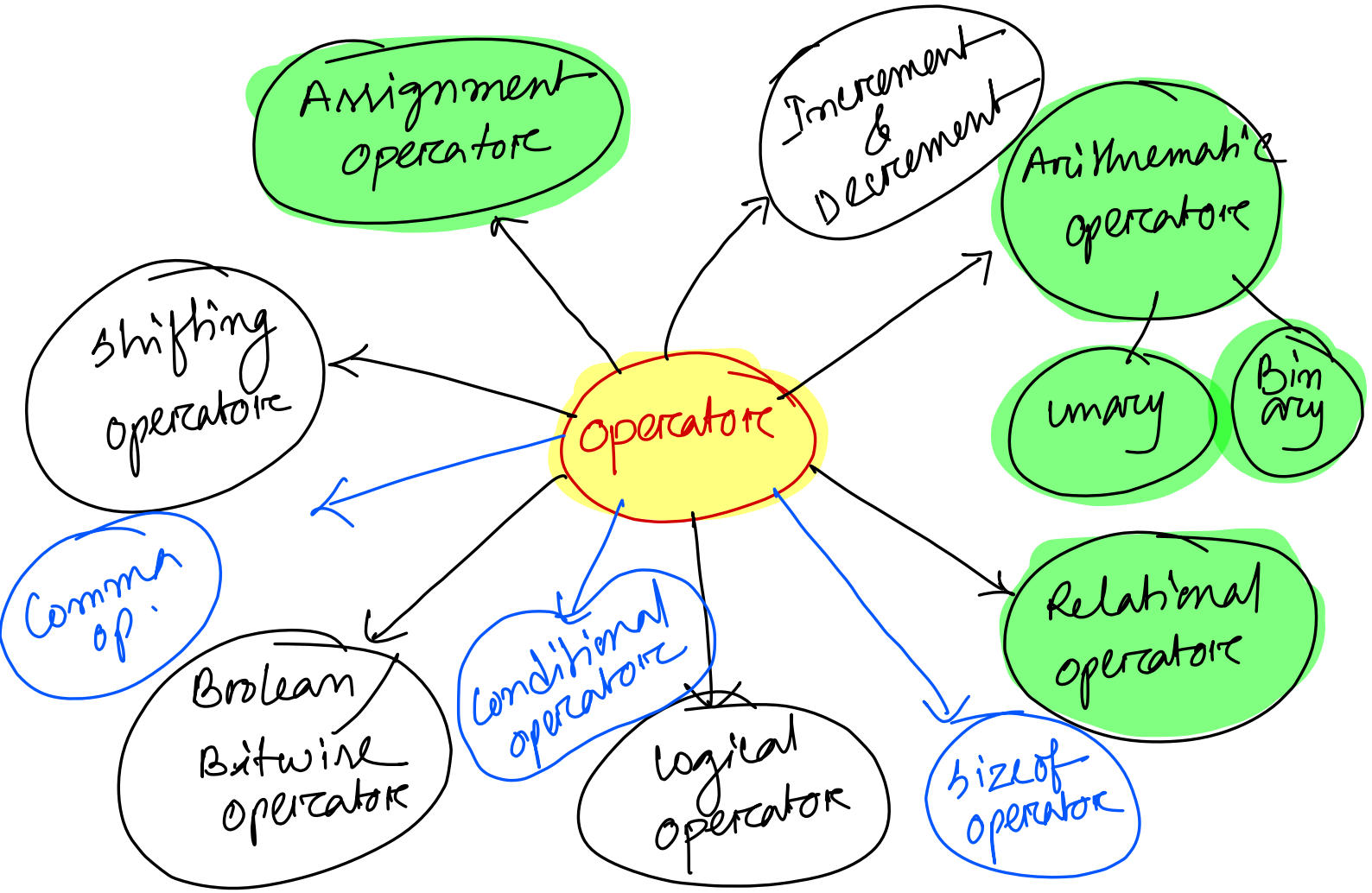
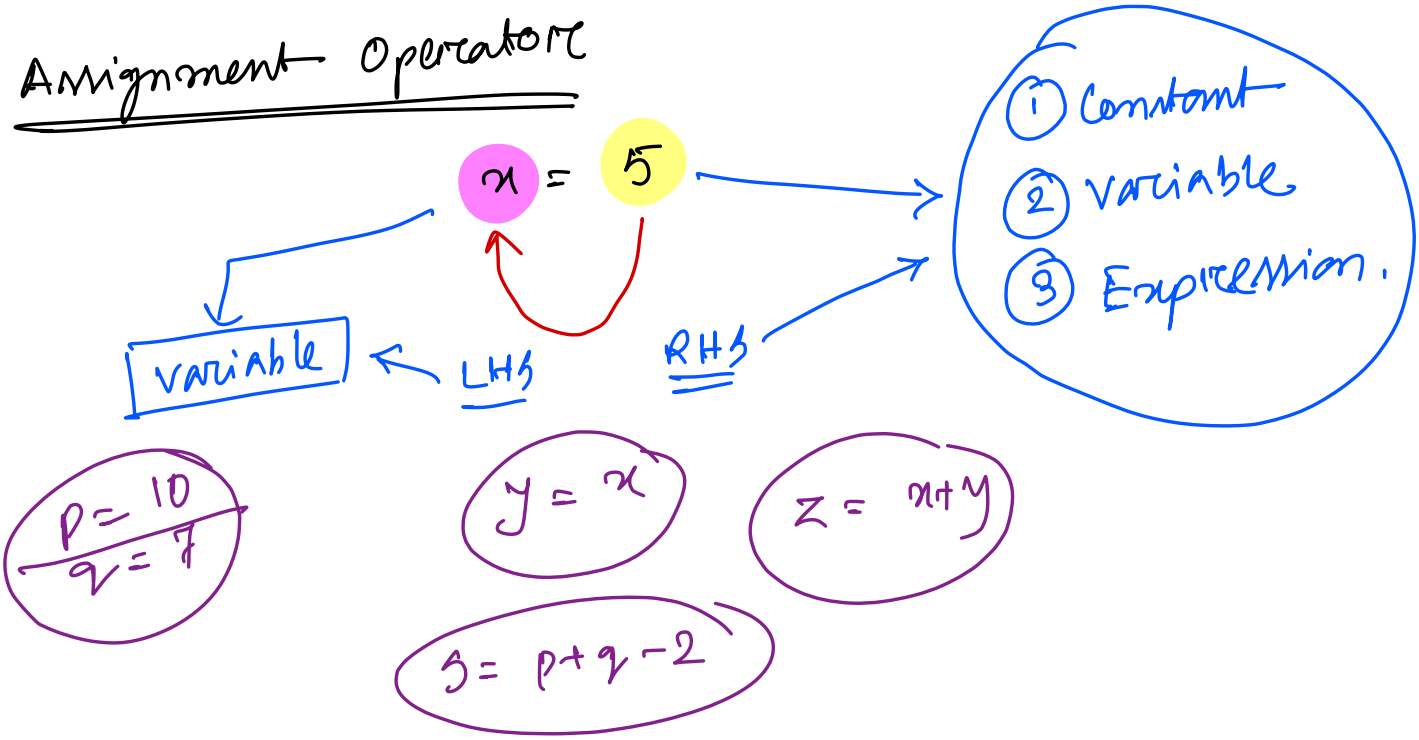
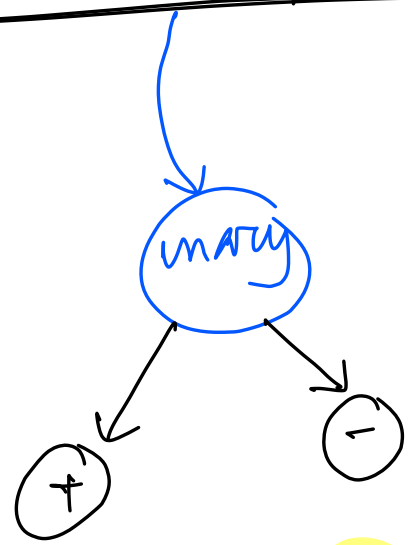
operators

