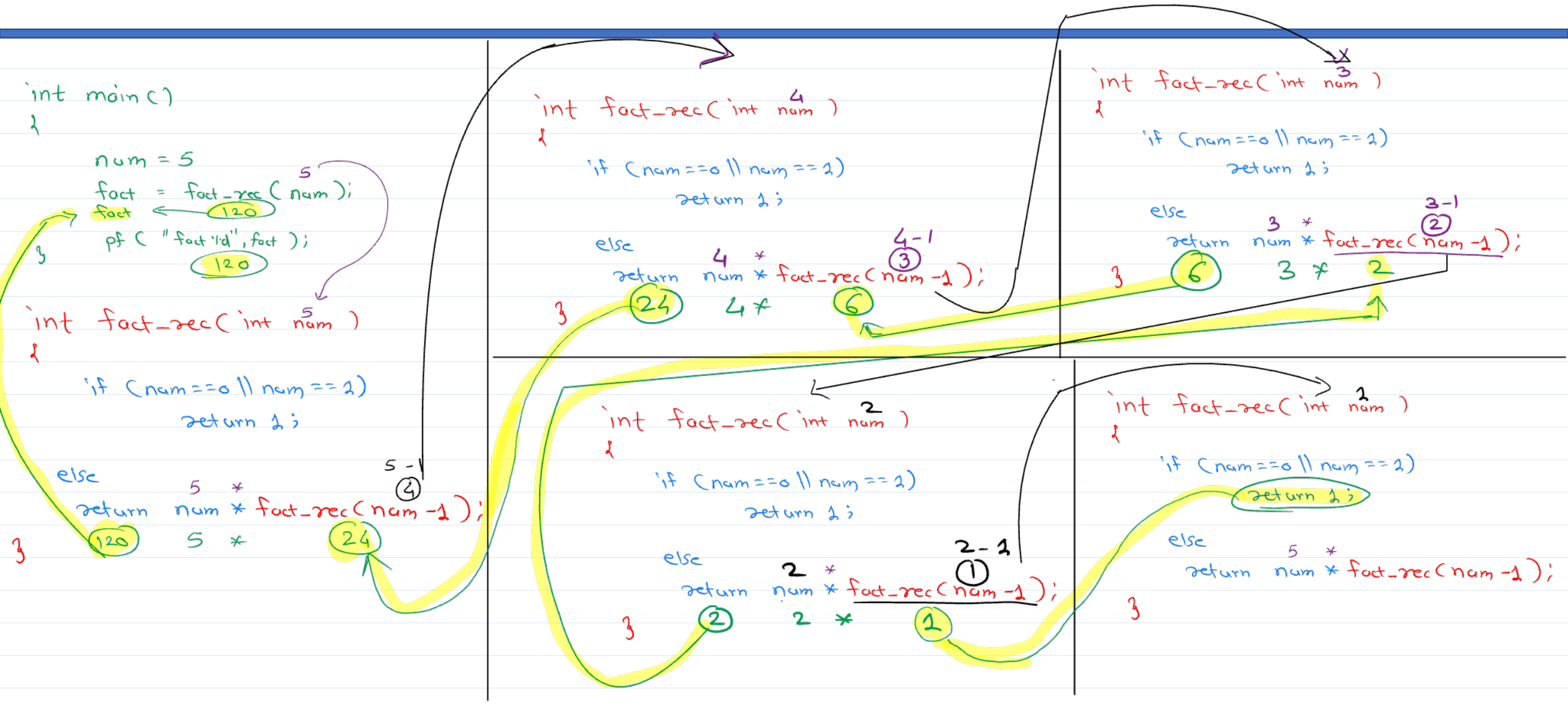
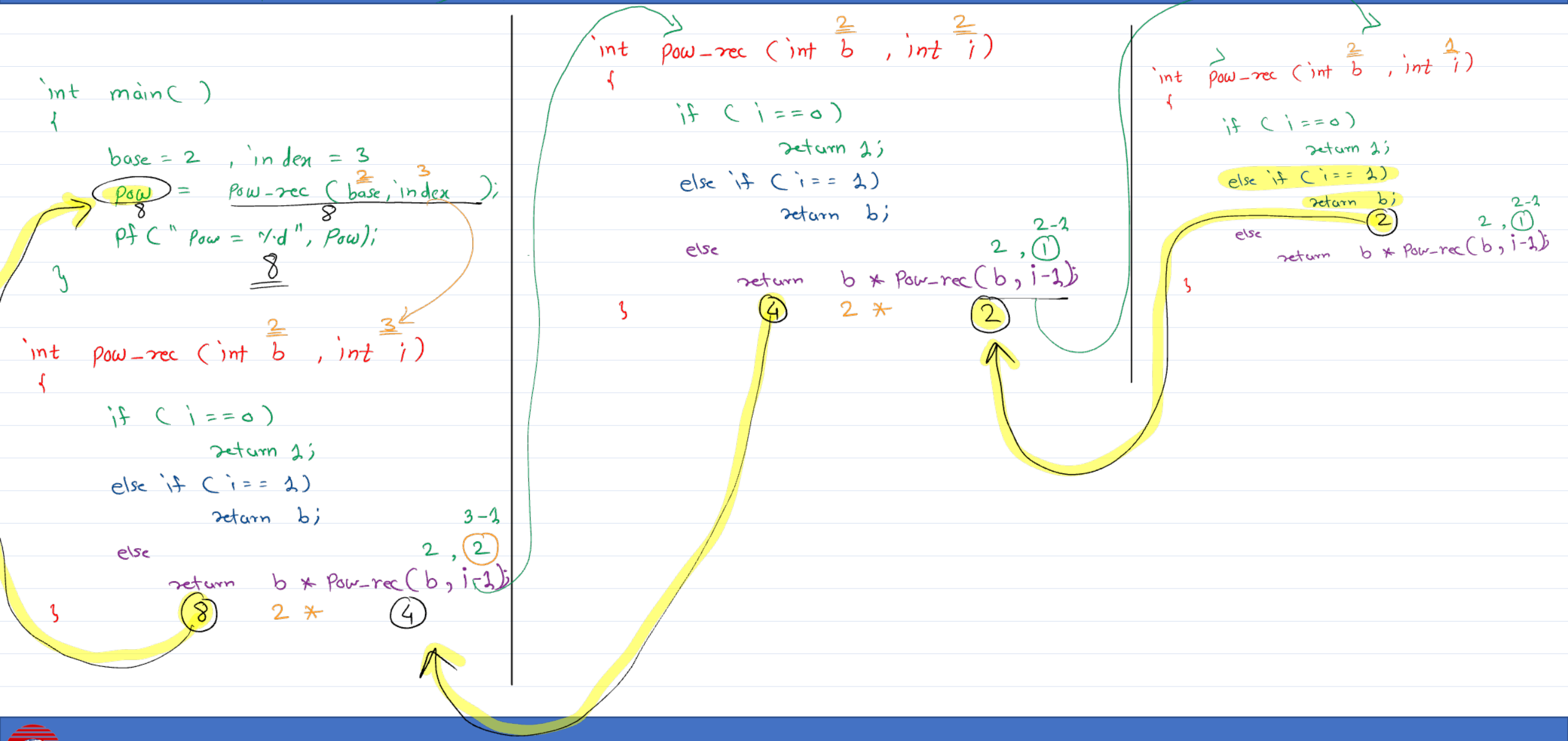
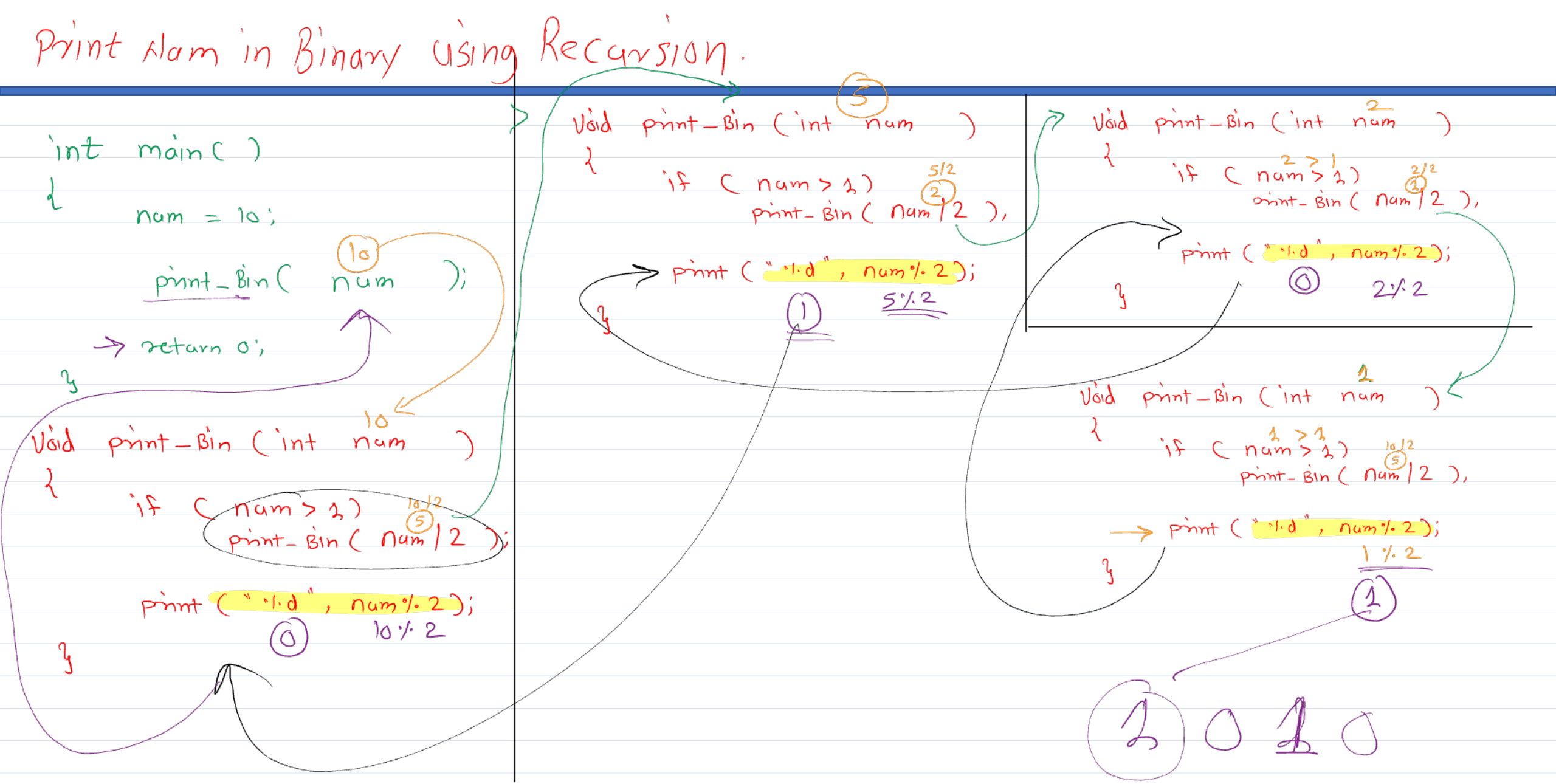
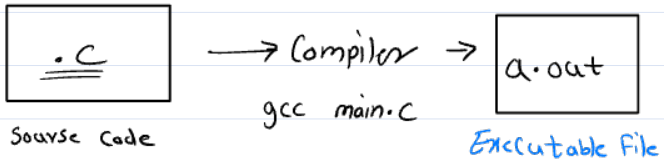

int main()
{
num = 5
fact = fact_rec(num);
fact
