Shifting
Suppose we are in eight-bit world. What is the result of the following:

(1101 1110) << 3

1111 0000

(1101 1110) >> 3 (arithmetic)

1111 1011

(1101 1110) >> 3 (logical)

0001 1011

(0010 0111) << 3

0011 1000

(0010 0111) >> 3 (arithmetic)

0000 0100

(0010 0111) >> 3 (logical)

0000 0100

Some bitwise operations
Evaluate the following, assuming 4-bit values:

1010 | 0101

1111

1010 || 0101

0001

1010 & 0101

0000

1010 && 0101

0001

~1001

0110

!1001

0000

Masking (credit to CSAPP)
Let x be an integer (type int). Write C expressions in terms of x. Do not use constants greater than 0xFF

A. The least significant byte of x, with all other bits set to 0

```
x & 0xFF
```

B. All but the least significant byte of x complemented, with the least significant byte left unchanged.

```
(x & 0xFF) | (~x & ~0xFF)  // one option
(x & 0xFF) | ~(x | 0xFF)   // another option
```

C. The least significant byte set to all ones, and all other bytes of x left unchanged

```
x | 0xFF
```

Does anything change if x is unsigned?

No. Bitwise operators operate on the value without regard to whether it has a signed or unsigned type.