Recursion-1

$n! = n * \underbrace{(n-1) * (n-2) * \ldots 1}$

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

→ Count zero (ache question hai)

Factorial

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

int main() {
    cout << fact(5);
}
```

n
5
4
3
2

- small output

fact(4) 24 → 120 waiting

fact(3) 6 → 24

fact(2) 2 → 6

fact(1) 1 → 2

120

· Power

4,0 → 1

eg. 3 4

3 * 3 * 3 * 3 = 81

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

x	n
3	4
3	3
3	2
3	1

81

x * pow(3,3) → 27

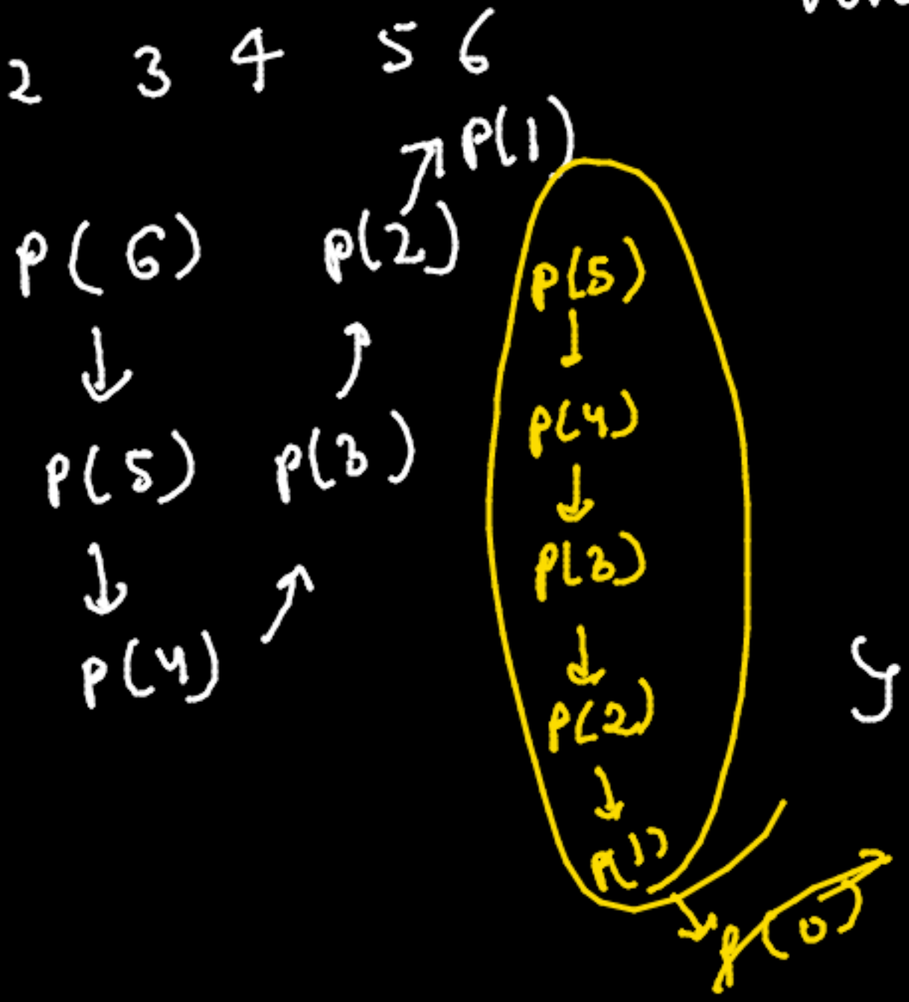
3 * pow(3,2) → 9

x * pow(3,1) → x

Print Numbed

eg. n = 6

o/p : 1 2 3 4 5 6



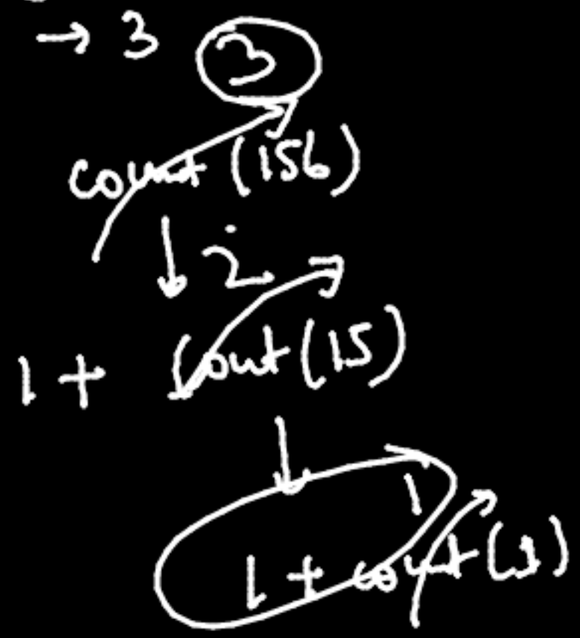

