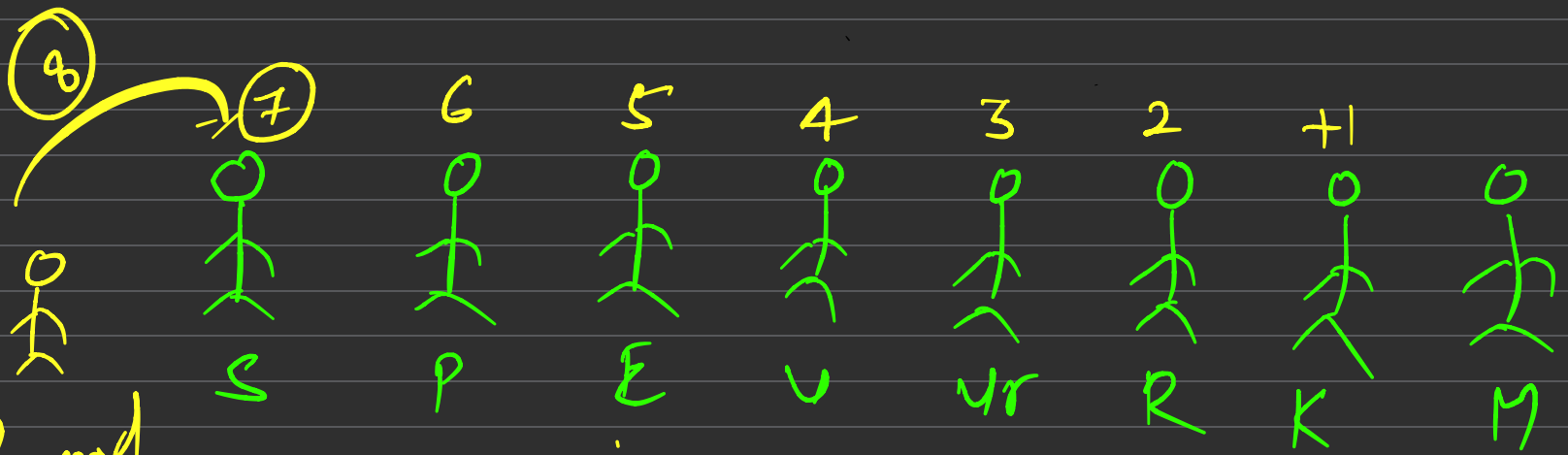
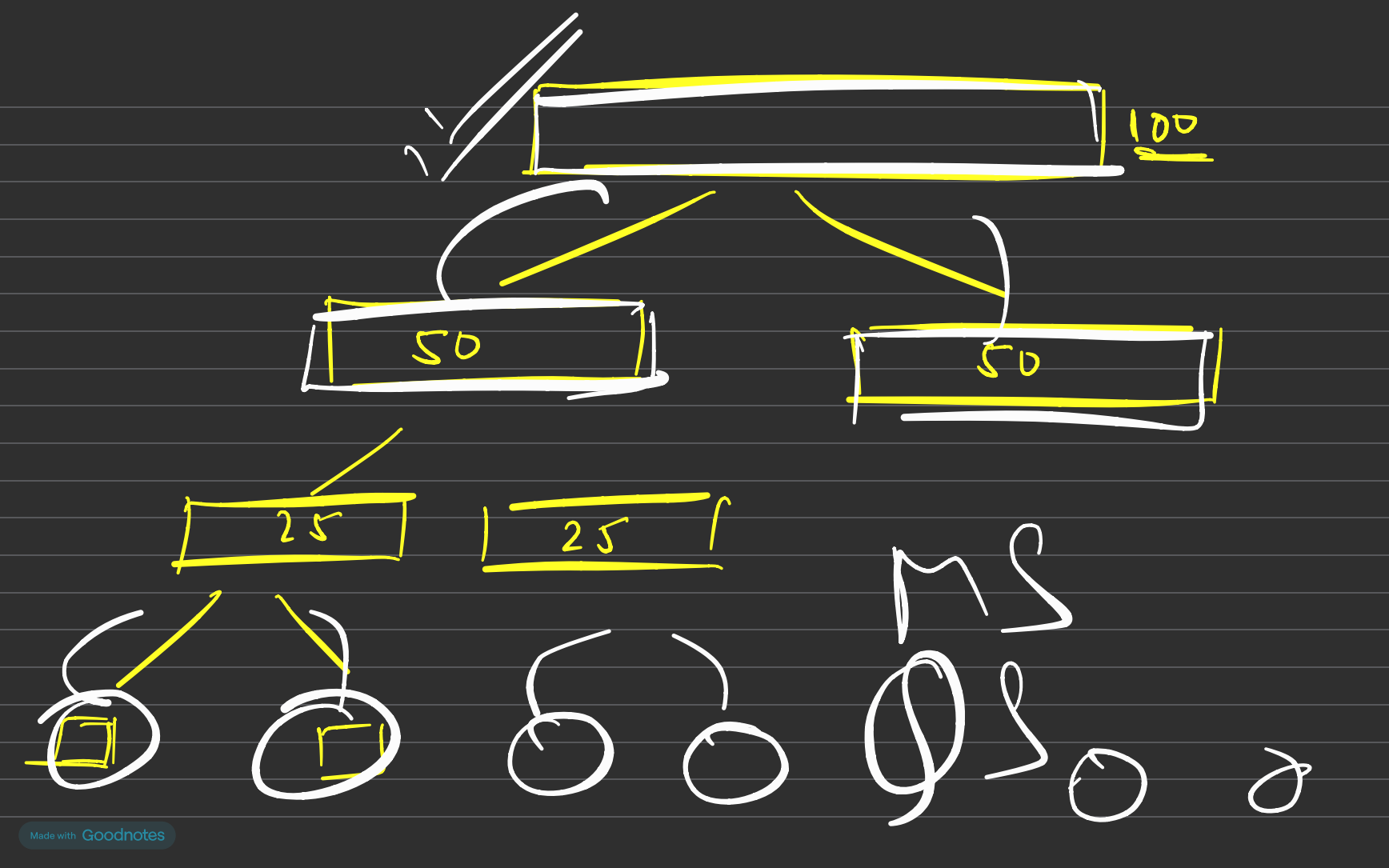
Learning is a lifelong process of keeping your mind open to new ideas and experiences.

