CS & IT ENGINEERING

C Programming

Function & Storage Classes

Lec- 05

By- Pankaj Sharma Sir


TOPICS TO
BE
COVERED
Recursion 01

n>0

i/p : 2

o/p : PankajPankaj

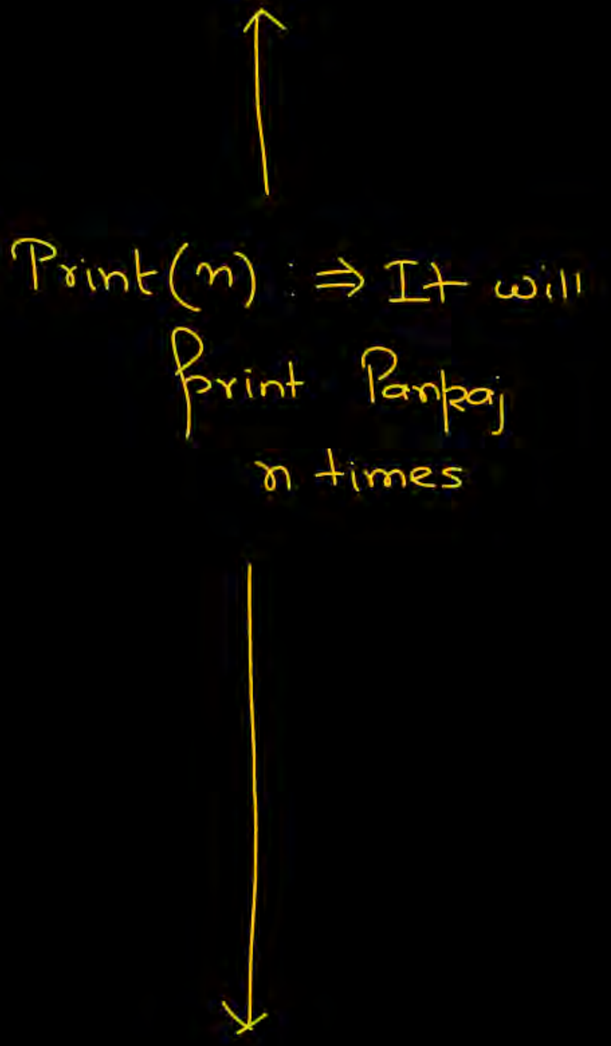
i/p : 5

o/p : PankajPankajPankajPankajPankaj

void Print(int n)
{

Print(n) :⇒ It will
print Pankaj
n times

}




```
void Print(int n)
{
```

what is that small value of n ?

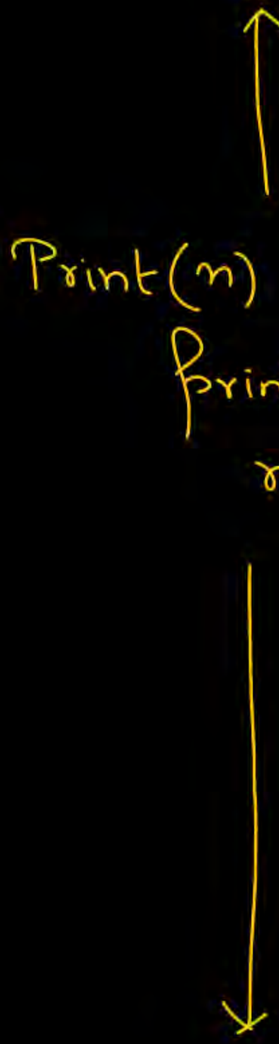
```
if(n is small){
```

* Easy case
* No recursion
* Can be answered directly

```
}
else {
```

* input is large
* Hard case
* Can not be answered directly
* Recursion is needed.

```
}
}
```



Print(n) : ⇒ It will print Pankaj n times

```
void Print(int n)
{
if( n==1 ){
    printf("Pankaj");
}
else {
    * input is large
    * Hard case
    * can not be answered directly
    * Recursion is needed.
}
}
```

```
void Print(int n)
{
if( n==1 ){
printf("Pankaj");
}
else {
}
}
```

Print(n) : ⇒ It will print Pankaj n times

Every recursive call perform small unit of task and remaining is done by recursion.

- input is large
- Hard case
- Can not be answered directly
- Recursion is needed.

```
void Print(int n)
{
if( n==1 ){
        printf("Pankaj");
}
else{
        printf("Pankaj");
← Print Pankaj n-1 times →
}
}
```

```
void Print(int n)
{
if( n==1 ){
printf("Pankaj");
}
else{
printf("Pankaj");
Print(n-1);
}
}
```

$n > 0$

i/p : 125

o/p : 8

i/p : 1269

o/p : 18

i/p : 7

o/p : 7

i/p : 541

o/p : 10

i/p : 3

o/p : 3

i/p : 9

o/p : 9

```
int sum_digits(int n)
{
    if (n is small)
    {
    }
    else {
    }
}
```

→ single digit

$n > 0$

i/p : 125

o/p : 8

i/p : 1269

o/p : 18

i/p : 7

o/p : 7

i/p : 541

o/p : 10

i/p : 3

o/p : 3

i/p : 9

o/p : 9

```
int sum_digits(int n)
{
    if( n>0 && n<=9)
    {
        return n;
    }
    else {
```

