Certainly! Let's break down **Question 3: Operators in C** in detail.

---

3. Operators in C

Operators are symbols used to **perform operations** on variables and values.

C provides the following types of operators:
1. **Arithmetic Operators**
2. **Relational (Comparison) Operators**
3. **Logical Operators**
4. **Bitwise Operators**
5. **Assignment Operators**
6. **Increment & Decrement Operators**
7. **Ternary (Conditional) Operator**
8. **Special Operators**

---

1. Arithmetic Operators

Used for mathematical calculations.

Operator	Meaning	Example
`+`	Addition	`x + y`
`-`	Subtraction	`x - y`
`*`	Multiplication	`x * y`
`/`	Division	`x / y` (integer division for `int`)
`%`	Modulus (Remainder)	`x % y` (remainder after division)

Example: Arithmetic Operators

```c
#include <stdio.h>
int main() {
    int a = 10, b = 3;
    printf("Addition: %d\n", a + b);
    printf("Subtraction: %d\n", a - b);
    printf("Multiplication: %d\n", a * b);
    printf("Division: %d\n", a / b);  // Integer division (result = 3)
    printf("Modulus: %d\n", a % b);   // Remainder (10 % 3 = 1)
    return 0;
}
```

---

2. Relational (Comparison) Operators

Used to compare values, returning `1` (true) or `0` (false).

Operator	Meaning	Example
`==`	Equal to	`a == b`
`!=`	Not equal to	`a != b`
`>`	Greater than	`a > b`
`<`	Less than	`a < b`
`>=`	Greater than or equal to	`a >= b`
`<=`	Less than or equal to	`a <= b`

Example: Relational Operators

```c
int a = 5, b = 10;
printf("%d\n", a > b);  // Output: 0 (false)
printf("%d\n", a < b);  // Output: 1 (true)
```

---

3. Logical Operators

Used for **logical operations** (AND, OR, NOT).

Operator	Meaning	Example
`&&`	Logical AND	`(a > 5 && b < 10)`
`||`	Logical OR	`(a > 5 || b < 10)`
`!`	Logical NOT	`!(a > 5)`

Example: Logical Operators

```c
int x = 10, y = 20;
printf("%d\n", (x > 5 && y < 30));  // Output: 1 (true)
printf("%d\n", (x > 15 || y < 25)); // Output: 1 (true)
printf("%d\n", !(x > 5));           // Output: 0 (false)
```

---

4. Bitwise Operators

Perform operations on **binary bits** of numbers.

Operator	Meaning	Example (for x = 5, y = 3)
`&`	Bitwise AND	`x & y` (0101 & 0011 → 0001 = 1)
`|`	Bitwise OR	`x | y` (0101
`^`	Bitwise XOR	`x ^ y` (0101 ^ 0011 → 0110 = 6)
`<<`	Left shift	`x << 1` (0101 → 1010 = 10)

Operator	Meaning	Example (for x = 5, y = 3)
`>>`	Right shift	`x >> 1` (0101 → 0010 = 2)

---

5. Assignment Operators

Used to **assign values** to variables.

Operator	Meaning	Example
`=`	Assign	`x = 10`
`+=`	Add & Assign	`x += 5` (x = x + 5)
`-=`	Subtract & Assign	`x -= 5` (x = x - 5)
`*=`	Multiply & Assign	`x *= 5` (x = x * 5)
`/=`	Divide & Assign	`x /= 5` (x = x / 5)
`%=`	Modulus & Assign	`x %= 5` (x = x % 5)

---

6. Increment & Decrement Operators

Used to **increase or decrease** values by `1`.

Operator	Meaning	Example
`++`	Increment	`x++` (Post-increment)
`--`	Decrement	`x--` (Post-decrement)

Example: Pre vs. Post Increment

```c
int a = 5;
printf("%d\n", a++);  // Output: 5 (post-increment)
printf("%d\n", a);    // Output: 6
printf("%d\n", ++a);  // Output: 7 (pre-increment)
```

---

7. Ternary (Conditional) Operator

Shorthand for `if-else`.

Syntax:

```c
(condition) ? value_if_true : value_if_false;
```

Example: Ternary Operator

```c
int a = 10, b = 20;
int min = (a < b) ? a : b;
printf("Minimum value: %d\n", min);  // Output: 10
```

---

8. Special Operators

Operator	Meaning	Example
`sizeof`	Returns size of variable/data type	`sizeof(int)`
`&`	Address-of operator	`&x` (gets memory address)
`*`	Pointer dereference	`*ptr` (access value at address)
`,`	Comma operator	`x = (a = 5, b = 10, a + b);`

Example: sizeof Operator

```
int x = 10;
printf("Size of int: %lu bytes\n", sizeof(x)); // Output: 4 bytes
```

Key Takeaways

✔ **Arithmetic**: +, -, *, /, %
✔ **Relational**: ==, !=, <, >, <=, >=
✔ **Logical**: &&, ||, !
✔ **Bitwise**: &, |, ^, <<, >>
✔ **Assignment**: =, +=, -=, *=, /=
✔ **Increment/Decrement**: ++, --
✔ **Ternary**: `(condition) ? true_value : false_value;`
✔ **Special**: `sizeof`, &, *

Would you like more examples or explanations?