(1) Bitwise &

```
   10     =>     1 0 1 0
&   5     =>  &  0 1 0 1
------        ------------
    0            0 0 0 0
```

```
   8      =>     0 1 0 0 0
& 15      =>  &  0 1 1 1 1
------        --------------
   8             0 1 0 0 0  ->
```

(3) Bitwise OR.

```
   10     =>     1 0 1 0
|   5     =>  |  0 1 0 1
------        ------------
   15            1 1 1 1 => 15
```

```
   8      =>     0 1 0 0 0
| 15      =>  |  0 1 1 1 1
------        --------------
   15            0 1 1 1 1 =>
                  (15)
```

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

ExOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

③ Bitwise ExOR (^)

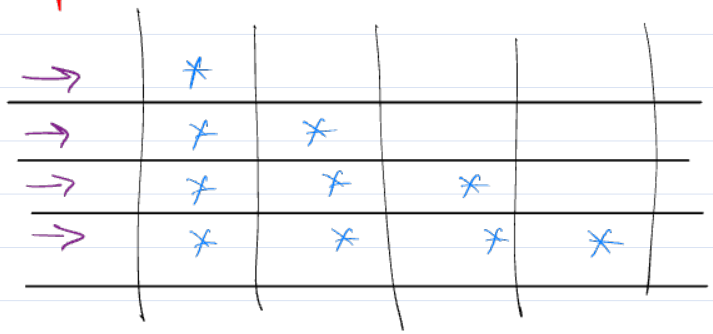
10 => 1 0 1 0
^ 5 => ^ 0 1 0 1
15 — 1 1 1 1 = 15

8 => 0 1 0 0 0
^ 15 => ^ 0 1 1 1 1
7 — 0 1 1 1 => 7

// 2 = 0000 0010
// ~2 = 1111 1101
// 2's = 1
1111 1110 → -2

(Row)

Row



```
for( i=1 ; i<= Row ; i++)
{
    for( j=1 ; j<= i ; j++)
    {
        printf( " * " );
    }
    printf( \n );
}
```

i

1 →	*	*	*	*
2 →	*	*	*	
3 →	*	*		
4 →	*			

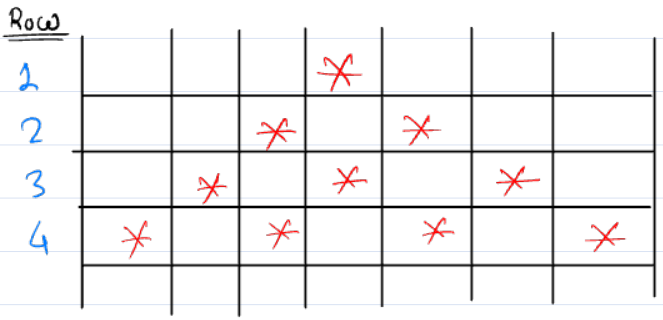
~~5 <= 4~~

4 <= 4

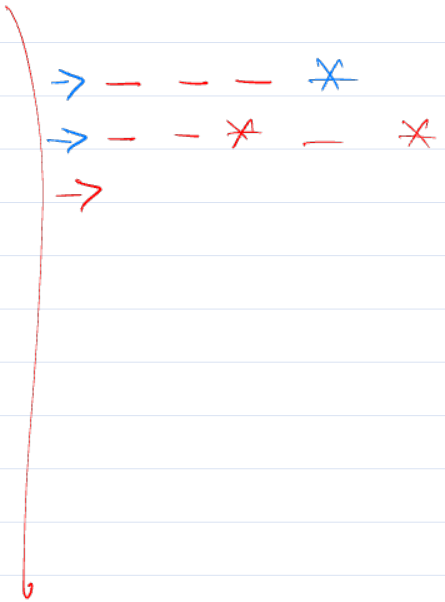
5

```
for( i=1 ; i <= Row ; i++ )
{
        j=4           3 >= 4
    for ( j=Row ; j >= i ; j-- )
    {
        printf(" * ");
    }
    printf( \n );
}
```

```
* * * *
* * *
* *
*
```




```
Row = 4

for( i=2 ; i<= Row ; i++ )
{
    for( j=1 ; j<= Row-i ; j++ )
    {
        printf( "  " );
    }
    for( k=1 ; k<= i ; k++ )
    {
        printf( "* " );
    }
    printf( "\n" );
}
```



Row = 4 , Col = 6

i						
1 →	*	*	*	*	*	*
2 →	*					*
3 →	*					*
4 →	*	*	*	*	*	*
	1	2	3	4	5	6 <= j

