Bitwise Manipulation

Operators:

1. AND

If any is false then entire is false. If both are true then it is true.

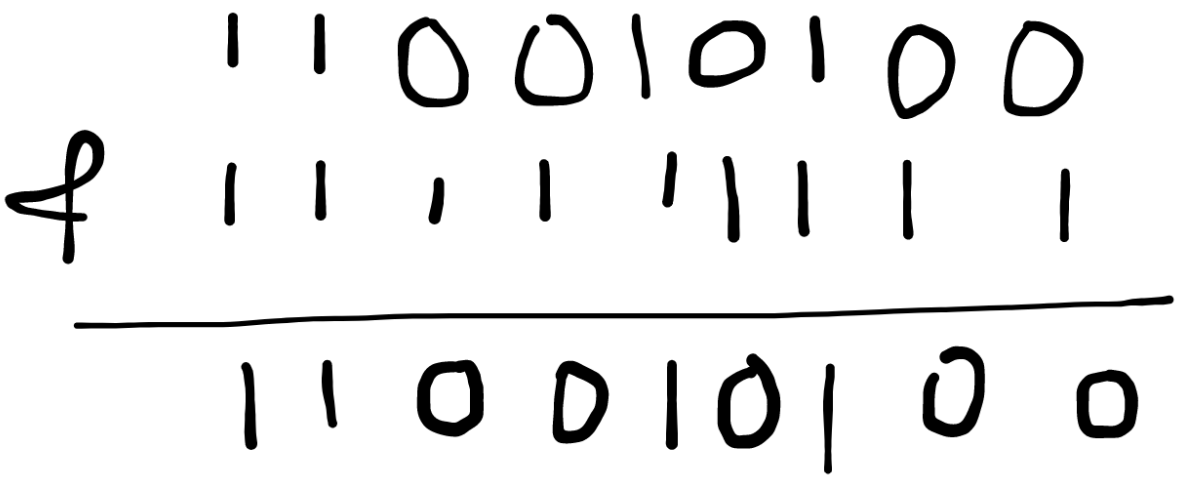
a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

Q: Given a number N find if it’s even.

Observation:

- When you & 1 with any number the digits remain the same.

Ex:



2. OR

If any one is true, then entire is true.

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

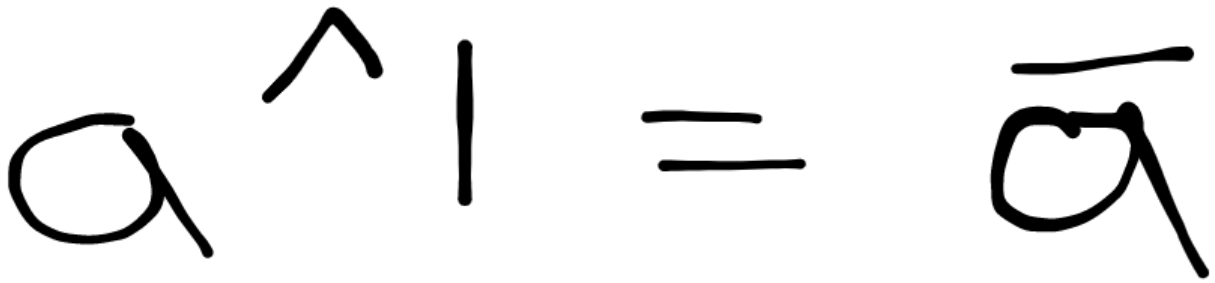
3. XOR (^) (Aka Exclusive OR)