3

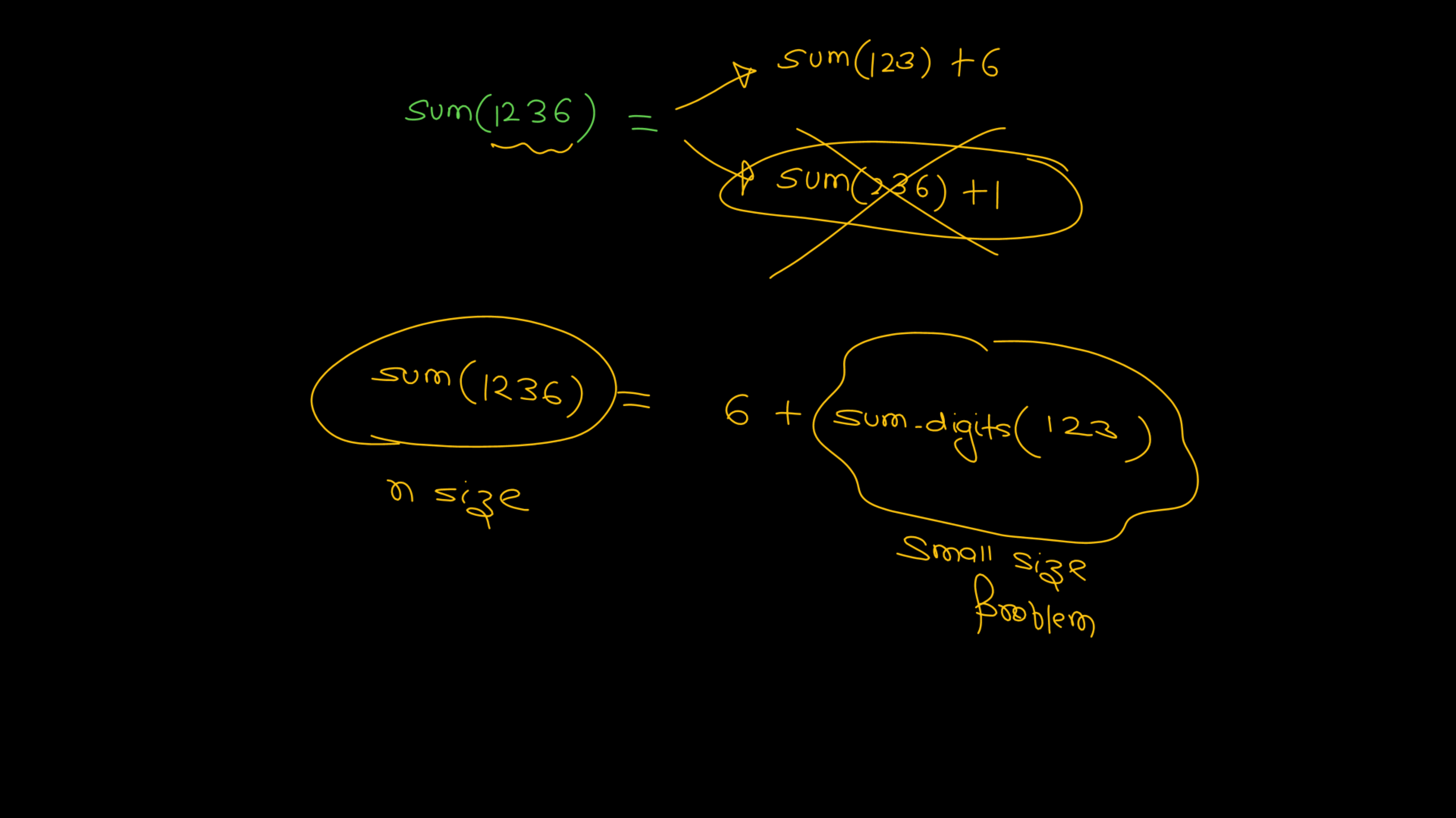
}

sum_digits(125) =

Problem
n size

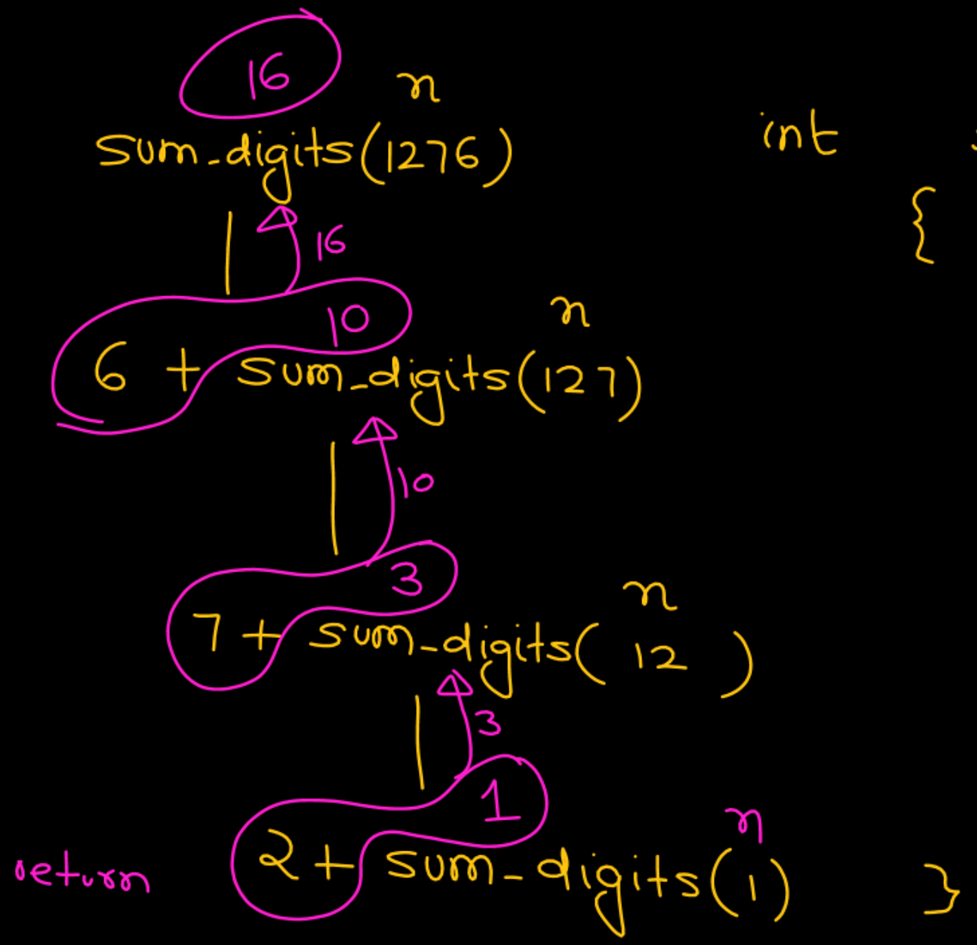
Always assume that you know the answer of small size problem.