2 → 2 ⟶ 5

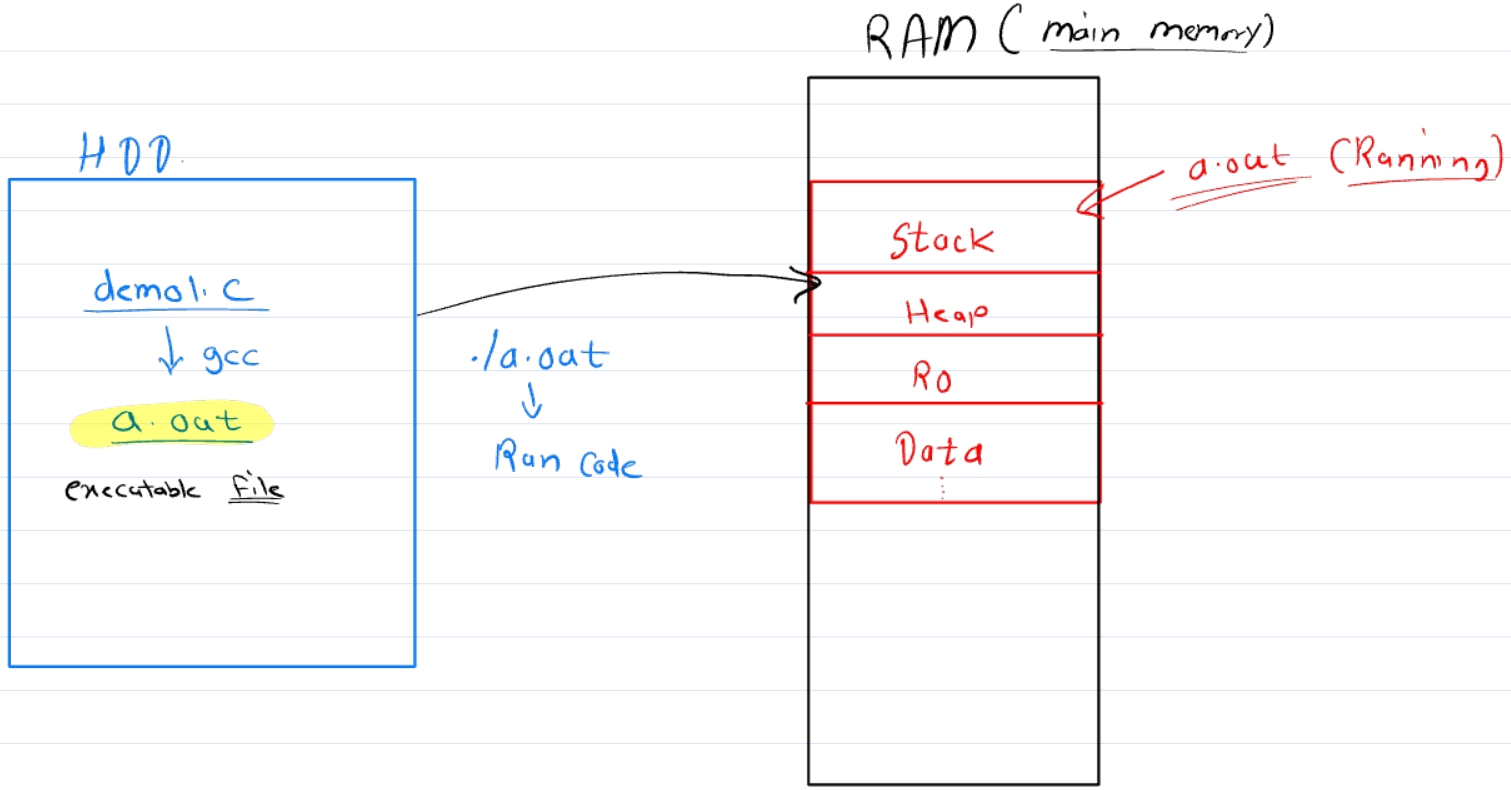
3 → 2 ⟶ 5

* * * * * *

* _ _ _ _ *

(1) 2 <= 4

```
for( i = 1 ; i <= Row ; i++ )
{
        1     6 <= 6
    for( j = 1 ; j <= col ; j++ )
    {
        if ( j == 1 || i == Row || i == 1 || j == Col )
        {
            pf ( " * " )
        }
        else
        {
            pf ( "  _  " );
        }
    }
    printf ( \n );
}
```


RAM (main memory)
HDD
demo1.c
gcc
a.out
executable File
./a.out
Run code
a.out (Running)
Stack
Heap
RO
Data