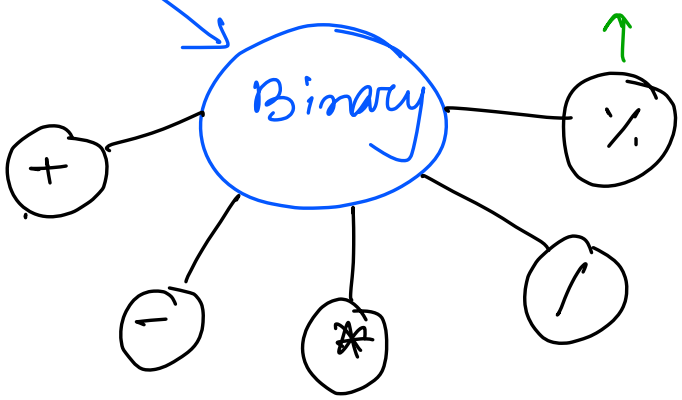


Assignment Operator



Arithmetic Operators





modulus

$x = x + (-y)$

a = 10, b = 5

a + b ⟶ 15

a - b ⟶ 5

a * b ⟶ 50

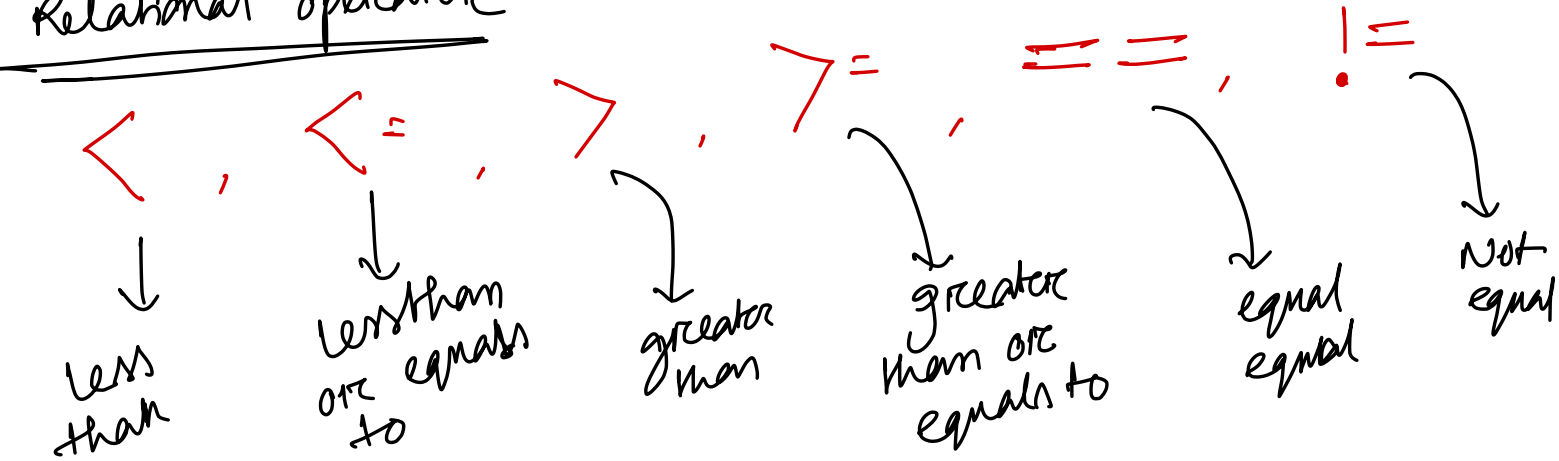
a / b ⟶ 2

a % b ⟶ 0

a = 7
b = 4
a % b = ?

