CS & IT ENGINEERING

Data Types -02

Introduction to C Programming

Lecture No. 4

Pankaj Sir

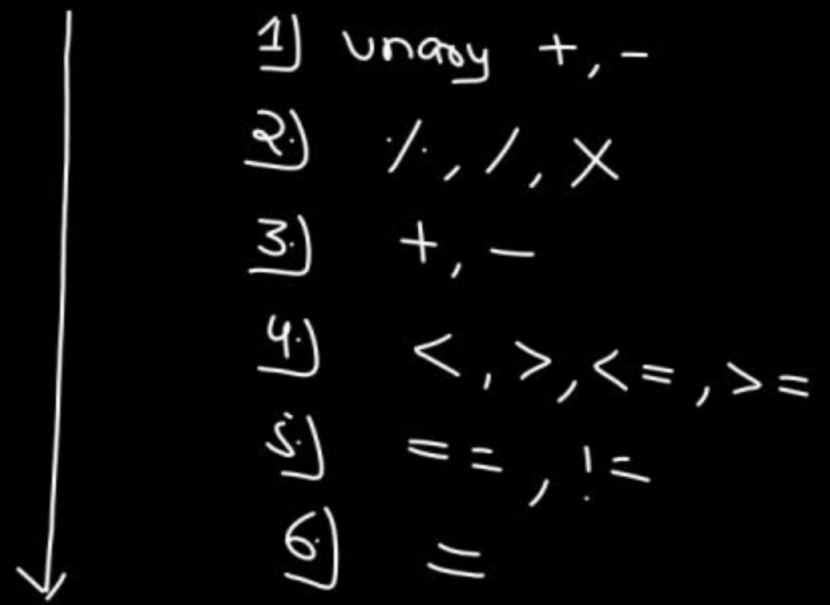
01 Logical Operators

02 Modify Operator

03 Bitwise Operators

Till now

- Assignment (=)
- Arithmetic operators (+, -, *, /, %)
- relational operators (<, >, <=, >=, ==, !=)



```
int i;
i = 3;
```
valid

value

```
int i;
i = printf("Pankaj");
```

value

↳ No. of characters/symbols printed

① Exp Evaluate

Q]
```
int a;
a = printf("%d", 4/2 + 3*2);
printf("%d", a);
```

("%d", 8)

①

4/2 + 3*2

2 + 3*2

2+6

8

81

Pankaj

```
int a;
a = printf("%d", printf("Gate 2023"));
printf("%d",a);
```

```
a = printf("%d",9);
printf("%d",a);
```
1

① printf("Gate 2023")
② value ⇒ 9
③ printf("%d",9)
④ a = 1
⑤ printf("%d",9);
1

Gate 2023 9 1

logical Operators

1) logical and (&&)
2) logical OR (||)

} binary operators

3) logical not (!) — unary operator

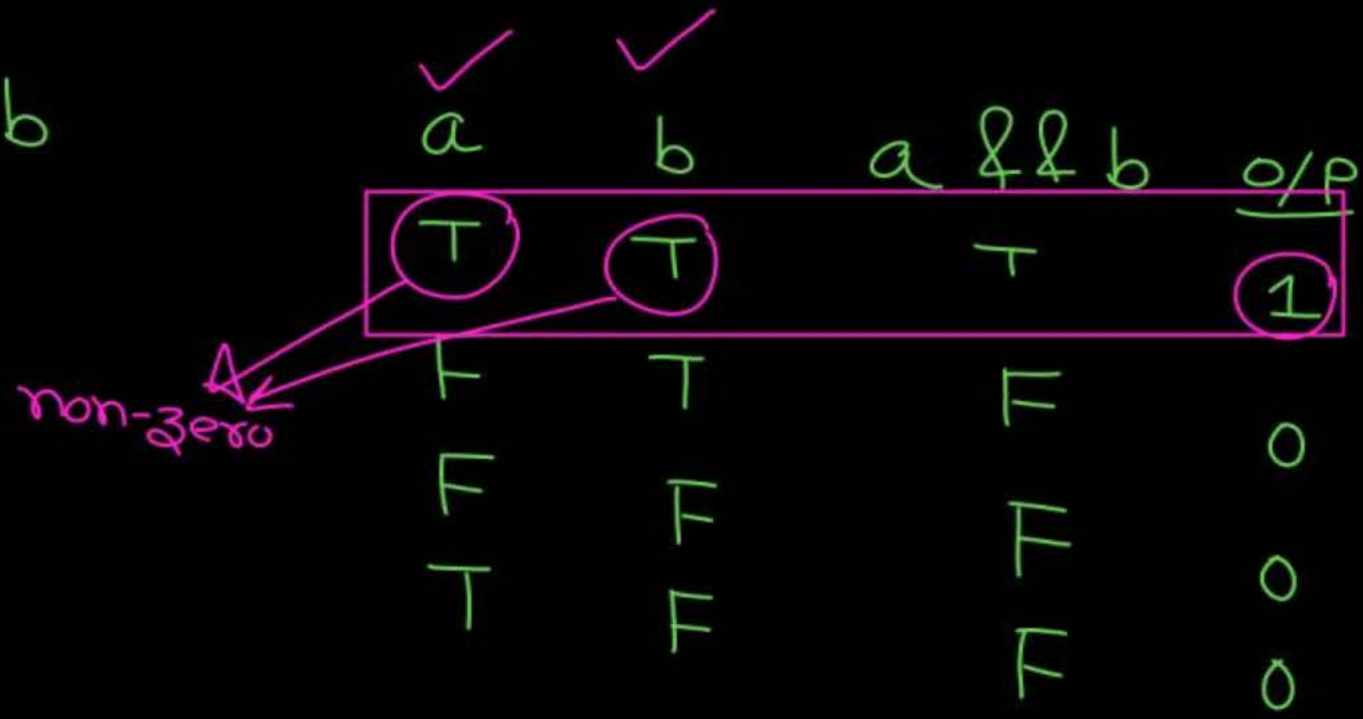
logical AND (&&)

AND → simultaneously

		result
10 < 20	True	1
20 < 10	False	0

a AND b

a && b



a × b = non-zero

Only when both a and b are non-zero.

