Part 1: section 3.4

Review: last time introduced the assignment and state
Frightening! Substitution model breaks down!

Suddenly variable is a place to hold a value !!!

So (f x) might change during the time of the program.
Pairs has identity as objects. Therefore two same pairs might be different with sharing concept.

Errors with bad sequencing and aliasing...

How we got I to this mess?

We wanted to build modular systems, systems fall apart into chunks that seems natural, pieces that mirrors the part of the system that represent real world.

Ex. Digital circuits, parts have identity and state.

Maybe the reason building systems like is introduce such a technical complication has nothing to do with computers!

Maybe the real reason we pay such a price to view the realty is that we have the wrong view of the reality.

Maybe time is just and illusion!! Similarly signal processing engineer look at the whole circuit we don't think of the state of each part of the circuit, we see the input signal and output signal, and this box as a filter as a whole.

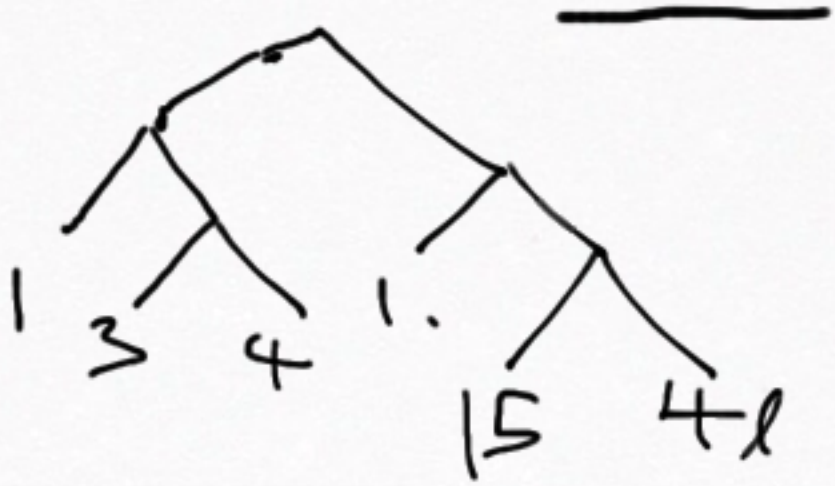
Today

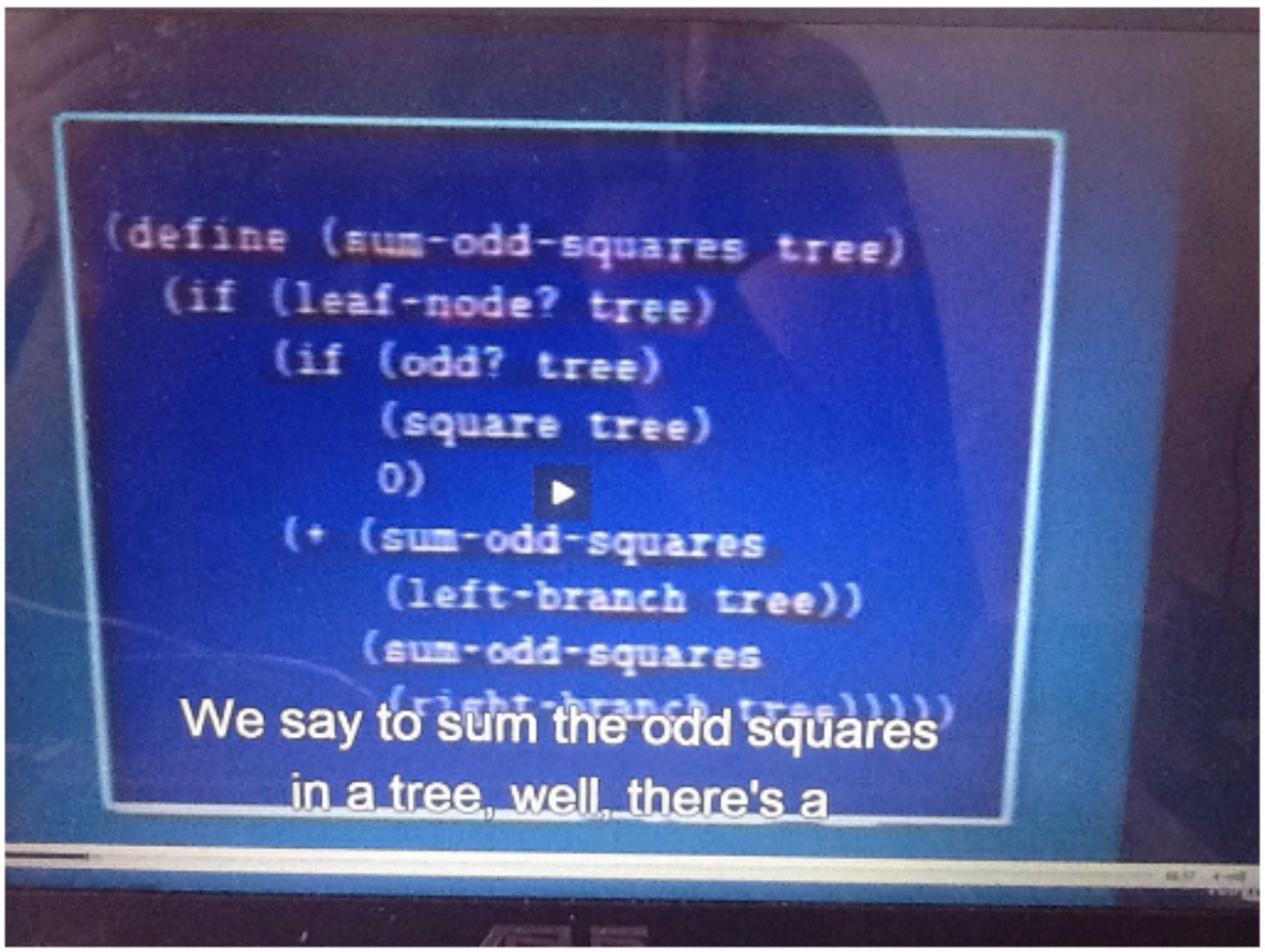
We are going to look at the another way to decompose systems that is more like the signal processing view of the world, called stream processing.

How we can make our program more unified by finding more commonalities and ignoring the time in our program.

