TO DO:-

1) Compile the code in editor
2) left shift → check with compiler (swift) → discarded or circled back. (in swift the edge (MSB & LSB) are discarded)

Operators

- Arithmetic: -, +, *, /, %
- Relational: >, <, >=, <=, ==, !=
- Logical: &&, ||, !
- ★ Bitwise: &, |, ^, <<, >>, ~
- Others: ?:, sizeof, =, -----

a	b	a&b	a|b	a^b	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	0	1	1	0	0

↳ if different → 1
if same → 0

5 : 0101
7 : 0111

```
int a=5, b=7;
int c = a&b;
print(c); = 0101 = 5
c = a|b;
print(c); = 0111 = 7
c = a^b;
print(c); = 0010 = 2
c = ~a; ──────────→
```

int is 4Bytes = 32bit so
5 is 00000------0101 (32bit)

print(c); = -

↳ ~a = ~5 = 11111 - - - 1010
↳ -ve

a & a = a	a & 0 = 0	a & 1 < even 0 odd 1
a | a = a	a | 0 = a	a | 1 < even a+1 odd a
a ^ a = 0	a ^ 0 = a	a ^ 1 < even a+1 odd a-1

} % moduler

a = 5 = 101
a&1 1
001

a=6 = 110
a&1 001
000

a=5 = 101
a|1 001
101

a=6 = 110
a|1 = 001
111

a=5 = 101
a^1 001
100

a=3=011
a^1 1
010

a=6 ⇒ 110
a^1 ⇒ 001
111

a=8 ⇒ ⇒ 1000
0001
1001

Associative and commutative properties (Bitwise)

- $a|b = b|a$
- $a|(b|c) = (a|b)|c$
- $a \& b = b \& a$

left shift & right shift

- <<

7 : 0 0 0 0 0 1 1 1

- discarded or circled back will be defendent on language and compiler.

7<<1 : 0 0 0 0 1 1 1 0 taken = 14 = 7x2

7<<2 : 0 0 0 1 1 1 0 0 = $28 = 7 \times 2^2$

7<<3 : 0 0 1 1 1 0 0 0 = $56 = 7 \times 2^3$

<< every bit is multiplied by 2

so $\boxed{a << i = a \times 2^i}$

- >>

20 : 0 0 0 1 0 1 0 0 → discarded

20>>1 : 0 0 0 0 1 0 1 0 = 10 ⇒ 20/2

20>>2 : 0 0 0 0 0 1 0 1 = 5 ⇒ $20/2^2$

20>>3 : 0 0 0 0 0 0 1 0 = 2 ⇒ $20/2^3$

>> every bit is divided by 2

$$\boxed{a>>i = a/2^i}$$

Question

i/P	O/P
N	
2	$2^2 = 4$
5	$2^5 = 32$
4	$2^4 = 16$
10	$2^{10} = 1024$

```
int  powerOf2 (int N) {
     return  ans = 1<<N
}
```

→ $N \leq 30$ as range of signed int $= 2^{31}-1$

(1u <<N)
type casting to unsigned to include N=31 as unsigned int range $= 2^{32}-1$

(1ll <<N)
type casting to long long $= 2^{63}-1 = \underline{\underline{62}}$

(1ull <<N) $= 2^{64}-1 = \underline{\underline{63}}$

Question:

i/p

N	i	check ith bit is 1
5	2	$\overset{2}{\underline{1}}\ \overset{1}{0}\ \overset{0}{1}$ → true
10	3	$\overset{3}{\underline{1}}\ \overset{2}{0}\ \overset{1}{1}\ \overset{0}{0}$ → true
8	2	$1\ \overset{2}{\underline{0}}\ \overset{1}{0}\ \overset{0}{0}$ → false
7	1	$0\ 1\ \overset{1}{\underline{1}}\ \overset{0}{1}$ → true

→ use right shift (i) and check the number is even or odd
false true

```
func ithbitOne(int N, int i) → Bool {

    return ((N>>i) & 1) != 0

or  return ((N>>i) % 2) != 0

}
```

other implementation

5 : 1 0 1   1 : 0 0 1

$\underset{(i)}{1<<2}$: 1 0 0

Now $5 \,\&\, (1<<\underset{(2)}{i})$

5   1 0 1

$1<<2$ & 1 0 0

$1<<2 \Leftarrow \underline{1\,0\,0}$ $!= 0$ → return true

leturn $\boxed{N \,\&\, (1<<i) \;!= 0}$

or eeturn $(N \,\&\, (1<<i)) == (1<<i)$

$1 \le N \le 10^9$, $0 \le i \le 30$

Question

How many bits are set?

