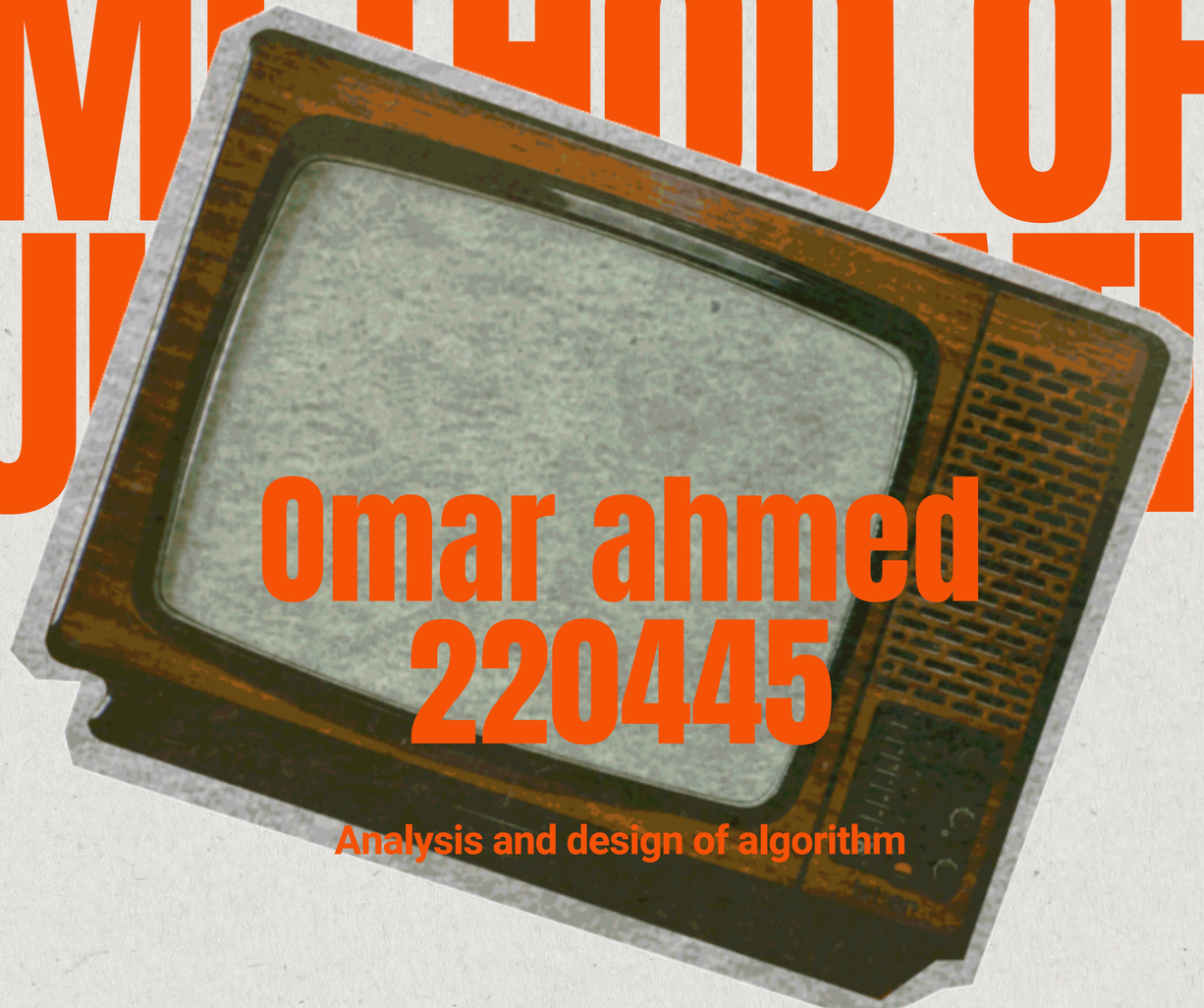
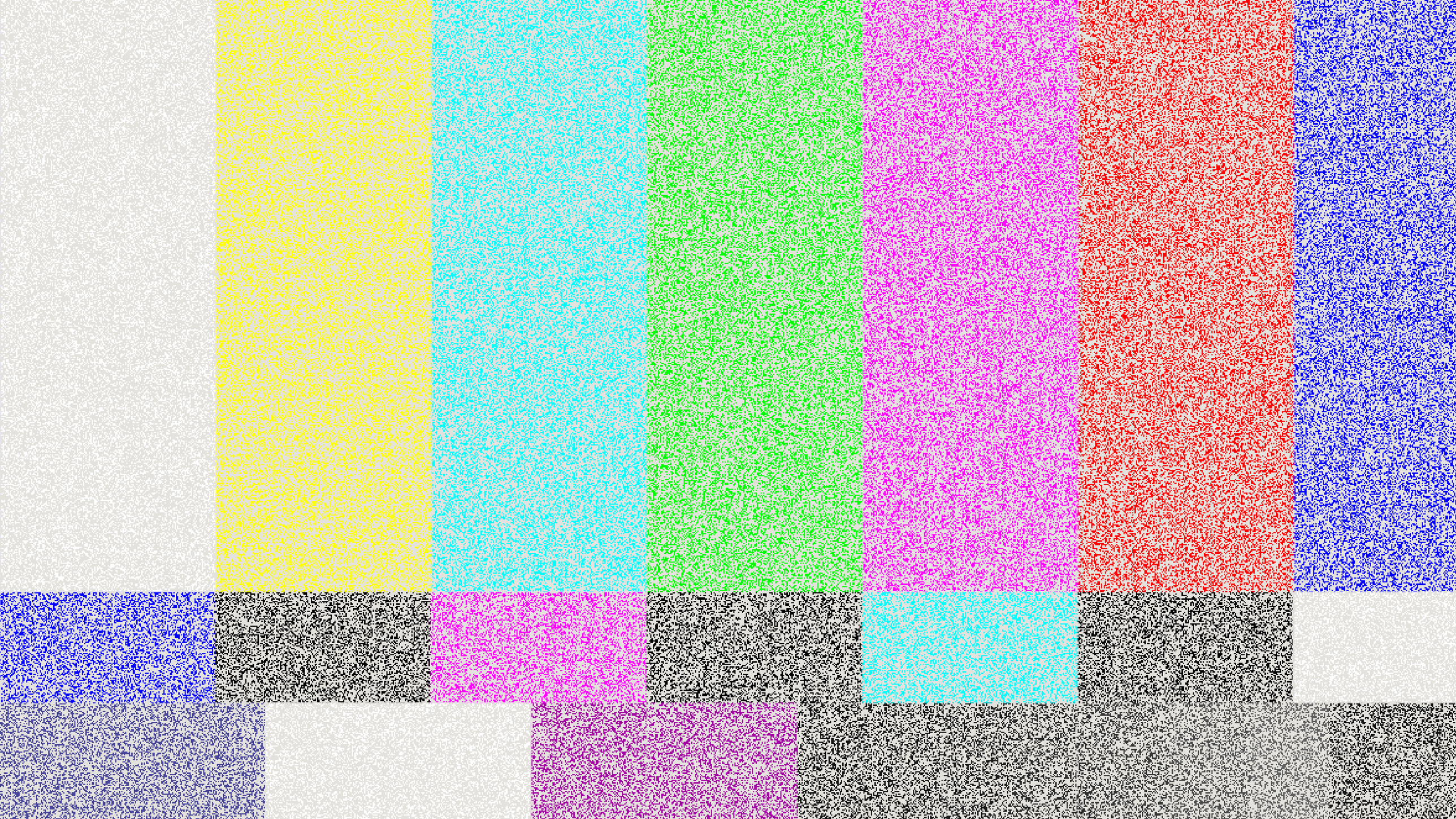
RUSSIAN PEASANT METHOD OF MULTIPLICATION