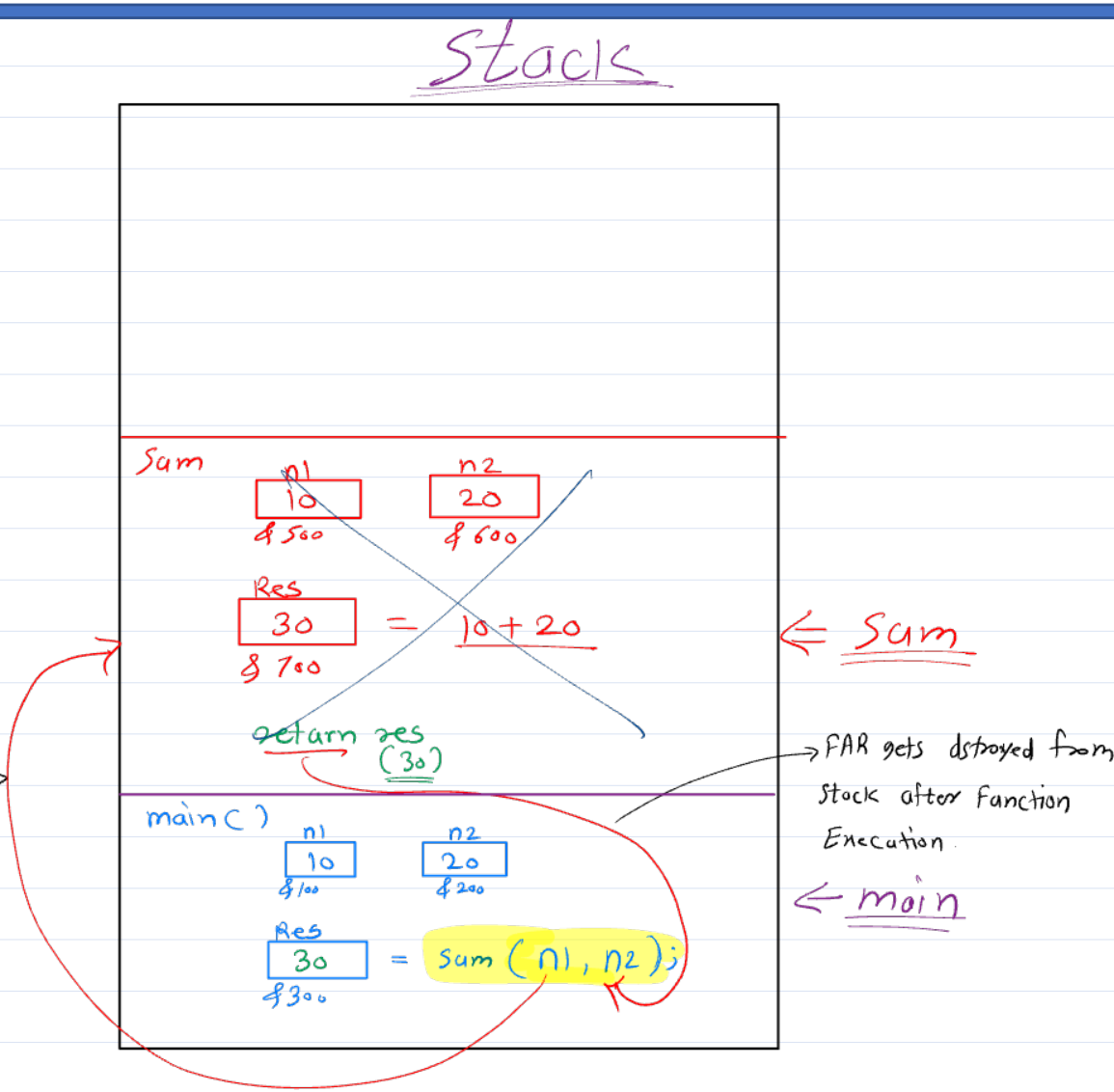
* Function Execution

```
int sum ( int n1 , int n2 );

int main ( )
{
    int n1 = 10 , n2 = 20; // local variable

    int res = sum ( n1 , n2 )

    printf ( " %d " , Res );
}

int sum ( int n1 , int n2 )
{
    int res;

    res = n1 + n2;

    return res;
}
```

FAR get Create on stack after function call

1. FAR
↓
main

2. FAR
↓
sum



```
int main( )
{
    int nam = 5;
    rec_print( nam );
    return 0;
}

Void rec_print ( int n1)
{
    if ( n1 < 1 )
        return ;

    rec_print ( n - 1 )
    printf( "%d" , n1 );
}
```