$\{ \frac{n}{2}, \frac{n}{3}, n-1, n-2 \}$


sum(1236) =
sum(123) +6
sum(236) +1
sum(1236)
n size
6 + sum_digits(123)
Small size
problem

```
void main(){
    int s;
    s = sum_digits(1276);
    printf("%d",s);
    }
```

```
int sum_digits(int n)
{
    if (n>0 && n<=9)
        return n;
    else{
    return n%10 + sum_digits(n/10);
    }
}
```




```
int sum_digits(int n)
{
    if (n>0 && n<=9)
        return n;
    else{
    return n%10 + sum_digits(n/10);
    }
}
```

Ex.3

a^b , $a>0$, $b\geq 0$

3^{100}

$3 \times 3 \times 3 \times 3 \ldots$

```
int Power(int a, int b)
{
    if(b is small)
    {

    else {

    }

    }
```

Ex.3

a^b , $a>0$, $b\geq 0$

3^{100}

$3 \times 3 \times 3 \times 3 \ldots$

$3^{100} = 3 \times 3^{99}$

```
int Power(int a, int b)
{
    if( b == 0 )
    {
        return 1;
    }
    else {
        // rec. call + small work
    }
}
```

Ex.3

a^b , $a>0$, $b\geq 0$

3^{100}

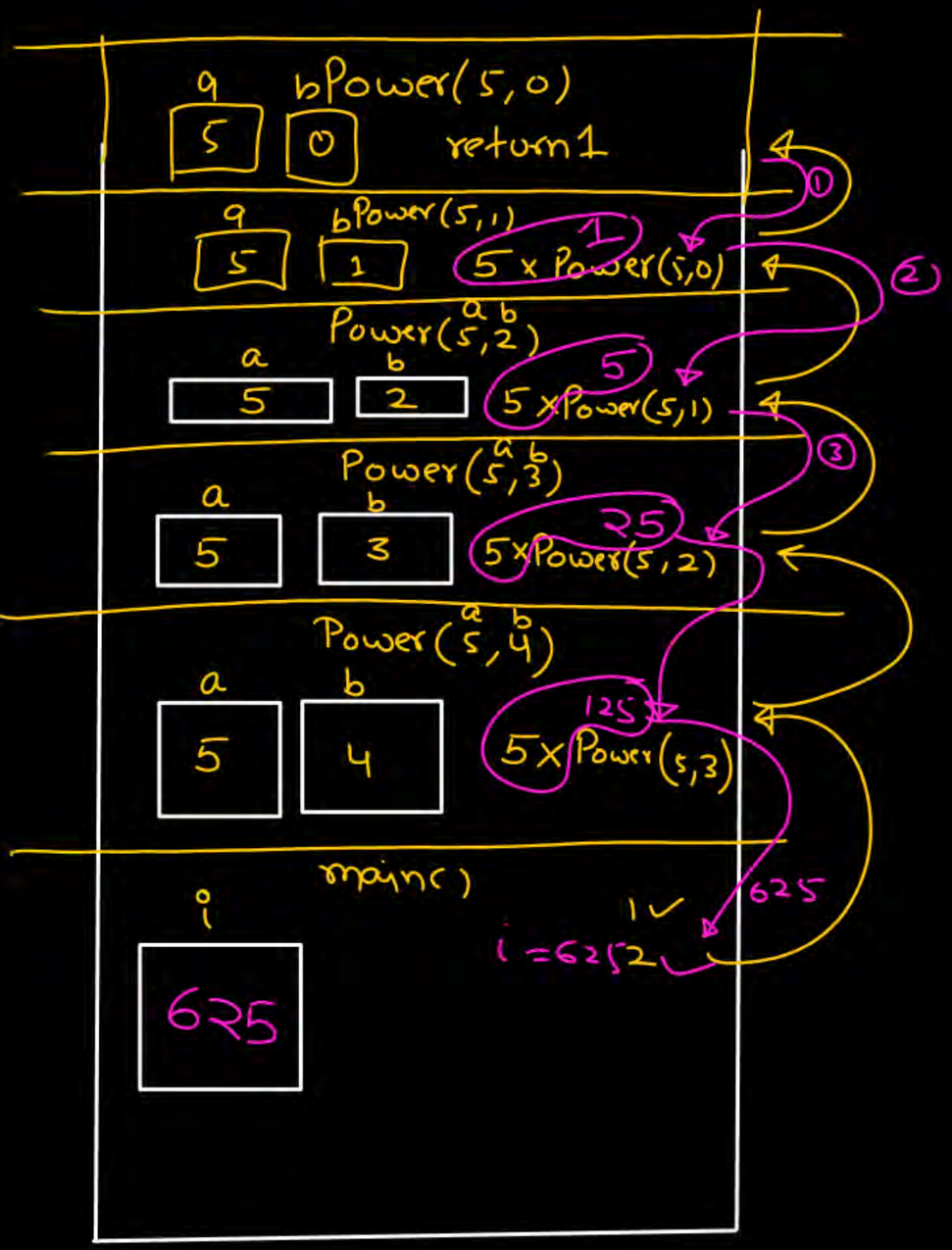
$3 \times 3 \times 3 \times 3 \ldots$

$Power(a,b) \Rightarrow a^b$

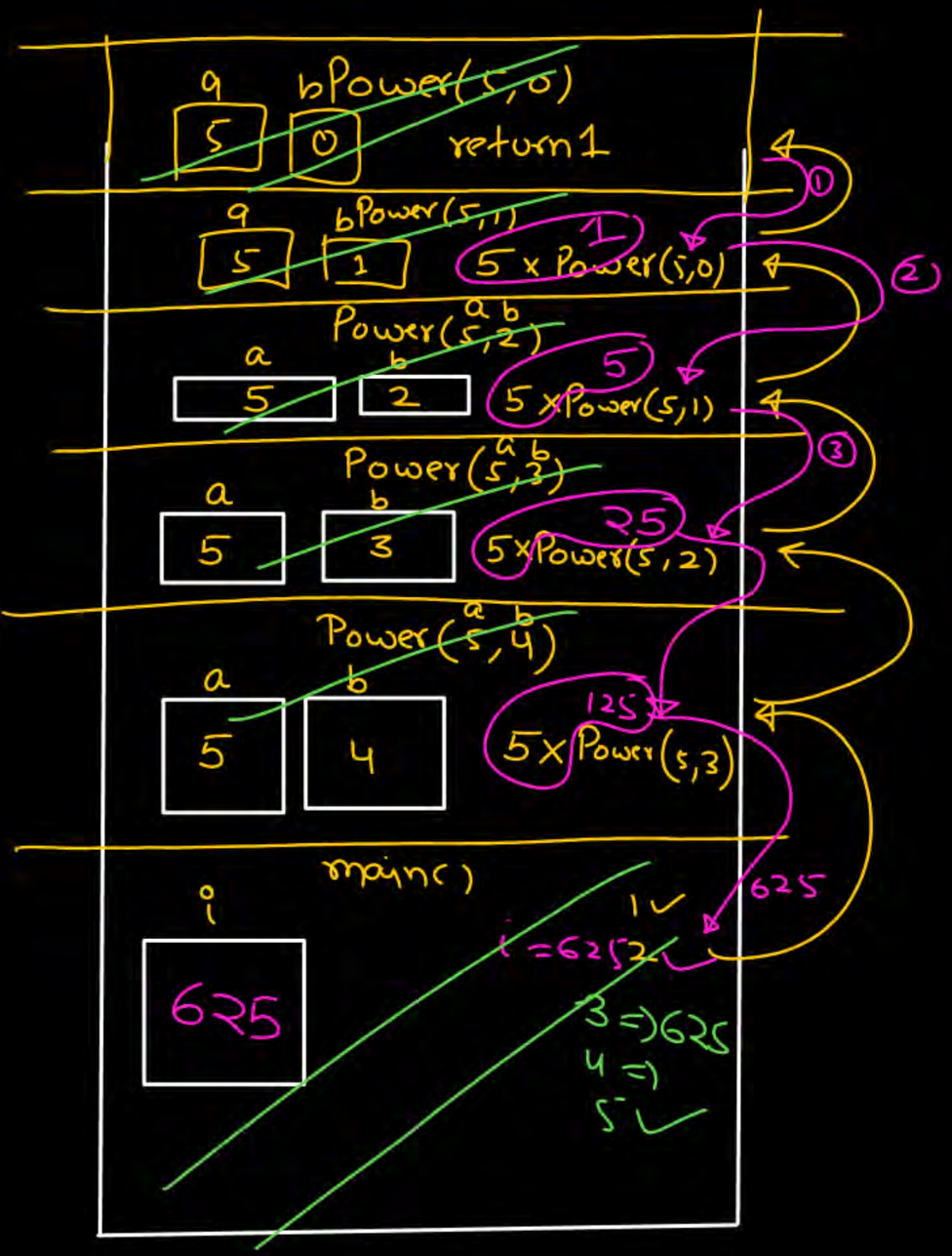
$Power(a,b-1)$ a^{b-1}

```
int Power( int a, int b)
{
    if( b==0 )
    {
        return 1;
    }
    else {
        return a x Power(a,b-1);
    }
}
```

```
int Power(int a, int b)
    {
        if(b==0)
            return 1;

    return  a * Power(a, b-1);

    }
int  main(){
1.      int i;
2.    i= Power(5,4);
3.    printf("%d",i);
4.      return 0;
5.          }
```

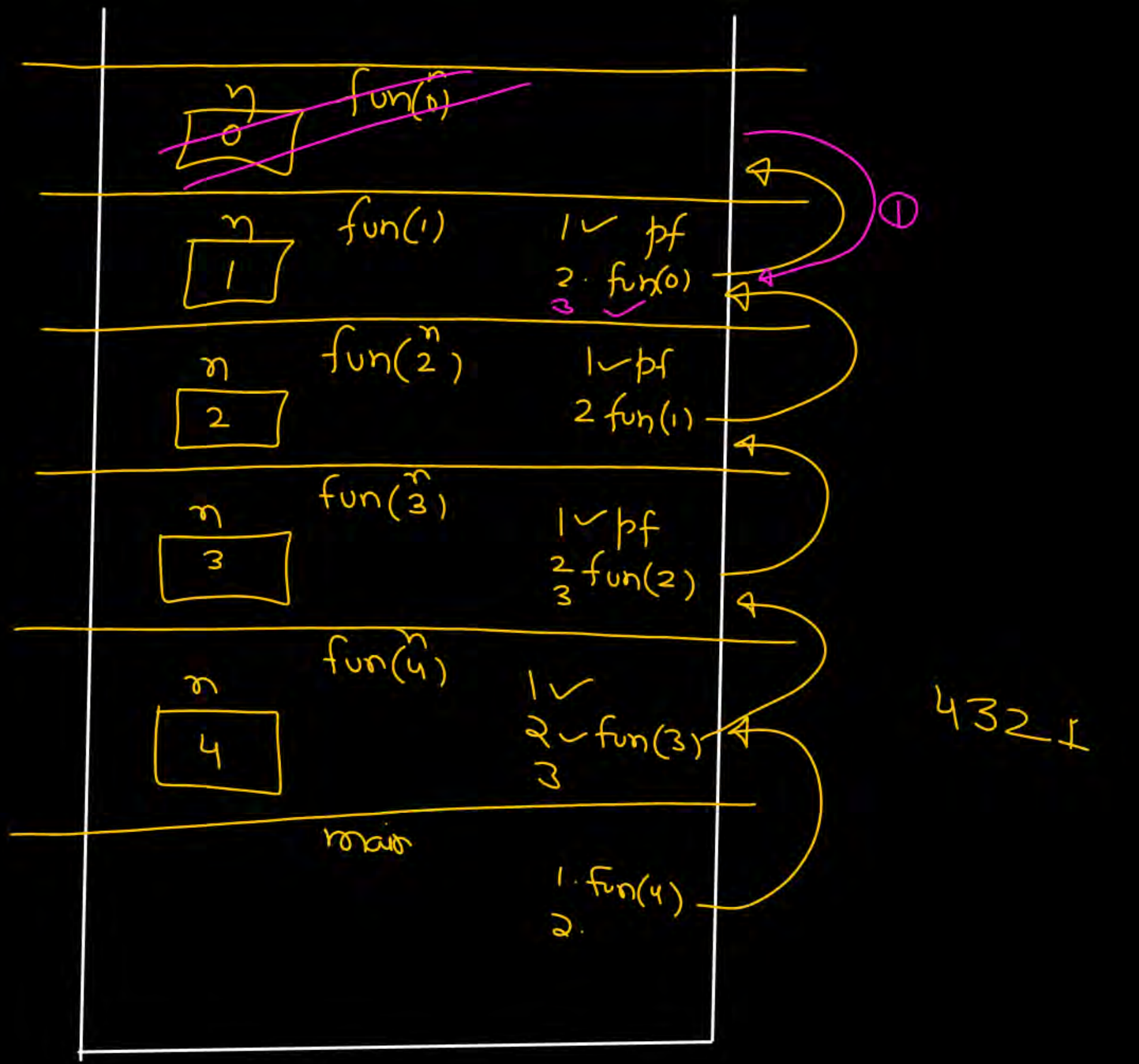


```
int Power(int a, int b)
    {
        if(b==0)
            return 1;

    return a * Power(a,b-1);
    }

int main(){
1.      int i;
2.  i= Power(5,4);
3.  printf("%d",i);
4.      return 0;
5.          }
```



