<10-09-2025> Wednesday.

2 ex n

and or

& | ~ >> <<

	16	8	4	2	1
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	1	1	1

```
 1 1 1 1  => 15
 0 1 1 0  => 6
---------
and 0 1 1 0  => 6
```

```
   1 1 1 1
and 0 1 1 0
-----------
   0 1 1 0
```

```
   1 1 1 1  = 15
or 0 1 1 0  = 6
-----------
   1 1 1 1  => 15
```

Left shift [<<]: This shifts all the bits to left & fills with 0 on the right.

Ex: 5<<1 = 10

5<<2 = 20

5 = 0 1 0 1 0 = 10

5 = 0 1 0 1 0 0 = 16+4 = 20

Left: $n * (2 \wedge k)$

$5 * (2 \wedge 1) = 5 \times 2 = 10$

$5 * (2 \wedge 2) = 5 \times 4 = 20$

Right: $n // (2 \wedge k)$

$5 // 2 \wedge 1 = 5/2 = 2$

$5 // 2 \wedge 2 = 5/4 = 1$

13 << 2

Left: — — — right & fills with 0 by left.

1 1 0 1 << 2 → 0 0 1 1 0 1 = 3

1 1 0 1 >> 2 → 1 1 0 1 0 0 = 52

$13 // 2 \wedge 2 = 13/4 = 3$

$13 * 2 \wedge 2 = 13 * 4 = 52$

Nested if:

Syntax:

```
if (condition 1):        # outer if
    if (condition 2):    # inner if
        Statements of condition 2
    else:                # inner if else
        Statements of inner if-else
else:                    # outer if else
    Statements of outer if-else.
```

Ex: Given no is +ve, -ve or zero

```
n = int (input ("Enter a num:"))
if (n>=0):
    if (n>0):
        print ("+ve")
    else:
        print("zero")
else:
    print("-ve")
```