Bits

=) Checking if number is even or odd

if (n&1) => 1 [odd]

(n&1) => 0 [even]

=) Detecting if 2 integers have opposite sign

x ^ y is negative if x & y have opposite sign

x ^ y is positive if x & y have same sign

=) Swap 2 numbers without 3rd variable

x = x^y

y = x^y

x = x^y

=) Setting a bit at ~~po~~ same position (pos)

num |= (1 << pos)

=) Unset / clear bit at same position (pos)

num &= (~(1<<pos))

=> Toggling bit at n^{th} position

num ^= (1 << pos)

=> Check if n^{th} bit is set or not

~~bot~~ bool bit = num & (1<<pos)

=> Stripping of the lowest bit

x = x & (x-1)

=> Getting lowest set bit of a number

x & (-x)

=> Check if given 32 bit integer is power of 2

return (n && !(n & (n-1)))

↓ Check for 0 (edge case)

=> Find log base 2 of 32 bit integer

```
int res = 0;
while (n >>= 1) res++;
return res;
```

=) Count set bit

```
int solve(n) {
    int count = 0;
  while (n) {
        n& = (n-1);
        count ++;
     }
     return count;
}
```

=) ~~n<<~~ n <<= 1 (multiply by 2)

~~n<<=~~

n >>= 1 (divide by 2)

In general

$n <<= i$ [multiplication by 2^i]

$n >>= i$ [division by 2^i]

=) find position of only set bit in a number

$\log 2(n)+1$

1) uppercase

```
ch |= ' '
```

2) lowercase

```
ch |= '_'
```

3) Invert alphabet's case

```
ch ^= ' '
```