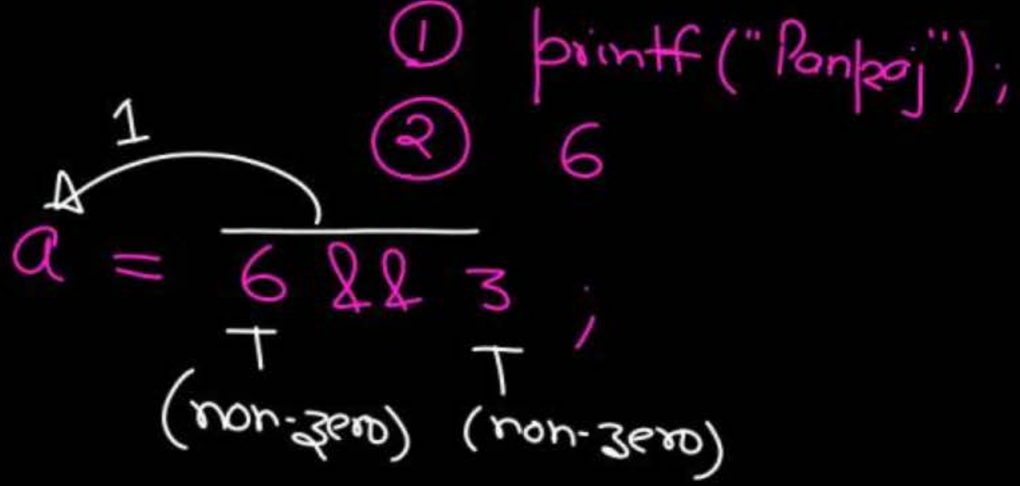
```
printf("%d", 2 && -5);    1
printf("%d", 2 && 12.5);  1
printf("%d", -12 && 0);   0
printf("%d", 0.0 && 15);  0
```

* If both operands are non-zero, the result is 1, otherwise result is 0.

```
int a;
a = printf("Pankaj") && 3;
printf("%d", a);
```

* Result of any logical operator is either 0 or 1.

1] &&
2] =



① printf("Pankaj");
② 6

Pankaj 1

logical OR (||)

OR : choice

Result is 1, if atleast one operand is non-zero (T)

Otherwise 0 (both are false/0)

a	b	a||b	o/p
T	T	T	1
T	F	T	1
F	T	T	1
F	F	F	0

```
printf("%d", 2||12);    1
printf("%d", 12||3.78); 1
printf("%d", 0||3);     1
printf("%d", 0||0.0);   0
```

logical NOT (!)

NOT (True) = False

NOT (False) = True

!(0) ⇒ ! (False)

⇒ True

O/P : 1

!(2.78) ⇒ !(non-zero)

⇒ !(True)

⇒ False

O/P 0

Q]

```
int a;
a = 0 && ☐
```

if 1st operand is 0, we need not evaluate 2nd operand

do we need to Evaluate this ?

```
a = 12 && ☐
```

we need this to answer

Short-circuit Evaluation

```
0 && 12      → X
0 && 13.5
0 && 0.0
```

```
int a;
a = 0 && printf("Gate");
printf("%d",a);
```

not Evaluated

o/p : 0

```
int a ;
a = 10 || [ ] ;
```

if first operand is non-zero, then 2nd operand never evaluated

do we need to Evaluate?

