Constructors

The fn which has name of the class;

ex → class student{

}

s1 student()

→ default const no Arguments

constructor

→ same name
→ No return type
→ NO input arguments

* we never creats a default constructor when class is made the default const. is ~~auto~~ autometaelly created.

True.

```
student() {

}
```

When we made object constructor(default) is autometically created

```
student s2;
student * s3 = new student;
*s3.student;
s3 → student;
```

→ he creats garbage value in the "a" data.

Camlin Page
Date / /

1. Work of constructor is that it initilize the default value to every data member of the class

2. for every obj there is a default constructor is created.

When Student S1; written at that pt. Constructor created

Parameterized Constructor.
↳ Constructor which takes Argument.

```
Student(int r)
    ↳ take parameter / Argum
{
cout << "Constructor 2 called!" << endl;
rollNumber = r;
}
```

When we won't want to created that our obj doesn't create default. const then we write in this form

S4.student() → defau.

student S4(10);
↳ value passed
{S4.student(10);} interpretated like this.
↳
No default constructor created.

dynamically

Student *ss = new Student (101);

ss → display().

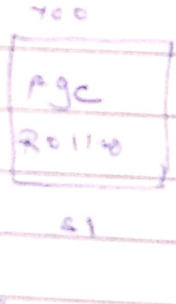
* This keyword

Holds the address of current object.

by ex:-

let we created

Student s1 (10, 1001);



object ka address ata gaya

Explain by example

Student (int rollnumber)
cout << "constructor 2 called <<endl;

This contains the address of our obj
which called the constructor

Si ki memory block ka address. This
maj

We can use it like

```
Student(int a, int r)
{
  cout<< " this: " << this <<endl;
  cout<< " Constructor is called !" <<endl;
   this -> age = a;
   this -> rollNumber = r;
}
```

More Effectively

```
Student (int rollnumber)
{
   this -> rollnumber = rollnumber
}
```

Copy Constructor

```
Let - Student S1(10,1001);
      Student S2;
one way { S2.age = S1.age;
          S2.rollno = S1.rollno;
```

easy way,

```
Student S2(S1);
```

↳ it creats copy constructor internally.

How to call.

{S2.student(S1)}

↳ Internal C.C

dynamically

```
Student *S3 = new Student(20,2001);
Cout<<"S3";
S -> display();
```

```
Student S4(*S3);
```

↳ deref * . pass kro

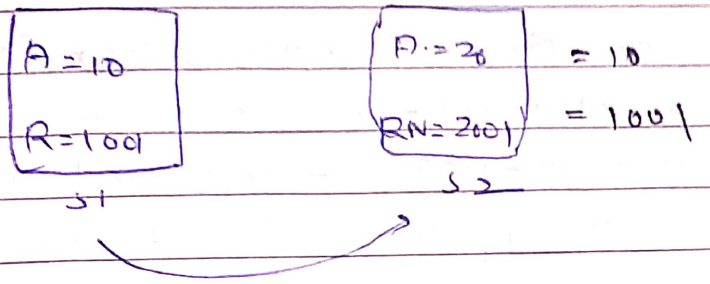
dynamic to dynamic → `Student *S5 = new Student(*S3);`

```
Student *S6 = new Student(S1);
```

↑ dynamic to non dynamic

Copy Assignment Operator (=)

```
Student S1(10,1001);
Student S2(20,2001);
```



if we `S2 = S1` → This operator do all work for us.

```
{ S2.age = S1.age;
  S2.RollNo = S1.RollNo; }
```

Ex

```
Student S1(10,1001);
Student S2(20,2001);

Student *S3 = new Student(30,3001);
```

no constructor created only copying value.

`S2 = S1;` (S2 mai S1 ki value)

`*S3 = S1;` (S3 mai S1 ki value)

they exist in memory

Destructor (To distruct/distroy memory)

- → Same name as our class
- → No return type
- → No input arguments.

```
~Student(){
}
```

tilda

* It calls when our obj is going to distroy
→ ek ~~f~~ distructor calls when obj ka scope khatam hone wala hota hai

distructor doesn't calls when the memory is dynamically alloted because the object is a pointer type and you can't ~~destroy~~ a if you want to distroy memory of ~~the~~ pointer variable then you have to write

delete s3.

Summary (constructor. 4dyst.)

```
int main() {

    Student s1;  //constructor 1 called
    Student s2(101);  //constructor 2 called
    Student s3(20,702);  //constructo 3 called
    Student s4(s3);  //copy-constructor
    s1 = s2;  //copy assignment Operator

    Student s5=s4;  //copy Constructor
```

↳ Student s5;
s5 = s4 → (X)

↳ Student s5(s4);

Function class.

