shrink

Linked List → input/shift → TC- O(1) [location tak jana padega]

Online Test case me corner case hote hai vo miss kar rhe hoge usko sambhalna hai

Types of Linked List

- circular doubly LL
- circular LL
- Doubly LL
- Singly LL.



Dynamic allocation ho ya static se object create karo, constructor call hoga. se object create kare

Jab bhi LL function me pass hogi using head pointer or tail pointer ya koi bhi pointer ho ai hum kabhi bhi original pointer ka use nhi karegye usko traverse karne ke liye.

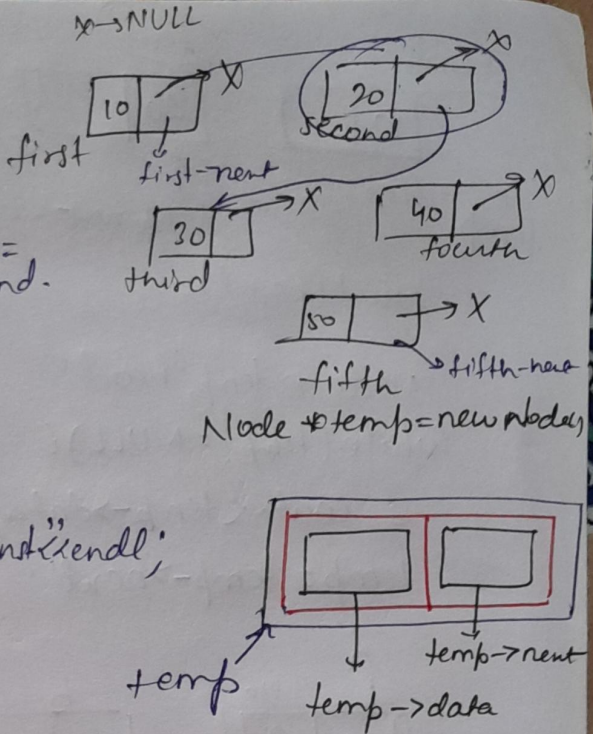
Pass by value hoya

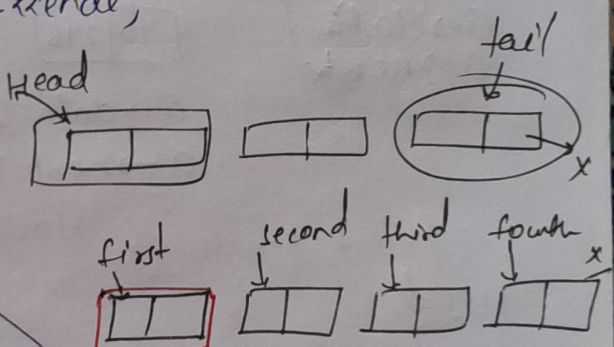
Pass by reference hum use nhi karegye

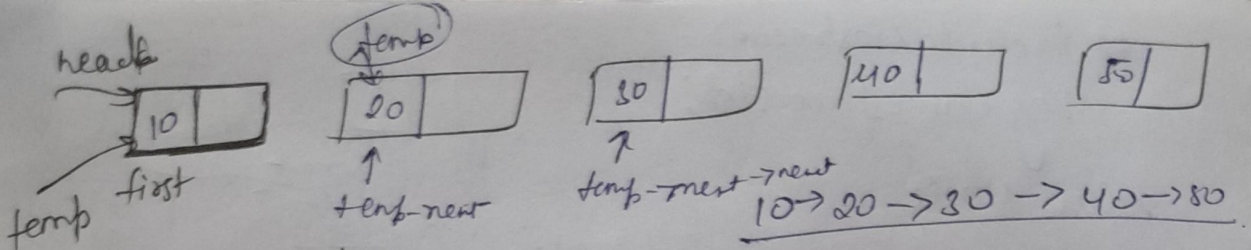
practice hogi hum temporary ya new pointer create karegye.