No

atleast one operand is non-zero

$\Rightarrow 1$

```
a = 4 || printf("2023");
1   =
```

×

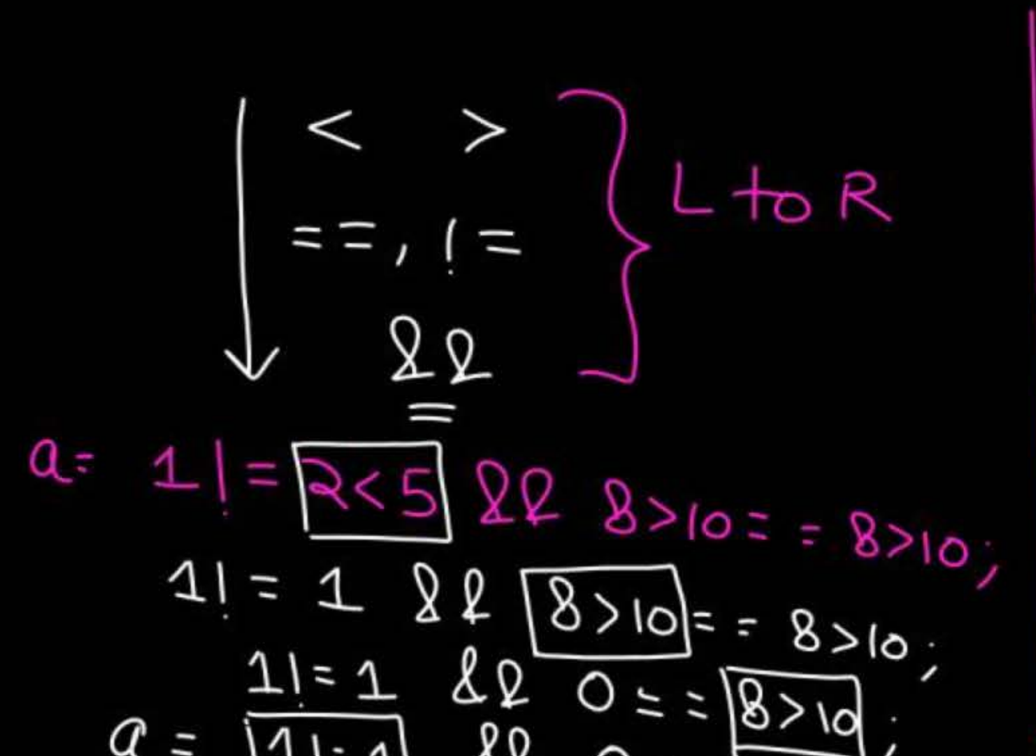
```
a = 0 || [ ]
```

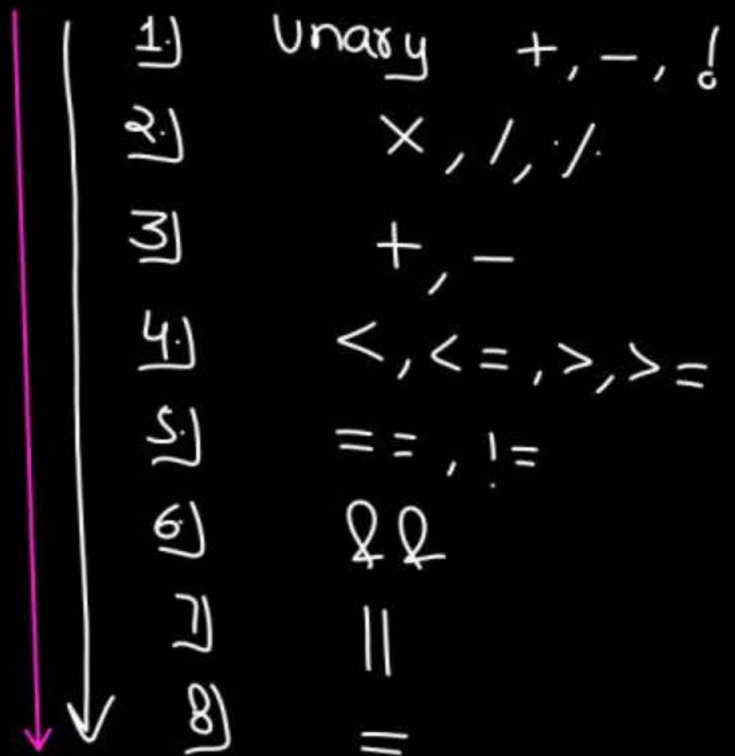
we need to Evaluate ✓

```
int a ;
a = printf("Gate") ||
                printf("2023");
printf("%d", a);
```

O/P : Gate 1

```
int a;
a = 2<5 != 2<5 && 8>10 == 8>10;
printf("%d",a);   0
```





Q1.

```
int a;
a = ((21 = 5) == 3) && ((6 > 10) == (10 > 6));
printf("%d", a);
```