120
pf ("fact %d", fact);
120
}
int fact_rec(int num)
{
if (num==0 || num==2)
return 1;
else
return num * fact_rec(num -1);
5 *
5 - 1
4
120
24
int fact_rec(int num)
{
if (num==0 || num==2)
return 1;
else
return num * fact_rec(num -1);
4 *
4 - 1
3
24
6
int fact_rec(int num)
{
if (num==0 || num==2)
return 1;
else
return num * fact_rec(num -1);
3 *
3 - 1
2
6
2
int fact_rec(int num)
{
if (num==0 || num==2)
return 1;
else
return num * fact_rec(num -1);
2 *
2 - 1
1
2
1
int fact_rec(int num)
{
if (num==0 || num==2)
return 1;
else
return num * fact_rec(num -1);
1

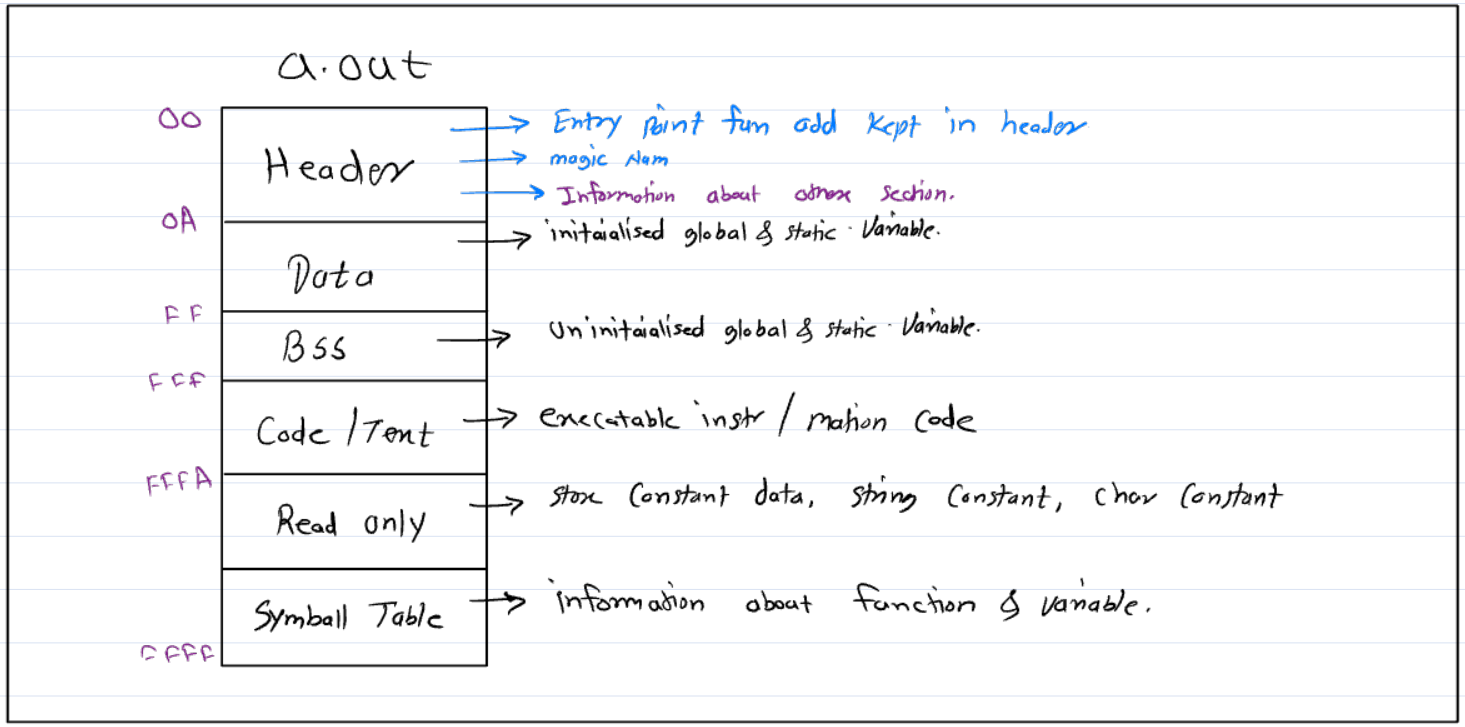
*Calcalate Power using Recursion.