Omar ahmed
220445

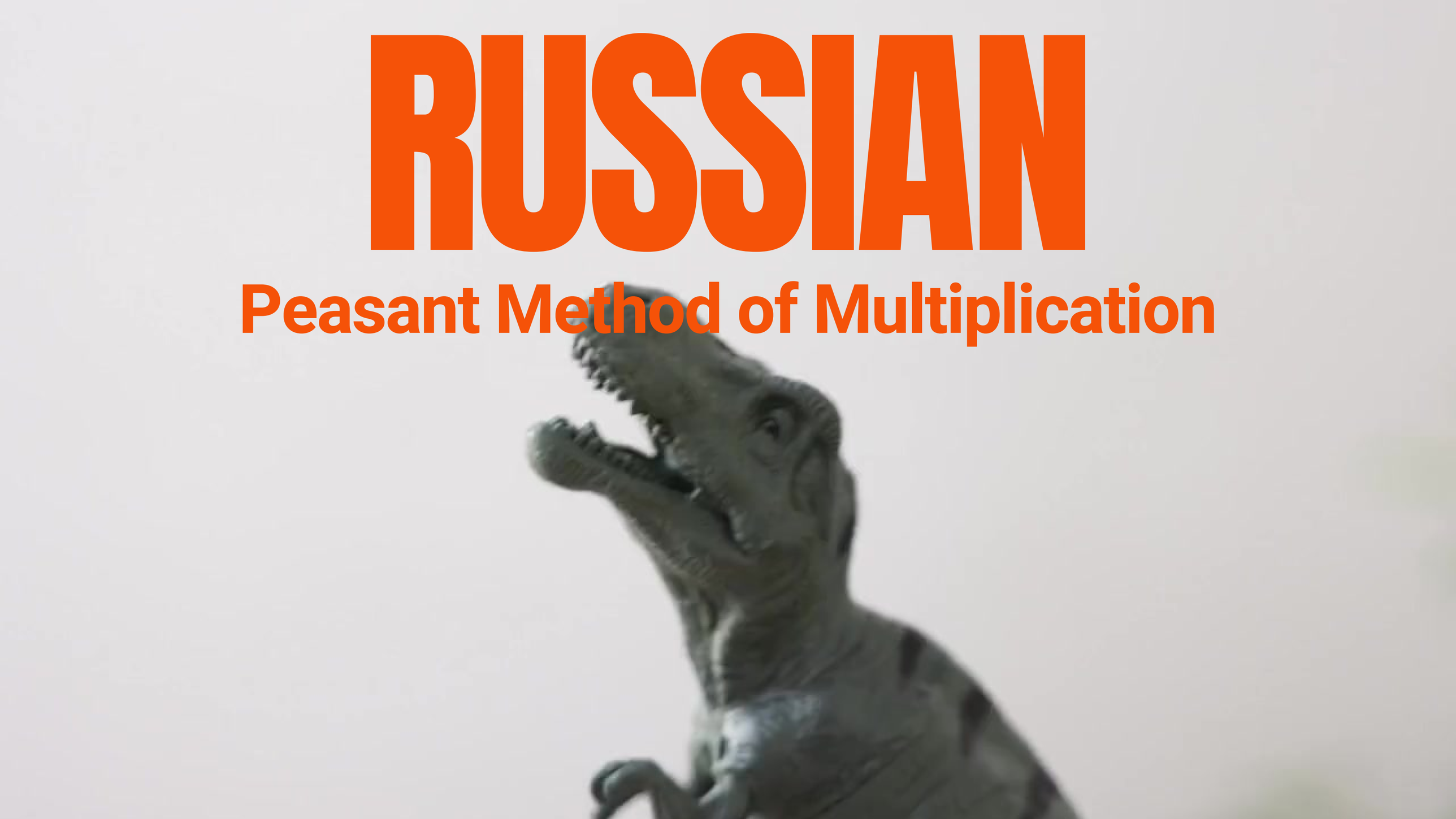
Analysis and design of algorithm




HOW TO SOLVE

RUSSIAN

Peasant Method of Multiplication



Lets assume that we need to solve this mathematical method

24 x 16

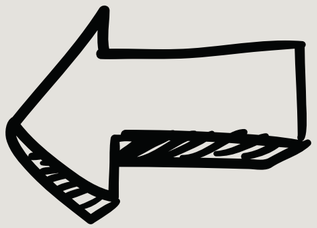
we will divide the equation to two parts

