10/9/24 class-18:-

→ In untyped languages, (( ), . . .) → quite possible.

→ Enumerating mixed up with all other parts.

→ larger programs are written using sequences.

12/9/24 class-19:-

→ (car (cdr (stream-filter prime? (enumerate-interval 10k 1Million

↓ elangant but inficient

→ we want to reach to efficiency where we start with 10k & end when 2nd prime detected.

STREAMS:-

→ the life changed a lot with assignment statements ⇛

→ Let us see this alternative view, where no assignment

→ chalk → have a path (x, y, z)

einstien → chalk is itself path.

* STREAMS as data structures:-

Constructor: (cons-stream a b) ⇒ outputs stream (say 's')

Selectors: (stream-car s) ⇒ a

(stream-cdr s) ⇒ b

Utility operations: (the-empty-stream ⇒ '() → empty list for now

(Steam-null? s)

⇒ (eq? s the-empty-stream)

(define (stream-ei low high))

(if (> low high)

the-empty-stream

(cons-stream low (stream-ei (+ low 1) high)))))

```
(define (stream-filter pred s))
  (cond ((stream-null? s) the-empty-stream)
        ((pred (stream-car s)) (cons-stream (stream-car s)
                                            (stream-filter (stream-cdr s))))
        (else (stream-filter (stream-cdr s)))))
```

streams just look like lists

```
(stream-car (stream-cdr (stream-filter prime? (stream-ei 10k 1M))))
```

What is different?

```
(cons-stream a b) = (cons a (delay b))
(stream-car s)    = (car s)
(stream-cdr s)    = (force (cdr s))
```

delay & force are IMP here

delay is like a promise

force → forces the promise

thunk → when you pass a delay to force, we get a data structure thunk

→ how to (delay <exp>)

done in church ... wala lab,

We put it inside a lambda, we evaluate it when lambda called

```
(delay <exp>) => (lambda () <expr>)
(force promise) => (promise)
```

→ (stream-ci 10k 1m) = (cons 10k (delay (stream-ei 10k+1 1m

This is a stream, which is a data structure which isn't evaluated completely.

I have gotten it. I can't show it to you.

(there is a pedalogical reason, why can't be written)

now filter does

(stream-filter (cons 10k+1 (delay (stream-ei 10k+2 1m

⋮

(stream-filter (cons 10007 (delay (stream-ei 10k+8 1m

= (cons 10007 (delay (stream-filter (cons 10008 (delay (stream-ei 10k+9 1m

⇒ stream-cdr forces this

(cons 10007 (stream-filter (cons 10008 (delay (stream-ei 10k+9 1m

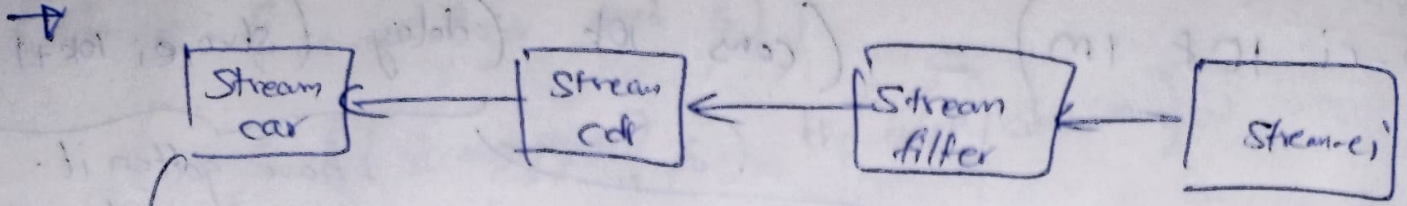
(cons 10007 (stream-filter (cons 10009 (delay (stream-ei 10k+9 1m

(cons 10007 (cons 10009 (delay (stream-filter (cons 10010 (delay ...

cdr gives

stream-car (cons 10009 (delay (— — —)))

"10009"



If cdr is kept
we get list of all prime number from
10k+10 to 1M

In haskel, ":" ⇒ cons