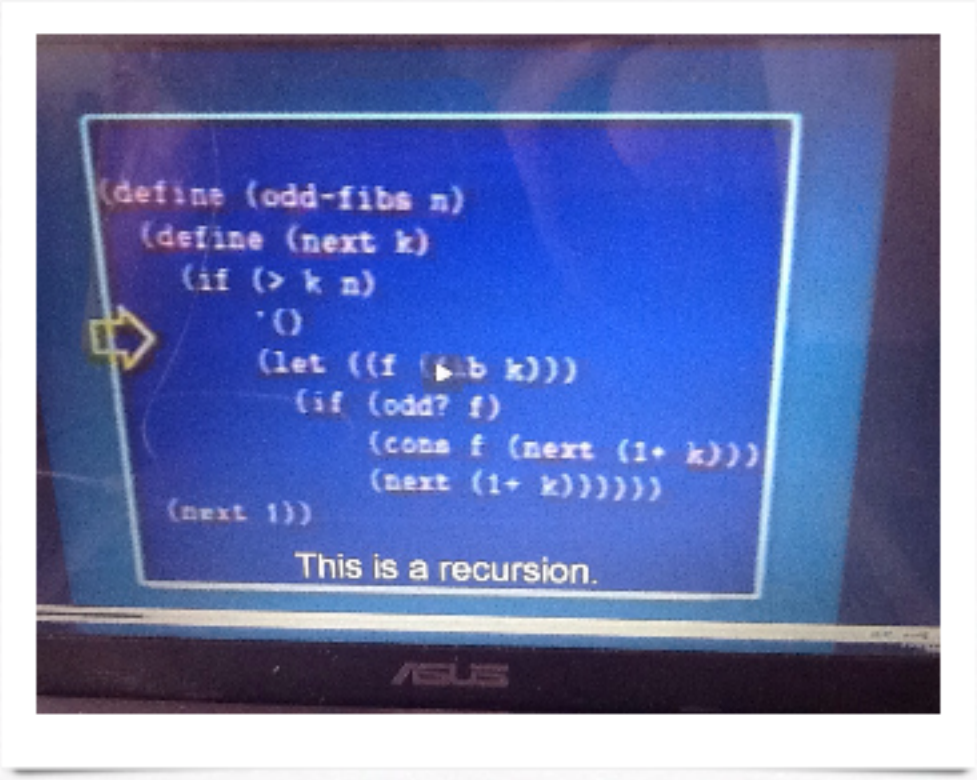
First example: there is a integer tree and we want to find square of odd numbers and sum them up.

We can do it recursively sum comes from left branch and right branch. if odd sum square if not sum 0

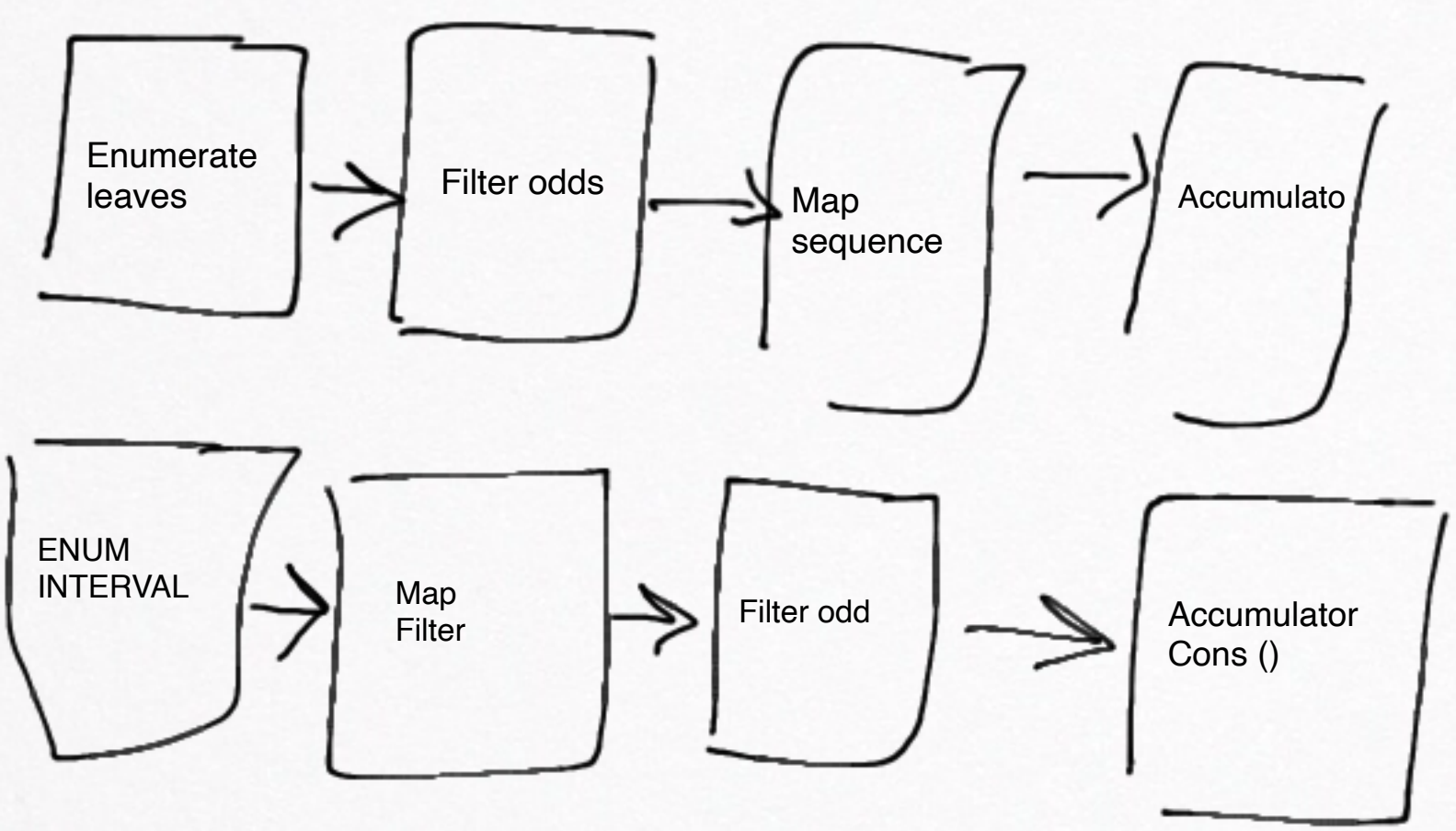




Now let's see the second program, for integer k find fibbonatchi number and find the odd numbers. Here is a procedure for doing that:



There are two procedures which seems different but are very similar! Let's look from signal processing view:



But when we look at the program we don't see this commonalities.

Ex: where is the enumerator?
It's embedded it the code, is not in the one place either.