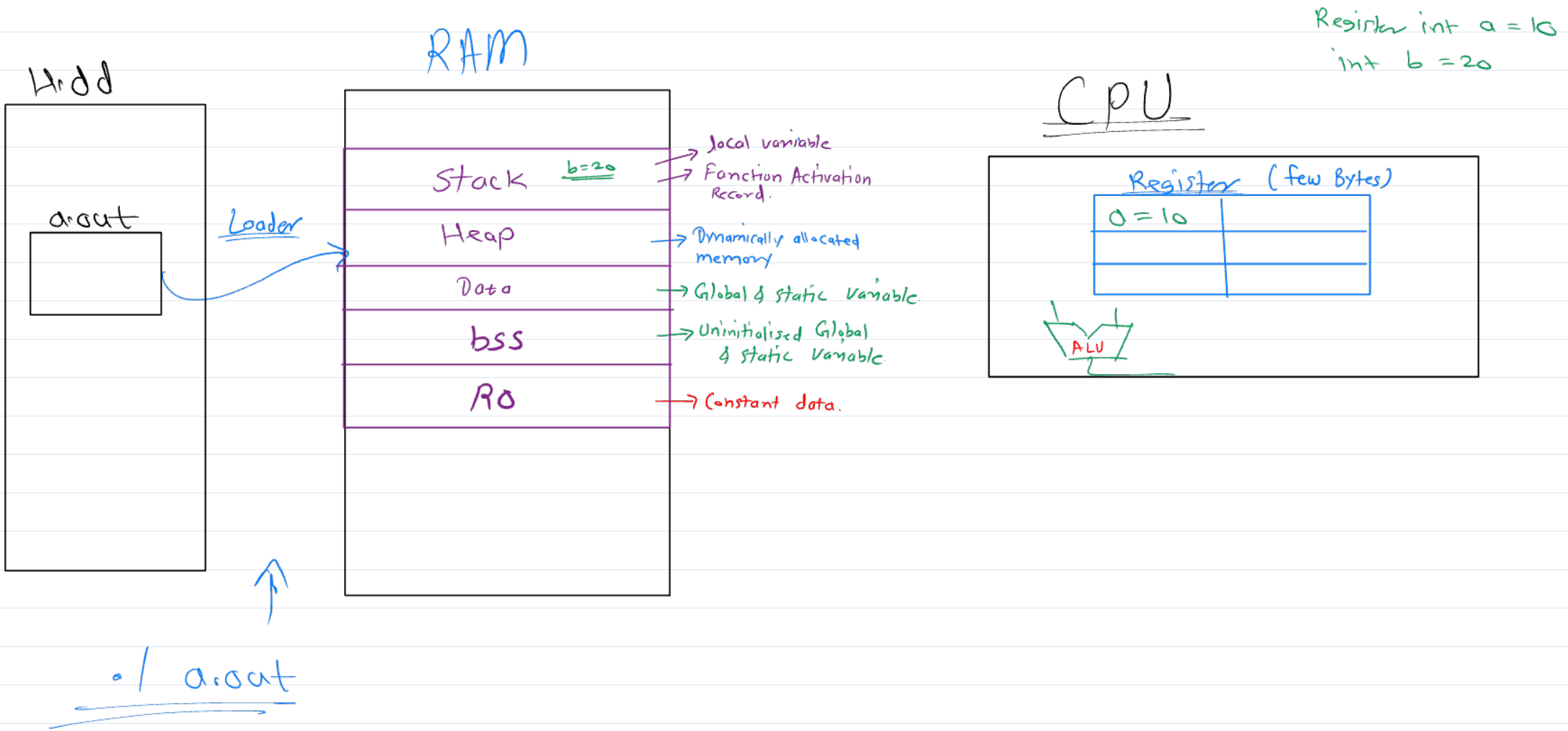


Print Num in Binary using Recursion.