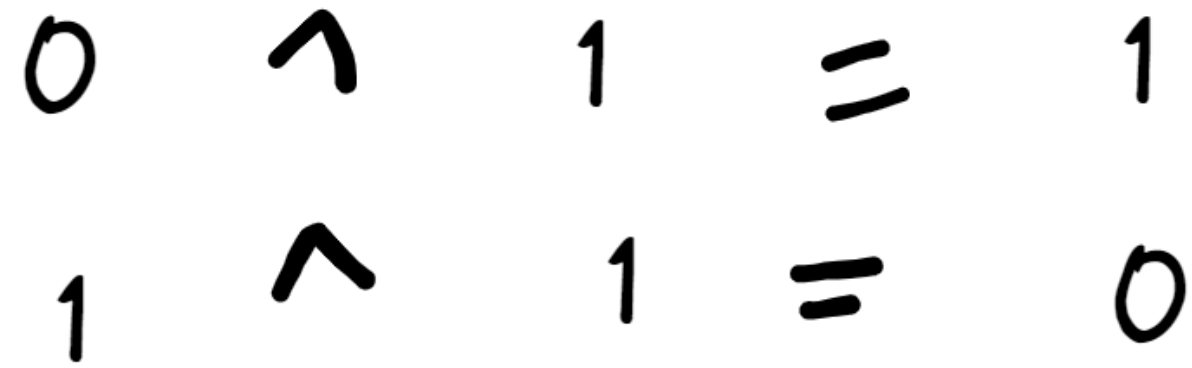
If and only if. (Only 1 should be true)

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

Observation:

- When you ^ 1 with any number we get the compliment of that number.





- When you ^ 0 with any number we get the number itself.

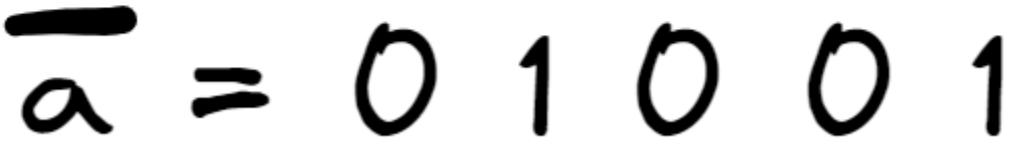
a ^ 0 = a

- When you ^ number with itself we get 0.

a ^ a = 0

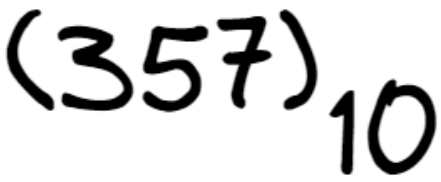
4. Compliment (~)

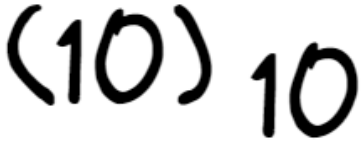
Opposite of the number.



Number Systems:

1. Decimal Number → 0, 1, 2, …. , 9 (Base 10)





2. Binary Number → 0 & 1 (Base 2)

$(10)_{10} = (1\ 0\ 1\ 0)_2$

3. Octal → 0, 1, 2, 3, …, 7 (Base 8)

Octal 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, ...

$(9)_{10} = (11)_8$

4. Hexadecimal → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (Base 16)

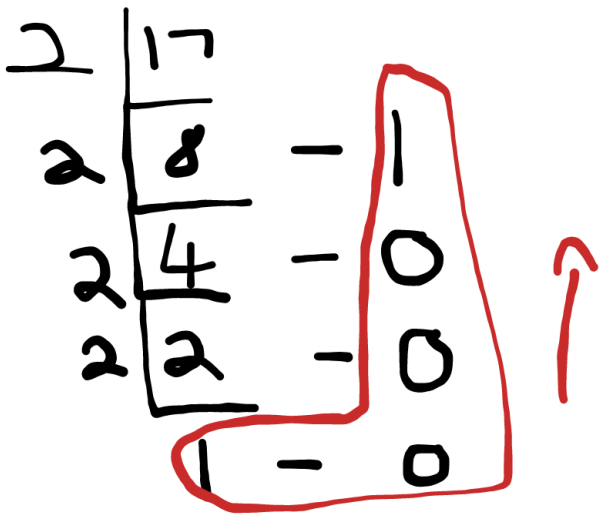
$(10)_{10} = (A)_{16}$ $(12)_{10} = (C)_{16}$

Conversion of Number Systems

- **Covert Decimal to any base b**

Convert $(17)_{10}$ to base 2

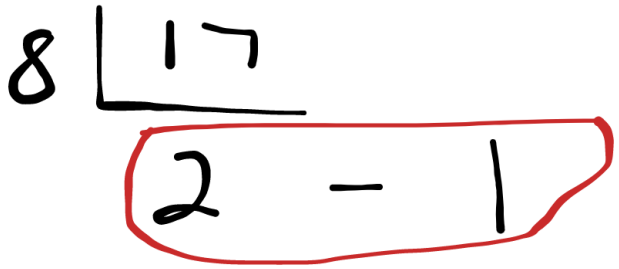
Keep dividing by base, take remainders, write it in opposite.