24 x 16

half

we will divide the nymber/2

24
/2 12
/2 6
/2 3
/2 1



stop when reaching 1

double

we will multiply the number by x2

16
x2 32
x2 64
x2 128
x2 256

remove the row for all the even numbers in left side

half		double
24	even	16
12	even	32
6	even	64
3		128
1		256

REMOVED

half		double
24	even	16
12	even	32
6	even	64
3		128
1		256

THEN WE WILL ADD THE NUMBERS ON THE RIGHT SIDE

HALF	DOUBLE
3	128
1	256

ADD

128

\+ 256

384

THE FINAL ANSWER

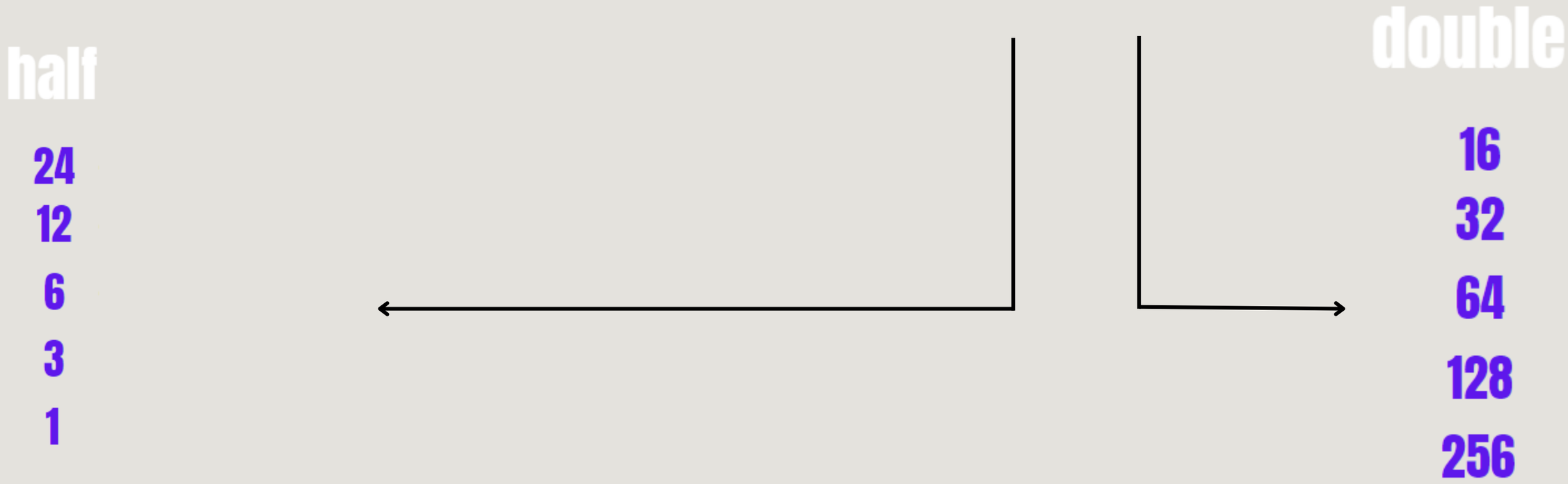
128
+ 256

24X16= → 384

So How can we Implement this with code

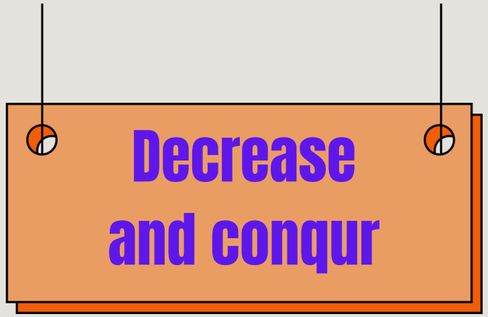
- Let's name the function **russian_peasant_multiplication.**

- Inside the function, initialize two variables to store the input numbers and a variable to store the result. Let's name the variables **num1**, **num2**
- so the base case is when **num1** reaches 1



- Check if num1 is odd (i.e., num1 % 2 == 1). If it is, add **num2** to **Result**.

- Divide num1 by **2** (i.e., num1 //= 2).
- Double the value of num2 (i.e., num2 *= 2).




```python
def russian_peasant_multiplication(num1, num2):
    if num1 == 1:
        return num2

    if num1 % 2 == 1:
        return num2 + russian_peasant_multiplication(num1 // 2, num2 * 2)
    else:
        return russian_peasant_multiplication(num1 // 2, num2 * 2)