```
void print (n) {
    if (n == 0)
        return;
    print (n-1);
    cout << n << " ";
}
```

Number of digits

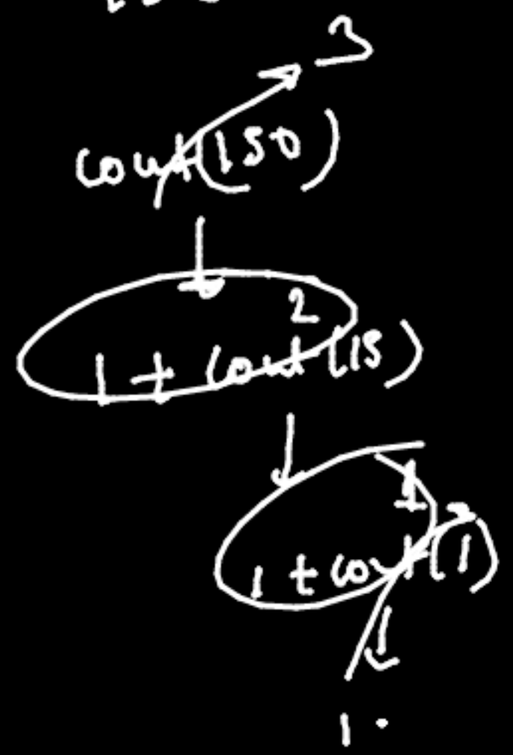
```
int count (int n) {
    if (n <= 9)
        return 1;
    return 1 + count (n/10);
}
```

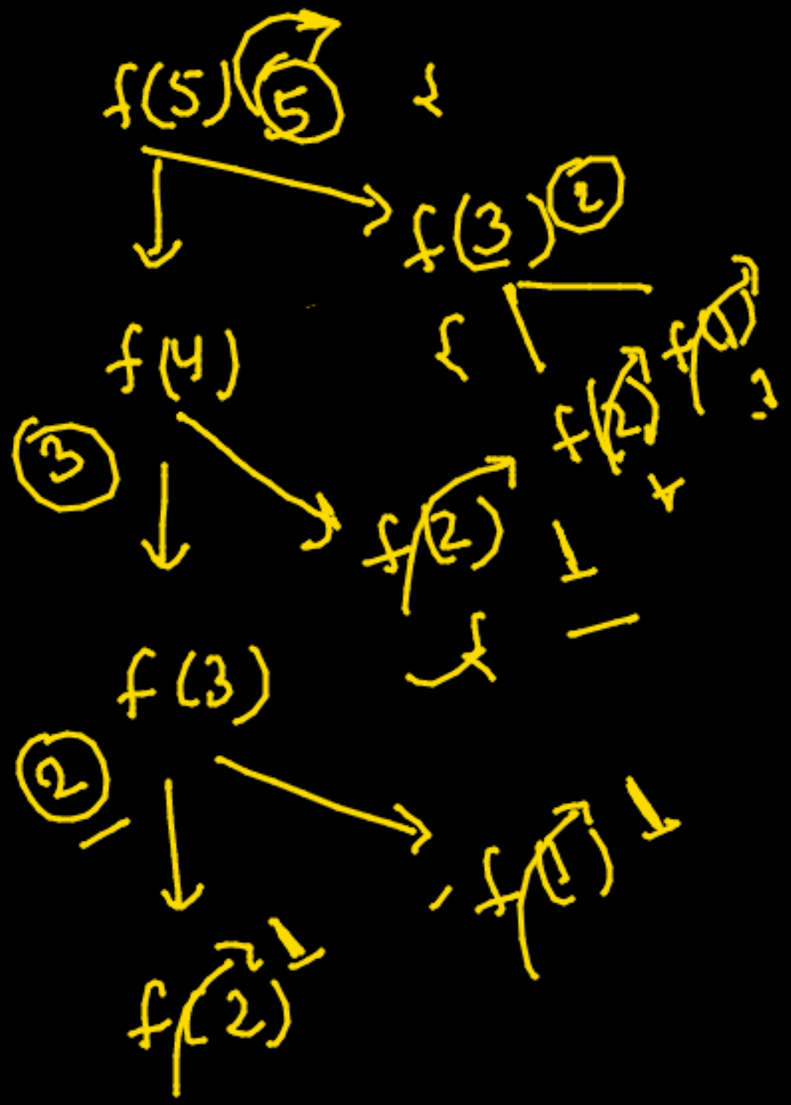
eg. 156

→ 3



150





Fibonacci Number

1 1 2 3 5 8 13 21 34 55

fibo (n) = fibo(n-1) + fibo(n-2)

