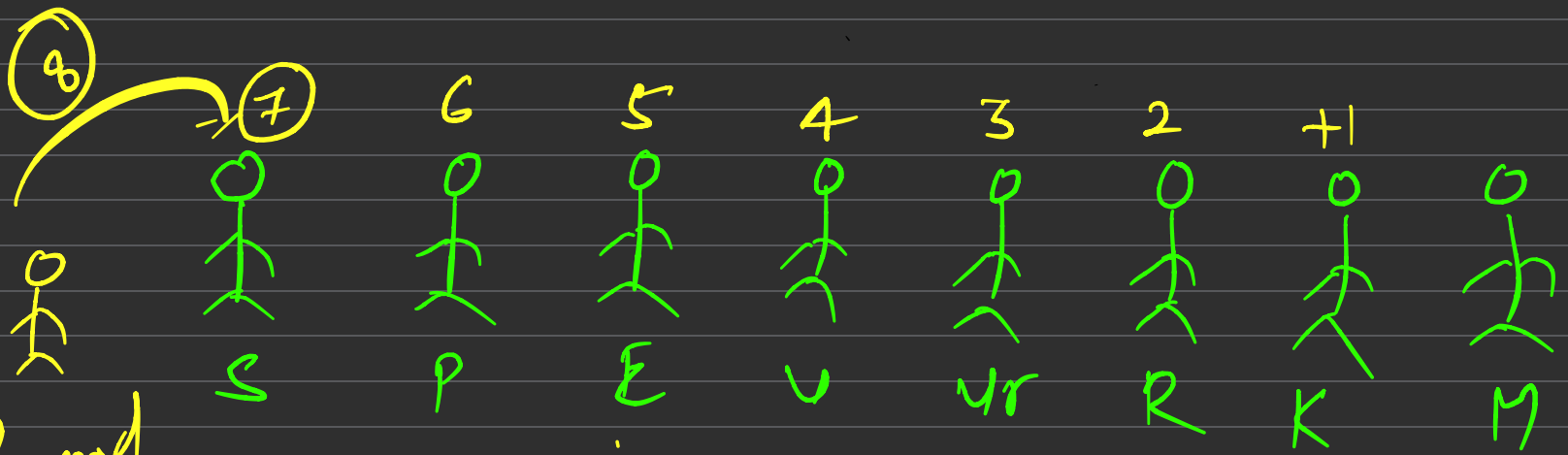
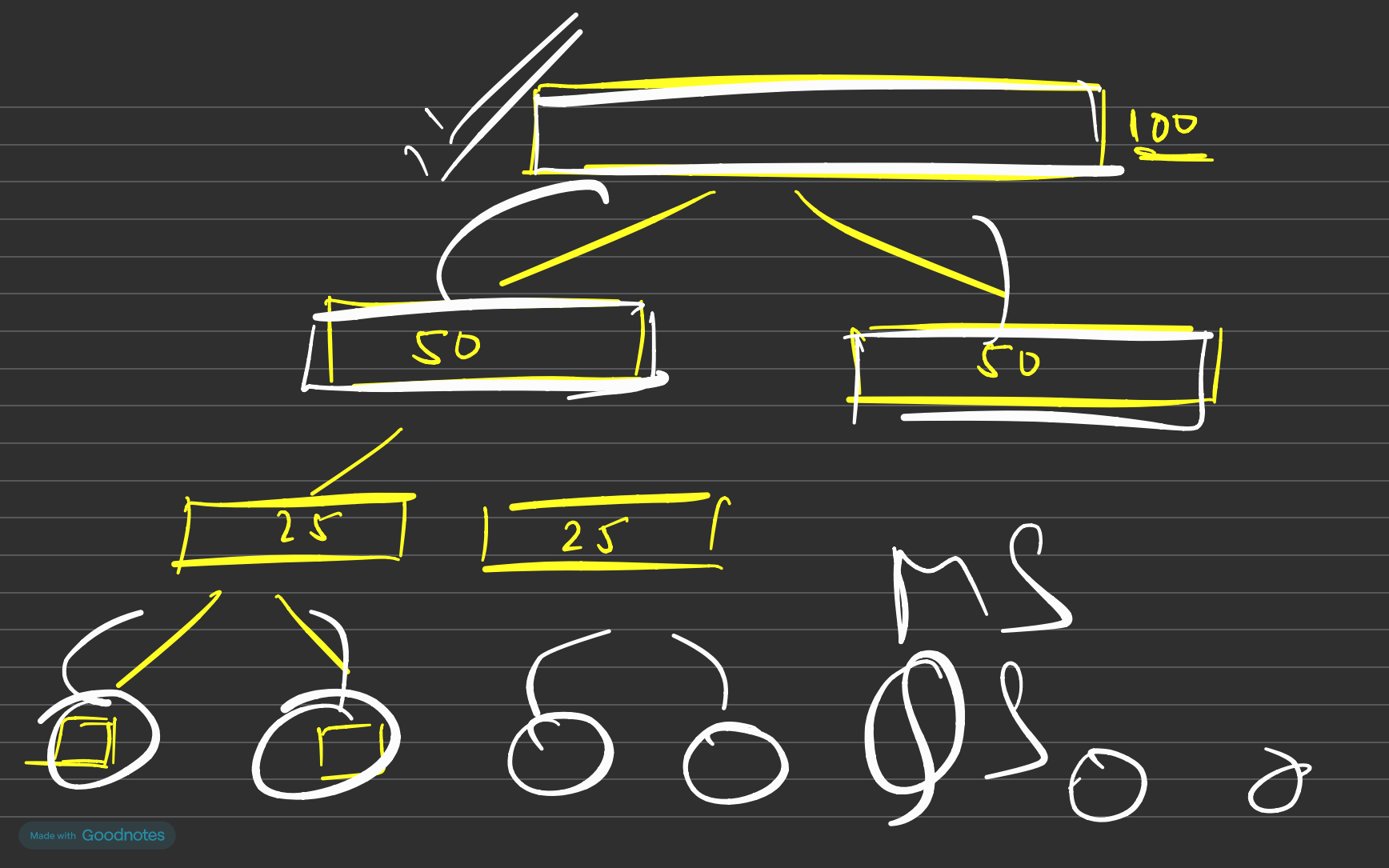
Learning is a lifelong process of keeping your mind open to new ideas and experiences.