4) 7 (1
4
3

Relational operators

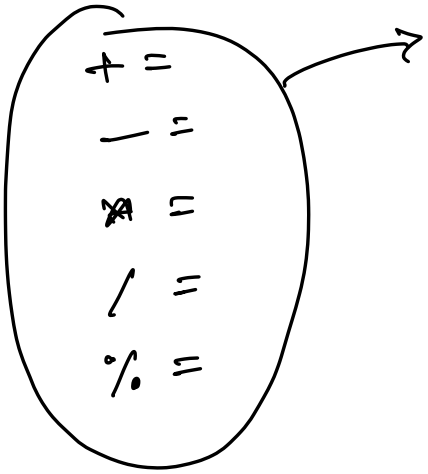


Assignment Operators :

```
int x, y, z;
x = y = z = 2;
```

```
x = 5;
x = x + 2;
```
// x = 7

x += 2



$x += 2 \rightarrow x = x + 2$

$x -= 5 \rightarrow x = x - 5$

$x *= (y+5) \rightarrow x = x * (y+5)$

Increment and Decrement Operators

++ , -- , prefix , Postfix

prefix Increment
prefix Decrement

Postfix Increment
postfix Decrement

```
x = 5
x++
printf("%d", x)
x = x + 1
```

5

6

```
++x
x = x + 1
printf("%d", x);
```

6

6

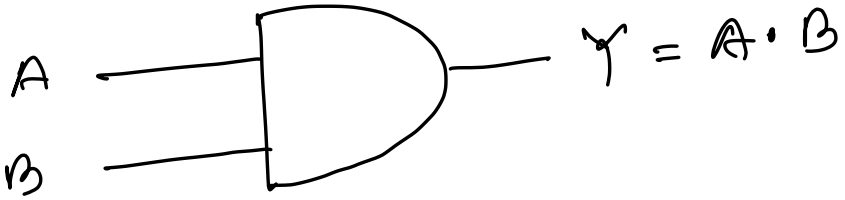
Logical Operators

AND → &&

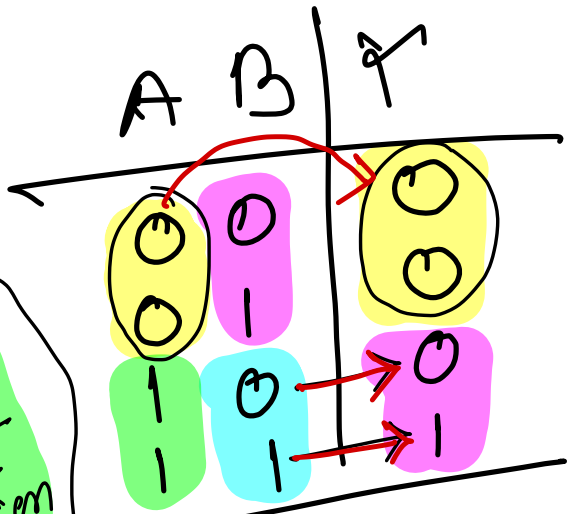
OR → ||

NOT → !

Logical AND



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



Except 0

Every variable or constant or Expression return True

x = 0

x = 0

x

F

x = 5 AND

T

x = -999

variable / constant / Expression.