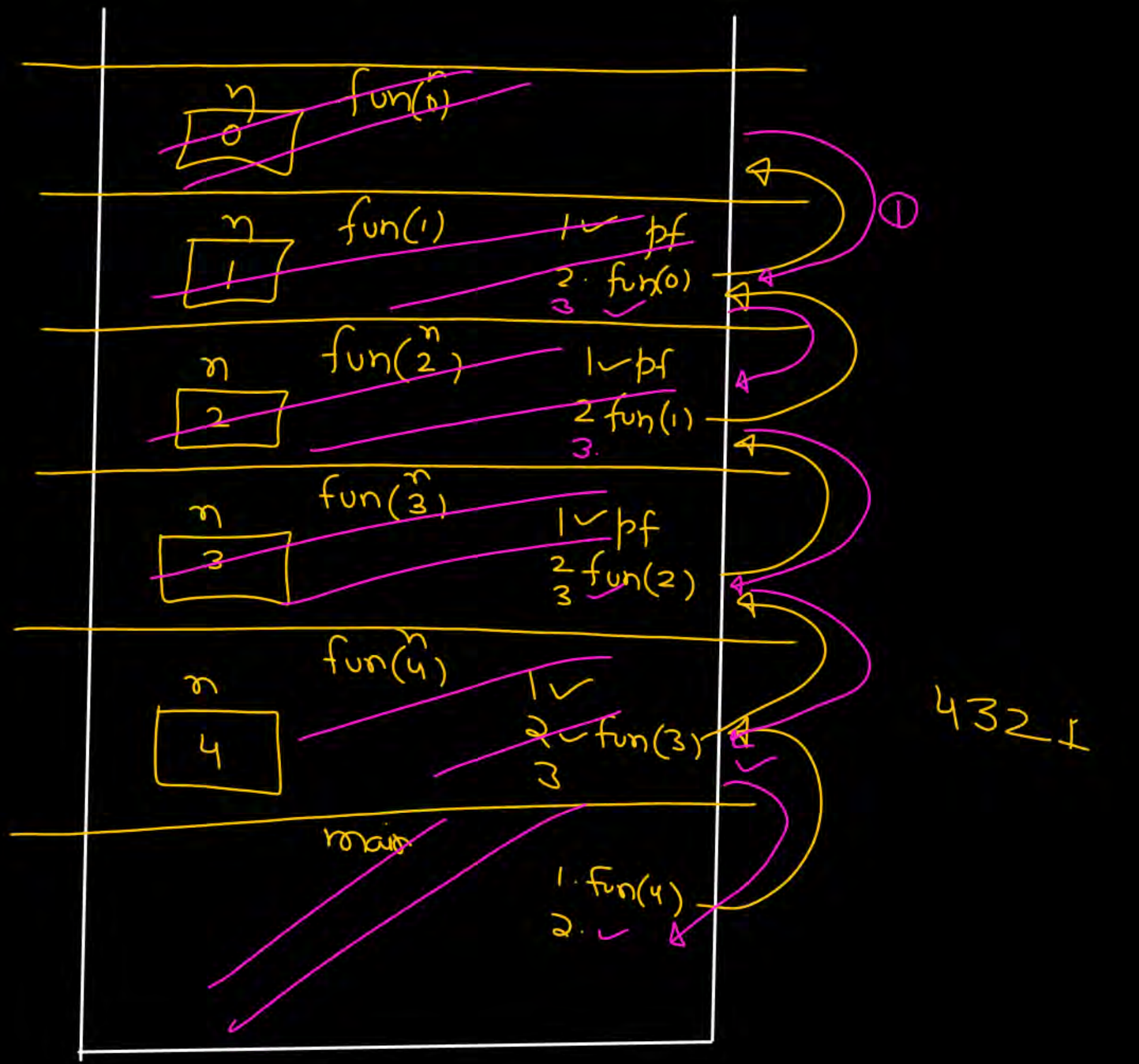
1.

```
void fun(int n){
    if(n<=0)
        return;
    else{
1.      printf("%d",n);
2.      fun(n-1);
3.  }
}
void main(){
1.  fun(4);
2.      }
```



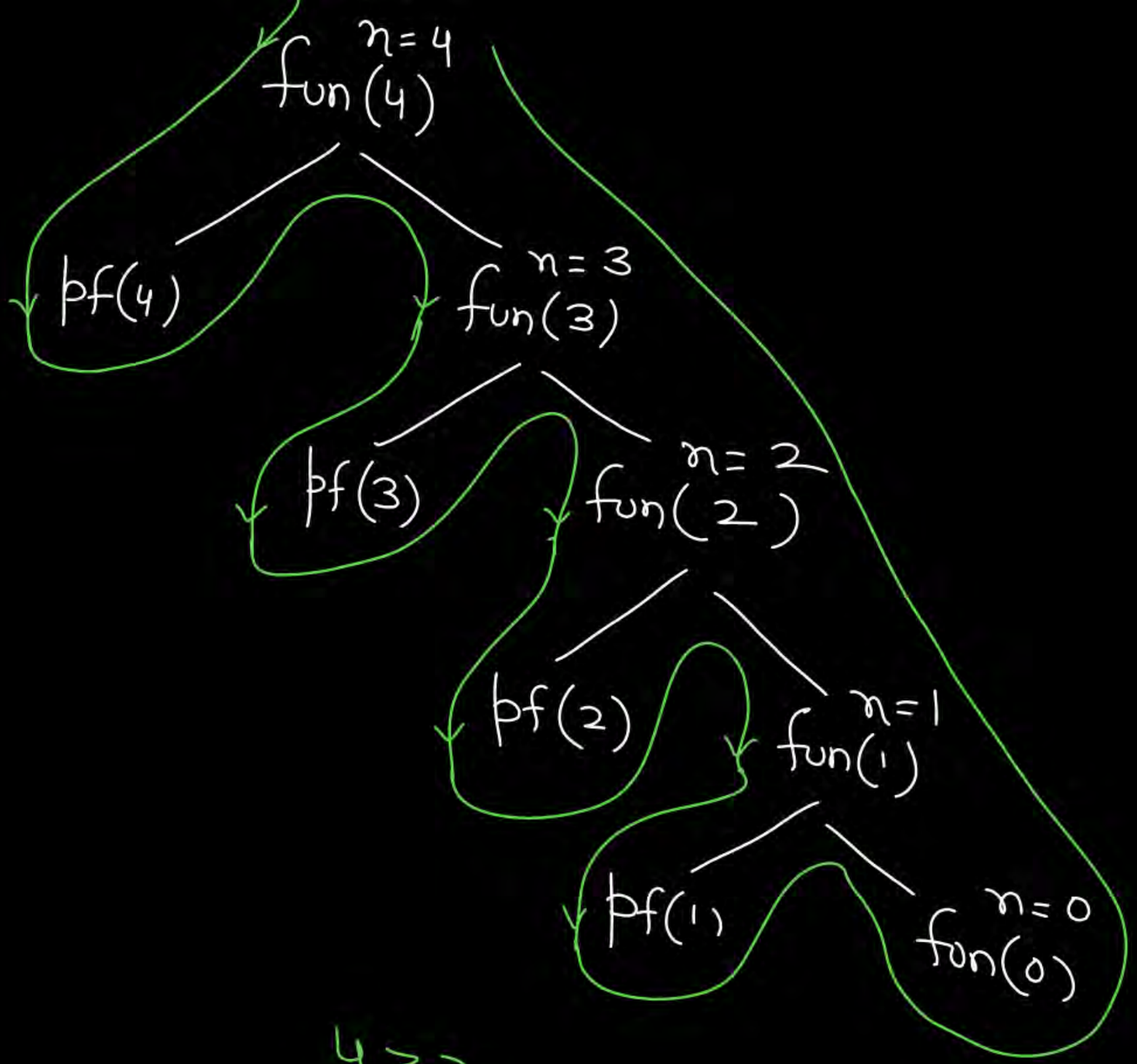
1.

```
void fun(int n){
    if(n<=0)
        return;
    else{
1.      printf("%d",n);
2.      fun(n-1);
3.  }
}
void main(){
1.  fun(4);
2.      }
```



