Lecture - 10

Programming in C

Operators - 04

- ✓ Unary op —
  - ++ ✓
  - -- ✓
  - - ✓
- Binary op —
  - Assignment operators ✓
    - =
    - += -= *= %= /=
  - Arithmetic Operators ✓ — + , - , * / %
  - ✓ Relational Operators
  - ✓ Logical Operators
  - Bitwise operator
- Trinary op — 'Shorthand if-else'

* Shift operators << , >>

Bitwise Operator :

Bit ← Binary digit

↳ Operators that works with Binary.

a) Bitwise AND (&)

↳ Perform AND operation on each bits of two binary numbers.

AND
↳ All inputs = True
then o/p = True

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

← Binary AND

[decimal Number System]

⇓

1

a = 5 → 0101

b = 3 → &0011

0001 ← $(0001)_2$

$(0001)_2 \rightarrow (r)_{10}$

r = 1

b) Bitwise OR (|)

↳ Perform OR operation on each bits of two binary Number

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

$r \to 7$

$| \leftarrow$ OR

"%d" ⇓ 7

int a = 5; ⟶ $a = 5 \Rightarrow 0\,1\,0\,1$

$b = 3 \Rightarrow$ OR $0\,0\,1\,1$

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

$(0\,1\,1\,1)_2 \longrightarrow (7)_{10}$

OR → Any one input = 1; O/P = 1

int ⇒ decimal

c) Bitwise XOR (^)

↳ equality detector → If both bits are different the o/p = 1 (high | True)

⇒ $A \oplus B$

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

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

$r \rightarrow 6$

$\downarrow$ 6

$$\begin{array}{rcccc} a = & 0 & 1 & 0 & 1 \\ b = \text{xor} & 0 & 0 & 1 & 1 \\ \hline & 0 & 1 & 1 & 0 \end{array}$$

$A \oplus B$

$(0110)_2 \rightarrow (6)_{10}$

int → 4 Bytes → 32 bits

$\underbrace{0 \ldots 000000}_{1} \; \underbrace{0101}$ ← $(5)_{10}$

$\Downarrow$

$\underbrace{1111 \ldots\ldots 1010}$ → 2's Complement

?

↳ $(-6)_{10}$

d) Bitwise NOT (~) ← Unary operator

↳ It inverts each bit of its operand

↳ Complement

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

$a = 5 \longrightarrow (0101)$

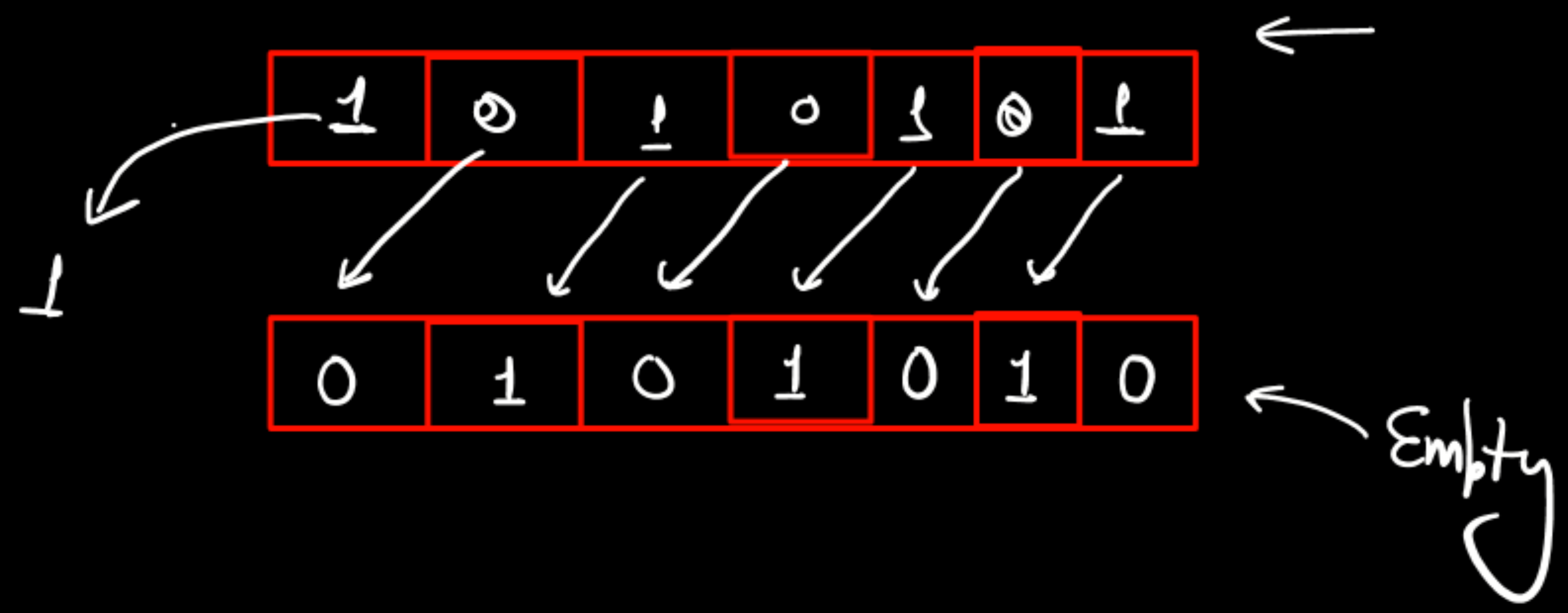
NOT a ⇒ $(1010)_2 \longrightarrow (r)_{10}$