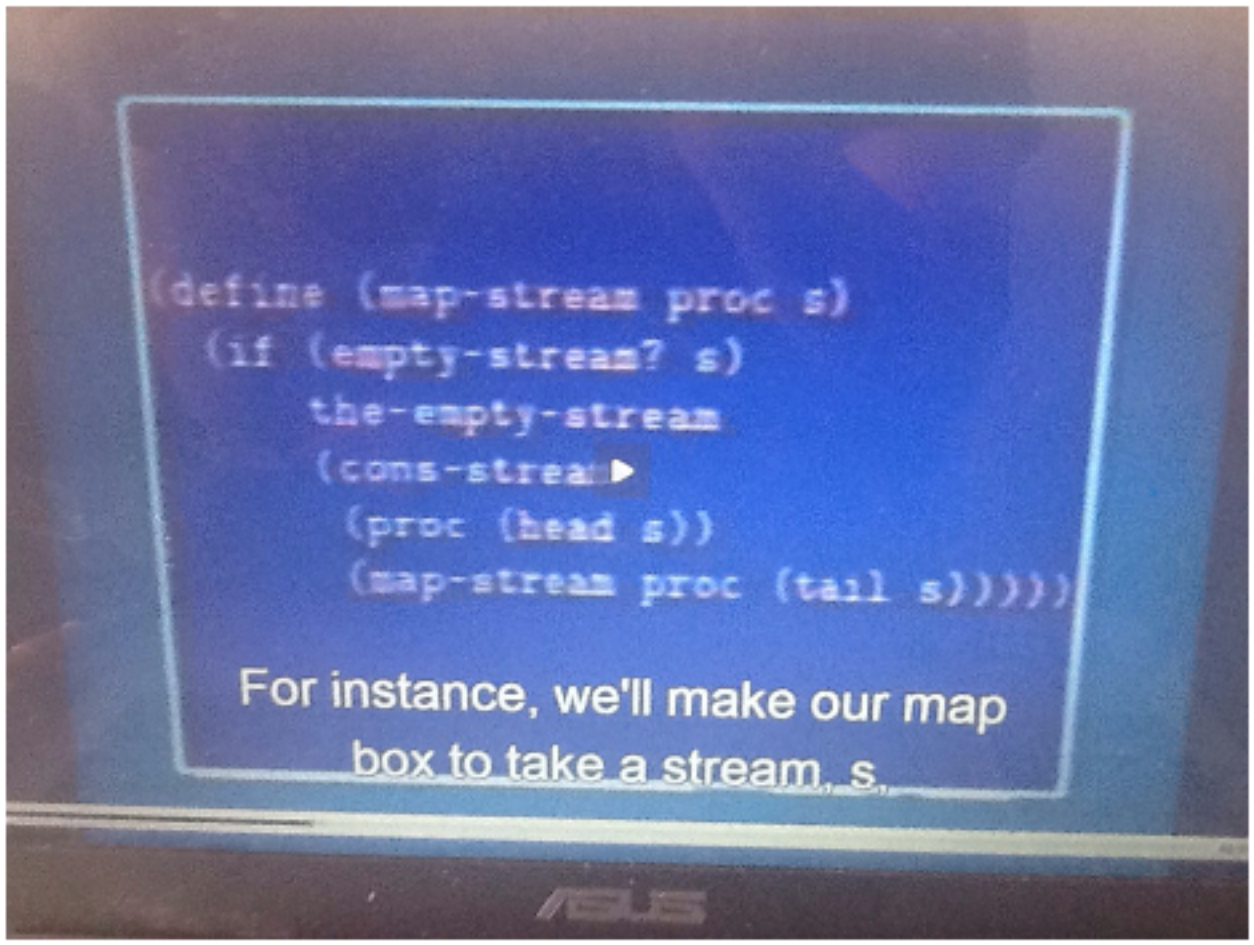
Mixed up, program
don't chops them in
right way!!!

Going to the fundamental principle of computer science, in order to control something you have to name it. Therefore we don't have the language to talk about streams right now.

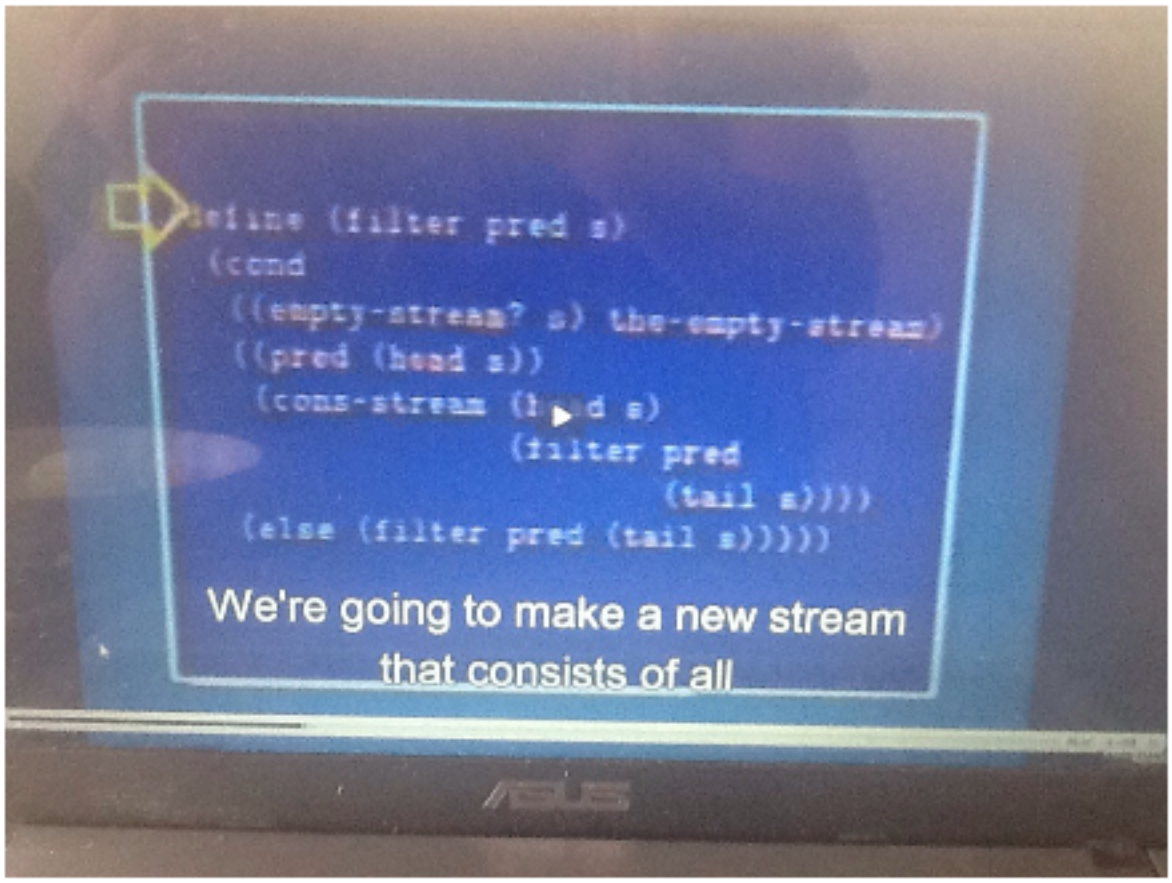
What's stream? Data abstraction with data selector and maker:
(Cons-stream x y) and (head s) and (tail s) and the empty-stream