Node *temp = head;

```
#include<iostream>
using namespace std;
class Node
public:
int data;
Node *next;

Node(){
cout<<"I am inside default const"<<endl;
 this->next = NULL;
}

Node(int data){
cout<<" I am inside Param const"<<endl;
 this->data = data;
 this->next = NULL;
};
void print(Node* head){

}
int main(){
//creation of node
 Node a;
 Node *first=new Node(10);
 Node *second= new Node(20);
 Node *third= new Node(30);
 Node *fourth= new Node(40);
 Node *fifth= new Node(50);

first->next = second
second->next = third;
third->next = fourth;
fourth->next = fifth;

Node *head = first;

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

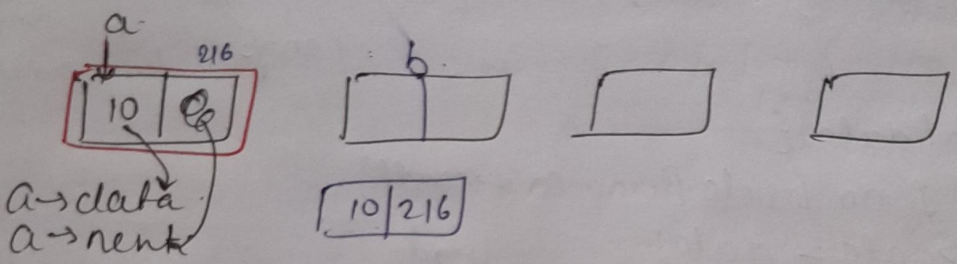






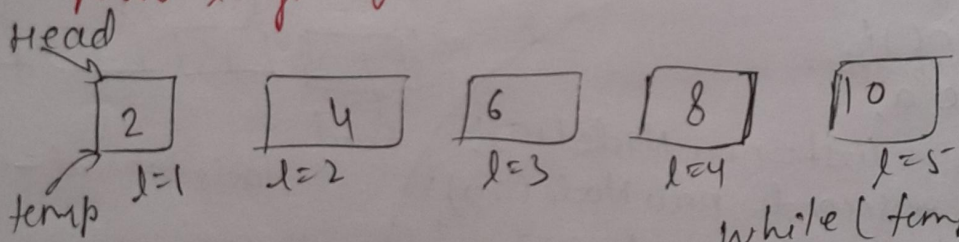

```
print(head)
{
Node * temp=head;
while(temp!=NULL){
    cout<<temp->data;
temp=temp->next.
```

Aisa koi sa tarika hai jisme linked list me in a way piche jaa sakte hai?
Think ↳ Recursion



a->data=>10
a->next->216.

Print length of Nodes



length=0.

```
int getLength(Node*a head
    Node * temp =head;
     int count=0;
    while(temp!=NULL){
        count++;
        temp = temp->next;
    }
    return count;
}
```

```
while(temp!=NULL){
    length++;
    temp=temp->next.
}
```

temp use karne ka content kya hai int main func^n ke andar ~~temp~~ head starting ll. ko darshta hai int main ne solve naamek func^n ko call kiya aur solve func^ head ko change kar diya aur jab tum use karo

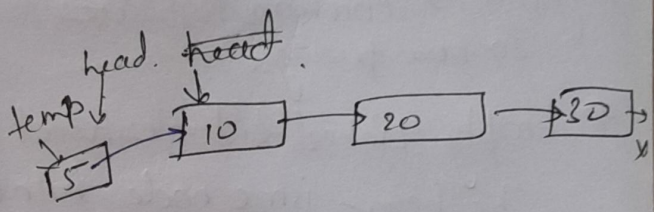
for se head ki value toh voh value change milegi. Isliye shi practice nhi hai

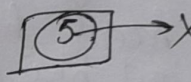
```
#include<iostream>
using namespace std;
class Node{
    public:
    int data;
    Node *next;
    Node(){
        this->data=0;
        this->next=NULL;
    }
    Node(int data){
        this->data=data;
        this->next=NULL;
    }
};
void print(Node *head){
    Node *temp=head;
    while(temp!=NULL){
        cout<<temp->data<<" ";
        temp=temp->next;
    }
}
int findLength(Node *&head){
    int count=0;
    Node *temp=head;
    while(temp!=NULL){
        temp=temp->next;
        count++;
    }
    return count;
}
```

Insertion :→

1. head → begin
2. tail
3. insert at any point

* Insertion → insert At head.





(A) create new node
(B) temp → next = head
(C) head = temp.