Constraints:

$1 \leq N \leq 10^9$

i/P		o/P
N		
5	101 →	2
16	10000 →	1
12	1100 →	2

```
func  countbits (   N: int ) → Int {
   var count = 0
   for i in 0...30 {
      if (N>>i & 1 != 0) {
          count +=1
      }
   return count

}
```

or

```
for i in 0...30 {
  if (checkbit(N,i)) {
     Count += i;
  }
return count
```

Now 5: $\underbrace{0000 \ldots}_{\text{all zeros}}$ 101 (30 bits) modification

```
while N !=0 {
        if  (N&1)!=0 {
            count +=1
        }
        N = N>>1;
}
```

to eliminate the extra checks for all the zeroes before $\underbrace{0000}_{\downarrow}\ldots101$

these zeroes

N	N-1	N&(N-1)
5 : 0101	4: 0100	4 : 0100
12 : 1100	11: 1011	8 : 1000
16 : 10000	15: 01111	0 : 00000
7: 0111	6 : 0110	6 : 0110

Observation: First set bit of N is being unset in N&(N-1)

N → 24 : 1 1 0 0 0 → unset

& N-1 → & 23 : & 1 0 1 1 1

16 1 0 0 0 0

Using this logic for previous problem countbits

```
func countBits ( N : int) → Int {
        var count = 0
        while (N != 0) {
            count += 1
            N = N & (N-1)
        }
        return count
}
```

Q : Generate a number with only ith bit & jth bit .

i	j	5	4	3	2	1	0	
5	2	1	0	0	1	0	0	→ 36
3	1	0	0	1	0	0	1	→ 10

4 3 0 1 1 0 0 0 → 24

```
func numberwithi&jbitSet( i:Int, j:Int )→ Int {

    return (1<<j) | (1 << i)

}
```

constraints $0 \le j, i \le 30$ int

$0 \le j, i \le 60$ long

(1ll << i)

Question: Generate a # which will have x 1's followed by y 0's

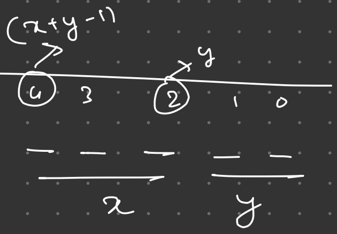
i/P		o/P
x	y	
3	2	11100 → 28
2	2	1100 → 12
4	3	1111000 →

```
func #x1y0 ( x : Int , y : Int ) -> Int {
        var out = 0
    while (x > 0) {
        out = out | 1<<x
        x -= 1
    }

        return out << y

}
```

alternative $x=3$ $y=2$



```
    var res = 0

    for i in y ...(x+y-1) {

        res += 1 <<i

    }

    return res
```

alternative

→ x1 is   y 0's

3   2

→ 11100

8 = 2^3 = 1000 $2^x - 1$

$8-1=7$: $=$ 0111

$$2^4 = 16 = 10000$$

$$2^4-1 = 15 = 01111$$

add $\underbrace{000}_{y}$ (shift << $\underbrace{3}_{y}$)

$$(2^x-1) << y$$

$$\boxed{((1<<x)-1) << y}$$

return $((1 <<x)-1) << y$

$$(2^x-1) << y$$

$$(2^x-1)\cdot 2^y$$

$$(2^{x+y}-2^y)$$

or $(1<<x)(1<<y) - (1<<y)$

or $(1 << (x+y)) - (1<<y)$

$1<<x = 1\cdot 2^x$

$a<<b = a\cdot 2^b$