Recursion

function calling itself is called Recursion.

```
int main()
  {
     printf(" Hello");
   return 0;
  }
```

```
int  fun( )
{
   ______
   ______
   ______
   fun(); ⟶ function call
 }
```

factorial :-

$\underline{|5} = 5 \times 4 \times 3 \times 2 \times 1$

$\underline{|4} = 4 \times 3 \times 2 \times 1$

$$\begin{tikzcd} \underline{|5} = 5 \times \underline{|4} \arrow[d] & & & \\ 4 \times \underline{|3} \arrow[d] & & & \\ 3 \times \underline{|2} \arrow[d] & & & \\ 2 \times \underline{|1} \arrow[d] & & & \\ \underline{1} & & & \end{tikzcd}$$

$$\lfloor 5 = 5\times 4\times 3\times 2\times 1$$
$$= 1\times 2\times 3\times 4\times 5$$

```
int  fact(int n)
  {
→    int  f = 1;
→    for( int  i = 2; i<=n; i++)
     {
          f = f*i;
              24
     }
→    return f;
  }
```

$\underline{\underline{O(n)}}$

```
int  main()
{
   int x = fact(4); 24

   cout<<x;

   return 0;
}
```

fact(4)

$\lfloor 4 = \underline{\underline{24}}$

n	f	i
4	~~1~~	~~2~~
	~~2~~	~~3~~
	~~6~~	4
	$\bigcirc$24	

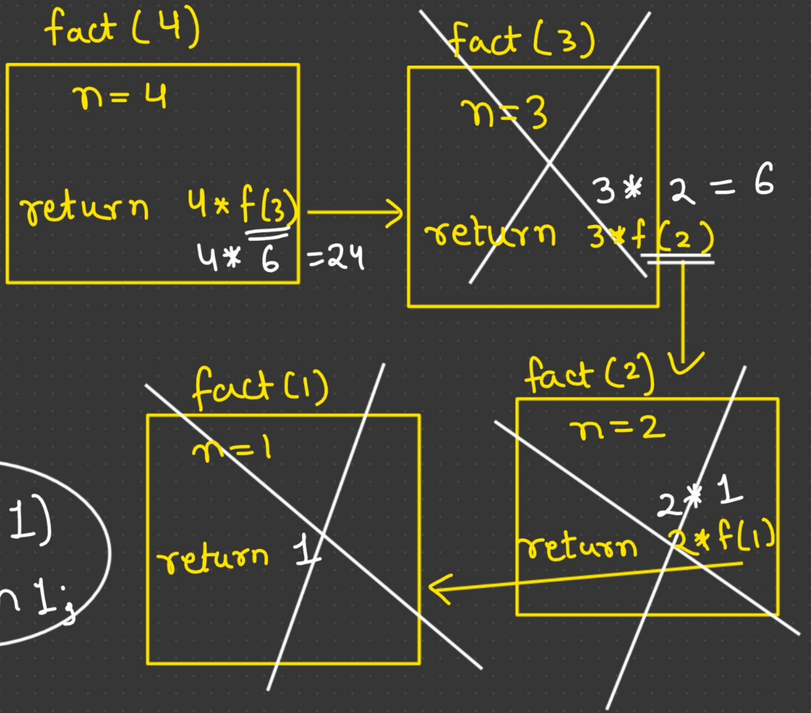
```
fact (4)
{
        4 x fact(3)
}
```

$\underline{4} = 4 \times \underline{3}$

```
int fact (n)
  {
    return n * fact (n-1);
  }
```

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

$O(n)$



Base condition

fact (1) = 1

fact (0) = 1

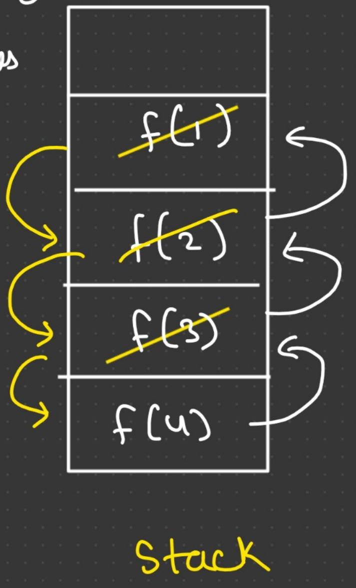
```
if(n==1)
  return 1;
```

```
int fun( parameter)
{
    Base condition

  return  -- fun( parameter-)
}
```

1) more memory

2) recursion takes more time



Application Recursion:-

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

Fibonacci Series

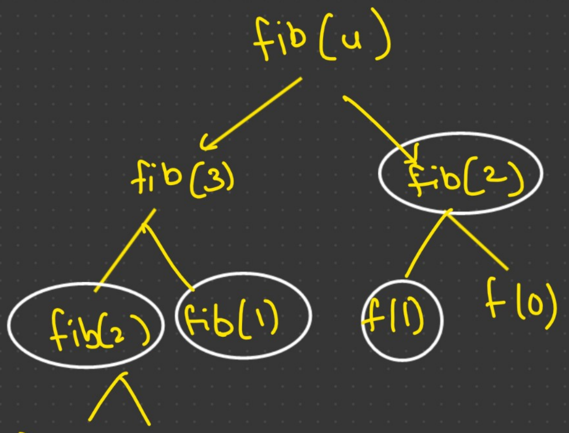
$$\left.\begin{array}{l} n = 0 \Rightarrow \text{return } 0 \\ n = 1 \Rightarrow \text{return } 1 \end{array}\right\}$$

