Bitwise Operators 11/17/21

Unsigned (binary)

int 32 bits $b_{31} b_{30} \cdots b_0 = \sum_{i=0}^{31} b_i \times 2^i$

long long 64 bits

treat 1 = true
0 = false

think of a number as a boolean array

bitwise Operator	symbol
and	&
or	|
xor	^

```
 86 =01010110
&35 =00100011
---------------
   =00000010 =2
```

```
 86 =01010110
^35 =00100011
---------------
   =01110101 =117
```

```
 86 =01010110
|35 =00100011
---------------
   =01110111 =119
```

$(a \mid b) - (a \,\&\, b)$ eq $(a \wedge b)$

AND = Shared qualities

Set intersection

```
Job 01110010
Cand 00000111 &
--------------
   =00000010
```

(# bits on represents # skills candidate has that the job wants)

Bitwise OR

Union

```
My CDs          10010110
|My wife's CDs  01110001
-----------------------
                11110111
```

(CDs we have together)

Grading T/F exam

```
Cor Ans :       10011000
My Ans : ^      00101100
-----------------------
                10110100
```

(0's are correct, 1's incorrect)

Bit shift op	Symbol
Right Shift	>>
Left Shift	<<

87 >> 3 = 10

↳ 1010|111 (chop off)

n >> k is int div by 2^k.

5 << 2 = 20

101 ⟹ 10100

mult by 2^k, k = # bits shifted

$5 \times 2^{(2)}$

Is the k^{th} bit of n on?

```
if (((1 << k) & n) != 0)
        n = 10101111
& 1 << k  = 00010000
```

Shines a flashlight on bit k.

k = 4

2's complement

most sign bit is negative

$$b_{31} b_{30} \cdots b_0 = -b_{31} \times 2^{31} + \sum_{i=0}^{30} b_i \times 2^i$$

8 bit 2's complement

$$10011011 = -2^7 + 2^4 + 2^3 + 2^1 + 2^0$$

$$= -101$$

~~If the positive part is~~

(1) Flip bits
(2) Add 1
(3) Negate

$$\begin{array}{r} 01100100 \\ + \qquad\quad 1 \\ \hline 101 \end{array}$$

$$-101$$

$\sim x \rightarrow$ flips all the bits

$\sim$ ↓ NOT

$$\sim x = \boxed{-x-1}$$

```
int numbitson(int n) {
    int res=0;
    for (int i=0; i<32; i++)
        if((1<<i) & n) != 0)
            res++;
    return res;
}
```

Problem

(n) array ints [12, 92, 13, 6, 47, 88]

target 145

```
int subsetsum(int * vals, int n, int target) {
    for (int mask=0; mask<(1<<n); mask++) {
        int sum = 0;
        for (int i=0; i<n; i++)
            if ((1<<i) & mask) != 0)
                sum += vals[i];
        if (sum == target) return 1;
    }
    return 0;
}
```

0000
0001
0010
0011
0100
0101
⋮

Exercise: Create array size 2^n where sum[i] stores sum of values for subset with mask i.