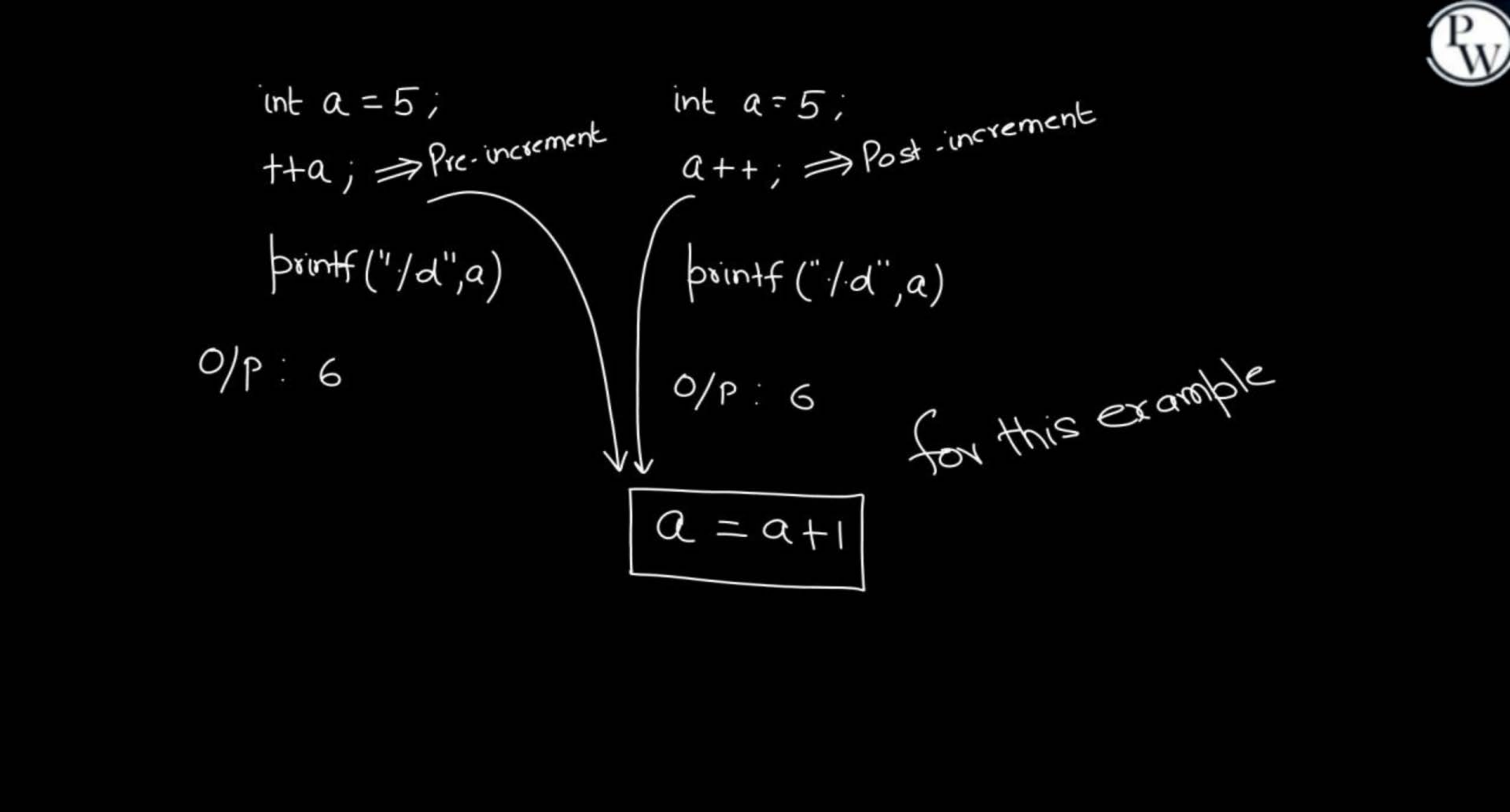
1 == 3

H.W.

Priority ⇒ $3 + 4 \times 6$

$3 + (4 \times 6)$

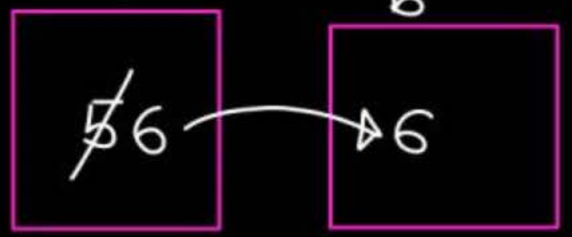
Modify Operator
Increament Operator {++}
Decrement Operator {--}
pre-increment
++Variable
post-increment
Variable++
Pre-decrement
--Variable
Post-decrement
Variable--


int a = 5;
++a; ⇒ Pre-increment
printf("%d",a)
O/P : 6
int a = 5;
a++; ⇒ Post-increment
printf("%d",a)
O/P : 6
a = a+1
for this example

Pre-increment ++a ⇒ (i) First increase
(ii) then use updated value

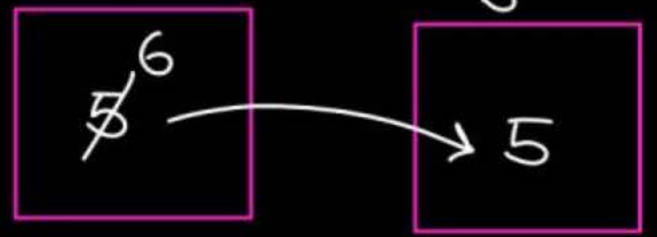
Post increment a++ (i) First use
(ii) Then increase

```
int a=5, b;
b = ++a;   (i) a = a+1;
           (ii) b = a;
printf("%d %d",a,b);
```



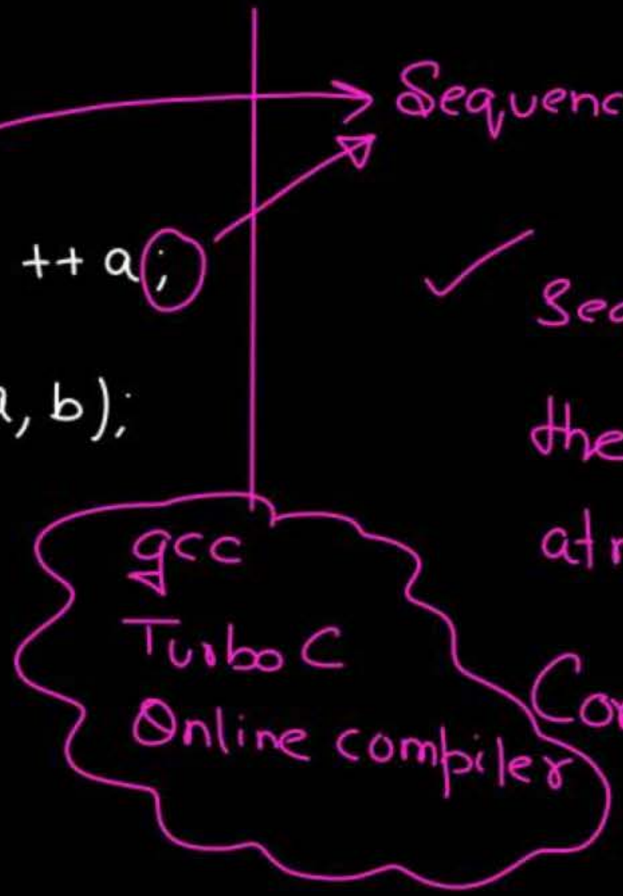
66

```
int a = 5, b;
b = a++;   (i) b = a;
           (ii) a = a+1;
printf("%d %d",a,b);
```



65

```
int a = 5, b;
b = ++a + a++ + ++a;
printf("%d %d", a, b);
```



Sequence points :

Between 2 successive seq point we can modify the value of a variable atmost one time.