.c
Compiler
gcc main.c
a.out
Sourse Code
Executable File

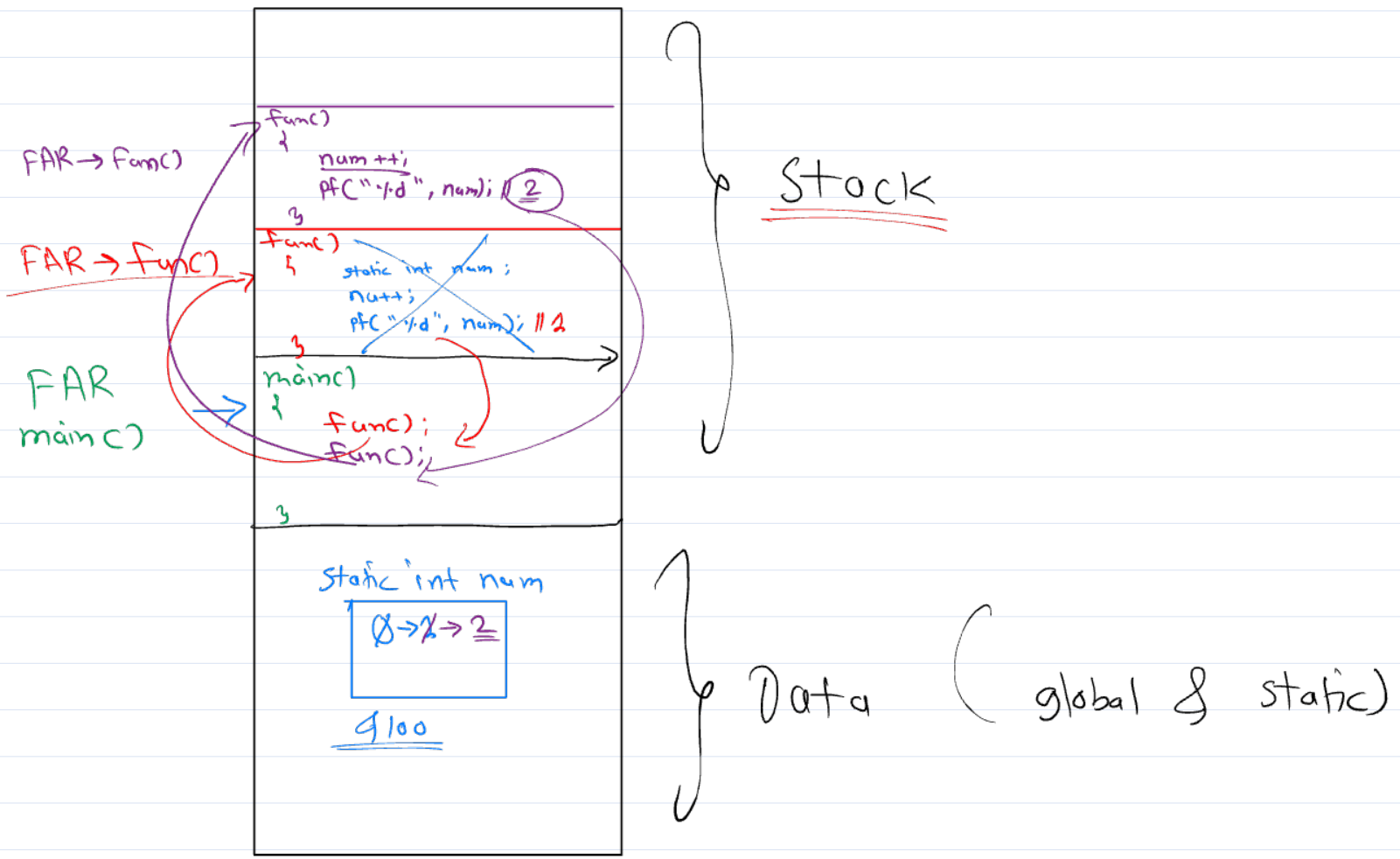

a.out
00
0A
FF
FFF
FFFA
0FFF
Header
Data
Bss
Code / Text
Read only
Symball Table
Entry point fun add kept in header
magic Num
Information about adress section.
initialised global & static Variable.
Uninitialised global & static Variable.
executable instr / mation code
Stor Constant data, String Constant, Char Constant
information about function & variable.