1.

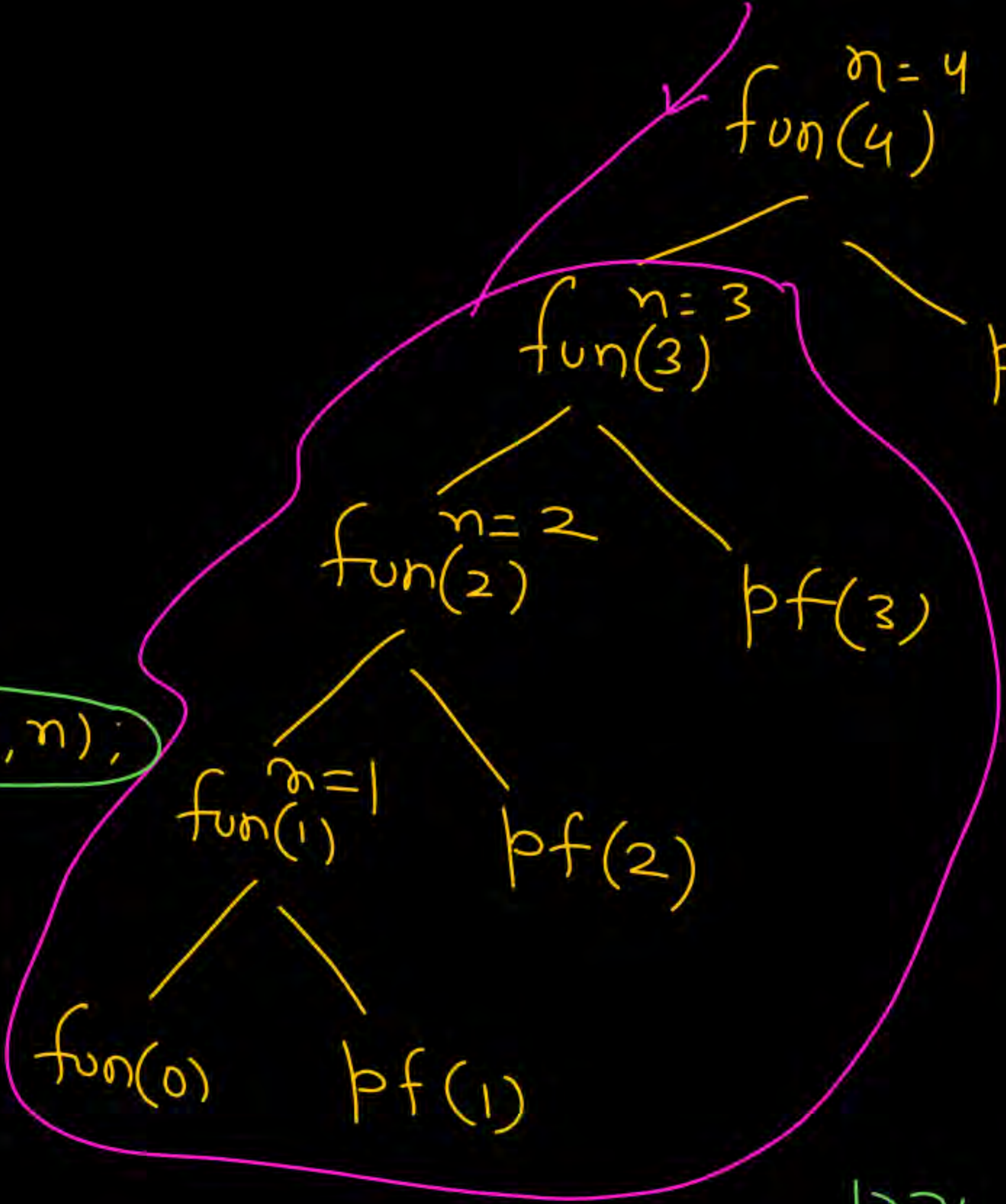
```
void fun(int n){
    if(n<=0)
        return;
    else{
        1) printf("%d",n);
        2) fun(n-1);
    }
}
void main(){
    1. fun(4);
    2. }
```



