class 1

# Recursion

Topic

1. what is Recursion
2. Example of Recursion
3. Tracing Recursion
4. stack used in Recursion
5. Time complexity
6. Recurence Relation

Function:

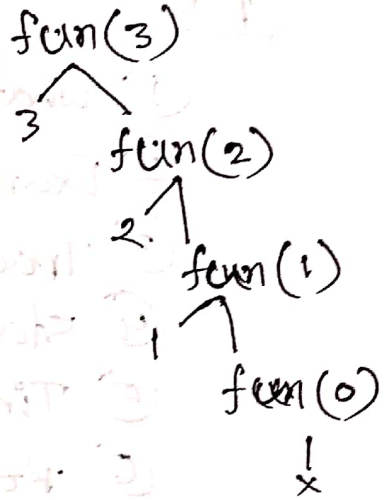
```
void fun (int n)
{
    1. ---
    2. ----
    3. ----
}

int main ()
{
    1. ------
    2. ----
    3  fun (n);
    4. ----
    5. ----
}
```

Recursion

```
Type fun (para..)
{
    if (base case)
    {
        1. ------
        2. fun (para);
        3. ----
    }
}
```

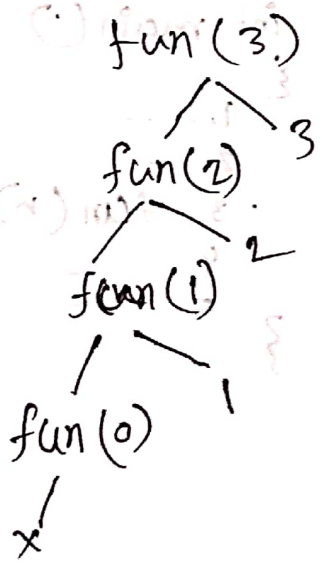
```
void fun(int n)
{
    if(n>0)
    {
        print(n);
        fun(n-1);
    }
}
```




```
int main( )
{
    int n=3;
    fun(n);
}
```

O/P: 3, 2, 1

```
void fun(int n)
{
    if(n>0)
    {
        fun(n-1)
        print(n);
    }
}
```




```
int main()
{ int n=3;
  fun(n);
}
```

O/P = 1, 2, 3

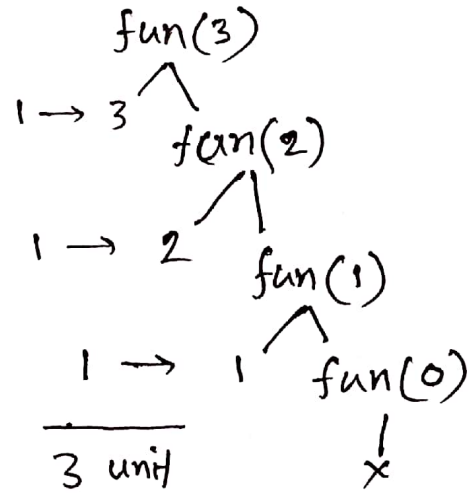
class 2 — Recursion

```
void fun(int n)
{
    if(n>0)
    {
        1. calling time          <-- Assending
        2. fun(n-1) * 2;
        3. returning time ↑      <-- Descending
    }
}
```

Class-4 : Every statement take 1 unit of time.

```
void fun(int n)
{
    if(n>0)
    {
        print(n);      --> 1
        fun(n-1);
    }
}

int main()
{
    int n=3;
    fun(n);
}
```