```
void insertAtHead(Node *&head, int data) {   // pass by reference
                                             // Node *&tail
    else {
        // create a new Node.
        Node *newNode = new Node(data);
        // create a new node
        newNode->next = head;
        // update head.
        head = newNode;
    }
}
```

Yha galti hui thi ki pass by reference karna jaruri hai.

Tab bhi LL ke andar func" me change kar rhe hai toh dhyan rakho pass by reference karna

toh isliye traversal ke time temp pointer bna rhe hai' head pointer ka use nhi kar rhe hai'

Galti → common testcase sochna padega

* Empty Linked List me head kha pe hoga — isme node hi nhi hai'

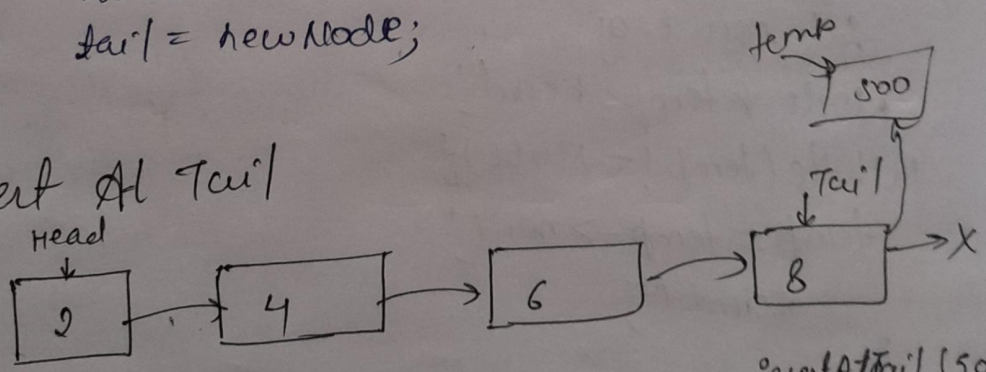
toh head hoga hi nhi yani head NULL ke upar hoga or tail bhi NULL ke upar hoga

head == tail

Single elt ke case me.

```
if (head == NULL) {
    // empty LL.
    // create new node.
    Node *newNode = new Node(data);
    // step 2:- update head.
    head = newNode;
    tail = newNode;
}
```

Insert At Tail



insertAtTail(500)

(A) create new Node

(B) tail->next = temp

(C) tail = temp

```
void insertAtTail(Node *&head, Node *&tail, int data){
    if(head==NULL){
//empty LL
// Step1: create toh karo Node
    Node *newNode = new Node(data);
// Step2:- //Single node hai entire list me that's why head
    // and tail ko ispar point karwado
     head= newNode;
     tail = newNode;
  }
  else{
    Node *newNode = new Node(data);
     tail->next = newNode;
      tail = newNode;
  }
}
```

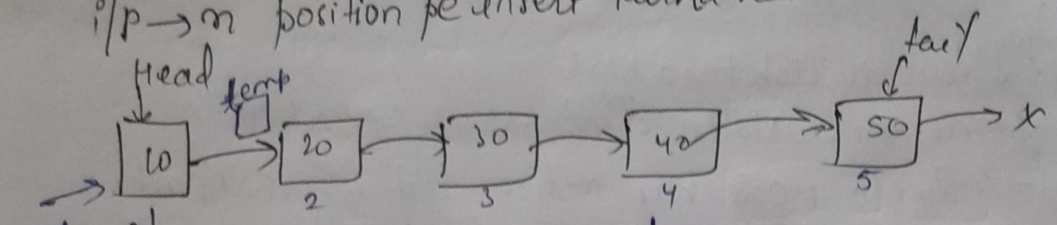
temp pointer ko head ke upar rakho aur usko end tak lejao then fir ye above logic lga do.

