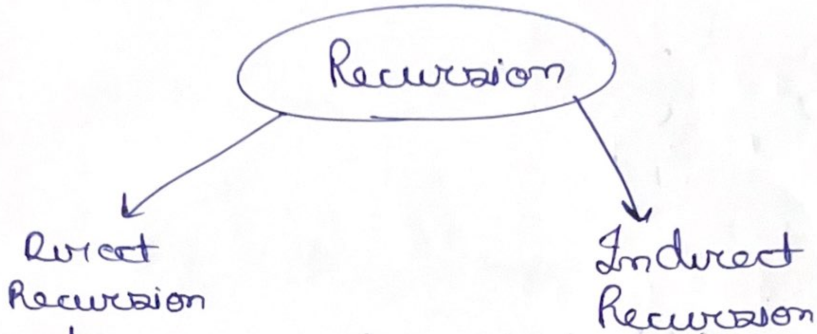
# Recursion

Recursion is a process in which a function call itself again and again
Recursion avoid Loop in a program



```
Void Sum (int a)
{


  Sum(a-1);
}
```

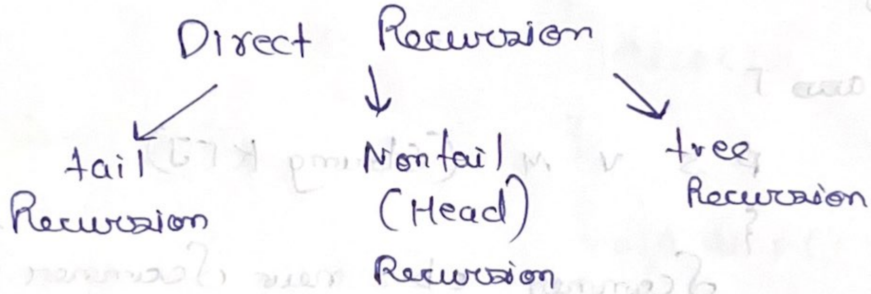
```
void show()
{
  disp();
}

void disp()
{
  show();
}
```

Create a Recursive function to display fibonacci Series.

```
import java.util.*;
class Calc
{
    static int fib (int a)
    { if (a<=1)
        return a;
      else
        return fib (a-1) + fib (a-2);
    }
}

class T
{   p s v M (String K[])
    {
      Scanner ob = new Scanner (System.in);
      SOPL ("Enter terms");
      int N= ob.nextInt();

    for (int i=0; i<N; i++)
        SOPL (Calc.fib(i);

    }
}
```

Enter terms 5

| i | output |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | fib(1) + fib(0) = 1 |
| 3 | fib(2) + fib(1)<br>1 + 1 = 2 |
| 4 | fib(3) + fib(2)<br>2 + 1 = 3 |



Tail Recursion

```
class A
{   static void disp (int a)
    {   if (a>0)
        {sopl (a);
         disp (a-1);
        }
    }
}
```

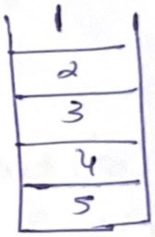
```
class T
{   P S V M (String K[])
    { A.disp(5);
  }
  }
```

O/P

5
4
3
2
1

## Non tail (Head) Recursion

```
class A
{
      static void disp (int a)
      {
          if (a>0)
          { disp(a-1);
             SOPL(a);
          }
      }
}
```



O/P

1
2
3
4
5

```
class T
{ P  S V  M (String K[])
  { A.disp(5);
  }
}
```

~~WAP that~~

Create a method that take a Parameter and return factorial

```
Class calc
 {
      static int fact (int a)
       {
            int i, f=1;
            for (i= 1; i<=a ; i++)
             {
                  f = f*i;

             }

             return f;

       }

 }


Class T
  {
       P S V  M  (String K[])
        {
            int r = Calc.fact(5);
            SOPL (r);
        }

  }
```

create a Recursive Method that takes a parameter and return factorial of given no.

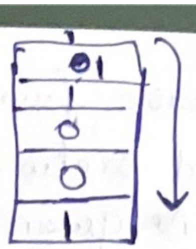
```
class Calc
{
     static int fact(int a)
     {
          if (a==0)
          return 1;
          else
           return a * fact (a-1);
     }
}

class T
{
     p s v m (string k[])
     { int r = Calc. fact(5);
       SOPL(r);
     }
}
```

```
5 * fact (4)
5*4 * fact(3)
5* 4 * 3 *(fact(2)
5* 4 * 3 * 2 *(fact(1)
5* 4* 3 * 2 * 1 * fact(0)
5* 4*3 *2*1*1
 => 120
 o/p
```