Q. Create a class fraction. (12/5).

Datamember -> numerator = 12
              -> denomentor = 5

fraction class

```
Class Fraction {

    Private:
        int numerator
        int denominator

    Public:
        fraction (int numerato, int denom)
        {
        this -> numerator = numerator;
        this -> denominator = denominator;
        }

    void print() {

    Cout << numerator << " / " << denominator << endl;

    }

Void add (fraction f2)
    {
        int lcm = denominator * f2.denominator
        int x = lcm / denominator;
        int y = lcm / f2.denominator;
        int num = x * numerator
                + (y * f2.numerator);
```

```
    numerator = num;
    denominator = lcm;
}

Void print() K neelu

Void Simplify()
    {
        int gcd = 1;
        int j = min(this -> numerator,
                    this -> denominator);
        for(int i = 1; i <= j; i++)
        {
            if(this -> numerator % i == 0 && this ->
                denominator % i == 0)
            {
                gcd = i;
            }
        }
    this -> numerator = this -> numerator/gcd
    this -> denominat = this -> denomm/gcd
}
```

Shallow & deep copy.

```
Class {
    int age;
    char* name;
    }

Public:
    Student (int age, char * name)
        {
        this → age = age;
        this → name = name;
        }

    void display ()
        {
        cout << name << " " << age << endl;
        }

→ Main ()
    {
    char name [] = "abcd"
    Student S1 (20, name);
    S1. display ();

    name [3] = 'e';
    Student S2 (24, name);
    S2. display ();

What happens next;
```

1st 780

name → [a|b|c|d|\0]

How 1st fy work.

S1. Student (20, name)

↑ iska age = 20

→ iska name = 780

When Student. S1 (20, name). Write

780

this → age =

this → name =

now,

name[3] = 'e'

[a|b|c|e|\0]

ye bhi change hoga

Student S2. (24, name)

this → age = 24

this → Name = 780 ← agar ye change to

Clone object same array ko point karta hai, " kese ek mai change clone mai

basically shallow copy

when we pass our array, rather then copying entire array, it copy the address of 0th index.

Now what we have to do.

when S1.student(20,name).

we have to copy whole array in name instead of coping address of 0th index

→ it is also called **deep copy**

this → name = new char [strlen(name) + 1];

strcpy(this→name, name);

(for null char)

How work?

name → 780

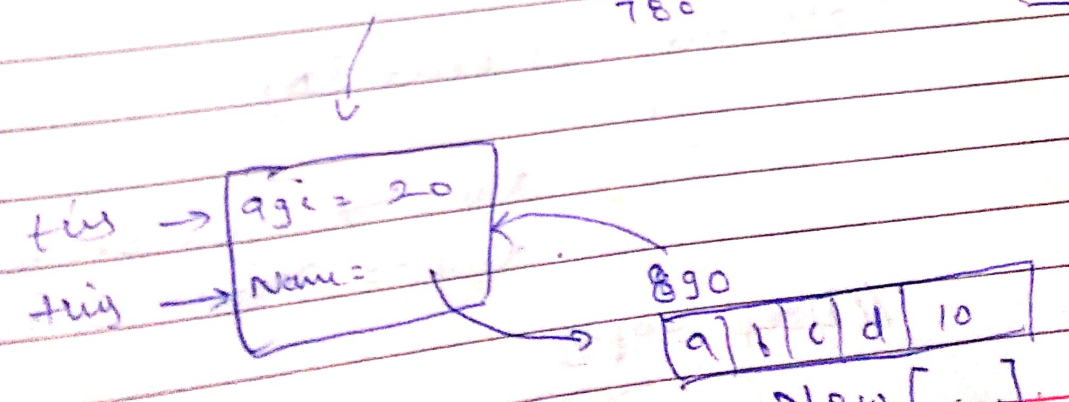
a	b	c	d	\0

student S1(20, name)

780

age = 20

name = 780

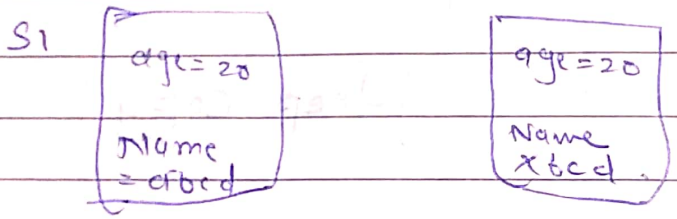


New [ ] copied to name

Applying Copy Constructor (How copy constructor works)

```
int main {

    Student S2(S1);
    S2.name[0] = 'x';
    S2.display();
```

→ Procw



S2.display() → 20 xbcd
S1.display() → 20 xbcd

Internally what happens.