[value of n  O(n)
n unit

[put book in table (n)
book's take (n) unit of tim

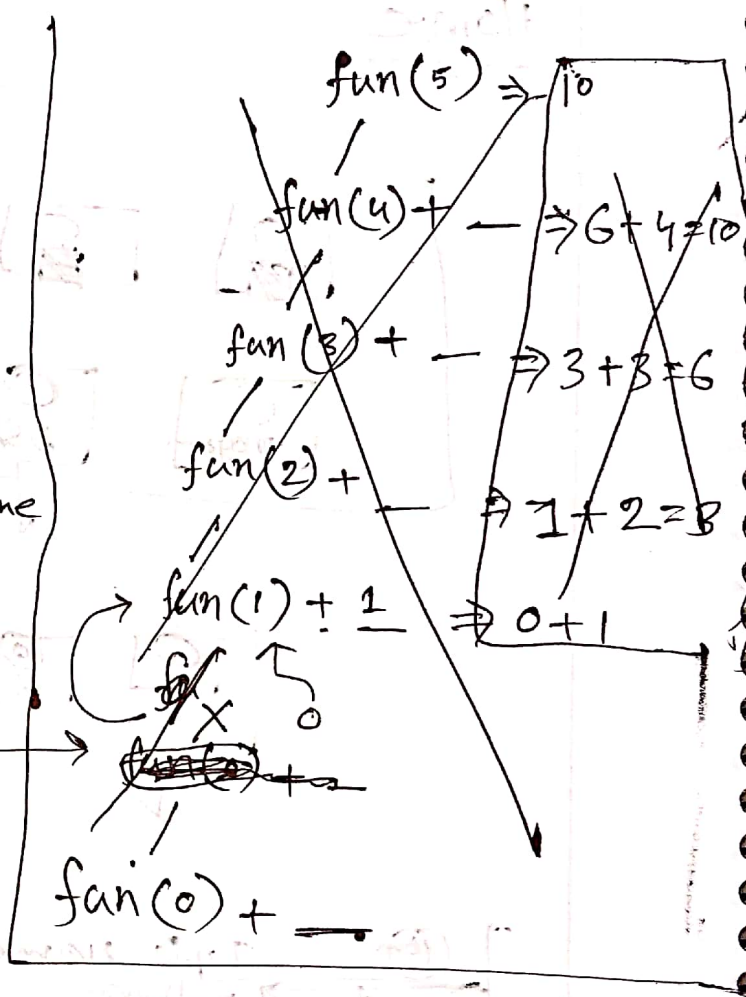
# static variable in Recursion

class-8:

```
int fun(int n)
{
    if(n>0)
    {
     return fun(n-1)+n;
    }
    return 0;
}

int main()
{ int a=5;
   fun(a);
}
```

↑ it is work in returning time




n=5
n=4
n=3
n=2
n=1

fun(5) ⇒ 15
fun(4) + 5 ⇒ 10+5=15
fun(3) + 4 ⇒ 6+4=10
fun(2) + 3 ⇒ 3+3=6
fun(1) + 2 ⇒ 1+2=3
fun(0) + 1 ⇒ 0+1

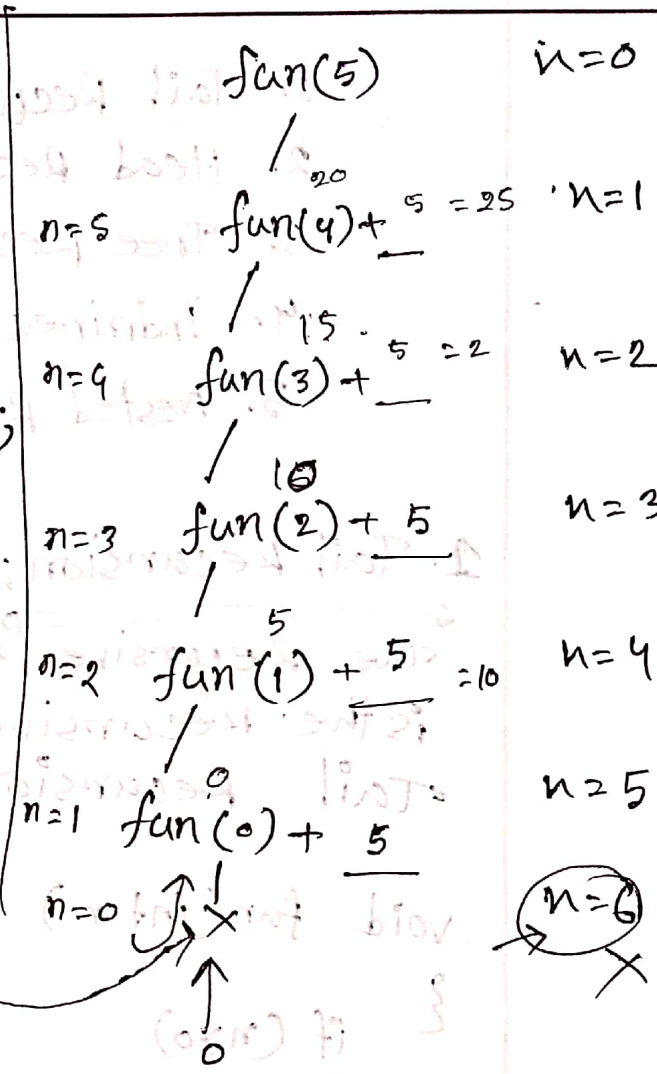
# static ⇒ like global variable

```
int fun(int n)
{
  static int x=0;

   if(n>0)
   { x++;
     return fun(n-1)+x;

   }

   return 0;
}
int main()
{
  int a=5;
  fun(a);
}
```