```

we are using **Recursion**

to implement **decrease and conqur**

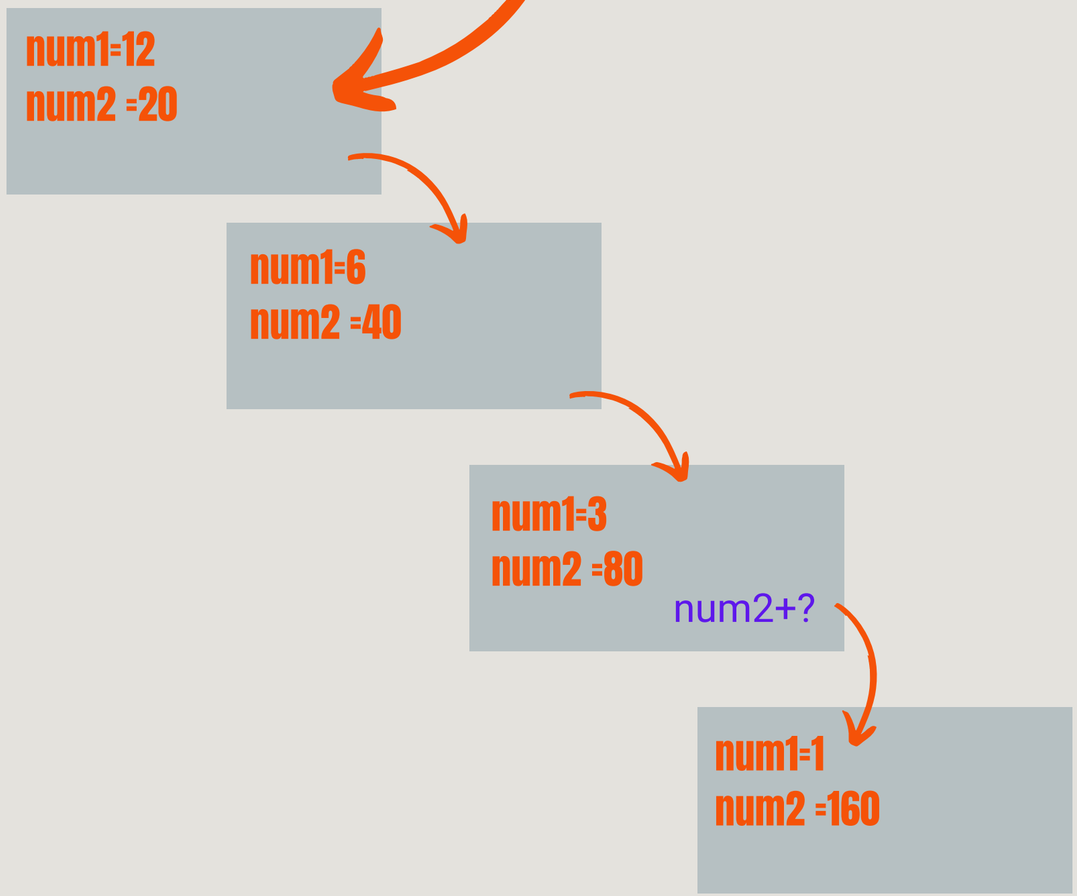
to make the problem into simpler sub problems

ex.

num1=12

num2 =20

X = ?