* Whenever we used copy constructor it used Shallow Copy.

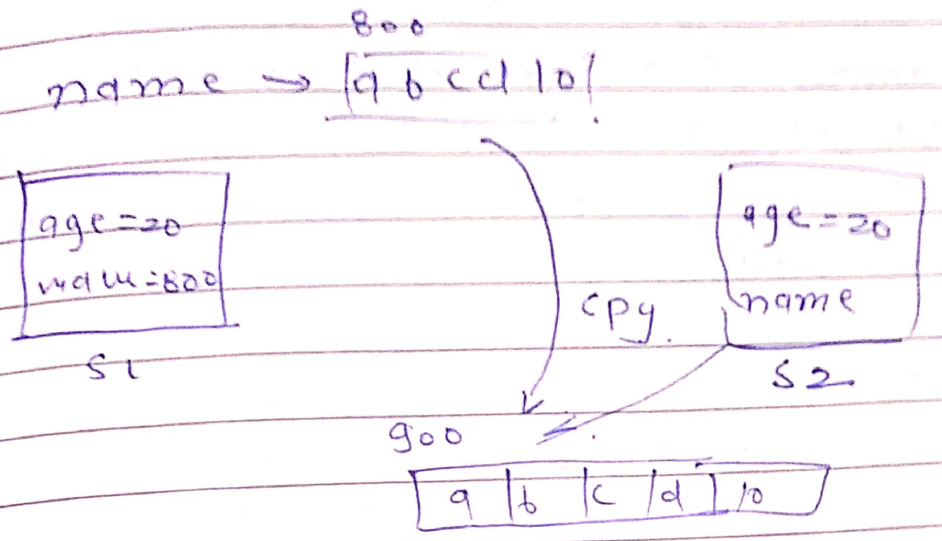
Making own copy constructor

```
Student (Student &S)
{
    this → age = S.age;
    this → name = S.name;
}
```

//Deep copy

```
this → name = new char[strlen(S.name)+1]
```

Now what happening



Student (Student s) ?

Student s2 (s1)

Student s = main (s1)

naya studn. bano. or main k s1 ki value copy karo

Initialization list

Hummare constant data members ko constructor k andar jane se pehele he initize (memory allocation k time) karta hai.

int a = 5;
int const b = a;
can't change b.

int a;
a = 5;
error

int const b; → have garbage
b = a

now you want to change, it can't possible

We have to initise const variable at time of its creation

int i = 5;
int &j = i;

Point 1.

but,

int &j;
j = i; → not possible

```
Class student {
    Public:
        int age;
        int const rollnumber;
        int &x;

    Student (int r, int age) : rollnumber(r),
                               age(age),
                               x(this->age)
    {

    }

int main ()
{
    #include <bits/stdc++.h>
    using namespace std;
    #include "initialisation_list_cls.cpp"

    int main()
    {
        Student S1(101, 20);

    }
```

Constant Func.

```
class function {

Private :
    int numerator;
    int denominator;

Public;
    function() {

    }
function (int numerator, int denominator)
    {
    this -> numerator = numerator;
    this -> denomentor = denominator;
    }

Int getNumetor() -> ye kese prop
                    const. change. nahi kar
    {
    return numerator;
    }
int getdenomintor() -> ye kres -
    {                  const
    return denominator;
    }
Void Set Numerator ()
    {
    numerator = n;
    }
Void set denominator (int d)
    {
    denominator = d;
```

void main()
{
function f1(10, 2)
function f2(10, 2)
}

function const f2 → This is our constant object
↳ fs se hum koye change nahi kar sakte

fs se koye f" bhi call nahi kar sakte jo humare property change kar deta ho.

compiler koye normal f" ko call karne deg a constant object ka.

