Topic9(Bitwise Operators)

Bitwise Operators

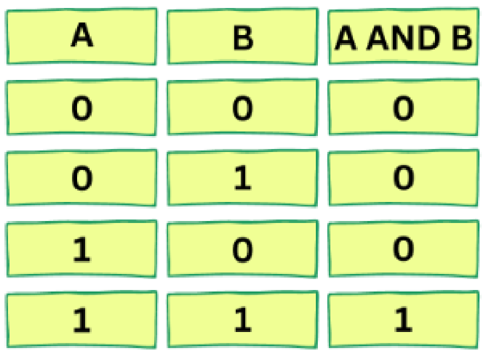
> Bitwise operations are asked frequently in programming interviews as well as competitive programming. Therefore it’s essential to practice problems that use a variety of approaches and algorithms.

Types Of Bitwise Operators

Operators	Name	Example
`&`	bitwise AND	`c = a & b`
`|`	bitwise OR	`c=a|b`
`^`	bitwise XOR	`c = a ^ b`
`~`	bitwise NOT	`c = ~ a`
`<<`	left shift	`c = a << 2`
`>>`	right shift	`c = a >> 2`

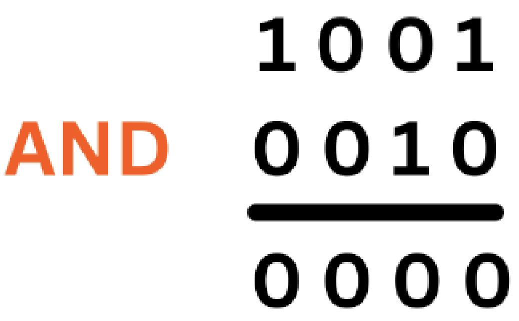
Bitwise AND

> It takes two numbers as input operands and does Bitwise AND on every corresponding bit of two numbers. If both operands are 1, the bitwise AND operator returns 1. Otherwise, it produces a value of 0.



A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

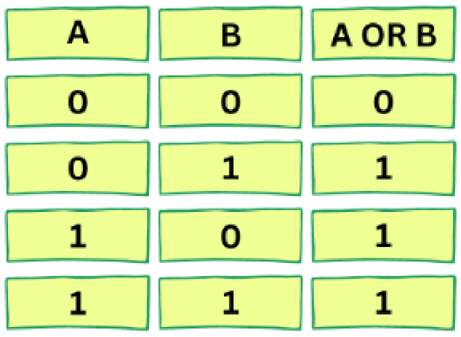
```
#include <iostream>
using namespace std;

int main()
{
    int a=9;
    int b=2;

    //And
    int c=a&b;

    //Output
    cout<<c<<endl;
    return 0;
}
```



```
      1001
AND   0010
      ----
      0000
```

Bitwise OR

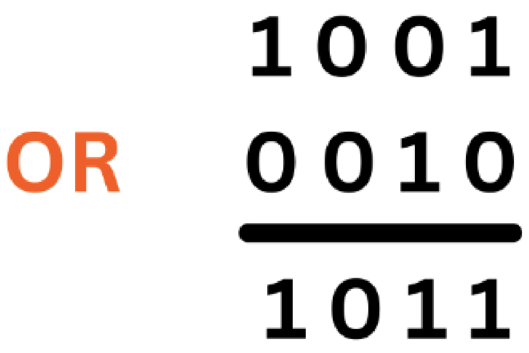
> It takes two numbers as input operands and does Bitwise OR on every corresponding bit of two numbers. If at least one bit is 1, the bitwise OR operator returns 1. Otherwise, it produces a value of 0.



A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

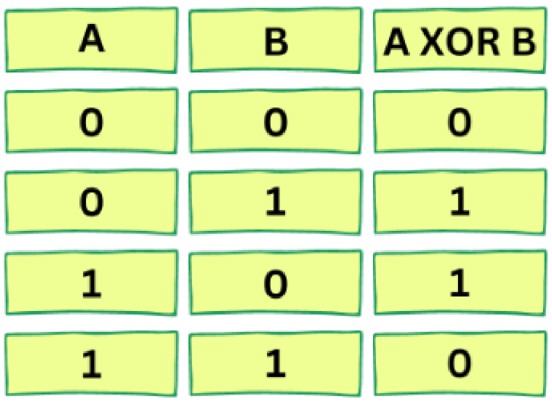
```
#include <iostream>

using namespace std;
```

```
int main()
{
    int a=9;
    int b=2;

    //OR
    int c=a|b;

    //Output
    cout<<c<<endl;
    return 0;
}
```