RECURSION & BACKTRACKING

- RECURSIVE TREE
- DRY RUN
- FUNCTION CALL



Pramid



$x = 1234$ → 10

↓

1 + (234)

↓ 9

2 + (34)

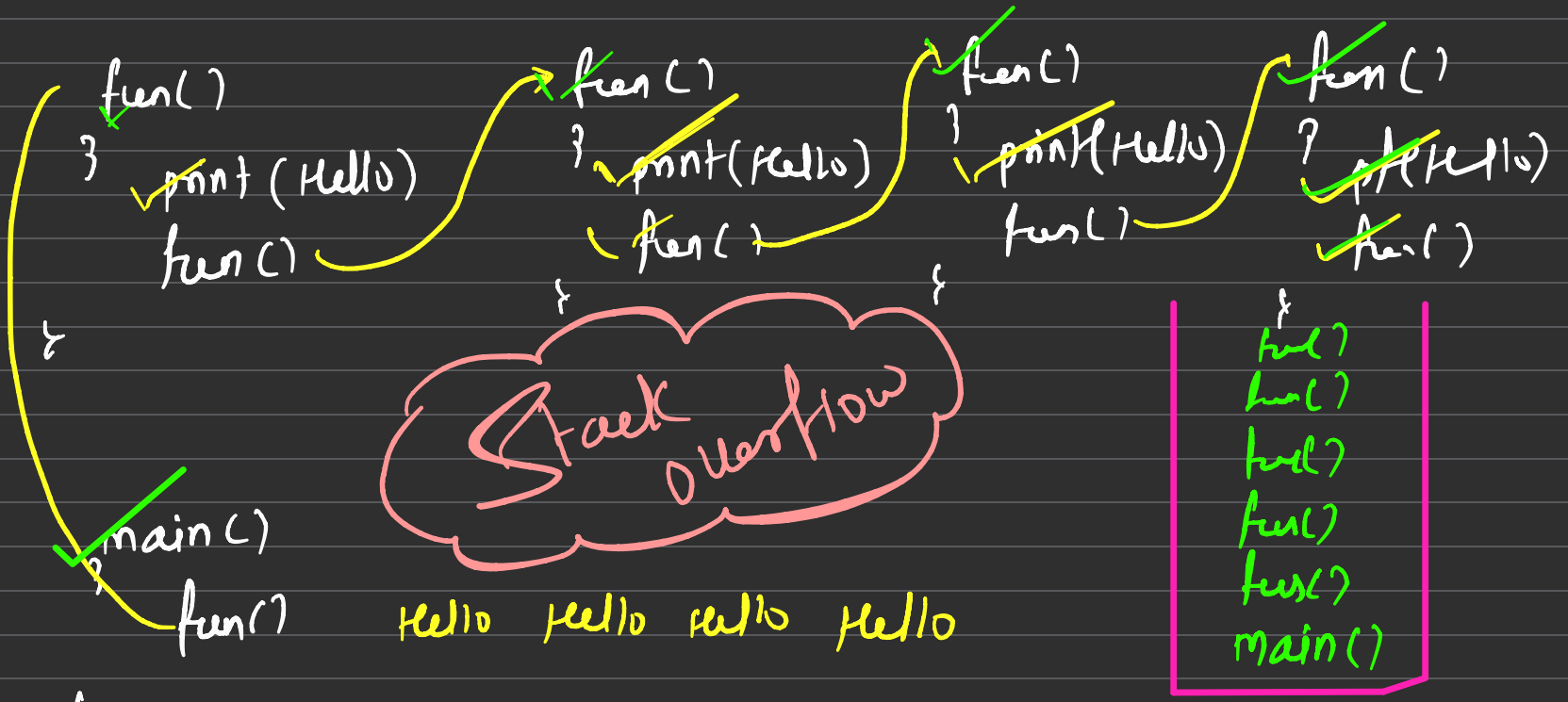
↓ 7

3 + (4)

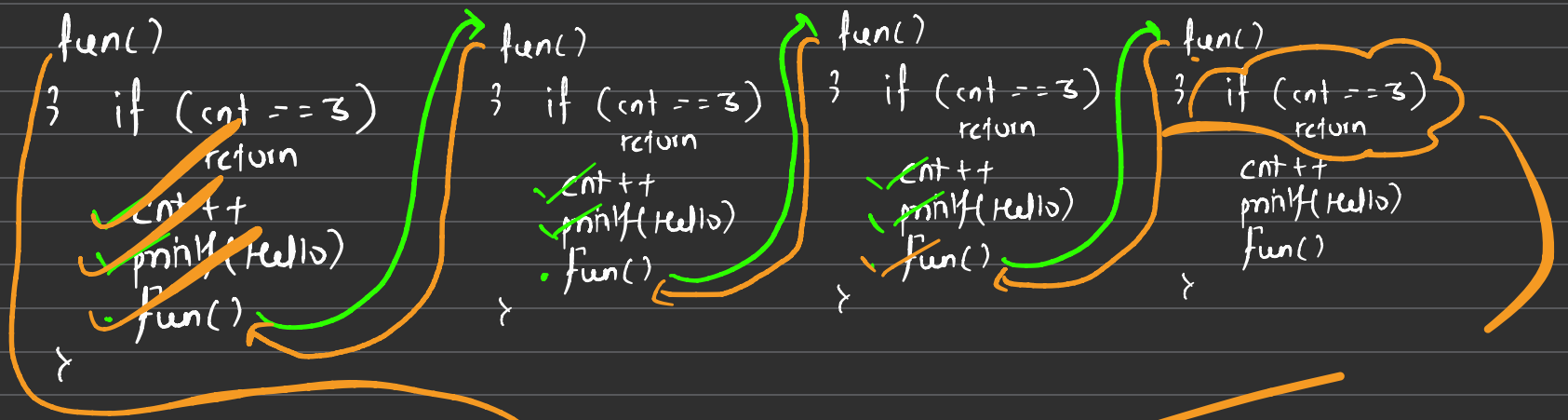
4

What is Recursion

Function calling itself until a specific condition is called as Recursion.