Static Member

class student {

Public:

int rollnumber;

int age;

Static int totalstudents; // total member of student

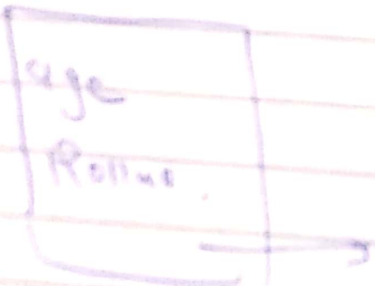
this property is not for different

this property not for obj it for class, this is not different for different obj

har object k leye seprately (sep nahi hoga

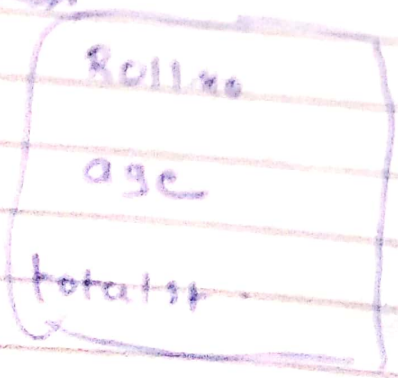
or

if we created Student S1;



→ No total student

Class



we can access non static member like

S1.age = ;
S1.RollNo =

But How to access static memory

Cout << Class name :: total student

scop.rel

How to initilize static d.M

int Class k bahar. Hota initilize

int Student :: totalStudent = 0;
//initilize

0 set ho gye

```
int main()
{
    Student S1;
    Cout << S1.rollnumber << " " << S1.age
    << endl;
    Cout << Student :: totalStudent << endl;
```

Static likhne jaai d.M. kt P copy nahi hoga kese bhi obj ki memory mai but hum uss property ko change kar sakte hai jese bhi d.M se, vo change jo bhi hoga vo class mai change hoga

How to update

Let's make constructor

```
Student() {
    total student++;
}
};
int Student :: total student = 0;
```

How to make & use static fn.

```
int getRollNumber()
{
    return rollNumber;
}
```

Static int getTotalStuden()
{
return total studen;
}

class ka fn hai

How to access it

```
cout << Student :: getTotalStudent();
```

fn name;

}

* Static ~~parameter~~ only access static data
* Static fn this pointer nahi hota kyu ki hum usse obj se access nahi kar re

Camlin | Page
Date / /

Operator Overloading

Kaise hum operator ko overload kar k use use kar sakte hai.

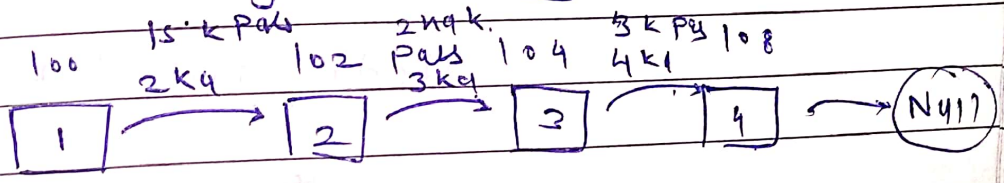
Dynamic Array

```
Class DynamicArray {

    int *data;
    int nextIndex;
    int capacity;        // total size

Public:
    DynamicArray()
    {
        data = new int[5];
        nextIndex = 0;
        capacity = 5;
    }

    Void add(int element)
    {
        if(nextIndex == capacity)
            int *newdata = new int[
                                2 * capacity
        }
        data[nextIndex] = element;

for(int i=0; i<capacity; i++)
        {
            newdata[i] = data[i];
        }
        delete [] data;
        data = newdata;
        capacity * = capacity;
                     2

int get(int i)
    {
        if(i < nextIndex)
            return data[i];
        else
            return -1;
    }
```

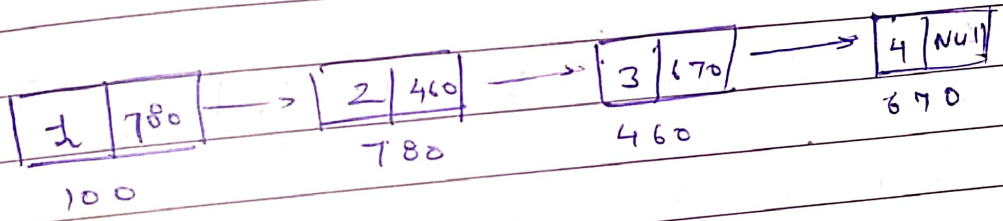
D.S.T.

Linked list

linked list kya bolte?
They are like array but not array.
They also store element but not continously
They store randomly in memory



hummare
Pass 1st
element k q
address

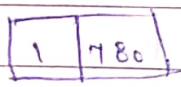


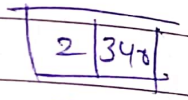
How to store element in linked list?
'X' → No pointer. (only address)
X → No array

We gonna build Class.