&&

V C E

V C E

T

F

0

✓

X

a = 10, b = 5, c = 0

(a >= b) && (b == 6) → F → 0

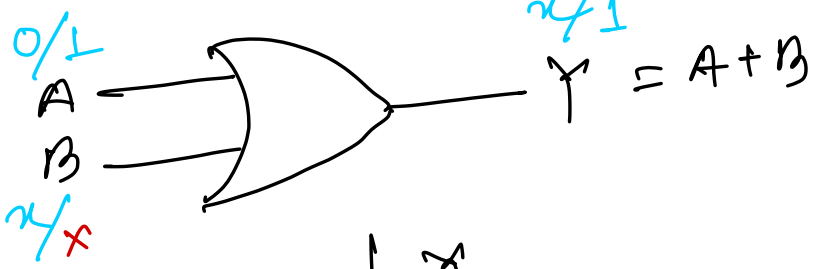
(a >= b) → T

(b == 6) → F

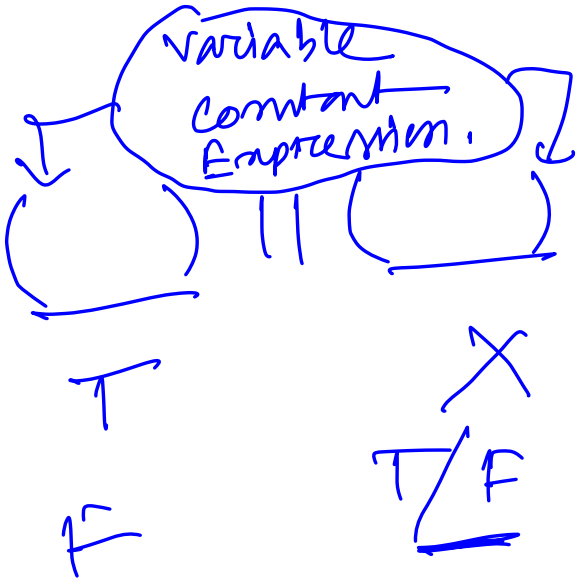
a && c → 0

a && b → 1

Logical OR → ||



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



```
int main(void){
    int a=3, b=4, c=3, d=4, x, y;
    x = (a=5) && (b=7);
    y = (c=5) || (d=8);
    printf("a=%d, b=%d, c=%d, d=%d, x=%d, y=%d", a, b, c, d, x, y);
    x = (a==6) && (b=9);
    y = (c==6) || (d=10);
    printf("a=%d, b=%d, c=%d, d=%d, x=%d, y=%d", a, b, c, d, x, y);
    return 0;
}
```

a: 5 → 5 → 5

b: 7 → 7 → 9

c: 5 → 5 → 5

d: 10 → 8 → 10

x: 0 → 1 → 0

y: 1 → 1 → 1

Bitwise Operators

1. Bitwise OR `|`
2. `||` OR
3. `||` NOT
4. `||` XOR

`~`

a = 10

b = 12

c = a | b

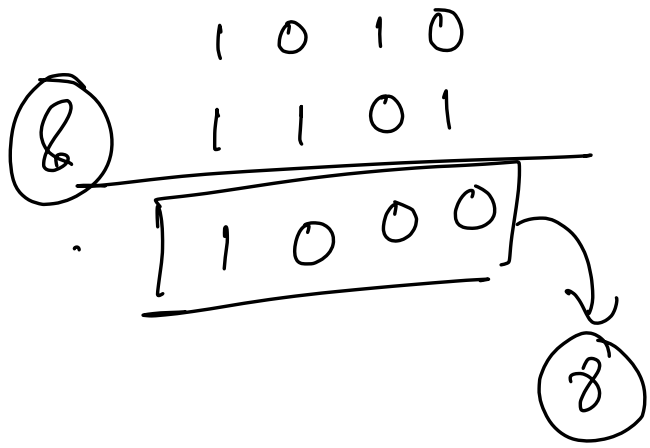
1	0	1	0
1	1	0	0
1	1	1	0

→ 14

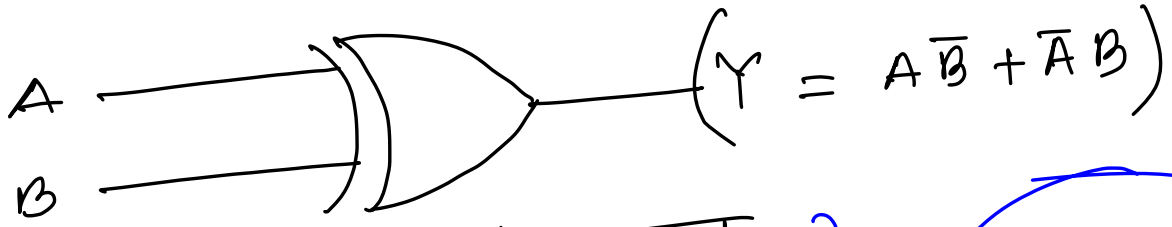
14

a = 10
b = 13

d = a & b



XOR gate



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

In equility checker

Bitwise XOR