$= (1\ 0\ 0\ 0\ 1)_2 = (17)_{10}$

Convert $(17)_{10}$ to base 8

Keep dividing by base, take remainders, write it in opposite.



$= (21)_8 = (17)_{10}$

- **Convert any base b to Decimal**

$(1001)_2 = (\ )_{10}$?

Steps:

Multiply and add the power of base with the digits.

$= 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0$

$= 16 + 0 + 0 + 0 + 1 = (17)_{10}$

$(21)_8 = (\ )_{10}$?

Steps:

Multiply and add the power of base with the digits.

$= 2 * 8^1 + 1 * 8^0$

$= 16 + 1 = (17)_{10}$

- If we are given to Convert Base 2 to Base 8, then first convert Base 2 to decimal and then to Base 8.

Operators Continuation

5. Left Shift Operator (<<)

$(10)_{10} = (1010)_2$

$10 << 1$

Steps:

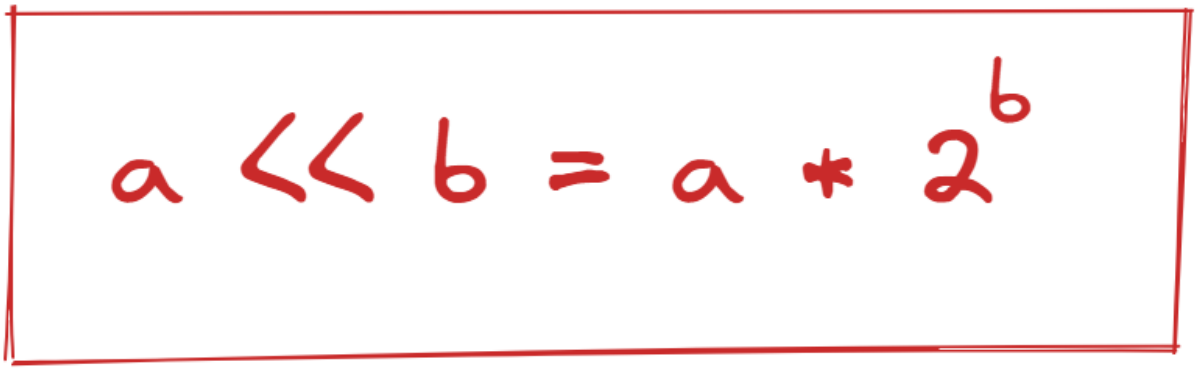
First Convert it into Binaary

$= 10100$

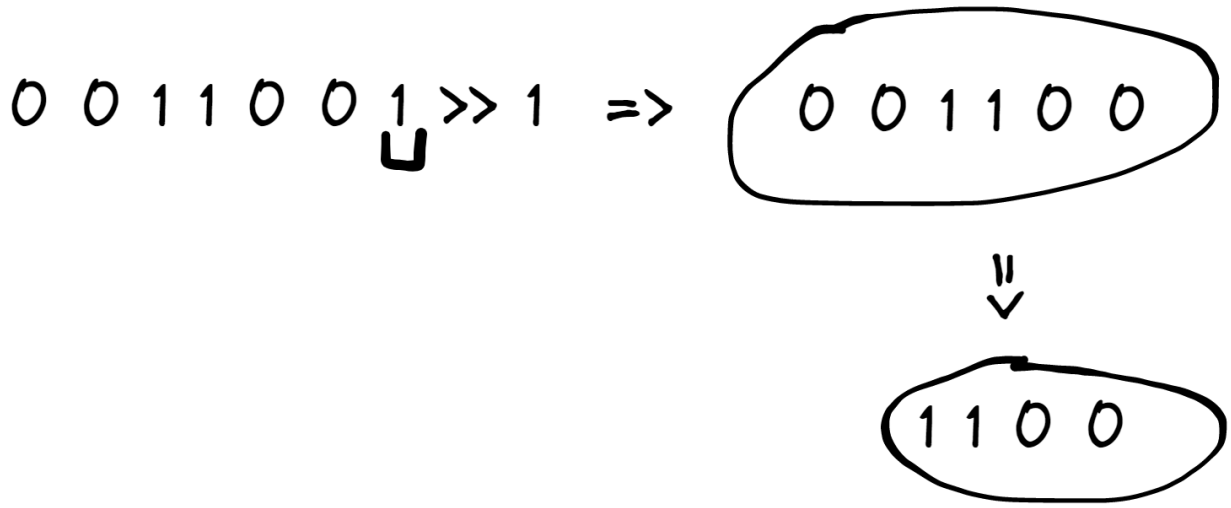
$= 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0$