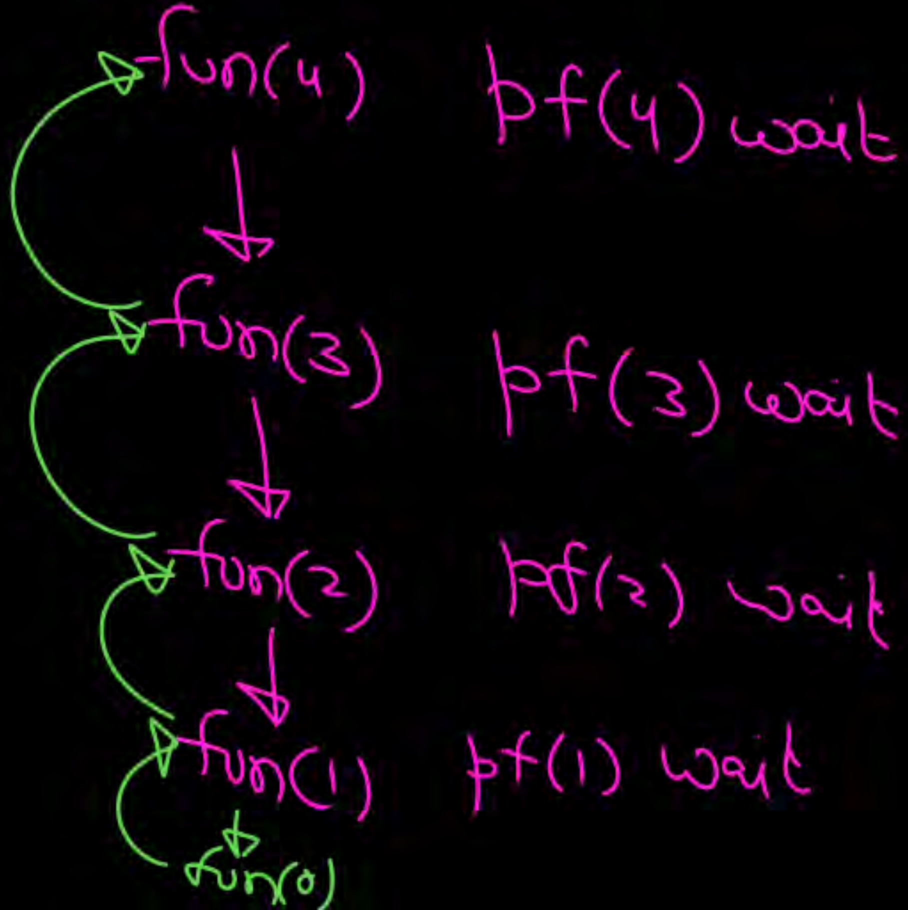
4321

1.
```
void fun(int n){
    if(n<=0)
        return;
    else{
        fun(n-1);
    2.  printf("%d",n);
    }
}
void main(){
1.  fun(4);
2.  }
```