cnt = 0 X 3



Hello Hello Hello

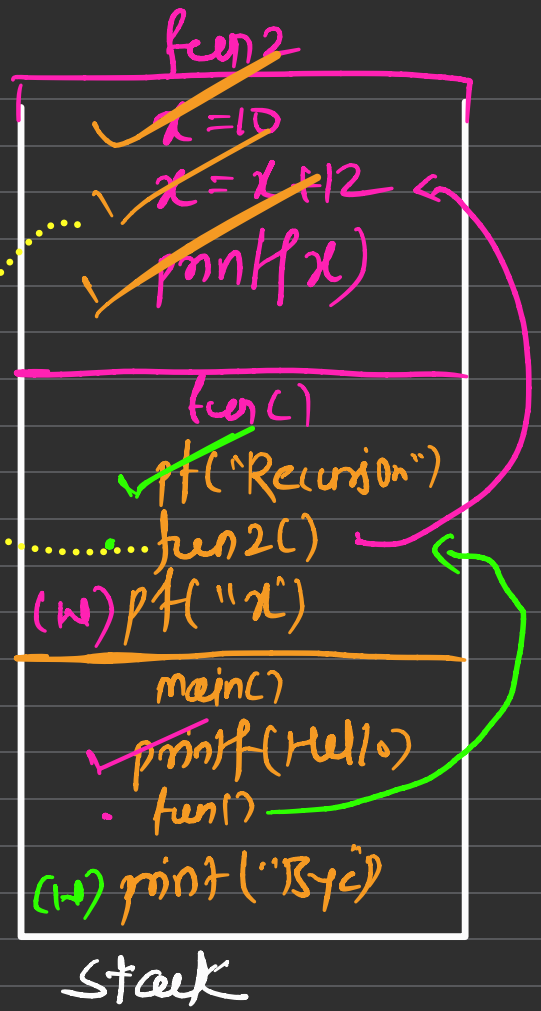
```
main()
{
    fun()
}
```

Base Case

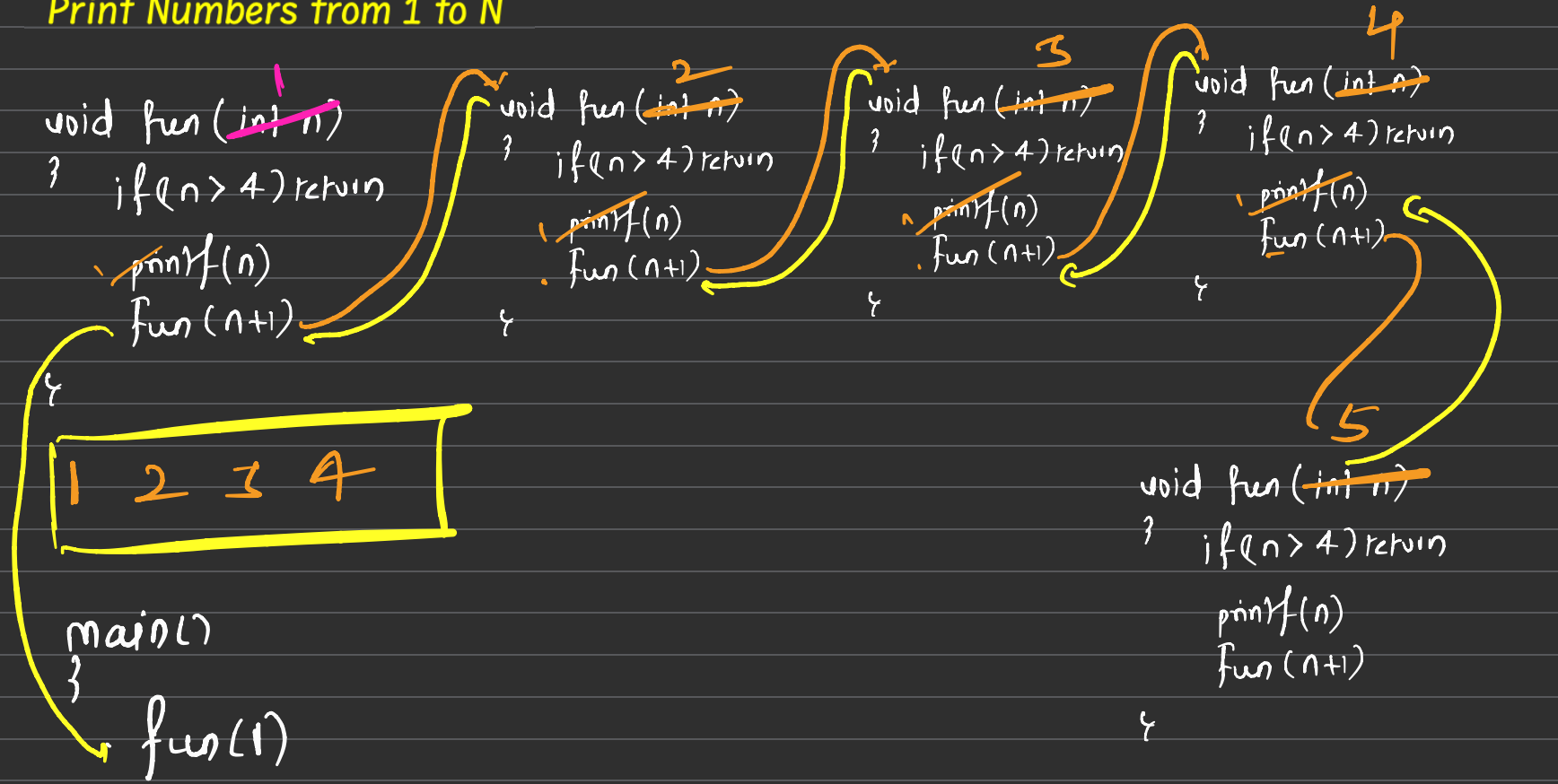
Activation Stack Record

```
main()
{
    print(Hello)
    fun()
    print(Bye)
}

fun()
{
    print("Recursion")
    fun2()
    print("x")
}

fun2()
{
    x=10
    x=x+12
    printf(x)
}
```