Always (x = 5)

coz x here like a global variable

Class - 8

# Types of Recursion

1. Tail Recursion
2. Head Recursion
3. Tree Recursion
4. Indirect Recursion
5. Nested Recursion

## 1. Tail Recursion:

The Recursive function the last statement is the Recursive call Then it call the Tail Recursion.

```
void fun(int n)
{
    if (n>0)
    {
        print(n);
        fun(n-1);
    }
}
```

fun(3)

convert loop

```
void fun(int n)
{
    while(n>0)
    {
        print(n)
        n--;
    }
}
```

fun(3)

Time → O(n)

Space O(n) ——— O(1)

class-9

## 2. Head Recursion

```
void fun(int n)
{
  if(n>0)
  {
    fun(n-1);
    ===
  }
}
fun(3)
```

```
void fun(int n)
{
  if(n>0)
  {
    fun(n-1)
    printf(n);
  }
}
```

fun(3)

O/P: 1 2 3

### loop

Head to loop conversion is not easy.

```
void fun(int n)
{
  int i=1;
  while(i<=n)
  {
    print(n);
    i++;
  }
}
```

fun(3)

# Tree Recursion

## Linear Recursion

```
fun(int n)
{
    if(n>0)
    {
        ===
        fun(n-1)
        ===
    }
}
```

call in one time

## Tree Recursion

```
fun(int n)
{
    if (n>0)
    {
        ---
        fun(n-1)
        ===
        fun(n-1)
        ===
    }
}
```

call in two time

## Tree Recursion

```
void fun(int n)
{
    if (n>0)
    {
        printf(n);
        fun(n-1);
        fun(n-1);
    }
}
```

# Tree Recursion

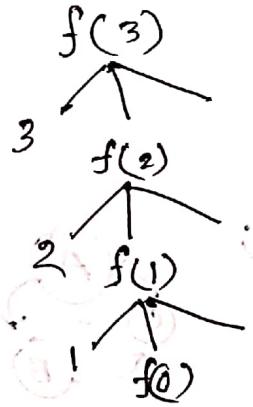
**tracing tree**

```
void fun(int n)
{
  if(n>0)
  {
    print(n);
    fun(n-1);
    fun(n-1);
  }
}
```

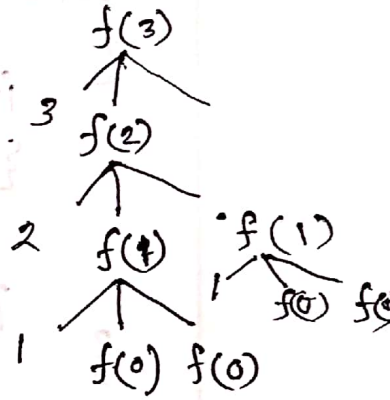
---

fun(3).