```
int fibo (int n){
    if (n==1 || n==2)
        return 1;
    int smalloutput1 = fibo(n-1)
    int smalloutput2 = fibo(n-2)
    return smalloutput1 + smalloutput2;
}
```

SUM OF ARRAY

Assuming all size in net 0

I/P

3	1	2
10	5	2

17

o/p → 17

n = 3

2 + sum(arr, 2) → 18

5 + sum(arr, 1) → 10

10 + arr[0]; → 10

```
int sum(int arr[], int n){
    if (n==0)
        return 0;

    return arr[n-1] + sum(arr,n-1);
}
```

```
int sum(int arr[], int n){
    return helper (arr,n-1);
}
int helper (int arr[], int n)
    if (n==0) return arr[0]

    return arr[n] + helper (arr,n-1);
}
```

Check Number.

present 9 8 10

9

→ true

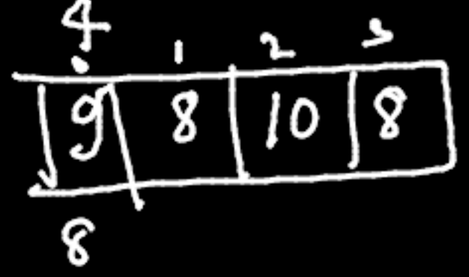
Not present 9 8 10

is

→ false

```
bool check (int arr[], int n, int x){
    return helper (arr, n-1, x);
}
bool helper (int arr[], int n, int x) {
    if ( n == -1) return false;
    bool smallOutput = (arr[n] == x);
    return smallOutput || helper(arr, n-1, x);
}
```

First Index of Number.



→ Index 1;


```
int firstIndex(int arr[], int size, int x){
    return helper(arr, 0, size, x);
}
int helper(int arr[], int i, int n, int x){
        if(i > n) return -1;
        if(arr[i] == x) return i;
        return helper(arr, i+1, n, x);
}
```

Last Index of Number

size = 4

x = 8

We are traversing from end and trying to find target, if found return it else, decrease the size and do it again using recursion.

```
int last (int arr[], int size, int x)
{
    if (size < 0)
        return -1;

    if (arr[size-1] == x)
        return (size-1);

    return last (arr, size-1, x);
}
```

Multiplication

m n
3 4
→ 15

```
int mul(int m, int n) {
    if (m==0 || n==0) return 0;
    if (n==1)
        return m;
    return m + mul(m, n-1);
}
```

3 + mul(3,4)

3 + mul(3,3)

3 + mul(3,2)

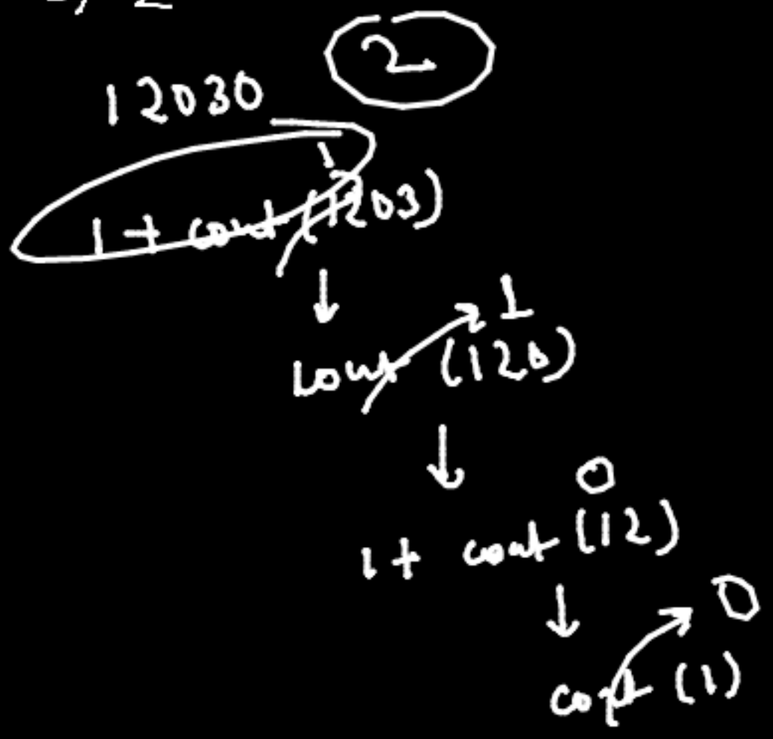
3 + mul(3,1)

Count Zero

0 → 1

60120 → 1

12030 → 2




```
int count (int n) {
    if (n<10) {
        if (n==0)
            return;
        return 0;
    }
    if (n%10 ==0)
        return 1 + count (n/10);
    return count (n/10);
}
```

Geometric Sum

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots \frac{1}{2^k}$$

$k = 3$

$\rightarrow 1.875000$

```
int fun (int k)
    if (k == 0)
        return 1;
    return (1/pow(2,k)) + fun (k-1);
}
```