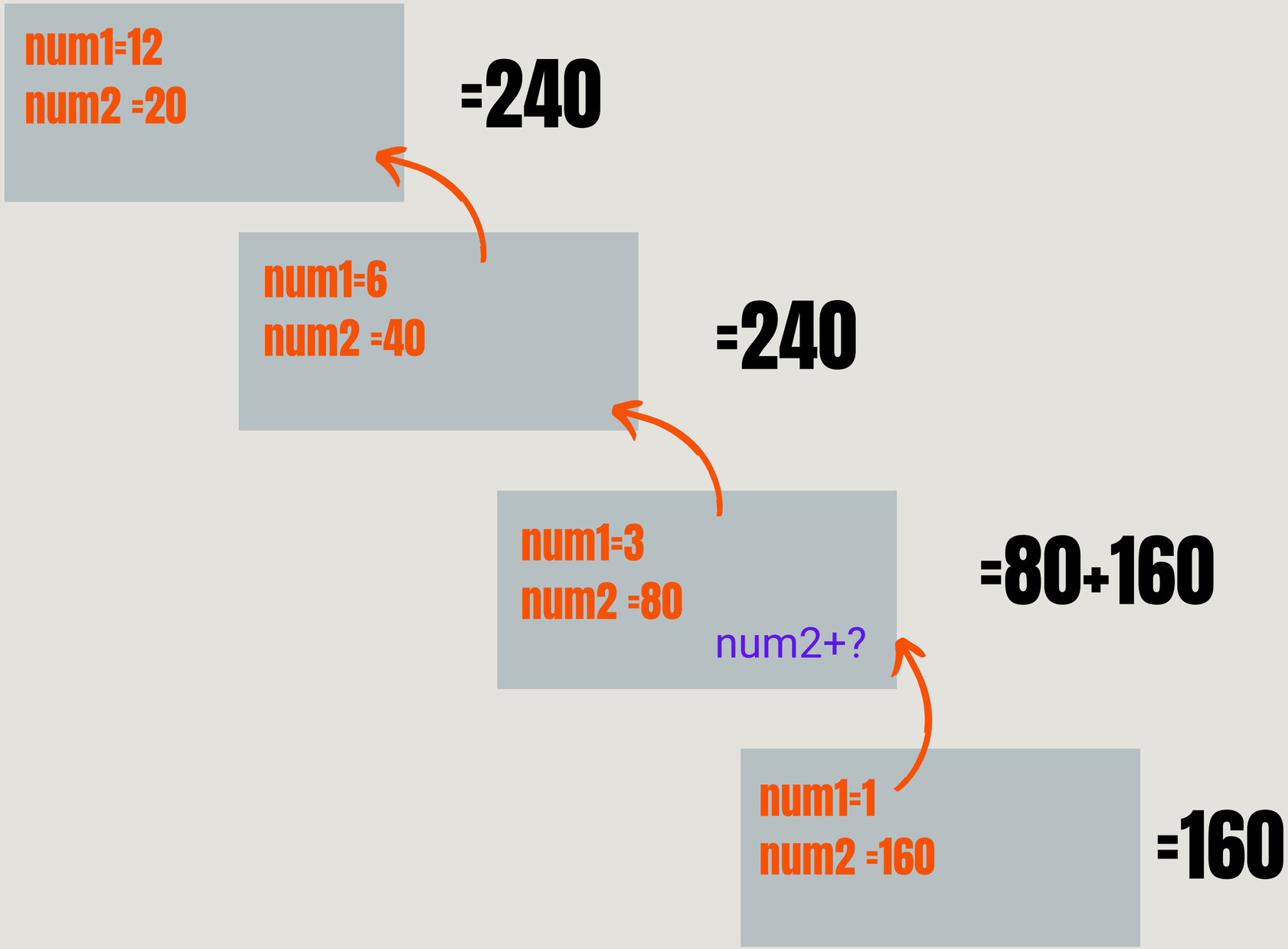
how the code works

```python
def russian_peasant_multiplication(num1, num2):
    if num1 == 1:
        return num2

    if num1 % 2 == 1:
        return num2 + russian_peasant_multiplication(num1 // 2, num2 * 2)
    else:
        return russian_peasant_multiplication(num1 // 2, num2 * 2)

```

Recursion


num1=12
num2 =20
=240
num1=6
num2 =40
=240
num1=3
num2 =80
num2+?
=80+160
num1=1
num2 =160
=160

TIME COMPLEXITY

Our Recursion code iterate depending on **num1**
the base case **num1 =1**

So our recurrence Relation

$$t(num1)=t(num1/2)+1 \qquad t(1)=0$$

TIME COMPLEXITY

t(num1)=t(num1/2)+1

t(num1/4)+2

t(num1/8)+3

t(num1/16)+4

t(1)=0

t(num1/2)= t(num1/4)+1

t(num1/4)=t(num1/8)+1

t(num1/8)=t(num1/16)+1

so there is a pattren

so we generalize $t(num1)=t(num1/2^{k})+k$

To Make t(num1/2)=1

$$num1=2^{k}$$

$$k=\log(num1)$$

$$t(num1)=t(num1/2^{\log/(num1)})+\log(num1)$$

$t(num1)=t(num1/2^{\log/(num1)})+\log(num1)$

$t(num1)=t(1)+\log(num1)$

$t(num1)=0+\log(num1)$

$t(num1)=\log(num1)$

so time complexity = O(log(num1))

THANKS FOR WATCHING

omar Ahmed