First call




Fist with second call



# Indirect type of Recursion

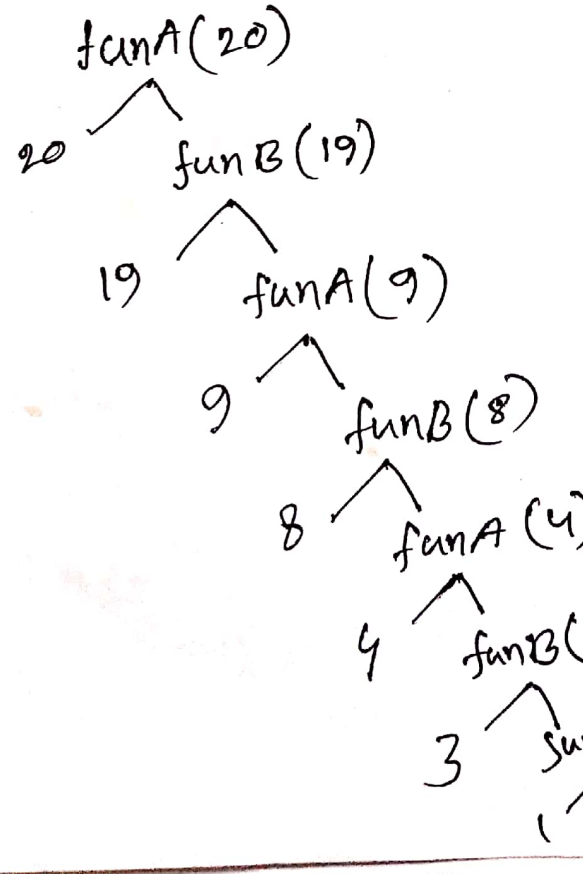
circular fasion

```
void A(int n)
{
  if(↔)
  {
    B(n-1);
  }
}
```

```
void B(int n)
{
  if(↔)
  {
    A(n-3)
  }
}
```


```
void funA(int n)
{
  if(n>0)
  {
    printf("%d", n);
    funB(n-1);
  }
}
```

```
void funB(int n)
{
  if(n>1)
  {
    printf("%d", n);
    funB(n/2);
  }
}
```



class-14 Nested Recursion

```
void fun(int n)
{
  if(---)
  {
    ---
    fun(fun(n-1));
  }
}
```

Recursion inside the Recursion when inside Recursion value obtain then outer Recursion call made.

```
int fun(int n)
{
  if(n>100)
    return n-10;
  else
    return fun(fun(n+11));
}

fun(95);
```

fun(95)
|
fun(fun(95+11))
|
fun(fun(95

10

Class - 16

# Sum of First 'n' Natural Number

1 2 3 4 5 6 7 ..

1+2+3- - - - +n

sum(n) = 1+2+3+ - - - - +(n-1)+n

sum(n) = sum(n-1) + n

$$sum(n) = \begin{cases} 0 & n=0 \\ sum(n-1)+n & n>0 \end{cases}$$

```
int sum(int n)
{
    if (n==0)
    { return 0;
    }
    else
      return sum(n-1) +n;
}
```

$\frac{n(n+1)}{2}$

O(n)

space O(n)

sum(5) = 15

sum(4) + 5 = 10 + 5 = 15

sum(3) + 4 = 6 + 4 = 10

sum(2) + 3 = 3 + 3 = 6

sum(1) + 2 = 1 + 2 = 3

sum(0) + 1 = 0 + 1

x = 0

class-18 Factorial of a Number

$n! = 1 * 2 * 3 * \dots * n$

$5! = 1 * 2 * 3 * 4 * 5 = 120$

$0! = 1$

$1! = 1$

$fact(n) = 1 * 2 * 3 * \dots * (n-1) * n$

$fact(n) = fact(n-1) * n$

$$fact(n) = \begin{cases} 1 & n = 0 \\ fact(n-1) * n & n > 0 \end{cases}$$

```
int fact (int n)
{
  if (n==0)
    return 1;
  else
    return fact(n-1) * n;
}
```

O(n)

space (n)

fact (5) ⇒ 120
|
fact (4) * 5 ⇒ 24 * 5
|
fact (3) * 4 ⇒ 6 * 4
|
fact (2) * 3 ⇒ 2 * 3
|
fact (1) * 2 ⇒ 1 * 2
|
fact (0) * 1 ⇒ 1 * 1
|↑
1

Q7 ⊛

class 20

# Exponent $(m)^n$ power

$2^5 = 2 * 2 * 2 * 2 * 2$

$m^n = m * m * m * \ldots$ for $n$ times

$pow(m, n) = m * m * m \ldots *(n-1)$ times $* m$

$pow(m, n) = pow(m, n-1) * m$

$$pow(m,n) = \begin{cases} 1 & n=0 \\ pow(m, n-1) * m & n>0 \end{cases}$$

```
int pow(int m, int n)
{
  if(n==0)
    return 1;
  return pow(m,n-1)*m;
}
```