Recursive call

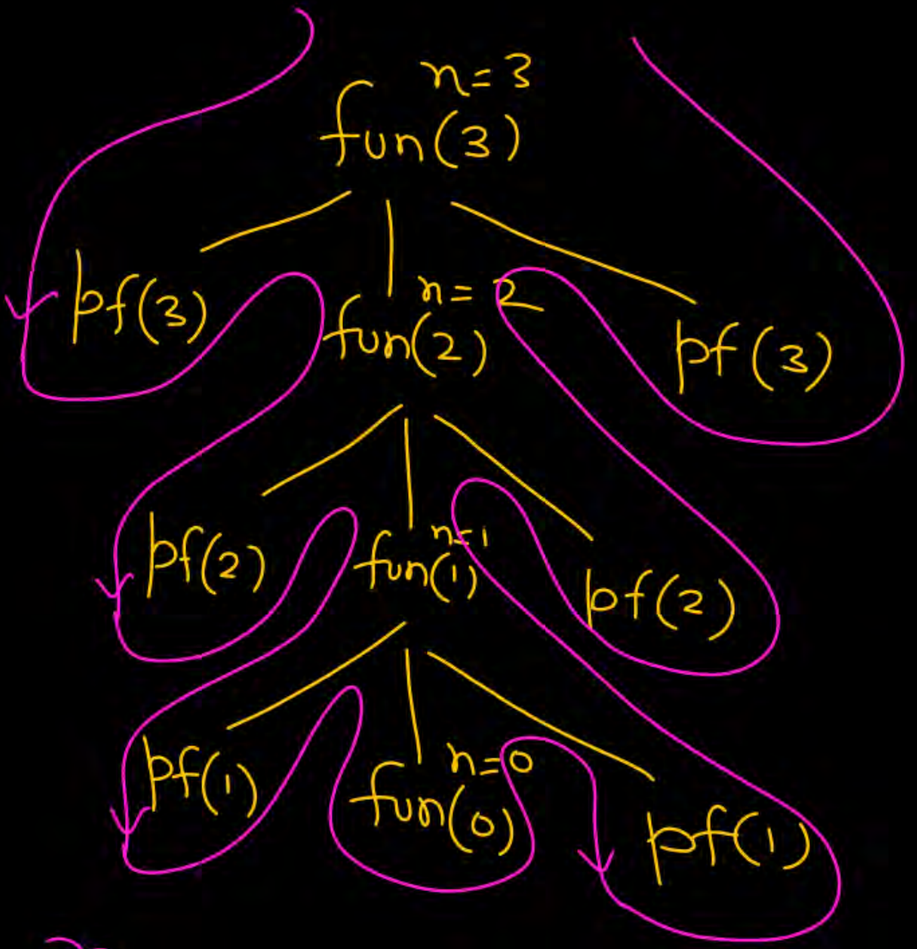


wait कर रहा है
fun(3)
⇒ रहते होते



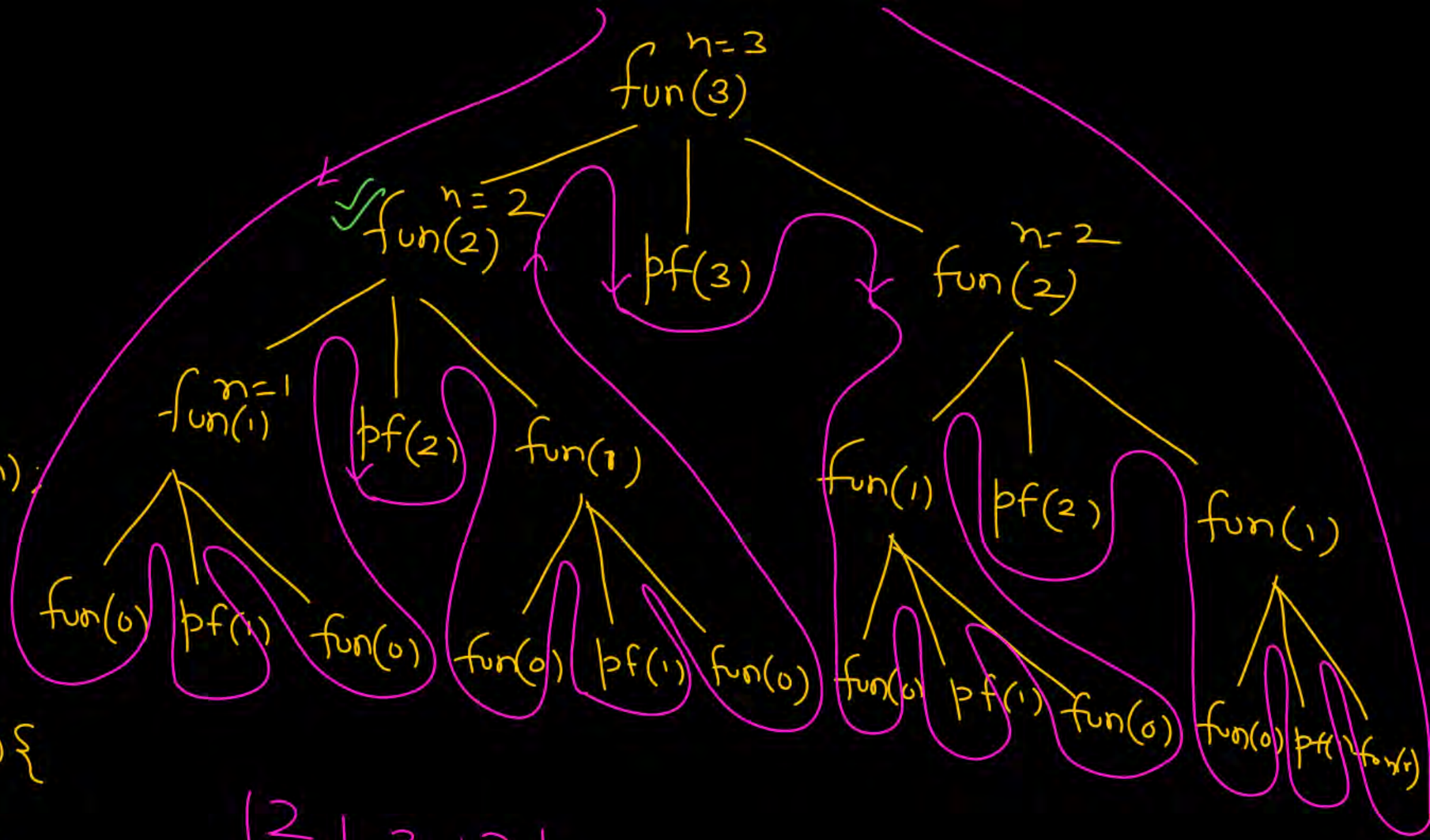
1234

```
void fun(int n){
    if (n<=0)
        return;
    else{
1       printf("%d",n);
2       fun(n-1);
3       printf("%d",n);
    }
}
void main(){
    fun(3);
}
```



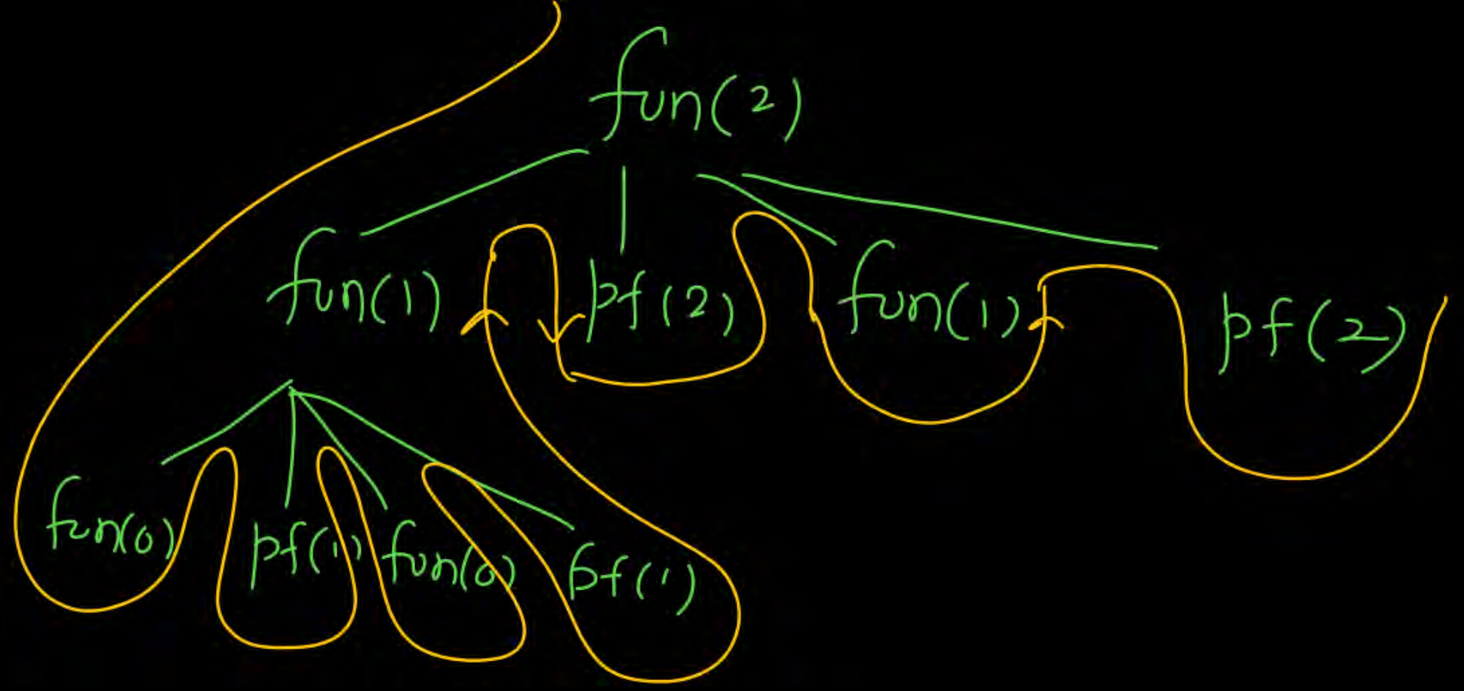
321123

```
void fun(int n){
    if (n<=0)
        return;
    else{
1       fun(n-1);
2       printf("%d",n);
3       fun(n-1);
    }
}
void main(){
    fun(3);
}
```



1 2 1 3 1 2 1

```
void fun(int n){
    if (n<=0)
        return;
    else{
1       fun(n-1);
2       printf("%d",n);
3       fun(n-1);
4       printf("%d",n);
    }
    void main(){
        fun(2);
    }
```



1 2 1 2


THANK YOU GW
SOLDIERS !