Compiler dependent.

Bitwise Operators

Number System :

① Decimal number system : 10 symbols
0 - 9
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

② Binary number system : 2 symbol
0, 1

③ Octal number System : 8 symbols
0, 1, 2, 3, 4, 5, 6, 7

④ Hexadecimal number System : 16 symbol
6 + 10
0, 1, .. - 9
a, b, c, d, e, f
A, B, C, D, E, F

decimal no. system

	value
2	2

$2\boxed{4}$ — Twenty Four (24)

$10^1\ 10^0$

$= 2\times10^1 + 4\times10^0$

$= 20 + 4 = 24$

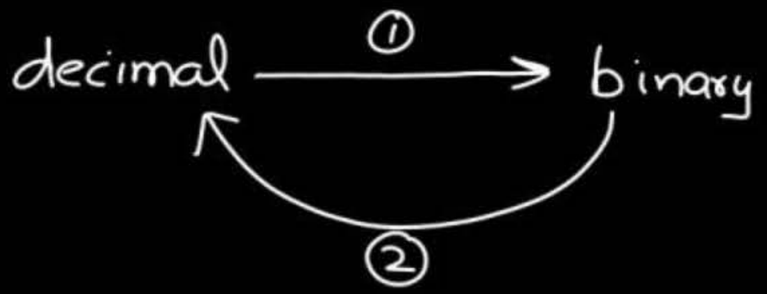
$24\boxed{3} \Rightarrow 2\times10^2 + 4\times10^1 + 3$

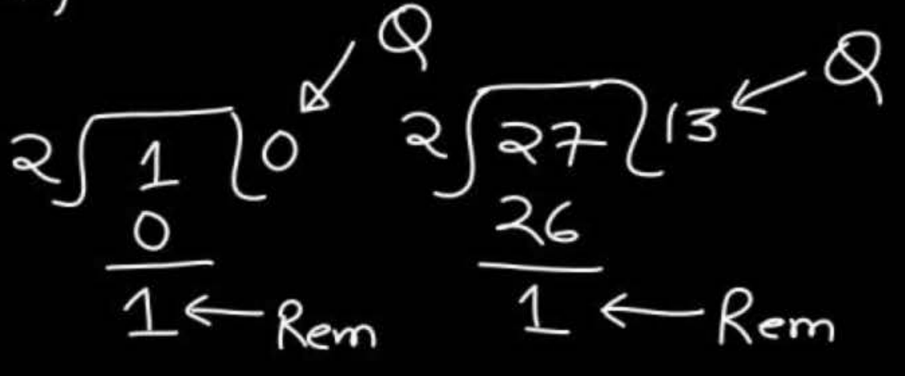
$10^2\ 10^1\ 10^0$

$= (\underline{2\times10^1 + 4\times10^0})\times10 + 3$

New value = (old value) × 10 + symbol

binary no. System (0/1)





decimal to binary: $(27)_{10} = (\quad)_2$

$(27)_{10} = (11011)_2$

2	27	Rem
2	13	1
2	6	1
2	3	0
2	1	1
	0	1

$(67)_{10} = (\quad)_2$

$(67)_{10} = (1000011)_2$

2	67	Rem
2	33	1
2	16	1
2	8	0
2	4	0
2	2	0
2	1	0
	0	1

STOP

binary to decimal

$(110011)_2 = (\quad)_{10}$

$1\ 1\ 0\ 0\ 1\ 1$
$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$= 1\times2^5 + 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$= 32 + 16 + 2 + 1$

$= 51$

$(1\,1\,0\,1\,1)_2 = (\quad)_{10}$

$1\;1\;0\;1\;1$
$2^4\;2^3\;2^2\;2^1\;2^0$

$= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$= 16 + 8 + 0 + 2 + 1$

$= 27$

$(11011)_2 = (27)_{10}$

$(1000011)_2 = (\quad)_{10}$

$1\ 0\ 0\ 0\ 0\ 1\ 1$

$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$1\times2^6+0\times2^5+0\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0$

$64+0+0+0+0+2+1$

$=(67)_{10}$

Bitwise Operators

1) Bitwise and (&)
2) Bitwise OR (|)
3) Bitwise XOR (^)
4) Bitwise NOT (~) tilde — Unary
5) Bitwise left shift (<<)
6) Bitwise Right shift (>>)