(Head (cons-stream x y)) x and (tail (cons-stream x y)) y

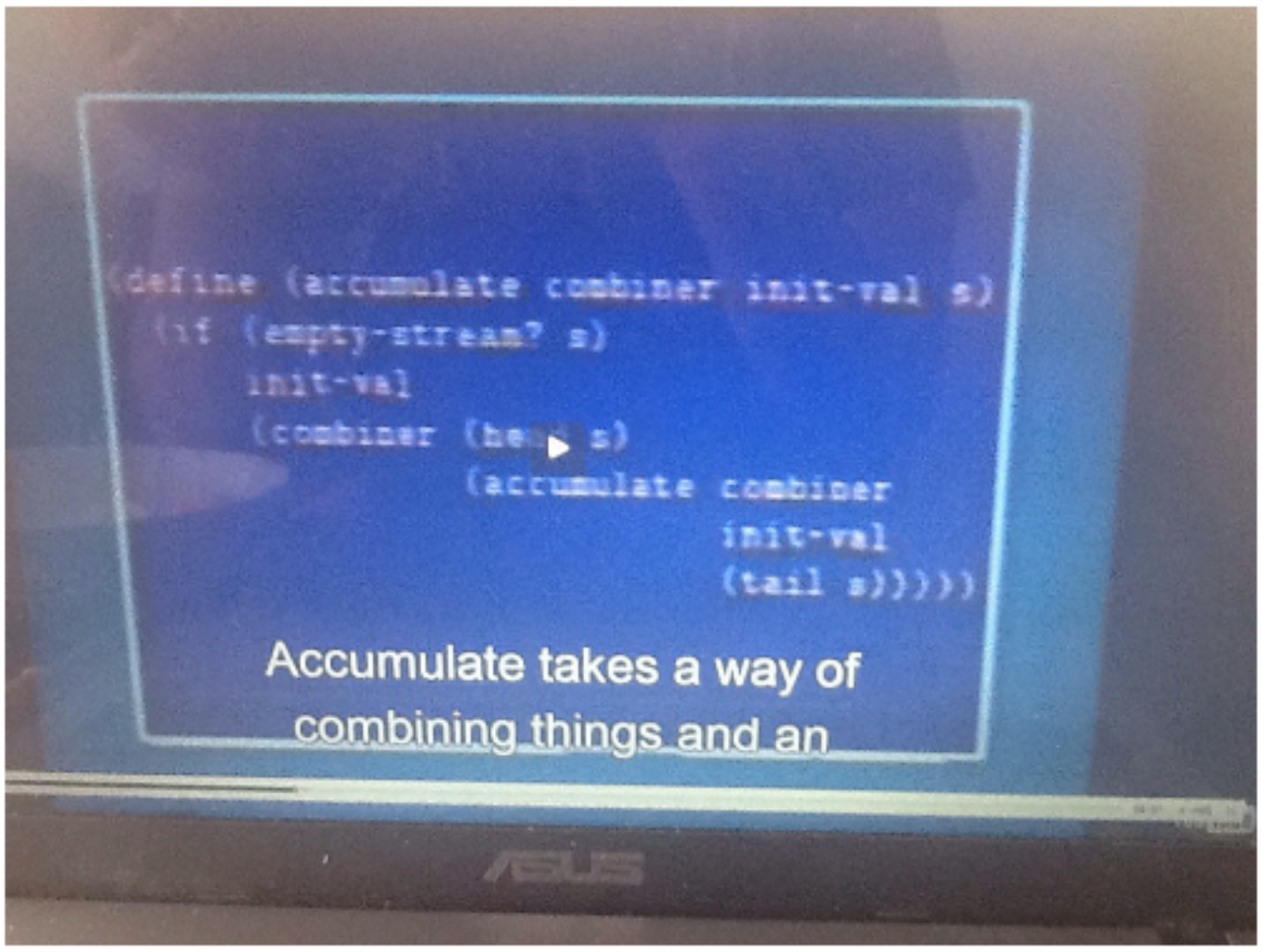
Now we can use the language to make useful things.
Our Map Stream:



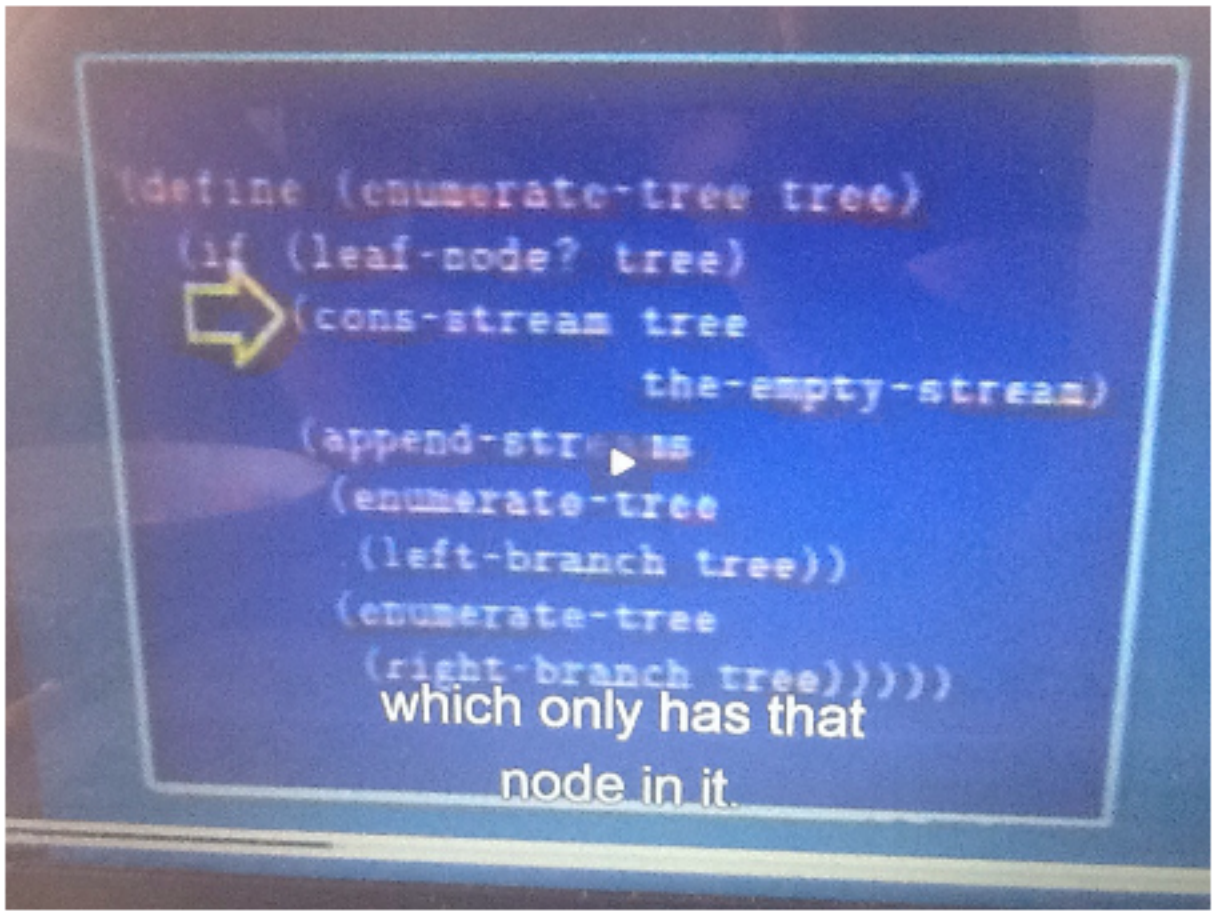
And here is our filter box:



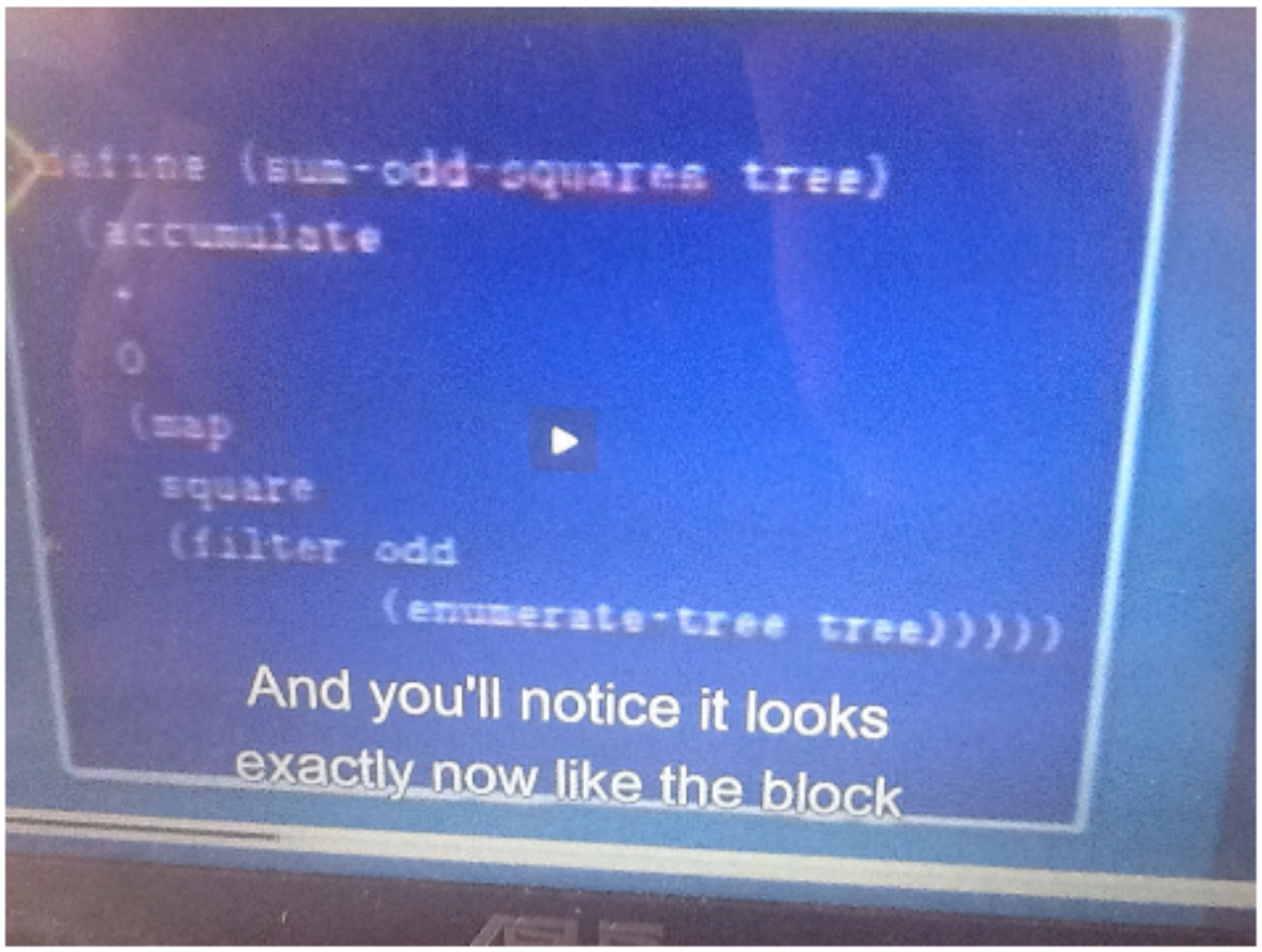
Here is the accumulate:



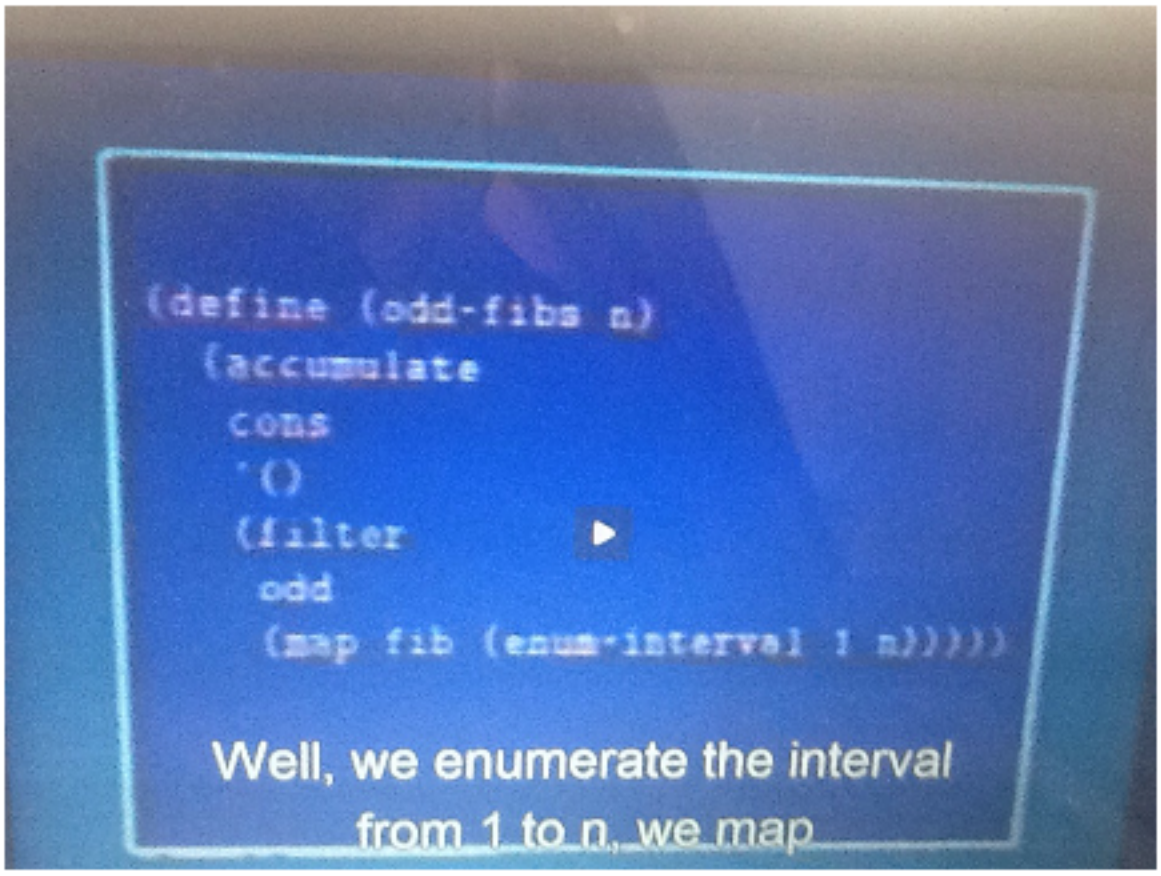
Enumerate the leaf of



Here now we have language to describe the first program:



Similarly for the second example:



Advantage?

Now we have parts to mix and match !!

Therefore we are establishing conventional interfaces, that allow us to glue things together and to see commonalities into our programs.

Part 2: Section 3.4

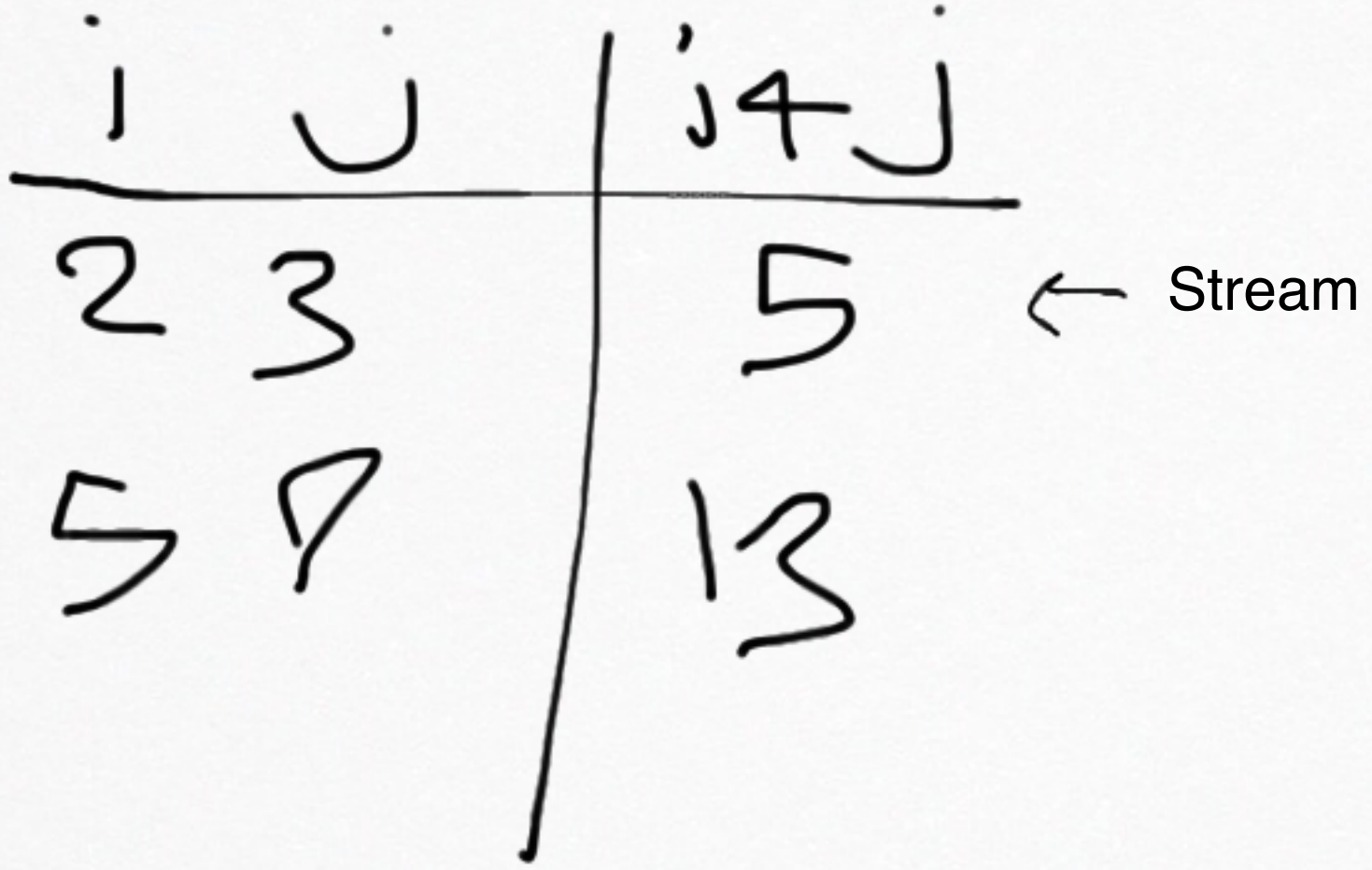
Now let's see so e more complicated examples.

Suppose we have stream of stream and we want to put all the stream into one stream.