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

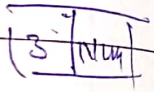
```
class Node {

    Public:
        int data;
        Node * Next;

        Node (int data)
        {
            This -> data = data;
            next = null;
        }
};

# include <iost ->

# includ "file name"

int main()
{
    Node n1 (1)
    Node * head = & n1

    Node n2 (2)
    n1.next = & n2
}
```





100

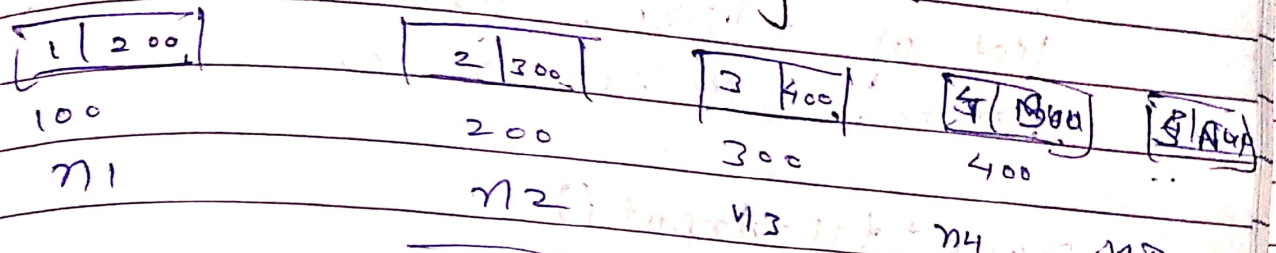
780



How to create dynamically

```
Node *N3 = new Node(10);
Node* Head = n3;

Node *N4 = new Node(20);
n3 -> Next = n4;
```

Q Create 5 nodes linked together.




```
Node n1 (1);
Node * head = &n1
Node n2 (2);
Node n3 (3);
Node n4 (4);
Node n5 (5);

n1.next = &n2;
n2.next = &n3;
n3.next = &n4;
n4.next = &n5;

print (head);
```

```
Void print (Node
            * head)
{
While (head != NULL)
{
cout << head -> data;
cout
head = head -> next;
}
}
```

We don't have to lost head. because we can use it in other.

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

Create a f^n which create linked list of 'n' size. User se input & head return. (Multiple node)

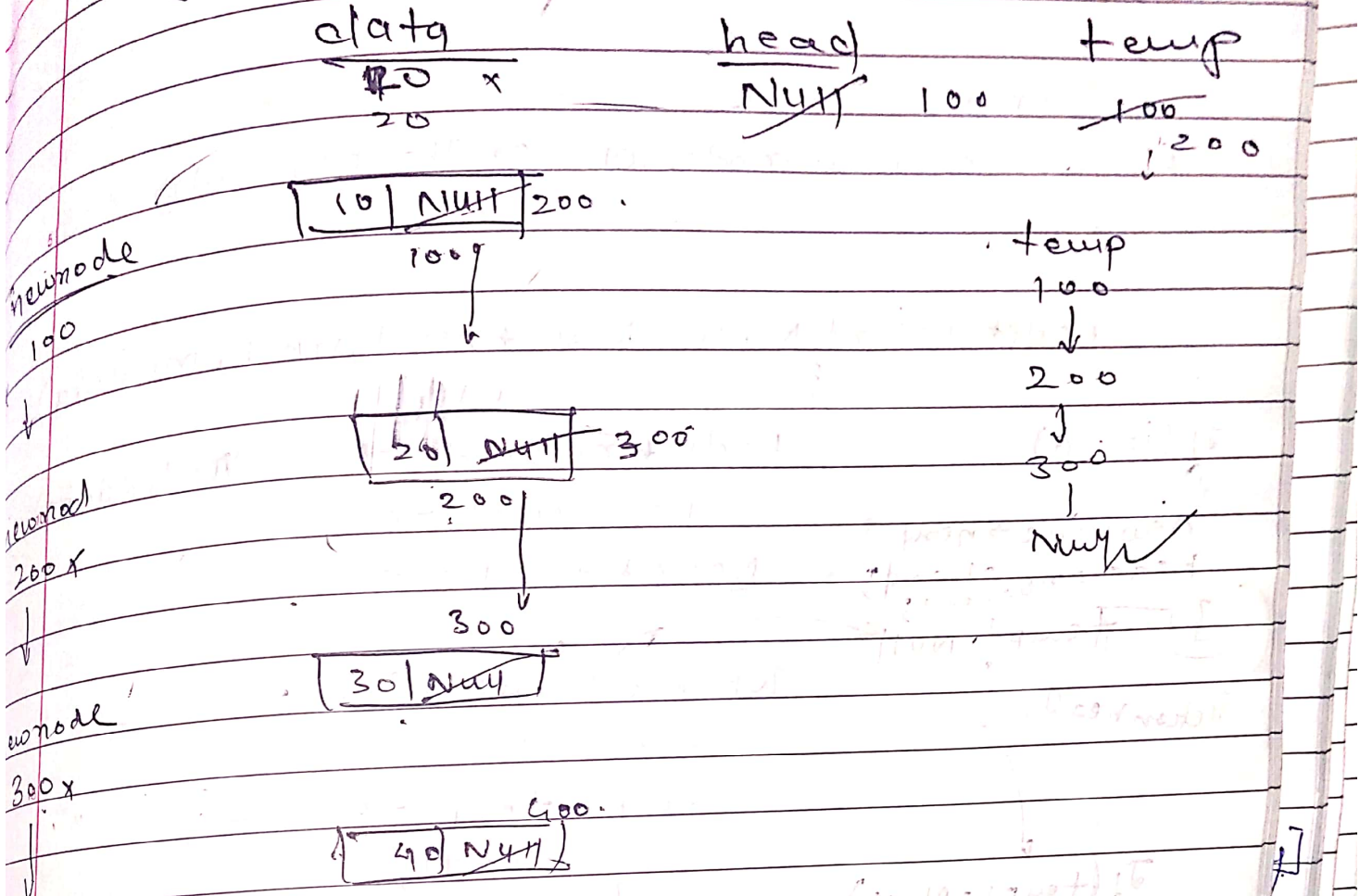
```
Node * takeInput()
{
    int data;
    cin >> data;
    node * head = NULL;
    while (data != -1)
    {
        node * newnode = new node(data);
        node.n(data); // → memory allocated.
        head = newnode;
        if (head == NULL)
        {
            head = new node;
        }
        else {
            Node * temp = head;
            } while (temp → next != NULL)
            {
                temp = temp → next;
            }
            temp → next = newnode;
        }
        cin >> data;
    }
    return head;
}
```