Q)

```
int a = 5, b = 17, c;
c = a & b;
printf("%d", c);
```

%d → decimal

1

binary of 5 ⇒ 0000000000000101

binary of 17 ⇒ 0000000000010001

c ⇒ 0000000000000001

1 & 0 = 0

0 & 0 = 0

0 & 1 = 0

1 & 1 = 1

2:

```
int a = 7, b = 21, c;
c = a | b;
printf("%d", c);
```

23

a ⇒ 0000000000000111

b ⇒ 0000000000010101

c ⇒ 0000000000010111

0 | 0 = 0

0 | 1 = 1

1 | 0 = 1

1 | 1 = 1

3] Bitwise XOR (^)

$$\left.\begin{matrix} 0\,\hat{}\,1 = 1 \\ 1\,\hat{}\,0 = 1 \end{matrix}\right\}$$

$$\left\{\begin{matrix} 0\,\hat{}\,0 = 0 \\ 1\,\hat{}\,1 = 0 \end{matrix}\right\}$$

a : 0000 0000 0000 0101

b : 0000 0000 0001 0001

a^b : 0000 0000 0001 0100

c

```
int a = 5 , b = 17, c ;
c = a ^ b ;
printf("%d", c);
```

20

Left shift (<<)

```
int a = 5<<1;
```

operand 1 → 5

operand 2 → 1

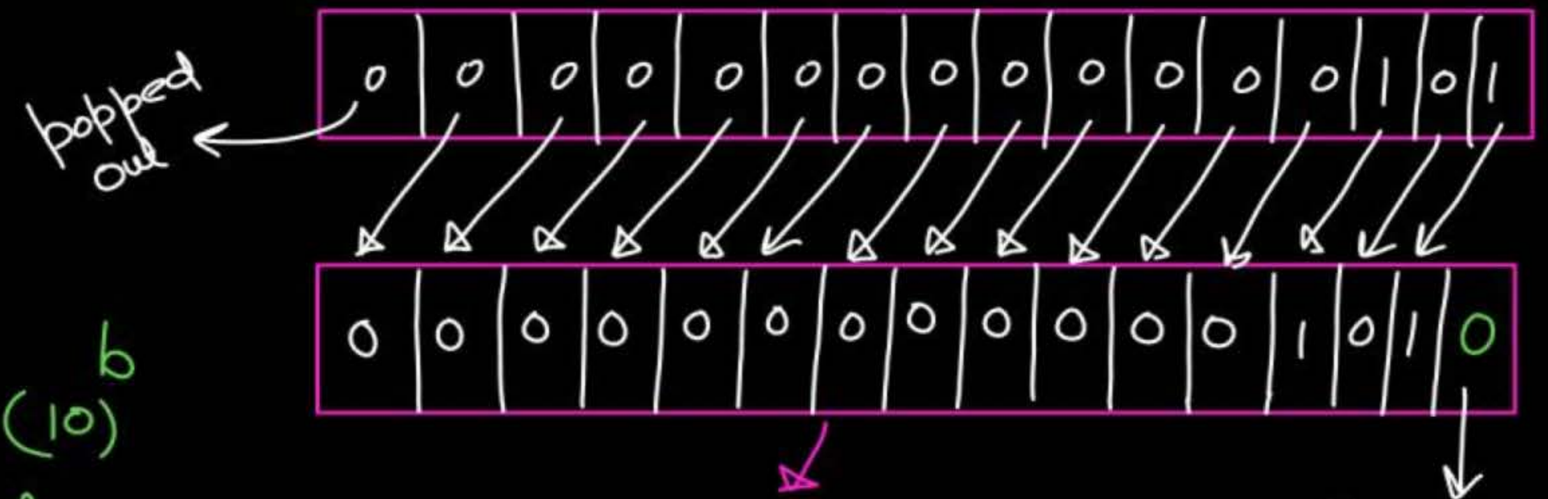
left shift 5 1 time

How many times

17 << 2

left shift 17 ⇒ 2 times

```
int a = 5, b;
b = a << 1 ;
printf("%d", b);
```