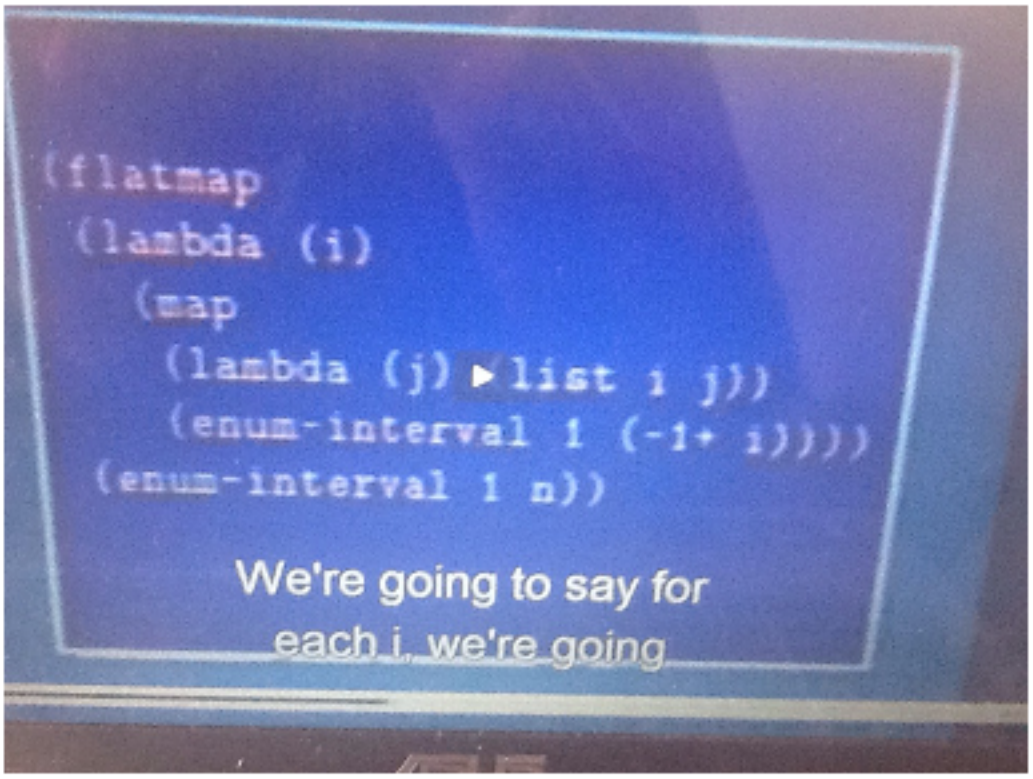
(Define (flatten st-of-st) (accumulate append-stream
The-empty-stream
St-of-st))

(Define (flatmap f s)(flatten (map f s))). But why to use it?

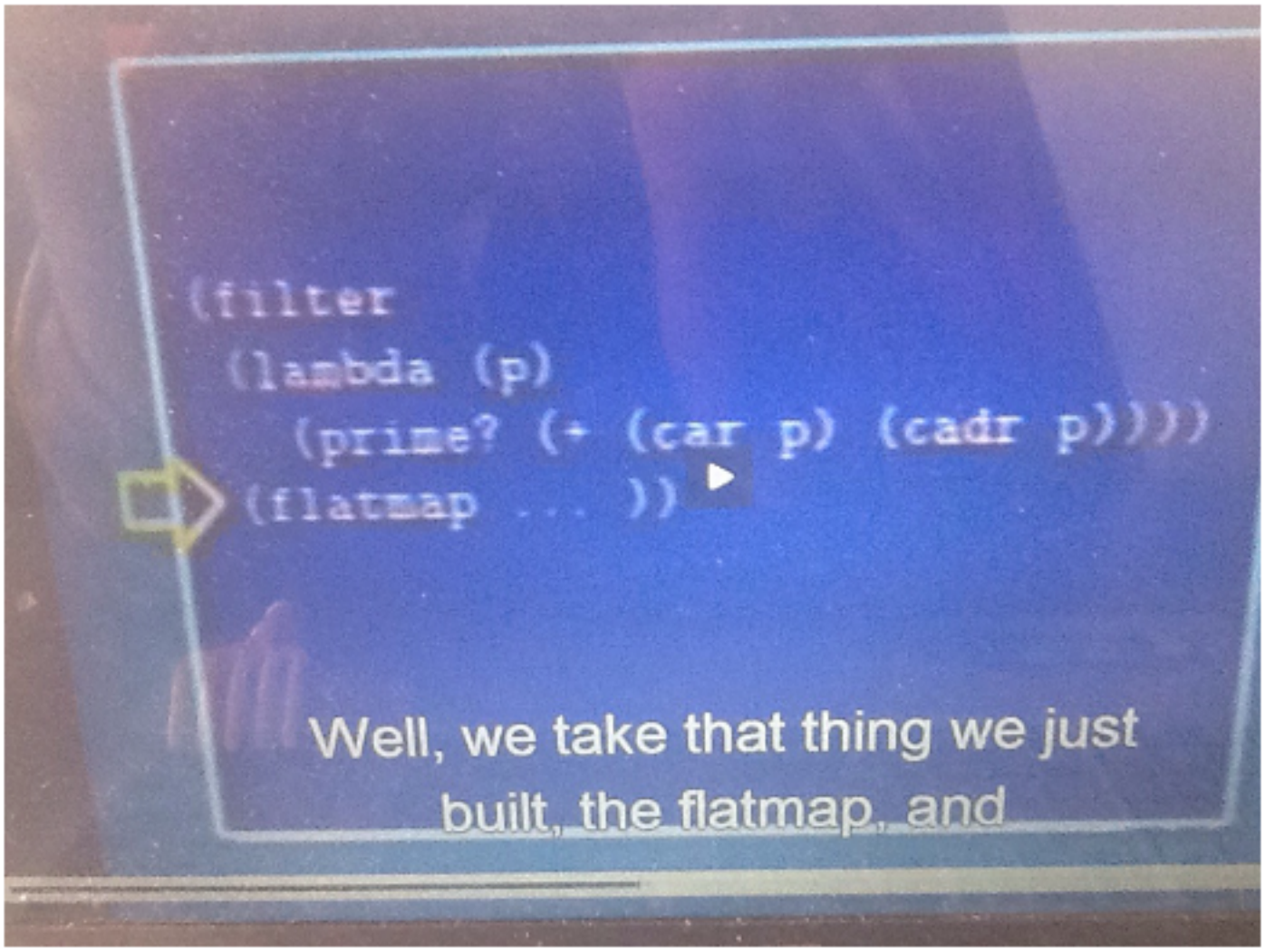
Problem: GIVEN N FIND ALL PAIRS 0 < j < n such that I+j is prime