For making better

```
{
    Node * tail = NULL;
    Node
}
```

Camlin Page
Date / /

Dry run:



it take complexity $O(n^2)$

we can also do it $O(n)$ by updating and making tail

```
if (head == NULL)
{
    head = newnode;
    tail = newnod;
}
else {
    tail->nex = newnode;
    tail = tail->next;
}
```

```
Void Print (Node *head)
{
    While (head != NULL)
    {
        cout << head->data << " ";
        head = head->next;
    }
}
```

Op.1

How To insert node b/w two element or at i^{th} index of a list

Step 1

* Create new node of the particular data

```
Node* insertNode(Node *head, int i, int data)
{
    Node *newnode = new Node(data);
    int count = 0;
    Node *temp = head;
    while (count < i-1)
    {
        temp = temp -> next;
        count++;
    }
    Node *q = temp -> next;
    temp -> next = newnode;
    newnode -> next = q;
    return head
}
```

```
if (i == 0)
{
    newnode -> next = head;
    head = newnode;
}
return head;
```

```
if (temp != NULL)
```

Overcoming some problem (Segmentation Problem)

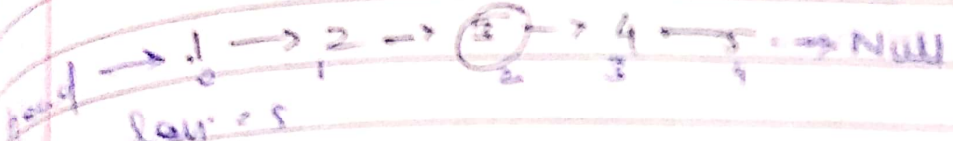
Op.2

Camlin Page
Date / /

deletion of node from list.

```
node * dele (Node *head, int j)
{
    Node *newnode = new Node(head);
    int count=0;
    Node *temp = head;
    while (count < j-1);
        {
            temp = temp->next;
            count++;
        }
    Node *a = temp->next;
    Node *b = a->next;
    temp->next = b;

    Cleleted;
}
return head;
}
```

Mid Point of linked list (mid element)