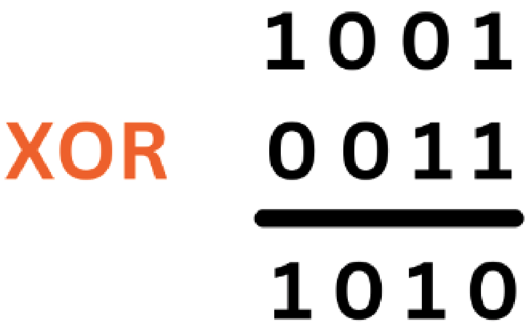
Bitwise XOR

> Also called Exclusive OR, it takes two numbers as input operands and does Bitwise XOR on every corresponding bit of two numbers. If both bits are different, the bitwise OR operator returns 1. Otherwise, it produces a value of 0.



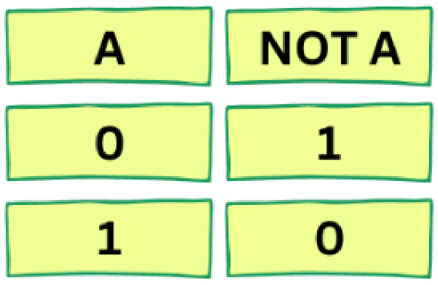
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

```
#include <iostream>

using namespace std;

int main()
{
    int a=9;
    int b=3;

    //Xor
    int c = a ^ b;

    //Output
    cout<<c<<endl;
    return 0;
}
```



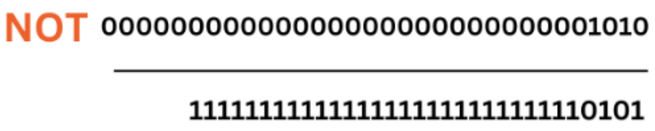
Bitwise NOT

> it takes one number as an input operand and inverts all the number bits. It means 0 will replace with 1 and vice versa.



A	NOT A
0	1
1	0

```
#include <iostream>

using namespace std;

int main()
{
    int a = 10;

    //Not
    int c = ~ a;

    //Output
    cout<<c<<endl;
    return 0;
}
```

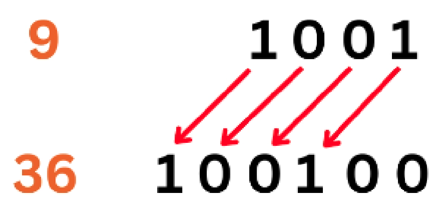


Left shift Operator

> It takes two numbers as input operands. The value of the left operand is shifted to the left by the number of bits given by the right operand and returned.

> If numbers are positive, the left-shift operators are identical to multiplication by two.

```
#include <iostream>

using namespace std;

int main()
{
    int a = 9;

    //Left Shift
    int c = a << 2;

    //Output
    cout<<c<<endl;
    return 0;
}
```



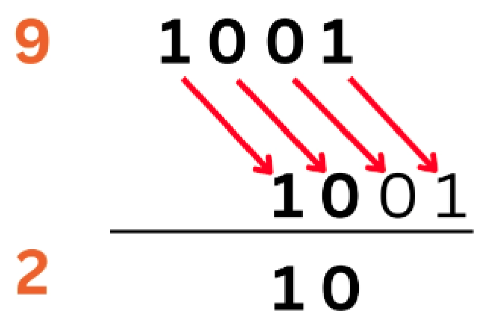
Left Shift by 2

Right shift Operator

> It takes two numbers as input operands. The value of the left operand is shifted to the right by the number of bits given by the right operand. The least significant bits are lost when we shift a number to the right, while the most significant bits are replaced with zeroes.

> Only if numbers are positive, the right-shift operators are identical to the division by two.

```
#include <iostream>

using namespace std;

int main()
{
    int a = 9;

    //Left Shift
    int c = a >> 2;

    //Output
    cout<<c<<endl;
    return 0;
}
```



Right Shift by 2