$5 \Rightarrow 10 \Rightarrow$

$(10)_{value}^{b}$

5

101

$101 \Rightarrow 1\times2^2 + 0\times2^1 + 1\times2^0$

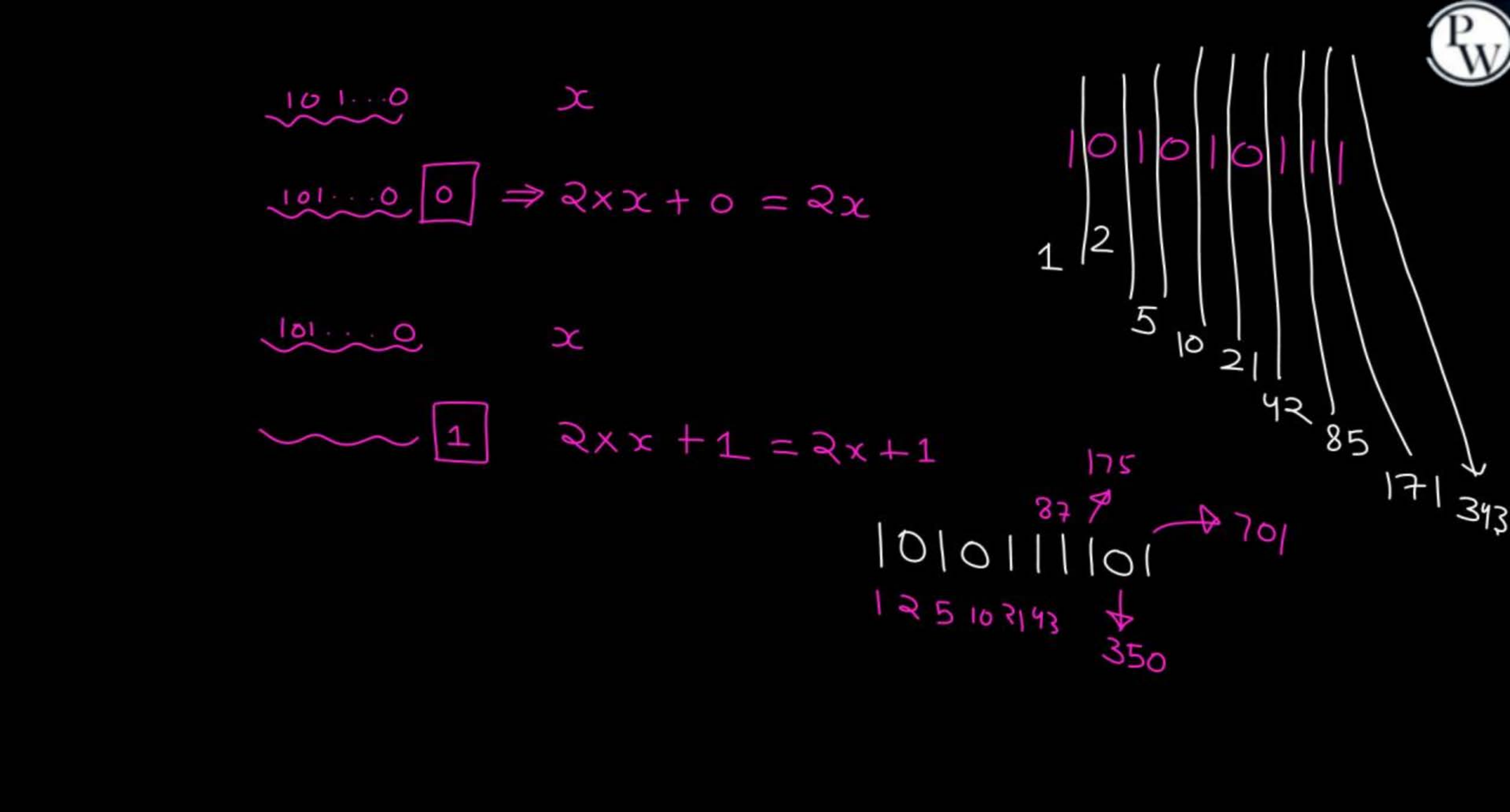
$1010 \Rightarrow 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$\Rightarrow (1\times2^2 + 0\times2^1 + 1\times2^0)\times2 + 0$

New no = old value $\times$ 2 + 0 (old no)

```
int a=5, b;
b = a<<3
printf("%d", b);
```

$\Rightarrow \quad a \times 2^3 \Rightarrow 5 \times 8 = 40$ ✓


10 1 ··· 0
x
101 ··· 0 0
⇒ 2x + 0 = 2x
101 ··· 0
x
1
2x + 1 = 2x + 1
1 0 1 0 1 0 1 1 1
1
2
5
10
21
42
85
171
343
175
87
701
1 0 1 0 1 1 1 1 0 1
1 2 5 10 21 43
350

Right shift (>>)

5 >> 1

Right shift 5
1 time

17 >> 3

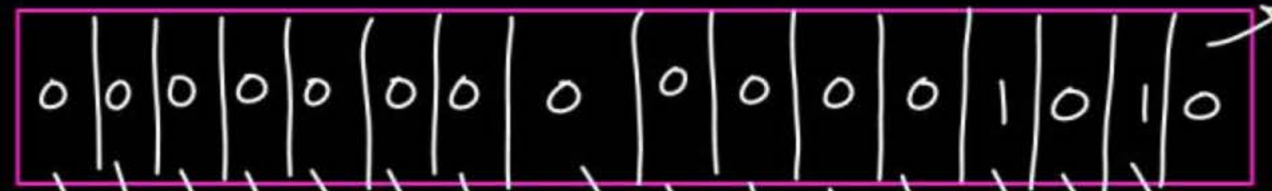
Right shift 17
3 times

Binary operator.

```
int a = 10;
int b;
b = a>>2;
printf("%d", b);
```

2 times

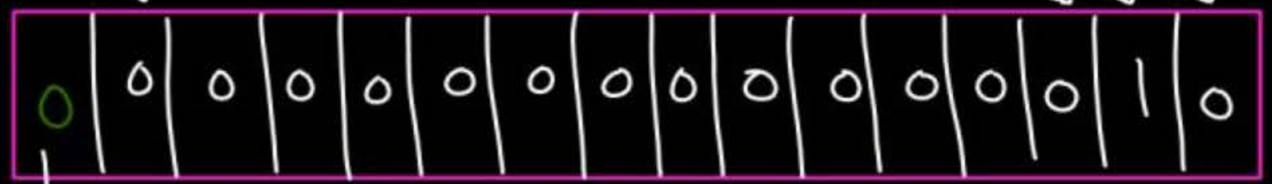
a : 10



5

$10 \Rightarrow 5 \Rightarrow 2$

Empty space is filled by 0



(2)

bitwise NOT (~)