```
void createTail(Node *head, Node *&tail){
     Node *temp = head;
       while(temp->next !=NULL){
          temp = temp->next;
       }                          // jab ye loop khtam hogya hoga then
       tail = temp;               // temp wala pointer khada hoga last
  }                               //   node par tab tail = temp karke
                                  //      tail ko lastnode pr le aao.
int main(){
  Node *head = NULL;
  Node *tail = NULL;
   insertAtHead(10, head, tail, 10);
   insertAtHead(20, head, tail, 20);
   insertAtHead(30, head, tail, 20);
   insertAtTail(head, tail, 50);
```

30->20->10->50->

Insert At Position

i/p → n position pe insert karna hai.



phele me!
beech me
last pe me.
yhi 3 teeno jagah insert kar sakte hai

pos < 1 → can't insert
pos = 1 → insertAtHead
pos = 5 → insertAtTail
pos > 1 && pos < 5 → logic find.
pos > 5 → can't insert.

Void insertAtPosition (Node *head, Node *tail, int data, int position) {

position = 2.

(A) create a node.

(B) traverse curr/prev to position

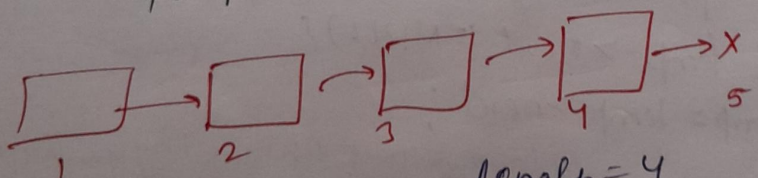
(C) prev → next = temp

(d) temp → next = curr.

if pos <= 1 insertAt Head

else if pos > length
↳ insertAt Tail

else → middle.

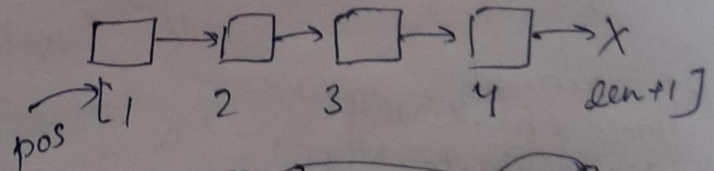


length = 4

[position = 1]
↳ insertAtHead

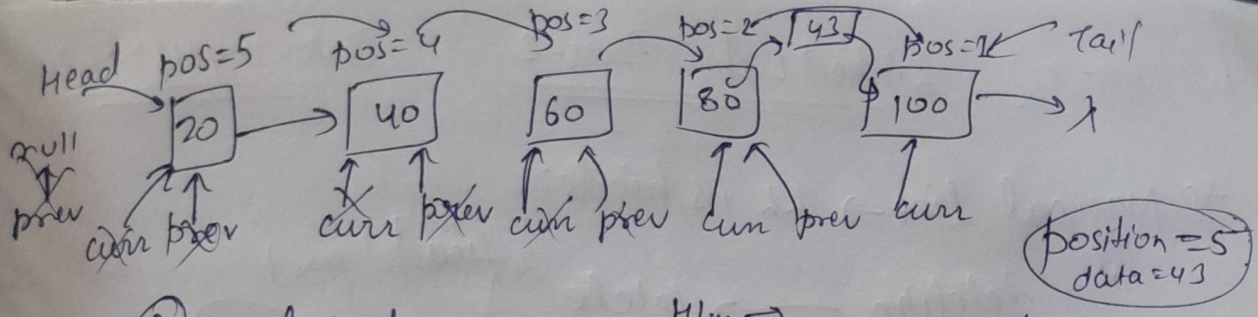
[position = len+1]
↳ insertAtTail()

pos [1 ↔ length+1].



pos [1, 2, 3, . . . , len+1]

insertAtHead
insertAtTail
insertMiddle → logic



(A) create node
(B) while(pos != 1)
(C) prev->next = newNode
(D) newNode->next = curr;

H/w → only previous pointer se solve karo.

```
void insertAtPosition(int data, int position, Node* &head,
        Node* &tail ){

    int len = findLength(head);
    if(position == 1){
        insertAtHead(head, tail, data);
            return;
    }
    else if(position > len){
        insertAtTail(head, tail, data);
          return;
    }
    else{
        Node* newNode = new Node(data);
        Node* prev = NULL;
        Node* curr = head;
          while(position != 1){
              position--;
              prev = curr
              curr = curr->next;
        }
        newNode->next = curr;
        prev->next = newNode;
    }
}
```