RECURSION & BACKTRACKING

- RECURSIVE TREE
- DRY RUN
- FUNCTION CALL



Pramid



$x = 1234$ → 10

↓

1 + (234)

↓ 9

2 + (34)

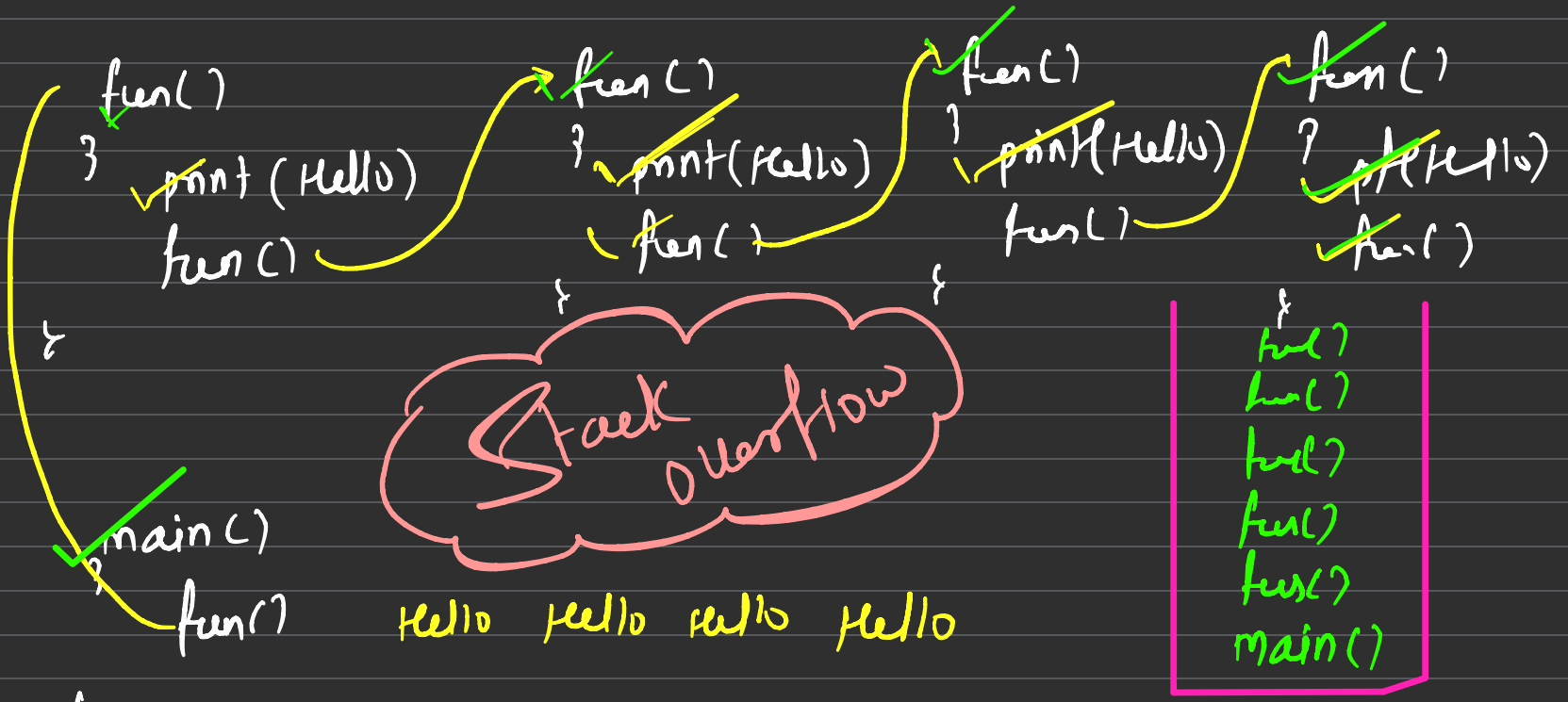
↓ 7

3 + (4)