$\sim a \Rightarrow -(a+1)$

```
int a = 5;
printf("%d", ~(a));
```

$\sim(5) = -(5+1)$

$= -6$

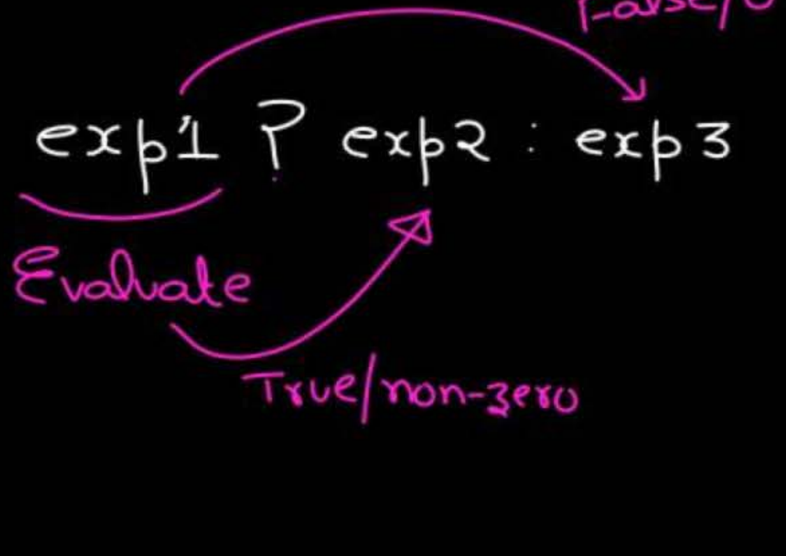
-6 ✓

```
int a = -5;
printf("%d", ~a);
```

$$\sim(-5) \Rightarrow -(-5+1)$$
$$= -(-4)$$
$$= +4$$

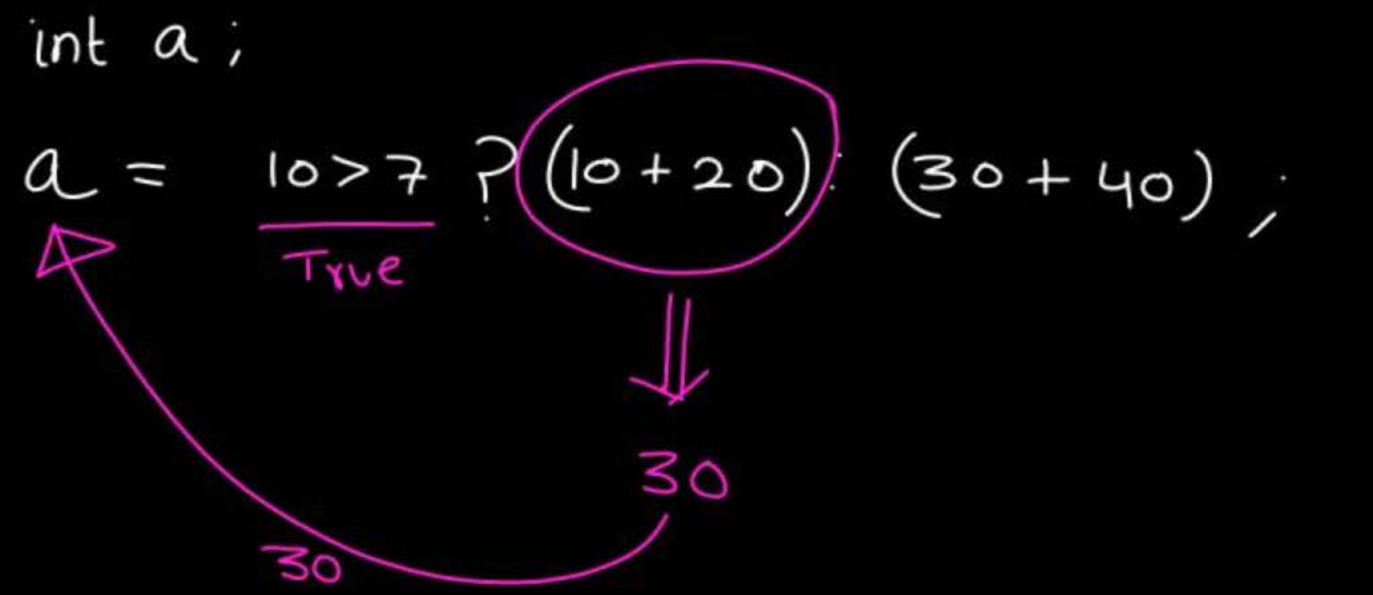
Ternary operator (? :)

3 operand



int a ; declaration statement

a = 10 ; Assignment statement


int a ;
a = 10>7 ? (10+20) : (30+40) ;
True
30
30

```
int a;
a = 9|=3>5 ? 1==8|=3 ? 10 : 20 : 30 ;
```

```
a = (9|=3>5) ? (1==8|=3 ? 10 : 20) : 30 ;
      L              M                R
```

9|=3>5

9|=0

1 ⇒ True

a = 10

```
a = (1==8|=3) ? 10 : 20 ;
      L         M    R
```

01=3

1

int a = 2, b = 2, c = 0, d = 2, e ;

e = ((a++ && b++) && c++) || d++

3 3 1 3 1

a	b	c	d	e
~~2~~ 3	~~2~~ 3	~~0~~ 1	~~2~~ 3	1

((2 && 2) && c++) || d++

(1 && c++) || d++

(1 && 0) || d++

0 || d++

e = (0 || 2)

1

```
#include<stdio.h>
void main()
{
    12;
    17.8;
}
```


THANK YOU GW
SOLDIERS !