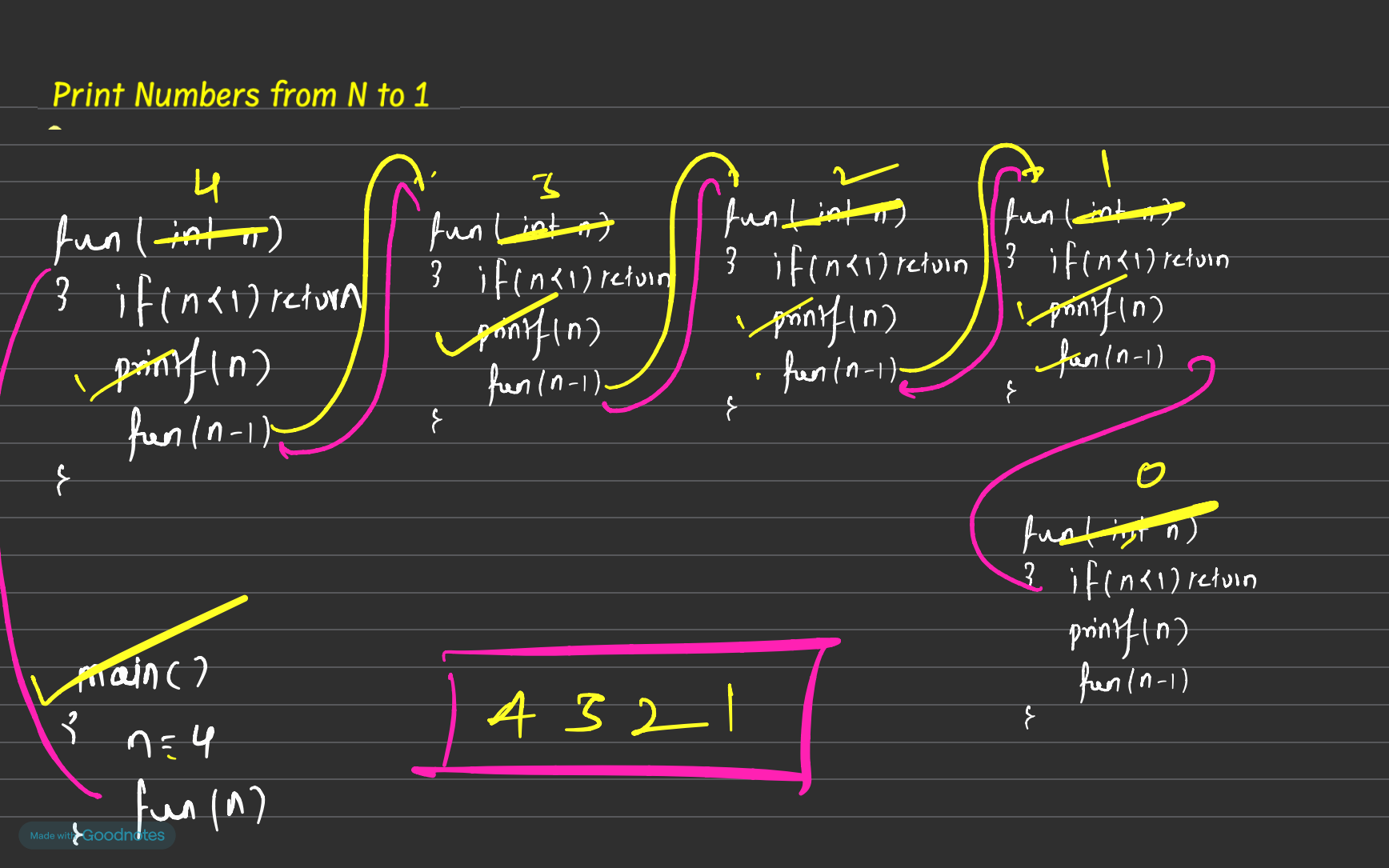
Hello
Recursion
12
x
Bye



Print Numbers from 1 to N



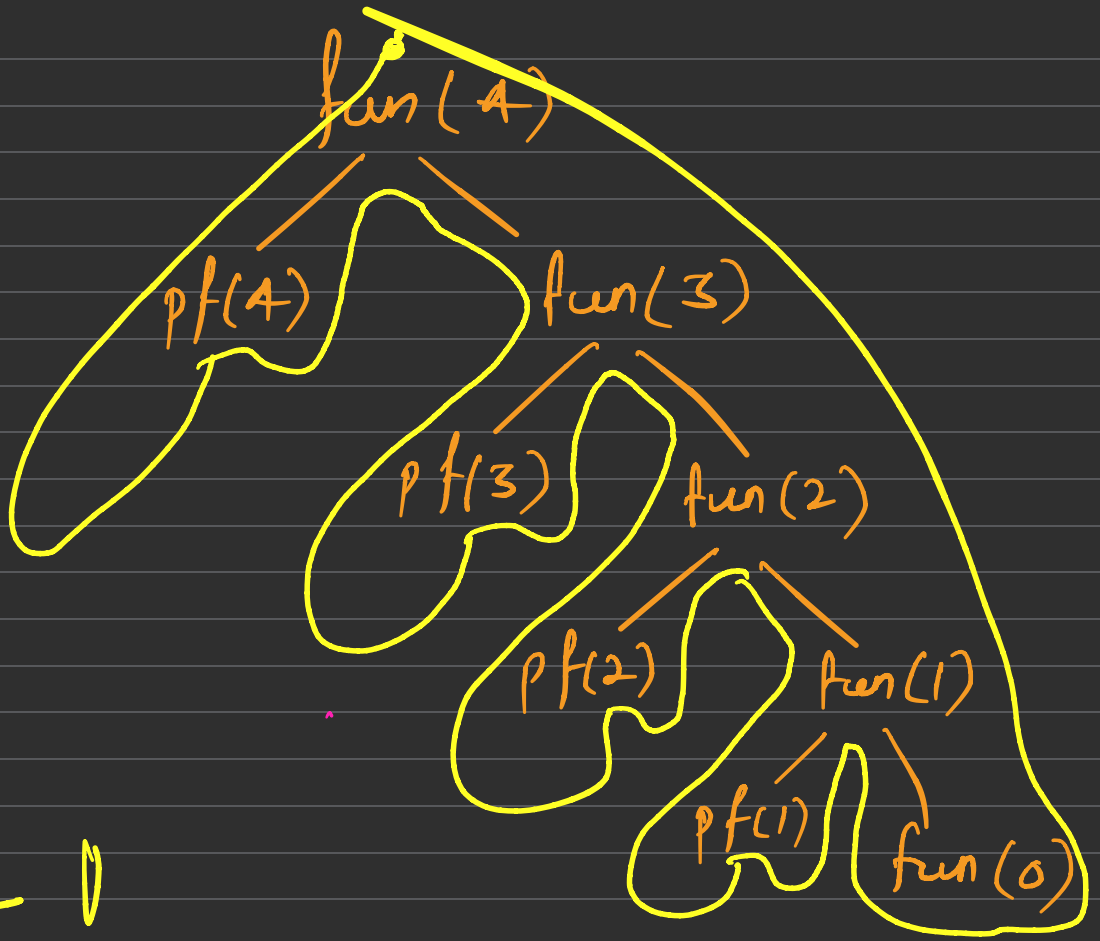
Print Numbers from N to 1



Recursive Tree

```
fun( int n)
{  if(n<1) return
   printf(n)
   fun(n-1)
}
```

N = 4



4 3 2 1

4 3 2 1

```
fun( int n)
{ if (n > 4) return
  fun(n+1)
  printf (n)
}
```

fun(1)

printf in same order

```
printf(N)
fun(N+1)
```

```
fun(N+1)
printf(N)
```

print in Reverse Order

Parameterised and Functional Recursion

Parameterised Recursion

Find the Sum of all the numbers from 1 to N

```
main()
{
    N = 4
    sum = 0
    fun(1, N, sum)
}
```

```
void fun(i, N, sum)
{
    if (i > N)
    {
        print(sum)
        return
    }
    sum = sum + i
    fun(i+1, N, sum)
}
```

N = 4

Sum = 0 ~~1 3 6~~ 10

1,

```
void fun(i, N, sum)
{
    if (i > N)
    {   print(sum)
        return
    }
    sum = sum + i
    fun(i+1, N, sum)
}
```

2

```
void fun(i, N, sum)
{
    if (i > N)
    {   print(sum)
        return
    }
    sum = sum + i
    fun(i+1, N, sum)
}
```

3

```
void fun(i, N, sum)
{
    if (i > N)
    {   print(sum)
        return
    }
    sum = sum + i
    fun(i+1, N, sum)
}
```

4

```
void fun(i, N, sum)
{
    if (i > N)
    {   print(sum)
        return
    }
    sum = sum + i
    fun(i+1, N, sum)
}
```

5

```
void fun(i, N, sum)
{
    if (i > N)
    {   print(sum)
        return
    }
    sum = sum + i
    fun(i+1, N, sum)
}
```

10

Find the Sum of all the numbers from 1 to N

Functional Recursion

Find the Sum of all the numbers from 1 to N

```
main()
{
    N = 4
    ans = fun(n)
    printf(ans)
}
```

10

```
int fun(int n)
{
    if(n==0)
        return 0

    return n + fun(n-1)
}
```