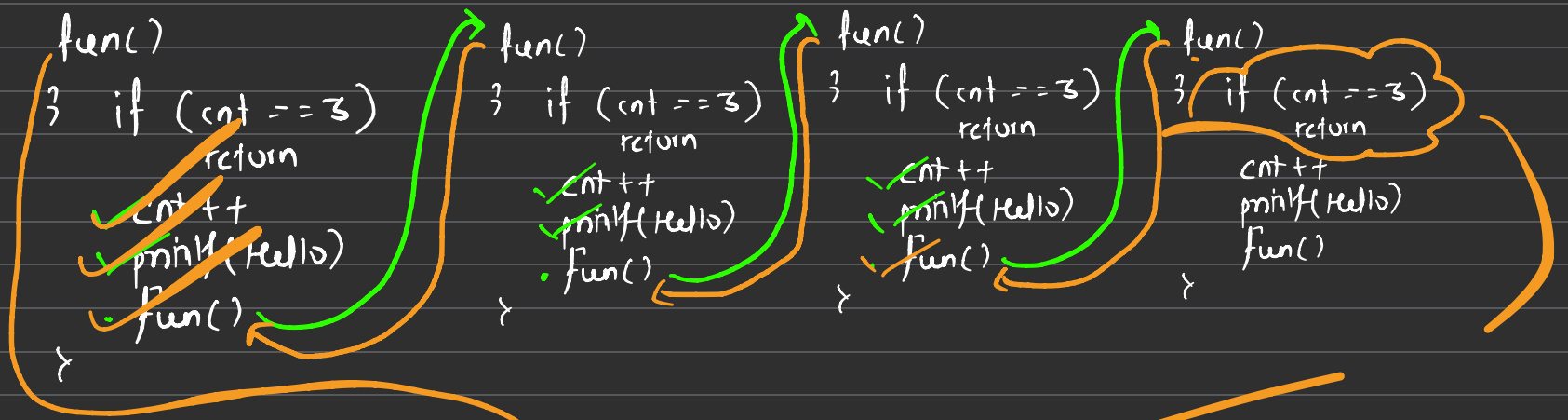
4

What is Recursion

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



cnt = 0 X 3



Hello Hello Hello

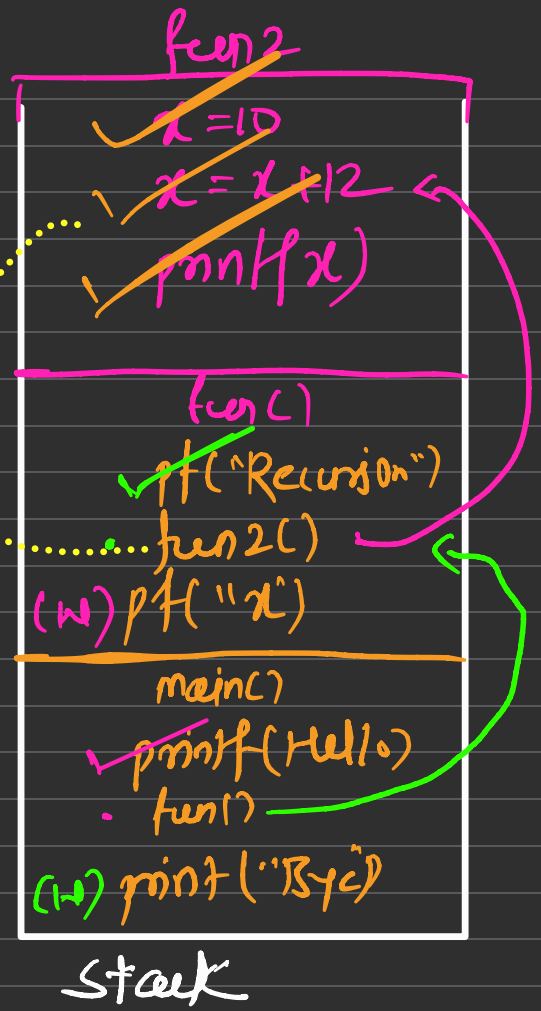
```
main()
{
    fun()
}
```

Base Case

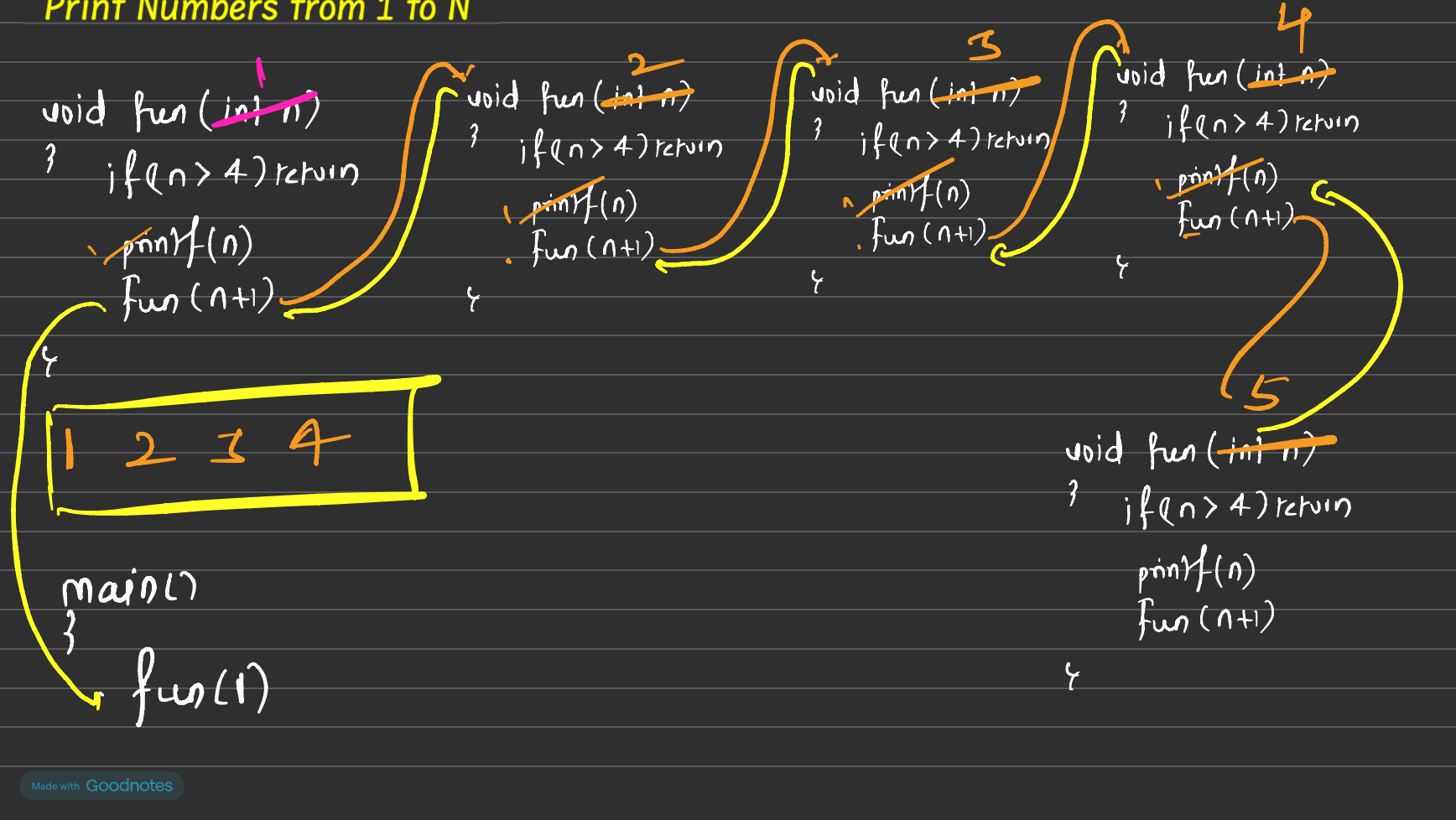