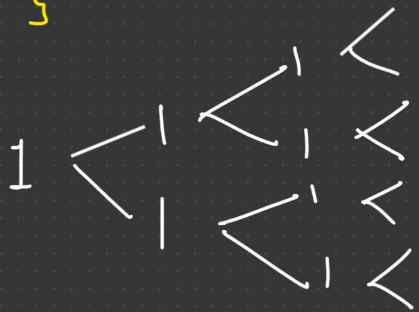
$$F(n) = F(n-1) + f(n-2)$$

0, 1, 1, 2, 3, 5

$$\begin{tikzcd}
F(5) = & \cancel{F(4)} \arrow[dl, "3"] & + & \cancel{F(3)}\ 2 \arrow[d] \\
2\ \cancel{F(3)} + \arrow[d] & \cancel{F(2)}\ 1 \arrow[d] & & \overset{1}{\cancel{F(2)}} + \overset{1}{\cancel{F(1)}} \arrow[d] \\
1\ \cancel{F(2)} + \underset{1}{\cancel{F(1)}} \arrow[d] & \underset{1 + 0}{\cancel{F(1)} + \cancel{F(0)}} & & \underset{1 + 0}{\cancel{F(1)} + \cancel{F(0)}} \\
\underset{1 + 0}{\cancel{F(1)} + \cancel{F(0)}} & & &
\end{tikzcd}$$

```
int fib( int n)
{
    if (n==0)
        return 0;
    if (n== 1)
        return 1;
    return fib(n-1) + fib(n-2);
}
```



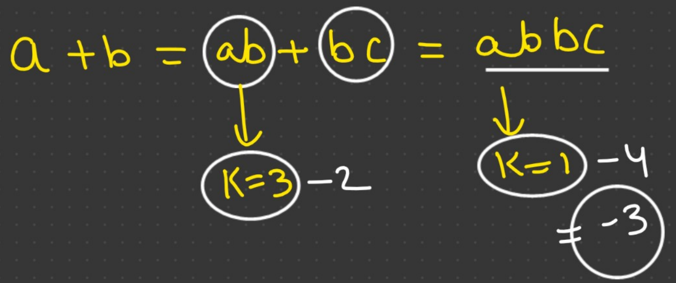


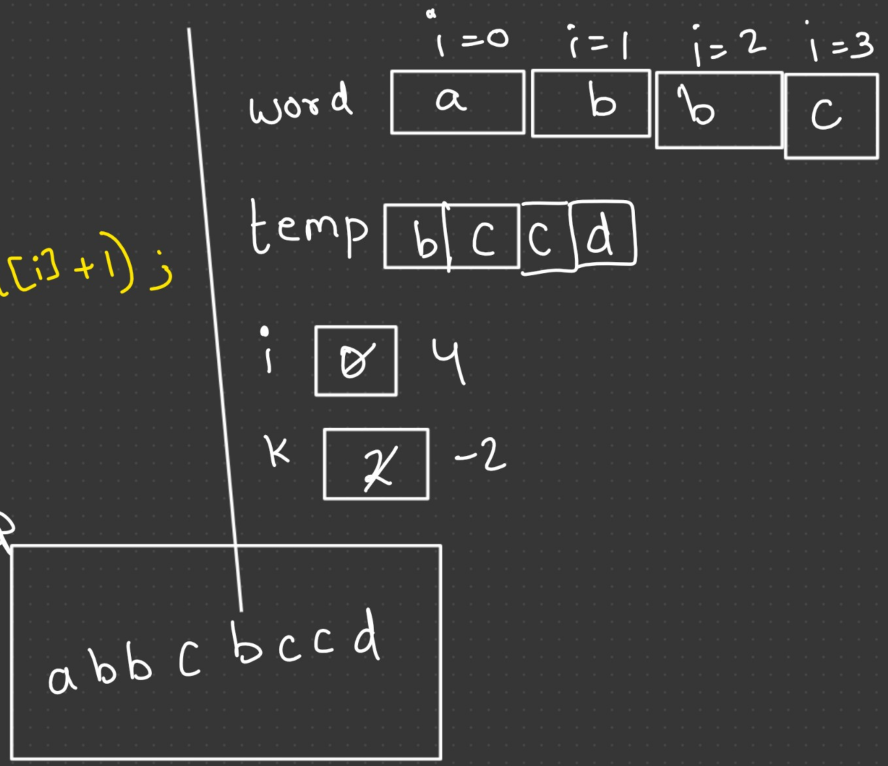
$2^0, 2^1, 2^2, 2^3, 2^4, \ldots 2^n$

$O(2^n)$

```
getword (int k)
{
    if ( K <=1)
        return;

    else
    {   int i=0;     string temp;
        while ( word[i] != '\0')
        {
            temp.append( 1, word[i]+1) ;
               i++; k--;
        }
        ~~word.append(temp);~~
    }      word= word + temp
```


a + b = ab + bc = abbc
K=3 -2
K=1 -4
≠ -3


i=0 i=1 i=2 i=3
word a b b c
temp b c c d
i 0 4
k 2 -2
abbcbccd