$= 16 + 4 = 20$

$a << 1 = 2a$



6. Right Shift Operator (>>)



$$a >> 1 = a / 2$$

$$a >> b = a / 2^{b}$$

Questions

```
Q1: Given a number n find if it is odd or even.
Ans: Check if the last bit (LSB) is 1 or 0.
If it is 1 then its odd, else even
LSB: Least Significant Bit.

static boolean isOdd(int num) {
        return (num & 1) == 1;
    }
```

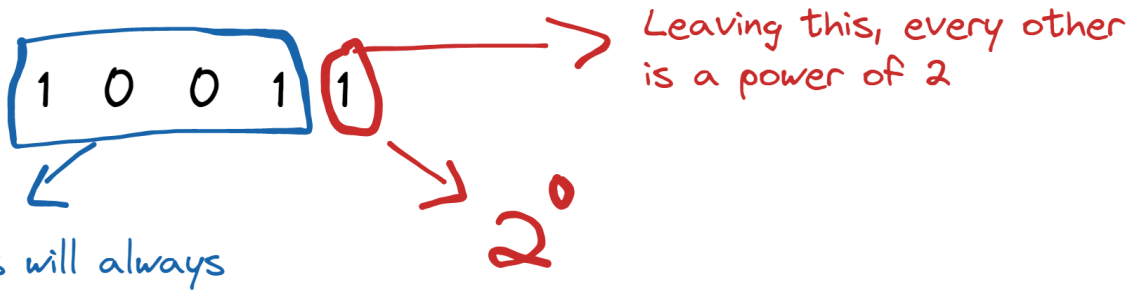
Q: Given a number n find if it is odd or even.

Point: Every no. is calculated in binary form internally

12 + 7 =>

$(19)_{10} = (10011)_2$

```
  1 1 0 0
+ 0 1 1 1
---------
1 0 0 1 1
```



Henc: If 2^0 place == 1 => The number is odd
== 0 => The number is even.

```
Q2:
Given a non-empty array of integers nums, every element appears twice except for one.
Find that single one.

Ans:
We know that any number xor with itself we get zero.
And we also know that any number xor with 0 will be the number itself.
So here we will xor the entire array and the resultant will be the single element.

public static int singleNumber(int[] nums) {
        int xor = 0;
        for (int num : nums) {
```

```
            xor ^= num;
        }
        return xor;
    }
```

```
Q3. Find the ith bit of a number.
Ans: AND that particular it with 1
Example: Find the 5th bit for: 10110110
    1 0 1 1 0 1 1 0
 &  0 0 0 1 0 0 0 0  --> This is called a mask.
 -------------------
    0 0 0 1 0 0 0 0

We need a mask with n - 1 zeros.
To create a mask we can use left shift with n - 1
1 << (n - 1)
1 << 4 => 10000

return num & (1 << ( i - 1 ));
```

```
Q4: Set the ith bit -> Turn it to 1
-> 0 --> 1
-> 1 --> 1
Ans: OR that particular ith with 1
Example: Set the 4th bit for: 1010110
    1 0 1 0 1 1 0
||  0 0 0 1 0 0 0  --> Mask
 -------------------
    1 0 1 1 1 1 0

We need a mask with n - 1 zeros.
To create a mask we can use left shift with n - 1
1 << (n - 1)
1 << 3 => 0001000
return num | (1 << ( i - 1 ));

static int setTheIthBit(int num, int i) {
        return num | (1 << ( i - 1 ));
    }
```