$2^8 = (2^2)^4$

$= (2*2)^4$

$2^9 = 2*(2^2)^4$

power reduce by half

pow(2,9)
|
pow(2,8) * 2
|
pow(2,7) * 2
|
pow(2,6) * 2
|
pow(2,5) * 2
|
pow(2,4) * 2
|
pow(2,3) * 2
|
pow(2,2) * 2
|
pow(2,1) * 2
pow(2,0) * 2

```
int pow (int m, int n)
{
    if (n==0)
        return 1;
    if(n%2==0)
        return pow(m*m, n/2);
    else
        return m* pow (m*m, (n-1)/2);
}
```

pow(2,9) ⇒ $2^9$

↓

2 * pow($2^2$, 4); ⇒ 2 * $2^8$

↓

pow($2^2$ * $2^2$, 4) $2^2 * 2^2 \Rightarrow 2^8$

↓

pow($2^8$, 1) ⇒ $2^8$

↓

$2^8$ * pow($2^{16}$, 0) ⇒ $2^8 * 1$

↓

1

# Taylor series $e^x$

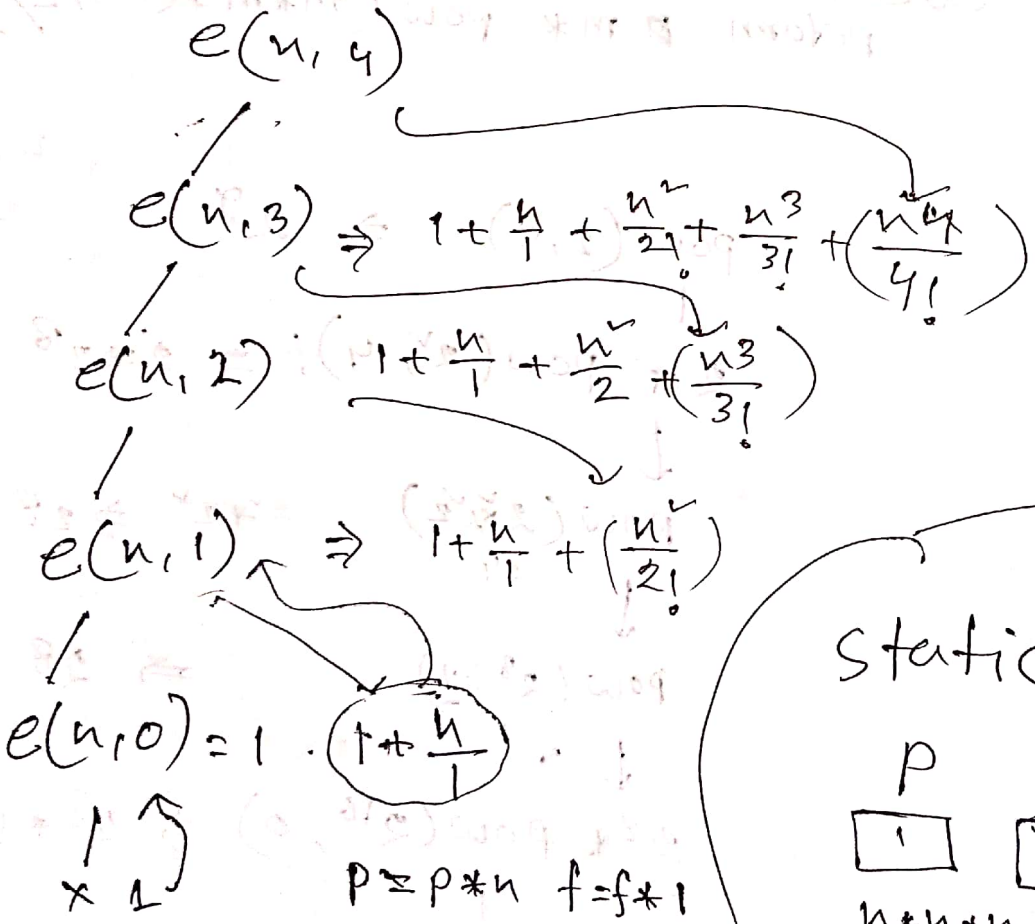
Class 22

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + n \text{ terms}$$

sum(n-1)*n ⇒ sum(n) = 1+2+3+---+n

fact(n-1)*n ⇒ fact(n) = 1*2*3*4*---*n

pow(x,n-1)*x ⇒ pow(x,n) = x*x*x*x---n times




Static variable

p f

1 1

x*x*x 1*2*3*4

```
int e(int n, int n)
{
    static int p=1, f=1;
    int r;
    if (n==0)
        return 1;
    else
    {
    r=e(n, n-1);
        p=p*n;
        f=f*n;
        return r+p/f;
    }
}
```

class-24

# Taylor series $e^x$

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + n \text{ terms}$$

mull $= 0 \quad 0 \quad \frac{x*x}{1*2} \quad \frac{x*x*x}{1*2*3} \quad \frac{3}{3} \quad \cdots$