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

Tree

Graph

Backtracking

Dynamic Prog.

fun(4) 10

4 + fun(3) 6

3 + fun(2) 3

2 + fun(1) 1

1 + fun(0) 0

Addition of digits in Number

Write a code to find the factorial of a number

Program to count digits in a number

Program to check if string is palindrome or not

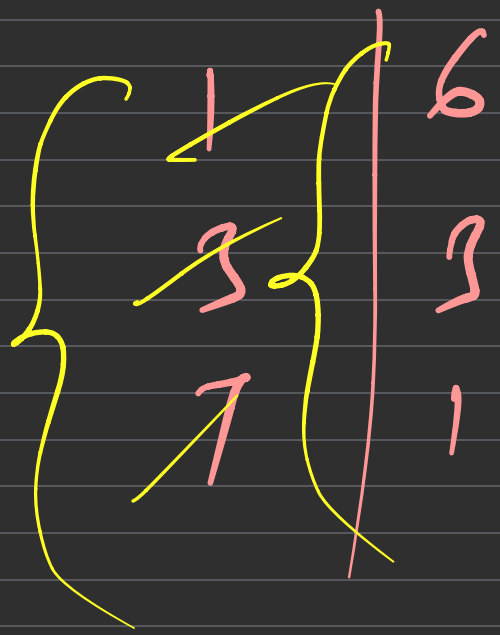
Program to Reverse array Recursively

```
int x=5, y=10, z=1
if(x>0 && y>z)
{
    printf("Pramod")
}
else
{
    break
}
```

```
int i;
for (i=0 ; i<=5 ; i++)
{
    int a = 10
    printf(a)
    a = a+1
}
```

```
main()
{
    int ans = fun(6)
    printf(ans)
}
int fun(int n)
{
    if (n <= 1) return n
    return 2 * fun(n/2) + 1
}
```

(1)



6

```
int fun(int n)
{
    if (n <= 1) return n
    return 2 * fun(n/2) + 1
}
```

~~fun(6)~~ ⑦

1

3

$2 \times$ ~~fun(3)~~ $+1$

1

$2 \times$ ~~fun(1)~~ $+1$

24

```
main()
{
    int ans = fun(12);
    printf(ans);
}
int fun(int n)
{
    if (n <= 1) return n;
    return 2 * fun(n/2) + n/2;
}
```

24

24

12	1
6	3
3	9
1	24

12

```
int fun(int n)
{
    if (n<=1) return n
    return 2*fun(n/2)+n/2
}
```

fun(12) 24

+

2 × fun(6) + 6 (9)

↓

2 × fun(3) + 3 (3)

↓

2 × fun(1) + 1 (1)

↓

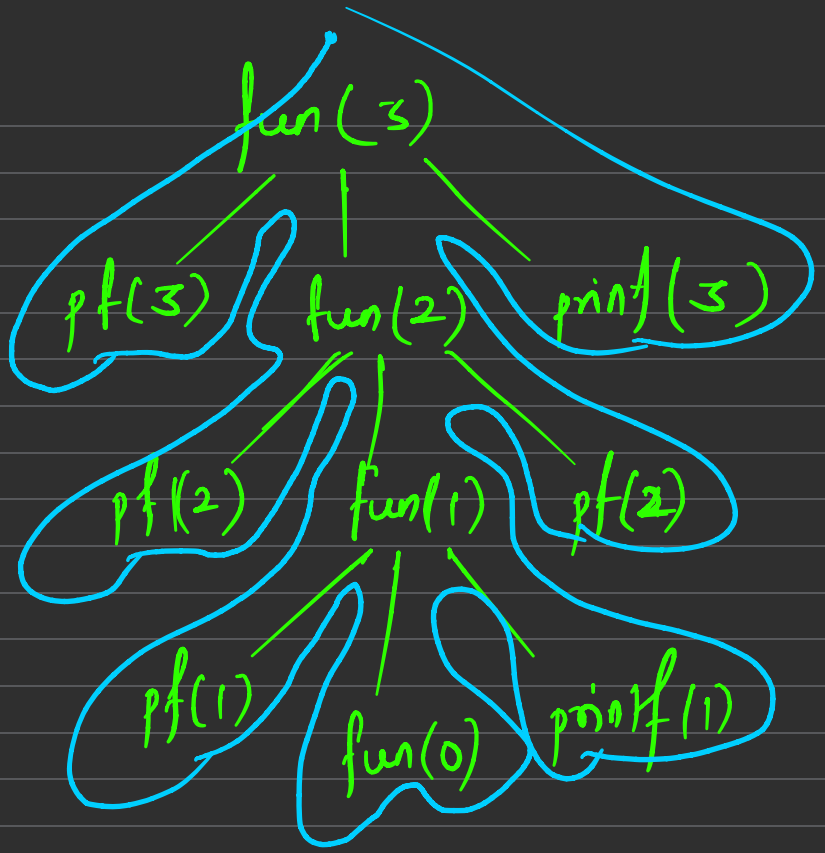
1

```
main()
{
  fun(3)
}
void fun(int n)
{
  if(n>0)
  {
    printf(n)
    fun(n-1)
    printf(n)
  }
}
```