Create a method that take a Parameter and display binary of given no

```
Class Calc
{
   static void bin(int a)
    {
       if (a>0)
       {   bin(a/2);
           SOP(a%2);
       }
    }
}
Class T
{  P S V M (String k[])
    {   Calc.bin(25);
    }
}
```

bin (25)
↓
bin (12)
↓
bin (6)
↓
bin (3)
↓
bin (2)
↓
bin (1)
↓
bin (0)



0|0

11001

Q Create a Recursive function that take a Parameter and return Sum of Natural No upto ~~A~~ given no.

```
class calc
{
        Static int Sum (int a)
        {
              if (a==1)
              return 1;

              else
              return a + Sum (a-1);
        }

}
class T
{      PS  v m  (String k[])
       {
            int r= calc.Sum(5);
            SOPL (r);
       }
}
```

Ques 3 create a Recursive method that tako No and Power ~~and~~ aas a Parameter and return result (N^P)

```
a=5
5 + Sum(4)
5 + 4 + Sum(3)
5 + 4 + 3 + Sum (2)
5 + 4 + 3 + 2 + Sum(1)
5 + 4 + 3 + 2 + 1
```

5, 4

```
class Calc
{
    static int ~~Sum~~ Power (int N, int P)
    {
        if (P==0)
            return 1;
        else
            return N * (N , P-1);
    }
class J
{
    P S V M (String K[])
    {
        int r= Calc.~~Sum~~ Power (5,4);
        SOPL (r);

    }
}
```

```
5 * (5 , 3)
5 * 5 * (5,2)
5 * 5 * 5 (5,1)
5 * 5 * 5 * 5
```

Q Create a Recursive method that take a Parameter and return reverse of given No;

```
class calc
{ int s=0
    static int ~~main~~ reverse (int a)
    {
        if(a <10)
            return a;
        else
            return a%10

        if (a>0)
        {
            s = s*10 + a%10;
            reverse (a/10);
        }
        else
        {
            return s;
        }
    }
}
```

a s

~~123~~

12

s

03

Q7 ~~Sum of digits~~ create a Recursive method that ~~th~~ take a Parameter and return Sum of digit.

```
class calc
{    static int Sum(int a)
     {
          if (a< 10)
             return a;
          else
             return a%10 + Sum(a/10);
     }
}
```

s=0

3

a=123

rev$(\overset{a}{12})$

s = 2

a=12

a = 1 2 3

3 + Sum(12)

3 + 2 + Sum(1)

3+2+1

=> 6

S = 0

S = 0

a > 0

a = 123/10

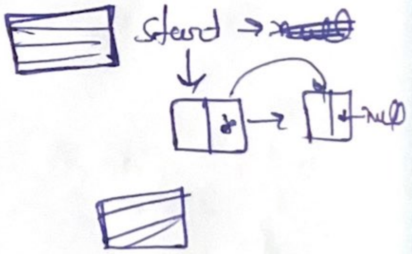
12

r = a%10

Sum=sn + r

Q Create a Recursive method that display Singly Linked List Values.

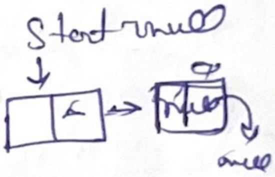
```
class ~~node~~ calc
{
    static void disp (node start)
    {
        if (start . ~~next~~ ==null)
            return ;
        else
        ~~return~~
      { SOPL (start.data);
        disp (start. next);
      }

    }
}
```



```
if ( start != null)
{
    SOPL ( start. data)";
  disp (start .New);
}
```

Q Create a Recursive method that display values of Linked List in reverse

```
class calc
{
        static void disp (node start)
        {       if (start== null)
                  return;
              else
                { disp (start. next)
                   SOPL (start. data);
                }

        }

}
```



Q Create a Recursive method that display reverse of given String.

```
class calc
{
        static void revstring (String a)
        {   ~~String~~

    if (a.length() <=1 )
      { sopL (a.charAt (0));

    }

    else
    {

    reversestring (a.substring (0, a.length-1))

      sop (a.charAt (a.length -1) );

    }

}
```

" Hello"

Hello
Hell
R0 1 2

a = Hello (4)

revst $\overset{a}{(Hello)}$

revst (O

revst (H)

calc. revString ("RAHUL");

↓

revString ("RAHU")

5

0 ↓

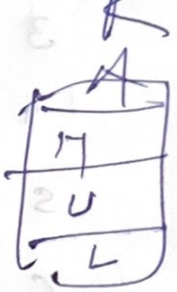
revs//("RAH"

↓

r("RA'

r(

("R"



(~~RAHUL~~)

RAHUL
0 1 2 3 4

L

5-4 ● O

$\boxed{0,1,2,3,4}$

$\boxed{0,4}$