$\frac{}{2} \quad \frac{}{3} \quad 6 \quad 8$

$$2\left[1+2+3+4-\right]$$

$$= 2\,\frac{n(n+1)}{2}$$

$$= n(n+1) \Rightarrow O(n^2)$$

multi

$$\Rightarrow 1 + \frac{x}{1} + \frac{x^2}{1*2} + \frac{x^3}{1*2*3} + \frac{x^4}{1*2*3*4}$$

$$\Rightarrow 1 + \frac{x}{1}\left[1 + \frac{x}{2} + \frac{x^2}{2*3} + \frac{x^3}{2*3*4}\right]$$

$$\Rightarrow 1 + \frac{x}{1}\cdot\left[1 + \frac{x}{2}\left[1 + \frac{x}{3} + \frac{x^2}{3*4}\right]\right]$$

$$\Rightarrow 1 + \frac{x}{1}\left[1 + \frac{x}{2}\left[1 + \frac{x}{3}\left[1 + \frac{x}{4}\right]\right]\right]$$

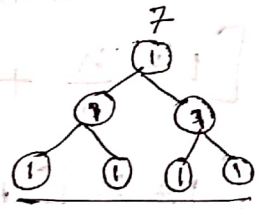
↑ 1 ↑ 2 ↑ 3 → mult

$O(n)$

nlogn

```
int e(int x, int n)
{  int s=1;
 for(;n>0;n--)
{ s=1+x/n*s;
}
 return s;
}
```

```
int e(int x, int n)
{  static int s=1;
   if(n==0)
     return s;
   s=1+x/n*s;
   return e(x,n-1);
}
```




100
750 * 50
=37500
500
75,000

700 * 200
140,000

2 3 4 5 6
1 1 2 2,3

2 3 4 5 6 7 8 9 10 11 12
1 2 2 3 3 4 4 4 5 5 5

13 14 15 16 17 18
5 5 5 5 5 6

# Fibonacci series

Class -27

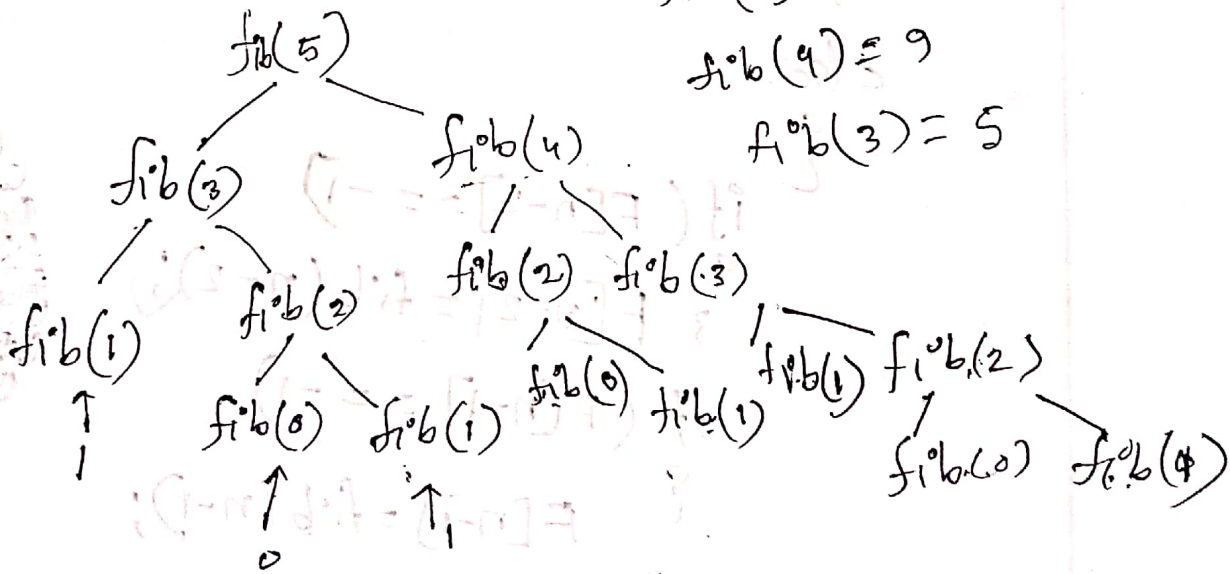
| fib(n) | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 |
|---|---|---|---|---|---|---|---|---|
| n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