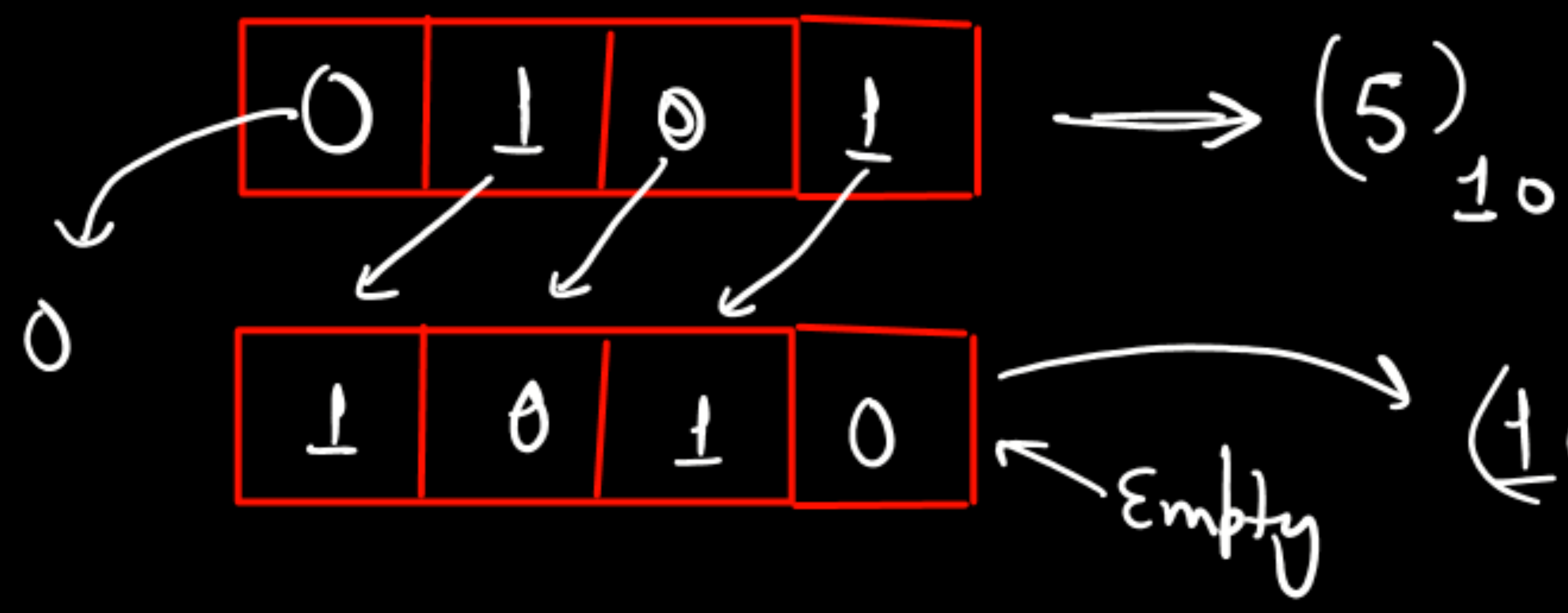
↳ $(10)_{10}$

e) Bitwise left shift operator (<<)

↳ it shifts the bits of a binary number to the left in 'n' place

* left shift of a binary number multiply the number by 2.



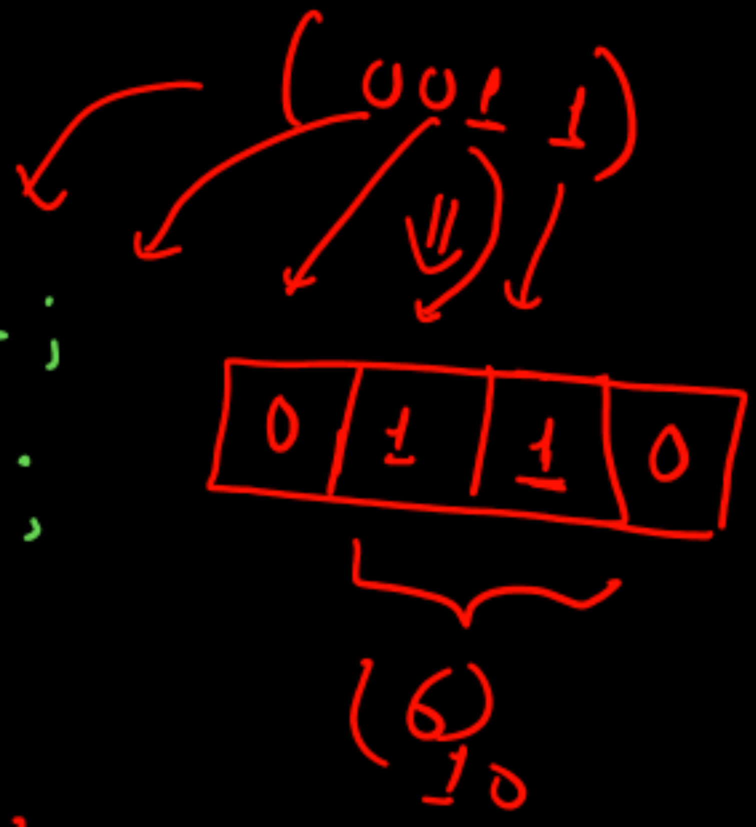


$(1010)_2 \Rightarrow (10)_{10} \leftarrow 5 \times 2$

```
int x = 3;
int r = x << 1;
printf("%d",r);
```

⇓

$6 \leftarrow 3\times2^1$



for 'n' of shift the result will be

$\underset{\sim\sim}{\text{result}} = \underset{\text{Binary}}{\underset{\downarrow}{\text{number}}} \times 2^n \quad \underset{\sim\sim\sim}{n < \text{size of (number)}}$