Activation Stack Record

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

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

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

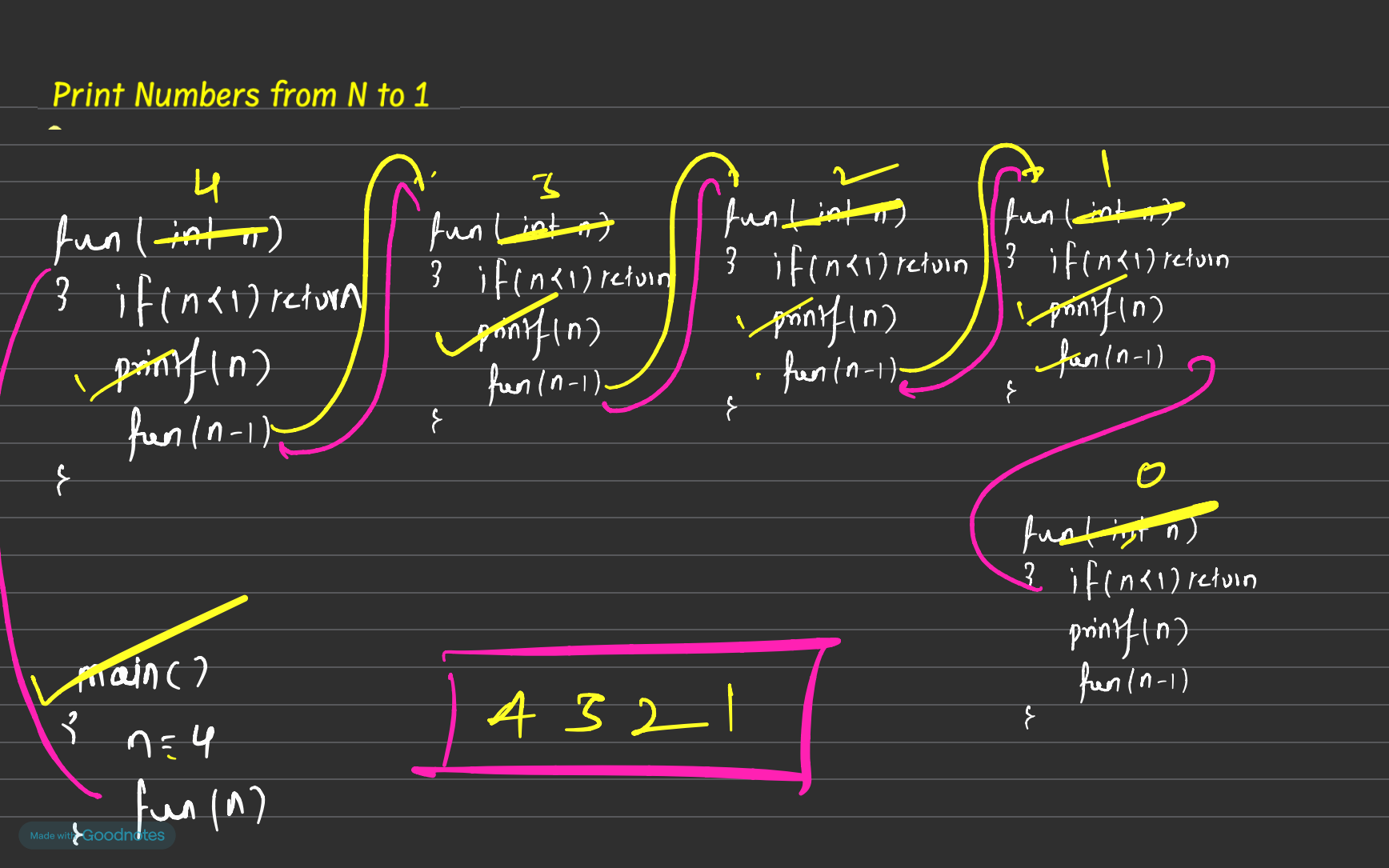
Hello