$$fib(n) = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ fib(n-2) + fib(n-1) & n>1 \end{cases}$$

```
int fib (int n)
{
    if (n <=1)
        return n;
    return fib(n-2)+fib(n-1);
}
```

2 fib(n)
$\Rightarrow O(2^n)$

fib(5) = 15 call
fib(4) = 9
fib(3) = 5

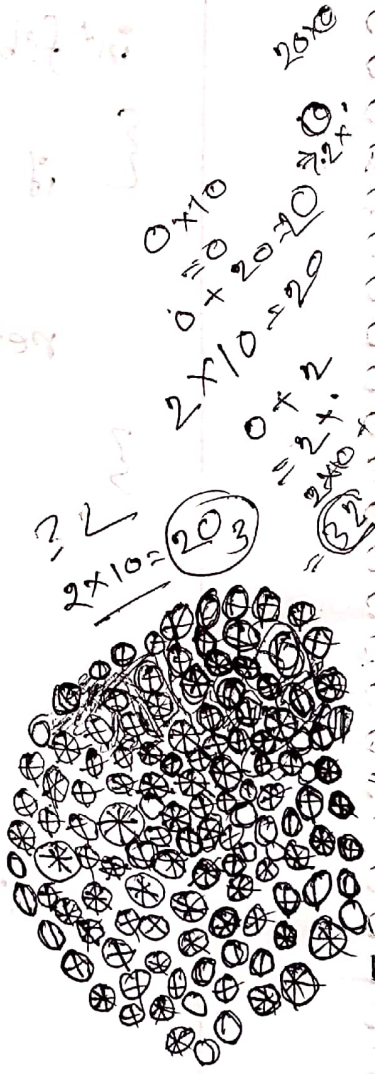


if the Recursion function ~~help the~~ call the same value many times then it called the Excursive Recursion

Using Memosization

```
int F[10];

int fib (int n)
{
  if (n <= 1)
  {
    F[n] = n;
    return n;
  }
  else
  {
    if (F[n-2] == -1)
      F[n-2] = fib(n-2);
    if (F[n-1] == -1)
    {
      F[n-1] = fib(n-1);
    }
    return F[n-2] + F[n-1];
  }
}
```



# combination Formula

Class-29

$$^nC_r = \frac{n!}{r!(n-r)!}$$

selection

$^5C_r \Rightarrow {}^5C_0\ {}^5C_1 \cdots {}^5C_5$

0-5

A B C D E F G

select 3

A B C
A B D
A C B

01789116343

hossainsabbir247@gmail.com

```
int C (int n, int r)
{
    int f1, f2, f3;
    f1 = fact(n);
    f2 = fact(r);
    f3 = fact(n-r);
    return f1/(f2*f3);
}
```

O(n)

sabh1

~~sabh123~~

sabb4321

sabb4321

# Combination Formula

$$^nC_r = \frac{n!}{r!\,(n-r)!}$$

```
int C(int n, int r)
{
    if (r == 0 || n == r)
        return 1;
    else
        return C(n-1, r-1)
             + C(n-1, r);
}
```

## Pascale triangle