Basic algo

slow = slow → next

fast = fast → next → next

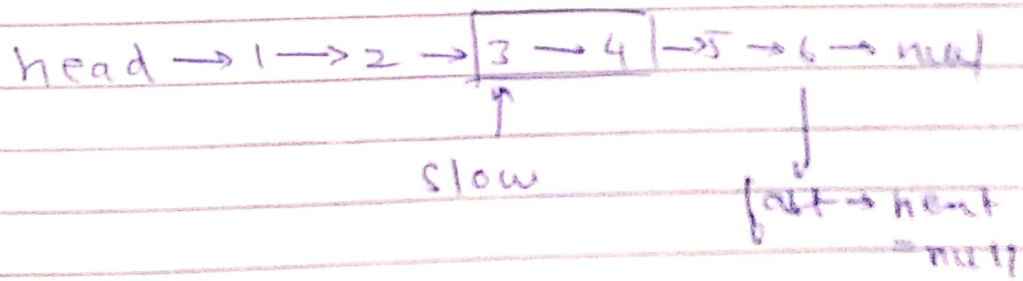
for odd value

fast == null

head → 1 → 2 → 3 → 4 → 5 → NULL

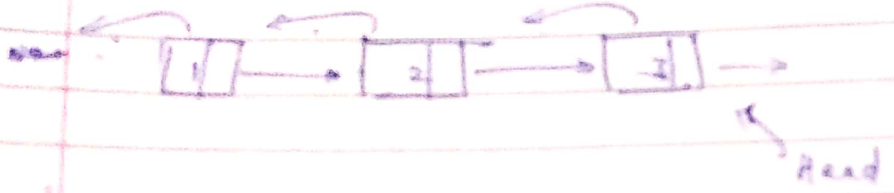
for even value

fast → next == null



Recursively

1. Reverse ~~Tree~~ of linked list



1. Base case → length 1 or 0
2. Rec call →
3. Small call

```
Node* reverse(Node* head)
{
    if(head == NULL || head->next == NULL)
    {
        return head;
    }
    Node* smallAns = reverse(head->next);
    Node* temp = smallAns;
    while(temp->next != NULL)
    {
        temp = temp->next;
    }
    temp->next = head;
    head->next = NULL;
    return smallAns;
}
```

Small ans.

Dry run

1 - 2 - 3 - 4

2 - 3 4

3 - 4

4 → head

Complexity

$T(n) = T(n-1) + n - 1$

$T(n-1) = T(n-2) + n - 2$

$T(n-2) = T(n-3) + n - 3$

$T(n) \leq (n-1) + (n-2) + (n-3) \ldots$

n^2

$T(1) = T(1)$

In time complexity $O(n)$

```
Class Pair {
    Public:
        Node * head;
        Node * Tail;
};

Pair Reverse (Node * head)
{
    if (head == NULL || head -> next == NULL)
    {
        Pair ans;
        ans.head; ans.Tail = head;
    }
}

Pair SmallAns = Reverse(head -> next);
SmallAns.tail -> Next = head;
head -> next = Null;
Pair ans;
ans.head = SmallAns.head;
ans.tail = head;
Return ans;
```

$T(n) = T(n-1) + k$

$n \neq 0$ (as)

Order of (n) O(n).

```
Node * Reversal_3(Node *head)
{
    if(head == NULL || head->next == NULL)
    {
        return head;
    }
    Node *small_Ans = Reversell(head->next);
    Node *tail = head->next;
    tail->next = head;
    head->next = Null;
    Return small_Ans;
}

Node* take_input()
{
    int data;
    cin data;
    node* head = Null; node* tail = NULL;
    While (data != -1)
    {
        node* newnode = new node(data);
        if (head == NULL)
        {
            head = newnode;
            tail = newnode;
        }
        else { tail->next = newnode;
               tail = tail->next;
        }
    cin >> data;
}
```

```
return head;
}

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

int main()
{
    Node* head = takeInput();
    print(head);
    head = reverse(head);
    print(head)
}
```