Recursion
12
x
Bye



Print Numbers from 1 to N



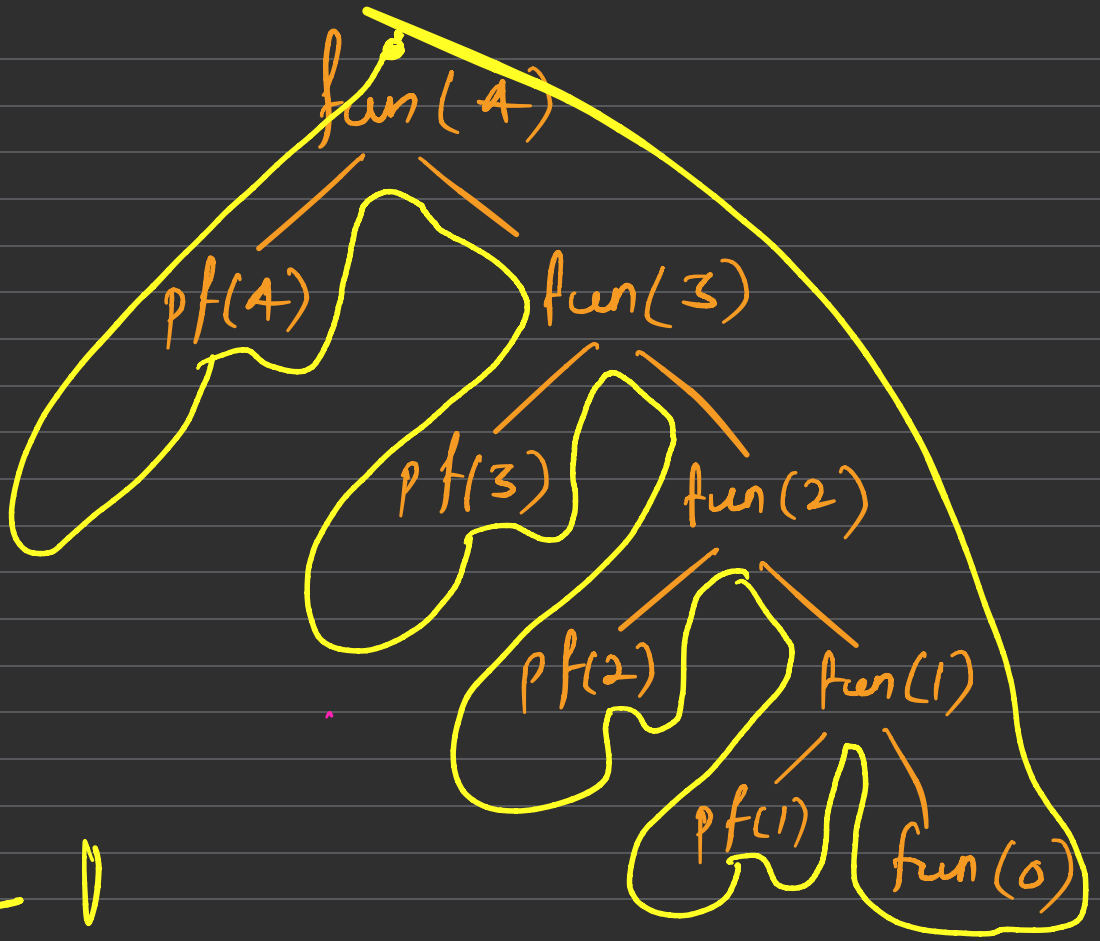
Print Numbers from N to 1



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

Find the Sum of all the numbers from 1 to N

Functional Recursion

Find the Sum of all the numbers from 1 to N

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

```
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
}
```

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

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