$1 \rightarrow n\times2$

$2 = n\times2\times2$

$3 = n\times\underbrace{2\times2\times2}$

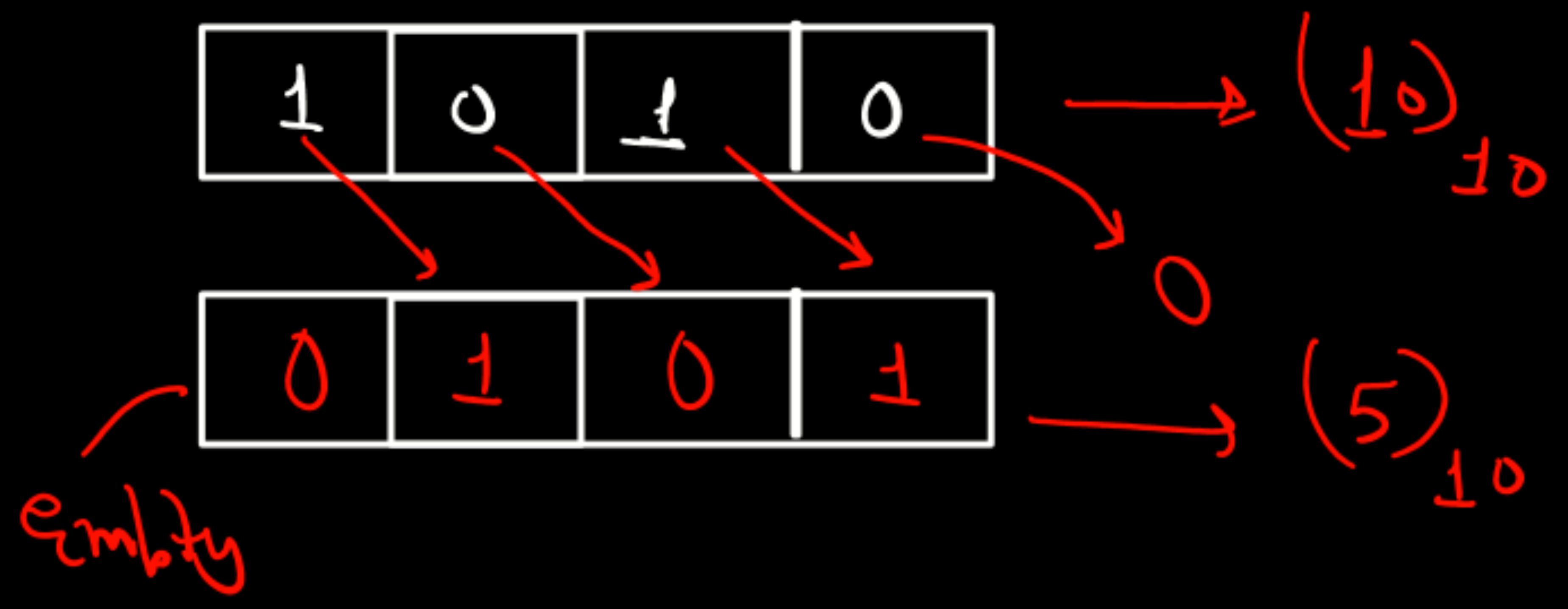
$4 = n \times \underbrace{2\times2\times2\times2} \quad 2^4$

f) Bitwise Right Shift operator (>>) :

↳ Shift the bits of a binary number to the right for 'n' place

Ex $x >> 2;$ ⟸ Shift the number for two places

⇒ Divide by 2



```
int a = 10
int r = a >> 1
printf("%d", r)
```

⇓

5

result = number x 2^{-n} for n shifts & n < size of (number)

resut = $10 \times 2^{-1} = \frac{10}{2} = (5)_{10}$

Trinary Operator → 3 operands, 2 operator

↳ Conditional operator, Short hand if else operator

Condition ← True — r_1

false → r_2

↳ Trinary op.

Syntax

Condition ? Expression 1 : Expression 2

⇓ True/false — ↓ if True — ↓ if false

```
int a = 5;
int b = 10;

int max;

max = (a > b) ? a : b;

printf("%d", max);
```

