WEEK 2

$\boxed{10/4}$ - 1, 2, 3, 4, ... are representations of numbers

- base 10, 60, 2, 8, etc.

- $\mathbb{N} = \{1, 2, 3, \ldots\}$
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\mathbb{Q} = \{\frac{a}{b} : a, b \in \mathbb{Z}\}$

- $\mathbb{R} = \mathbb{Q} \cup$ (irrational #s) $(\mathbb{R} - \mathbb{Q})$
- $\mathbb{C} = \mathbb{Q} \cup$ (imaginary #s)
- $\pi, e, \varphi \in \mathbb{Q}$
- $\mathbb{C} = i$, $a + bi$ tho $a, b \in \mathbb{R}$

- $\mathbb{Z}$ = int, $\mathbb{Z}^+$ = + int, $\mathbb{Z}^-$ = − int
- $0 \neq$ + or −
- natural #s $\mathbb{N} = \{0\} \cup \mathbb{Z}^+$

bits	min	max
8	-128	127
16	-32768	32767
32	-2147483648	2147483647
64	-9223372036854775808	9223372036854775807

bits	min	max
8	0	255
16	0	65535
32	0	4294967295
64	0	18446744073709551615

- computers do arithmetic in finite fields (large set of #s)
- digit for computer = bit (either 0 or 1)

- INTS

unsigned } short, long, long long } int

unsigned } char

char = 8 bits
short = 16 bits
int = 16 bits
long = 32 bits
long long = 64 bits

```
#include <stdint.h>
```

signed	unsigned	bitsize
int8_t	uint8_t	8
int16_t	uint16_t	16
int32_t	uint32_t	32
int64_t	uint64_t	64

Binary Arithmetic

0+0=0 1+1=10 0×0=0 1×1=1 101+11=1000
1+0=1 1×0=0 101×101=11001

- finite fields have additive inverses for every #
  - ex) $i = \#$, $\bar{i}$ = add. inverse $\Rightarrow i + \bar{i} = 0$
  - k bits $\Rightarrow$ can have every (+) int from 0 to $2^k - 1$
    - ↳ half will be add. inv so range = $-2^k - 1$ to $2^k - 1$

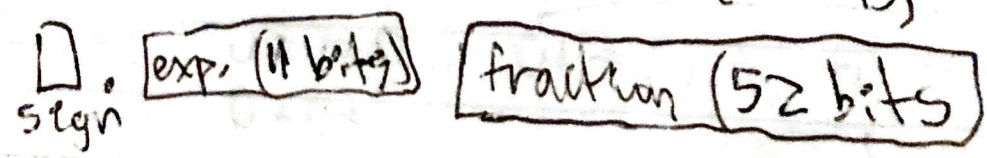
- Two's Complement (2C)
  - to get add inverse of m:
    - flip bits of m (0 to 1 & 1 to 0) (~ operator)
    - add 1 to result → add. inverse
    - if 2C bits > OG #, drop/carry highest-order bit
- Real #s = continuous or uncountably infinite
- Floating Point #s
  - proper subset of real #s ($\mathbb{F} \subsetneq \mathbb{R}$)
  - proper subset of rational #s ($\mathbb{F} \subsetneq \mathbb{Q}$)
  - proper subset of ints ($\mathbb{F} \subsetneq \mathbb{Z}$)
  - not reals or rationals $\Rightarrow$ FPNs are approximations

- float = single precision = 32 bits (IEEE 754 single)
- double = double precision = 64 bits (IEEE 754 doub
- long double = extended precision = 128 bits (IEEE 754 quad)

- float $\not\subset \mathbb{Q}$

- Single Precision =

- double precision =



- intel extended precision [1] . [exp. (15 bit)] [(1 bit)] [fraction (63 bit)]
  sign — int part

- quad precision = 1 bit sign · 15 bit exponent · 112 bit fraction

- floating point limits
  - float = 8 bit exponent, 23 bit mantissa
  - double = 11 exp, 52 mant
  - long double = 15 exp, 112 mant

- Little Endian = lowest address byte
- Big Endian = highest address byte } has least sig bit
  ↳ LE = [78][56][34][12] vs. BE = [12][34][56][78]

- modulo = % = ints only

- x/%
  +-
  << >>
  < <= > >=
  == !=
  &
  ^
  |
  &&
  ||
  } Left associativity

  ?:
  =
  +=
  -=
  x=
  /=
  %=
  } right associativity

- 8 bits = 1 byte

REMINDERS FOR ASCAN2

- $\log(x) = -\log\left(\frac{1}{x}\right)$ for $x > 0$ } invert exponents
- $\log(e^x) = e^{(\log x)}$
- $x^{1/k} = \sqrt[k]{x}$
- binary search solutions = monotonic
- $x_{k+1} = \frac{1}{2}\left(x_k + \frac{n}{x_k}\right) \Rightarrow$ sqrt_newton
- computer arithmetic approximates #s
  - ↳ round to finite representations
- round ↑ = ceiling, ↓ = floor, to 0 = truncate

- (+) = shift & ~~[illegible]~~ addition
- (/) = shift & subtraction
- (+) = more than exclusive - XOR

- abs(x) = if $x < 0$, then $-x$; else x

- libraries should be numerical precise, fast, & portable

- Taylor series $\Rightarrow f(x) = \sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n$
  - ex) e^x centered at $a=0$ is $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!}$ $-\infty < x < \infty$

- $\sum^{n}$ should be for loops, $\sum^{\infty}$ should be while loop

- computing e^x
  - $k! > e^k$ for all x
  - when k is large enough
    - $\frac{x^k}{k!}$ gets smaller w/ each step, can stop computing when small enough

- Computing sin (periodic $[-2\pi, 2\pi]$)
  - $\sin(x) = x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \frac{x^9}{362880} \cdots + O(x^{12})$
- series converges vvv slowly unless $x \approx 0$
- don't directly compare floating point #s
  ↳ check to epsilon ε (absolute error)
- $\frac{|x^* - x|}{|x|} < \varepsilon$ (relative error → more accurate)