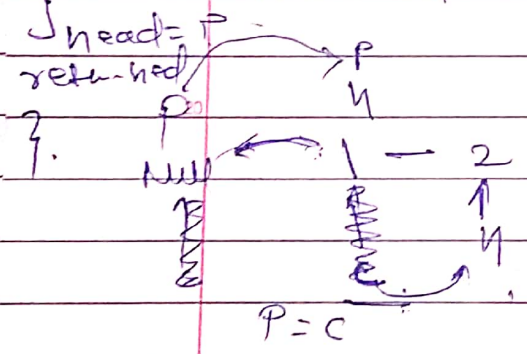
Iteratively

1 — 2 — 3 — 4 — Null

4 — 3 — 2 — 1 — Null

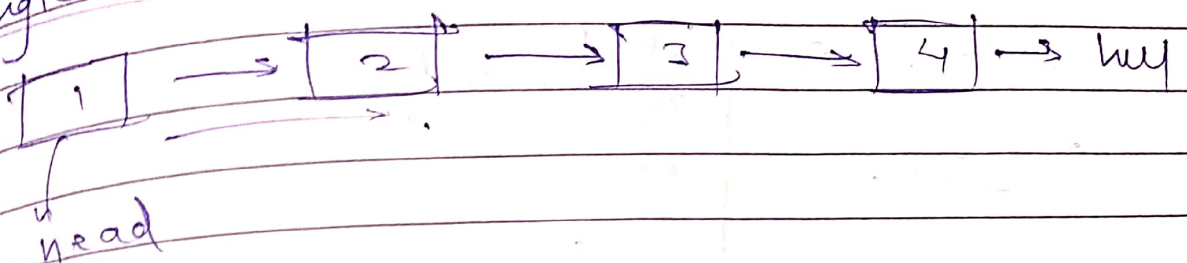
```
Node* Iteratively(Node* head)
{
    Node* current = head;
    Node* prev = NULL;
    Node* n;
    while(current != NULL)
    {
        n = current->next;
        current->next = prev;
        prev = current;
        current = n;
    }
    head = prev;
    return head;
}
```

```
Node* Iteratively(Node* head)
{
    while((current != NULL)
    next = current->next;
    Node* current = head;
    Node* prev = NULL;
    next = current->next;
    current->next = prev;
    current = next;
    prev->next = head;
    head = head->next;
}
```

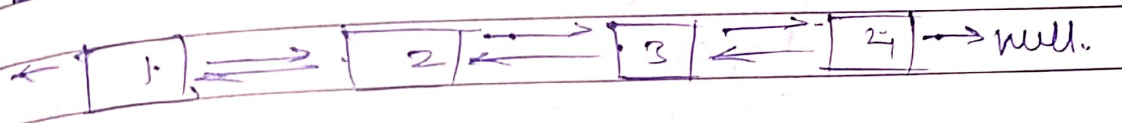


Camlin Page
Date / /

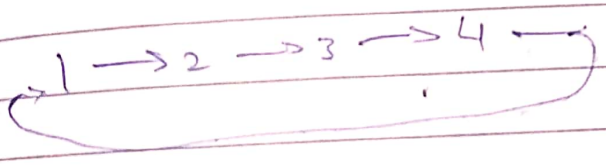
1. Single linked list



2. double linked list



3. Circular singly LL.

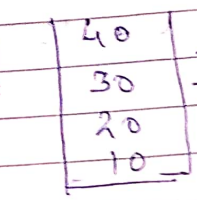


4. circular doubly linked list

* Reverse a linked list in grp of given size

Stacks & Queues

Why stack ?
(LIFO)



fun. → Insert() → push()
del → Pop() (nextInd - 1)
top() → (nextIndex - 1)
size() → nextIndex;
isEmpty() →

```
Class StackUsingArray
{
    int *data;
    int nextIndex;
    int capacity=0;

    Public:
    StackUsingArray (int totalSize)
    {
        data = new int[totalSize];
        nextIndex = 0;
        capacity = totalSize;
    }

int Size()
{
    return nextIndex;
}
```

```
bool Empty()
{
    if(nextIndex==0)
    {
        return true;
    }
    else
    {
        return fa
    }
    return 0; nex
}

// Insert element
void Push(int
{
    data[ne
    nextI
    if(
    {
        co
        re
    }
    data
    ne
}
```

```
bool Empty()
{
/* if(nextIndex==0)
   {
     Return true;
   }
else
   {
     Return false;
   }
*/
Return nextIndex==0;
}
//Insert element
void Push(int element)
   {
   data[nextIndex] = element;
   nextIndex++;
   if(nextIndex == capacity)
   {
   cout << "Stack full" << endl;
   return;
   }
data[nextIndex] = element;
nextIndex++;
}
```

// delete element

```
int pop() {
        nextIndex--;
        return data[nextIndex];

    if(isempty())
        {
        cout<<"Stack is empty"<<endl;
        return INT_MIN;
        }
    nextIndex--
    return data[nextIndex];
    }

int Top()
    {
    if(isEmpty())
        {
        cout<<"Stack is empty"<<endl;
        return INT_MIN;
        }
    nextIndex--;
    return data[nextInd-1]
    }
```