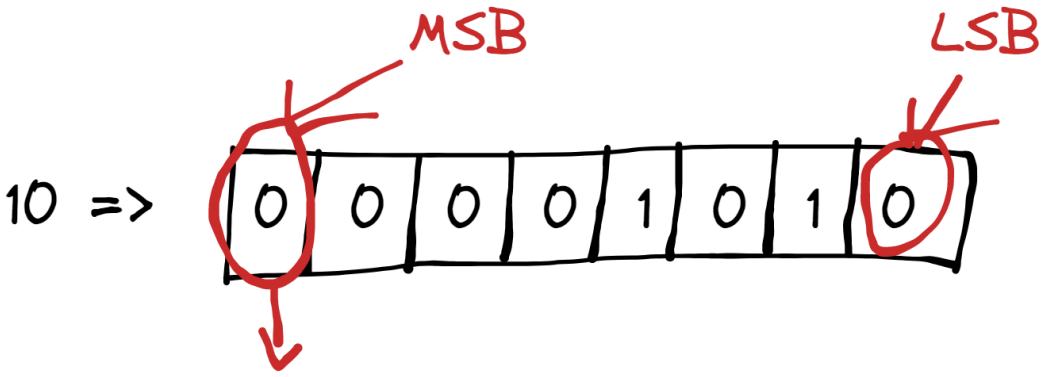
```
Q5: Reset the ith bit -> Turn it to 0
-> 0 --> 0
-> 1 --> 0
Ans: AND everything with 1 except that particular ith with 0
Example: Reset the 5th bit for: 1010110
    1 0 1 0 1 1 0
&   1 1 0 1 1 1 1  --> Mask
 -------------------
    1 0 0 0 1 1 0


To create a mask we can use left shift with n - 1
1 << (n - 1)
1 << 3 => 001000
Then take the complement of it, which is: 110111
Mask: !(1 << (n - 1))

static int setTheIthBit(int num, int i) {
        return num & ~(1 << ( i - 1 ));
    }
```