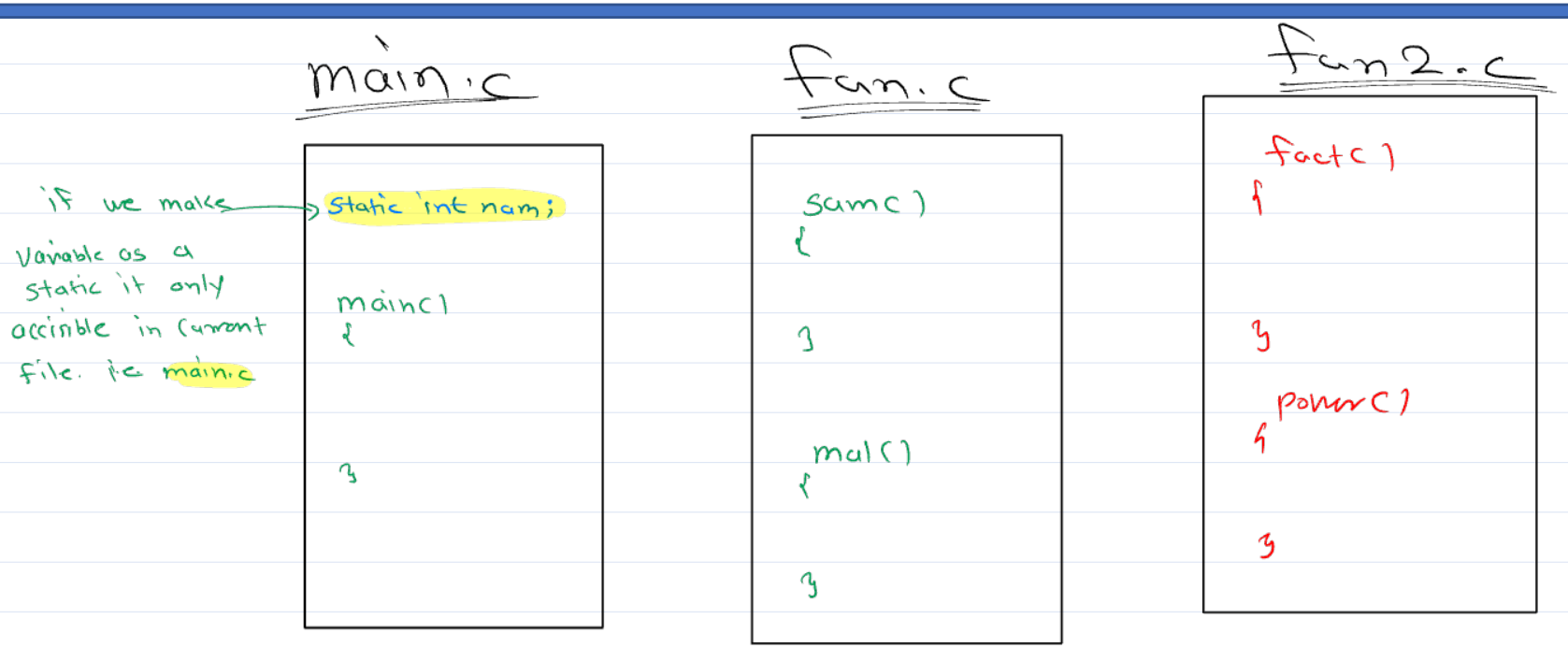

Hdd
a.out
Loader
RAM
Stack
b=20
Heap
Data
bss
RO
local variable
Function Activation Record.
Dynamically allocated memory
Global & static variable
Uninitialised Global & static variable
Constant data.
./ a.out
Register int a = 10
int b = 20
CPU
Register (few Bytes)
a = 10
ALU

process

```
Void func()
{
    static int Num;
    num++;
    pf ("%d", Num);
}

int main()
{
    func();
    func();
    func();
}
```




main.c
fun.c
fun2.c
if we make
static int nam;
variable as a
static it only
accisible in current
file. i.e main.c
main()
{
}
sam()
{
}
mal()
{
}
fact()
{
}
power()
{
}