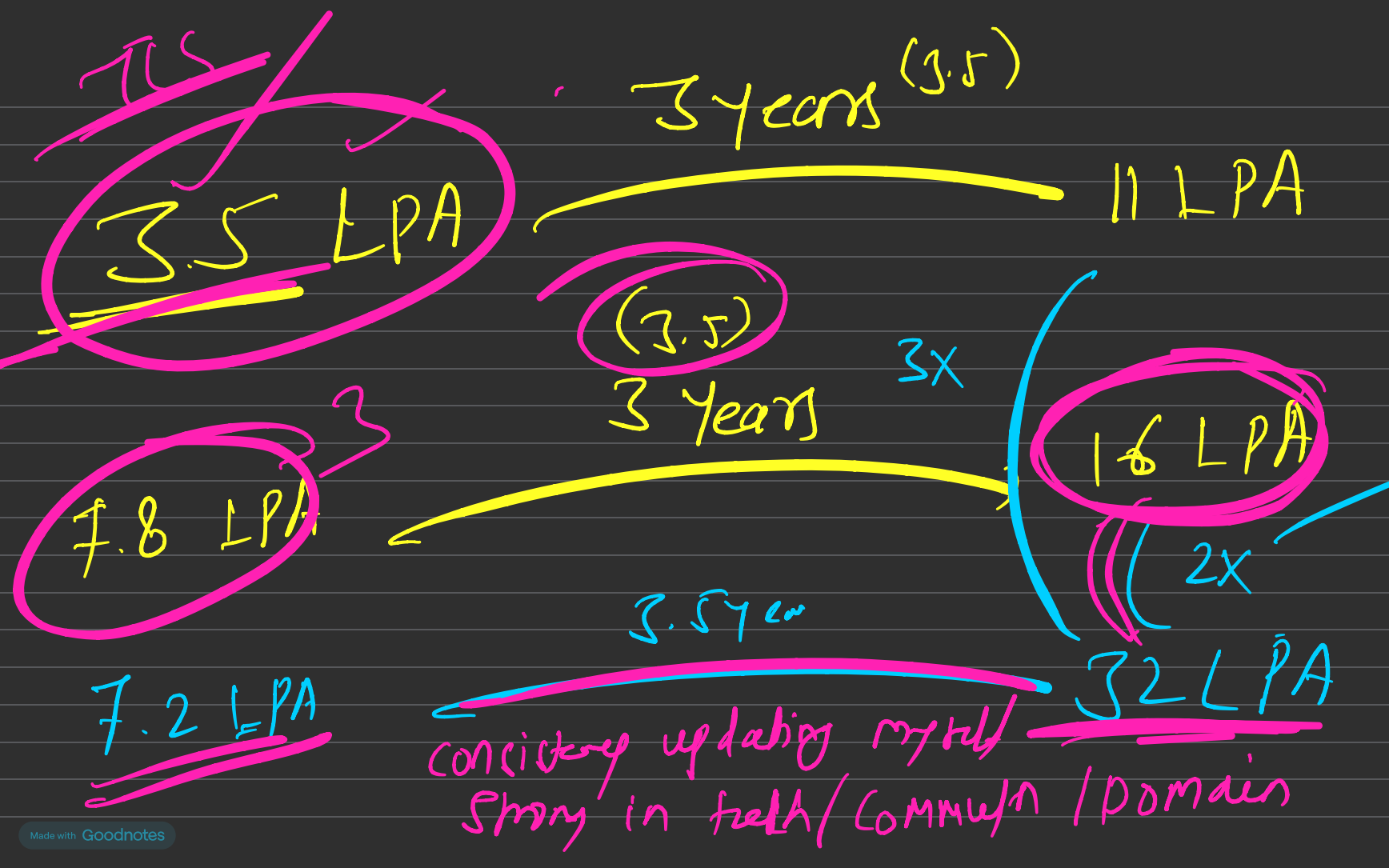
321

321123

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



3 2 1 1 2 3


3.5 LPA
3 years (3.5)
11 LPA
(3.5)
3 Years
3x
7.8 LPA
16 LPA
2x
3.5 yrs
7.2 LPA
32 LPA
consistency updating myself
strong in tech/ common / domain