(a > b) → True; a → max; b → max

a = 5

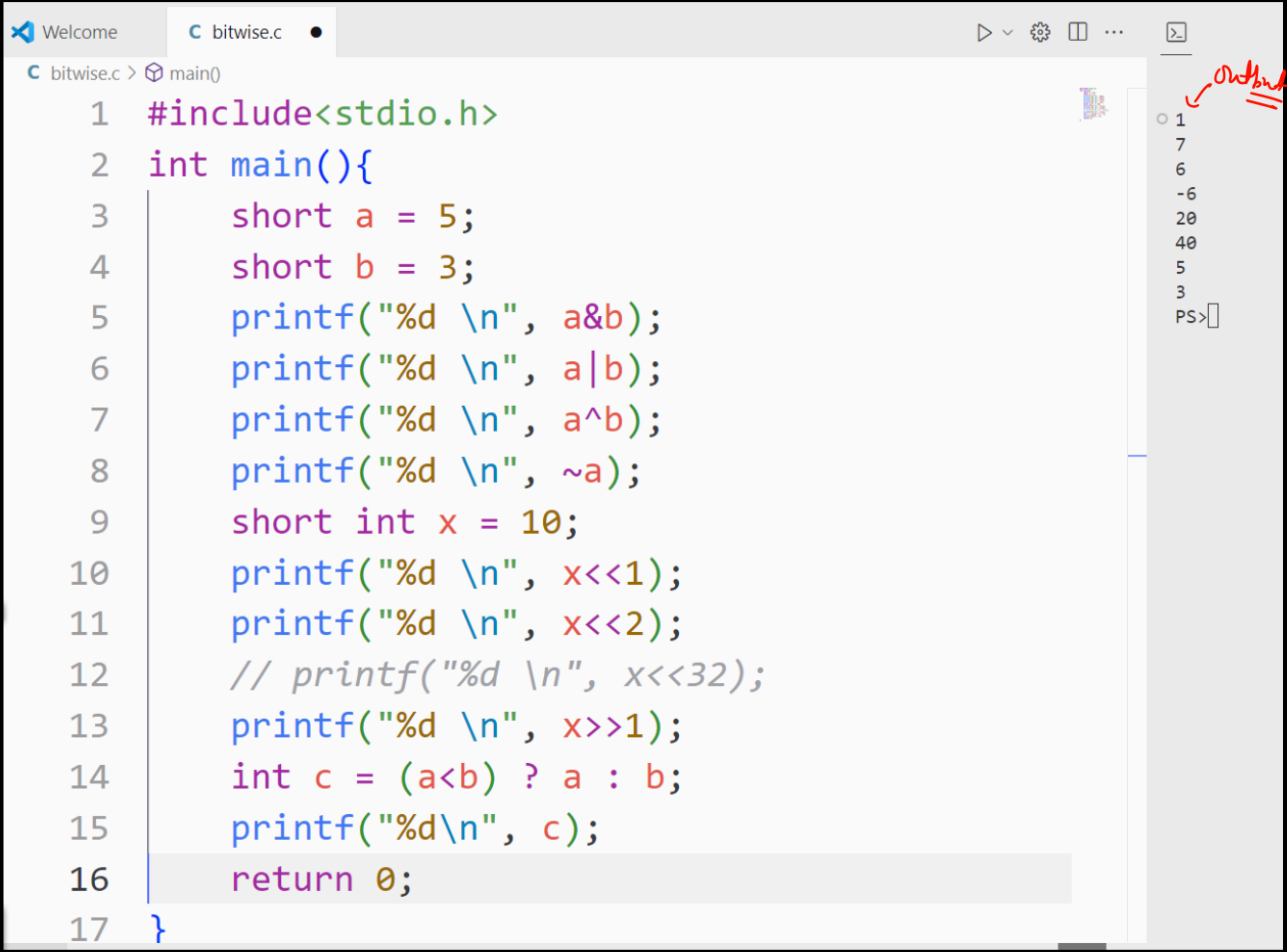
b = 10

max = a > b

↳ max = a

वरना

max = b



```c
#include<stdio.h>
int main(){
    short a = 5;
    short b = 3;
    printf("%d \n", a&b);
    printf("%d \n", a|b);
    printf("%d \n", a^b);
    printf("%d \n", ~a);
    short int x = 10;
    printf("%d \n", x<<1);
    printf("%d \n", x<<2);
    // printf("%d \n", x<<32);
    printf("%d \n", x>>1);
    int c = (a<b) ? a : b;
    printf("%d\n", c);
    return 0;
}
```

Output

```
1
7
6
-6
20
40
5
3
PS>
```

[Arrays & Pointers]

[Operator Precedence & Associativity]

[[60 MCQ → Token & Intro]
[100 MCQ → Operators]]

⇓

1 Video