COP 3502 11/12/25

② Bitwise Operators

① Base Conversion?

$$967 = 9\times10^{2} + 6\times10^{1} + 7\times10^{0}$$

10 ↑ base (# symbols you're using)

Symbols: 0-9, A-Z (base 36)

Generally ≤ 16 base

~~Ster~~ Alg 1: Convert from base b to base 10

$$432_6 = 4\times6^{2} + 3\times6^{1} + 2\times6^{0}$$

6 ↑ base

$= 4\times36 + 18 + 2$

$= 144 + 20$

$= 164$

$6\times(6\times(4+3)+2)$

res 0 → $6\times0+4$ = 4 → $6\times4+3$ = 27 → $6\times27+2$ = 144

```
int convert (char* num, int base)
{
    int res = 0;
    int len = strlen(num);
    for (int i = 0; i < len; i++)
        int value = convert(num[i]);
        res = base*res + value;
    }
    return res;
}
```

Alg #2 : Convert base 10 to base b

$164_{10} = ---_4$

$= a\times4^3 + b\times4^2 + c\times4^1 + d\times4^0$

(Divisible by 4)

$= (4)X + (d)$ → isolate d via mod remainder

4 | 164		
4 | 41	R0	
4 | 10	R1	
4 | 2	R2	
0	R2	

2210

$= 2\times4^3 + 2\times4^2 + 1\times4^1 + 0\times4^0$

$= 128 + 32 + 4 = \boxed{164}$

$692_{10} = ---_{16}$

16 | 692		
16 | 43	R4	4
16 | 2	R11	(B)
0	R2	2

16 = hexadecimal

0xf3 byte

1111 0011

2 = binary

3 = ternary

8 = octal

Alg #3 : Base b_1 to Base b_2

$b_1 \rightarrow 10 \rightarrow b_2$ (use algs above)

Alg 4: b_1 to b_2 when both are powers of 2

$b_1 \in \{2, 4, 8, 16\}$

$b_2 \in \{2, 4, 8, 16\}$

$b_1 \rightarrow 2 \rightarrow b_2$

padding prepend

$736_8 2104115_8 =$

$16 = 2^4$

00 111 011 110 010 001 000 100 001 001 101 $_2$

3 B C 8 8 8 4 D $_{16}$

$a8^2 + b8^1 + c$

$a \cdot 2^6 + b \cdot 2^3 + c$

$(a_1 2^2 + a_2 2^1 + a_3 2^0) \times 2^6 + (b_1 2^2 + b_2 2^1 + b_3 2^0) 2^3 + (c_1 2^2 + c_2 2^1 + c_3 2^0)$

$\hookrightarrow a_1 2^8 + a_2 2^7 + a_3 2^6 + b_1 2^5 + b_2 2^4 + b_3 2^3 + c_1 2^2 + c_2 2^1 + c_3 2^0$

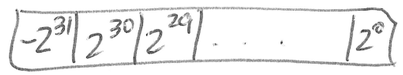
this is why this WORKS!

2^8 2^4 2^0

HEX

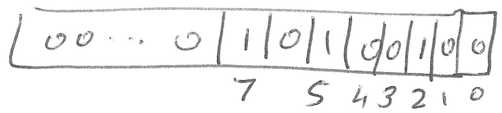
Bitwise Operators

In the computer, integers are stored in binary (2's complement)



We can think of an integer as a ~~binary~~ boolean array of size 32

$$164 = 2^7 + 2^5 + 2^2$$



128
16

If you need a boolean array of size 32 or less, you can use a single integer instead.

Learn how to

(a) read a particular bit

(b) write a particular bit

Several bitwise operators that will help us with (a) + (b), AND make certain calculations very easy (with a array loop would be required but with operators only + single stmt is needed)

Ops	
&	bitwise and
|	bitwise or
^	bitwise xor
>>	bitwise right shift
<<	bitwise left shift

```
  10110010  →  128+32+16+2 = 178
& 01101011  →  64+32+16+3 = 115
------------
  00100010      178 & 115 evaluates to 34
    ↓   ↓
   32   2
```

If these were 2 arrays, I'd have to do a loop lists which items are common to both groups

X & Y
↳ friends of person 1
↳ friends of person 2

X & Y represents subset of common friends btw person 1 person 2

bitwise or |

```
  10110010   178 | 115
  01101011
------------
  11111011   = 255-4 = 251
```

Essentially set union