24/09

Sieve of Eratosthenes - to find prime numbers.

list (infinite) of numbers . . .

~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ 7 . ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13
~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ ~~25~~
~~26~~ ~~27~~ ~~28~~ 29 ~~30~~ 31 ~~32~~ ~~33~~ ~~34~~ ~~35~~ ~~36~~ 37
~~38~~ ~~39~~ ~~40~~ 41 ~~42~~ 43 ~~44~~ ~~45~~ ~~46~~ 47 ~~48~~ 49.
- - - -

remove numbers divisible by 2, then 3, then 5 - - .

```
> (define (sieve s)
      (cons-stream (stream-car s)
                   (sieve .
                     (stream-filter
                        (lambda (x) (not (divisible? x
                                   (stream-car s))))
                     (stream-car s))))).

> (define primes (sieve (integers-starting-from 2))).
> (stream-ref 5 primes).
  13
> (stream-ref 100 primes)
  547 .
```

0	1	2	3	4	5
2	3	5	7	11	13

```
> (define primes
     (cons-stream 2
        (stream-filter prime? (integers-starting-from 3)))
```

this also works

using the smallest divisor procedure

```
>(define (prime2? x)
    (define (iter ps)
       ((cond ((> (square (stream-car ps)) x) true).
              ((divisible? x (stream-car ps)) false).
              (else (iter (stream-cdr ps))))).
    (iter primes)).
>(define primes (cons-stream 2 (stream-fitter prime2?.
                 (integers-starting-from 3)))).

>(stream-ref 5 primes)
  13
>(stream-ref 100 primes)
  547
```

Defining primes ~~[illegible]~~ by checking for prime2? ness.

prime2 - checking primeness using primes we've found till now.

Can generate 2 infinite streams using each other

primes had 2. promise initially.

The numbers we are checking divisibility with are already generated

Stream of guesses to compute square root

```
>(define (sqrt-stream x)
    (define guesses
       (cons-stream 1.0.
          (stream-map (lambda (guess)
                         (sqrt-improve guess x))
                      guesses))).
  guesses).
```

```
>(define sqrt2s (sqrt-stream 2)).
> sqrt2s
>(1.0. #<proc-------->).
> (stream-ref 1 sqrt2s)
 1.5
> (stream-ref 5 sqrt2s)
 1.414213562373095
```

Merging withdraw amounts for joint accounts
fair way - round robin).

One can't withdraw money if they just had and the other still hasn't

Time doesn't go well with the ~~s~~stream paradigm

Haskell

Function: Map from values in a domain D to range R

Type signature gives contract for a function

List in haskell is written in [ ].

Input: an integer, list of characters.

Int, [char).

Output: [Char).

commonWords :: Int → ([Char] → [Char]).

In Haskell, internally all functions take only one input at a time like the second way of writing 'add'.

This is called currying.

They supply functions forward.