Negative of a number in binary form

1 byte = 8 bits



Tells us if number is positive or negative

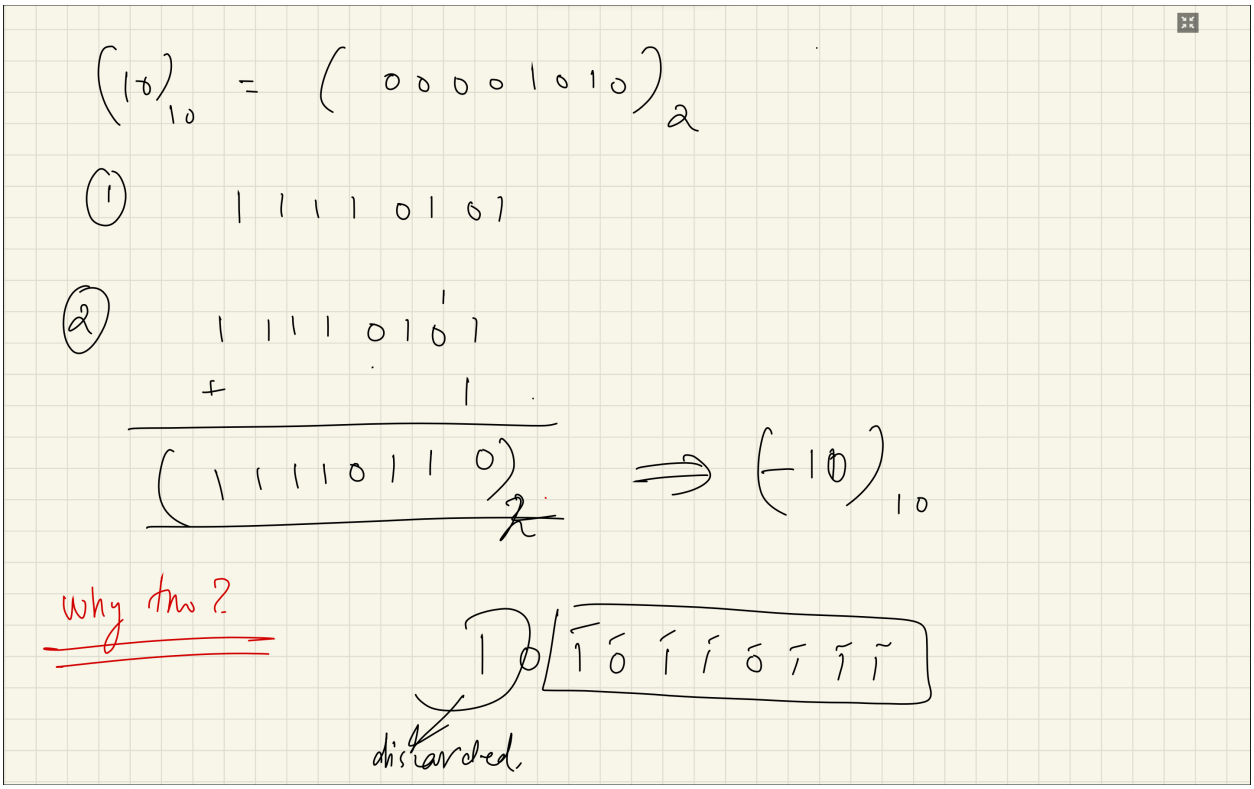
1 -> -ve
0 -> +ve

How to find negative of a number:-

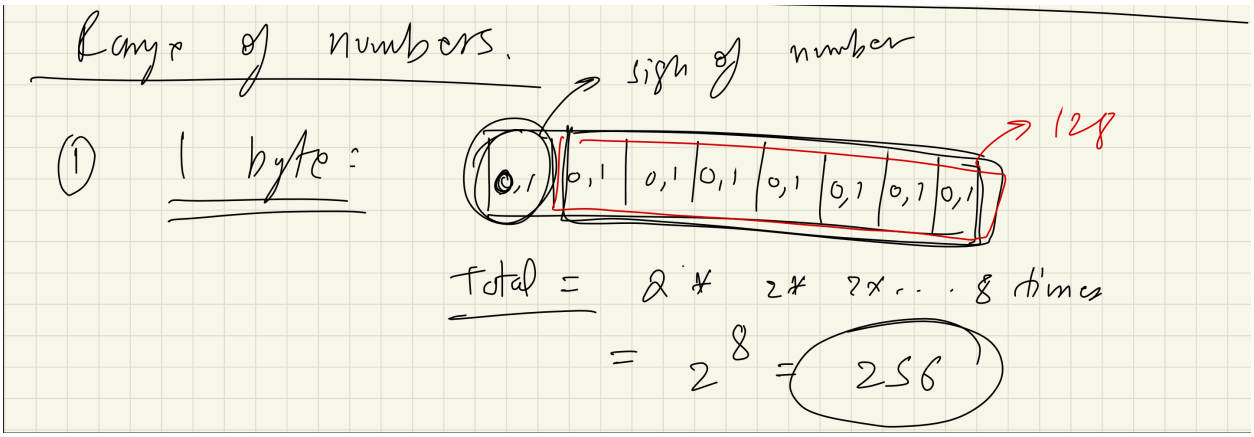
Steps:

1. Take complement of the number.
2. Add 1 to it.

} 2's Compliment



Range of Numbers:-



actual no is stored in bits = n-1 → total bits

In 1 byte : 7 bits

Total 1 can make from 7 bits

$= 2^7 = 128$

−128 to 127

→ 0 is here as well

① ⇒ 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1

② 1 1 1 1 1 1 1 1

\+ 1

1 0 0 0 0 0 0 0 0 ⇒ 9 bits

will be discarded.

→ −0 = 0

Range formula: for n bits

-2^{n-1} to $2^{n-1} - 1$