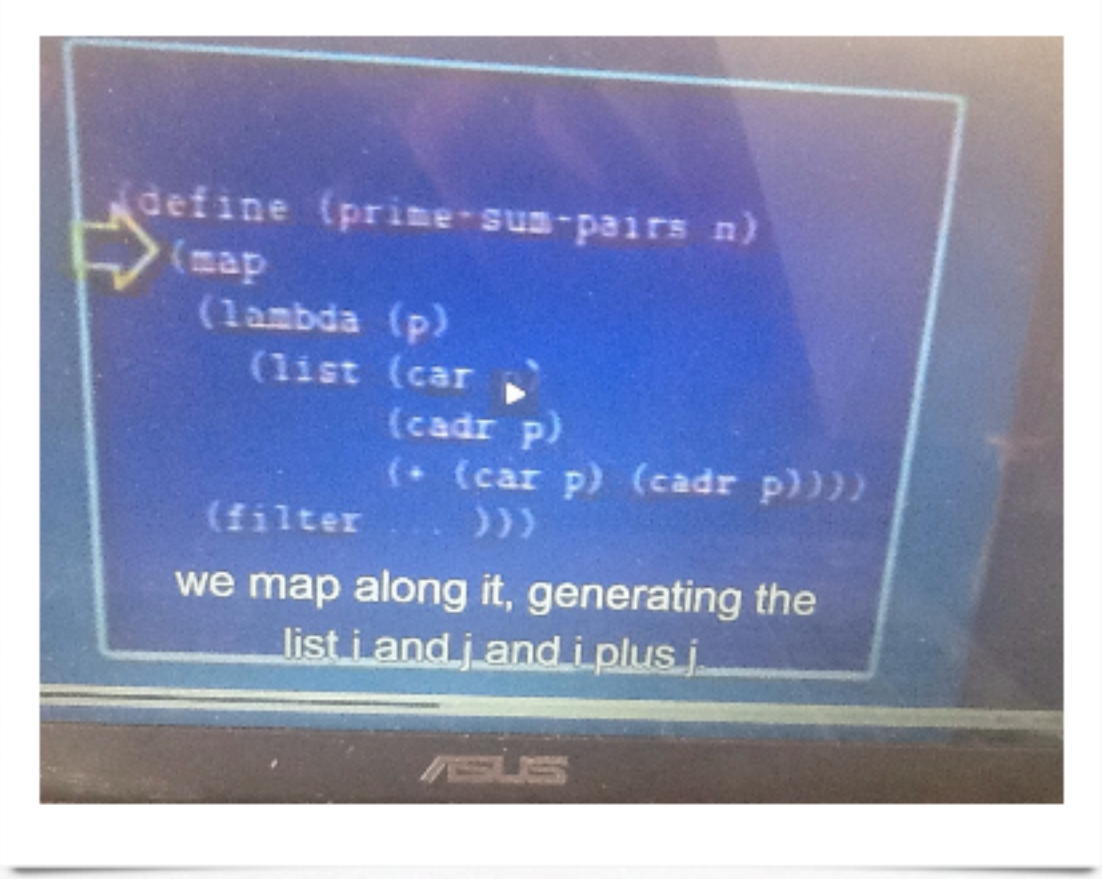
Here is the program:



Now we are going to test it:

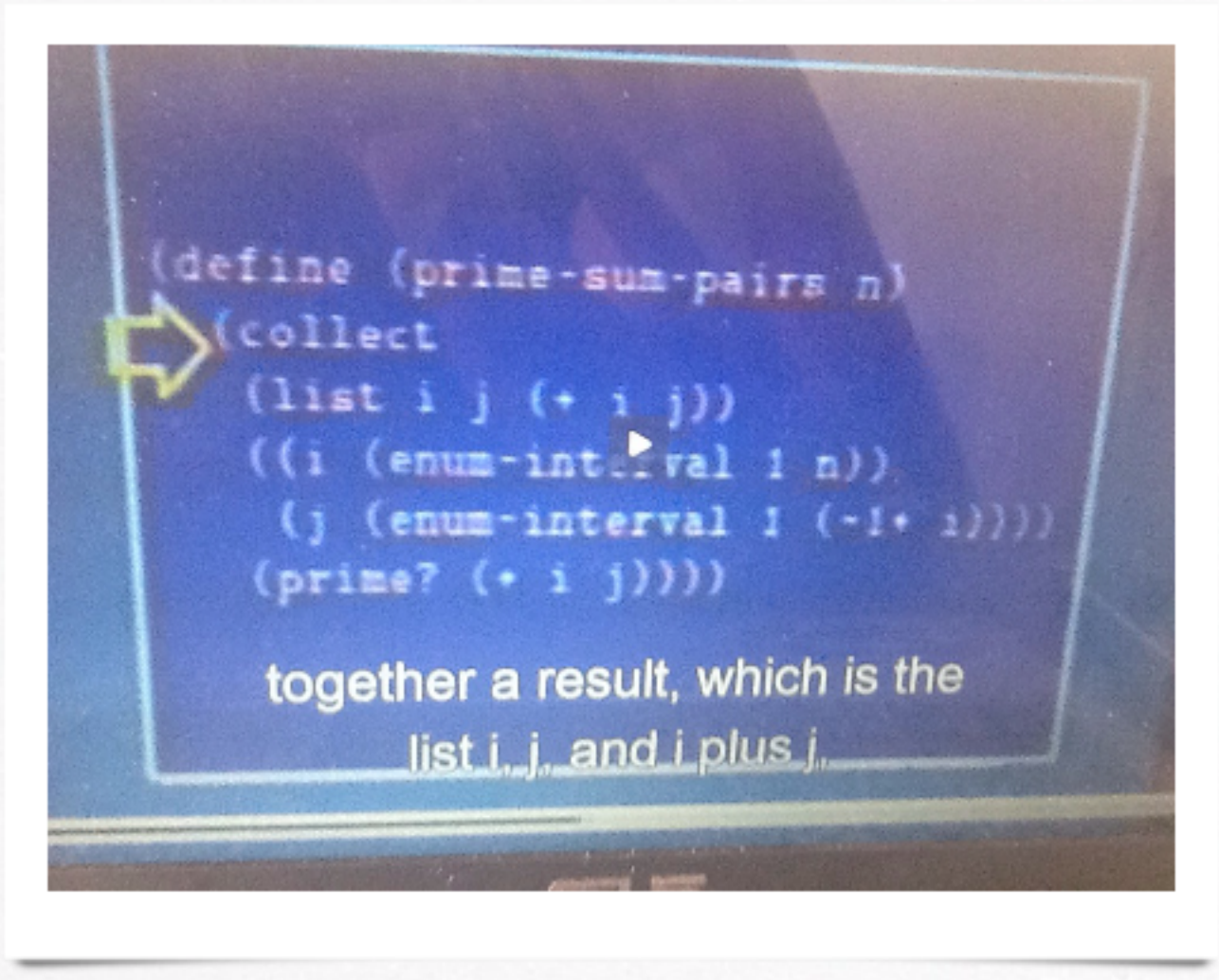


And the we are going to map it:



So the whole procedure is (map (filter (flatmap ...

So we can do this nested loops of flatmap and flat, but we can syntax sugar it with a function named collect:



And here is another problem named eight queens problem in no same column and row and diagonal.

First we should go to the George's level to find a way to represent the board. Assume we have function called safe that says is it Ok to have queen in this spot?

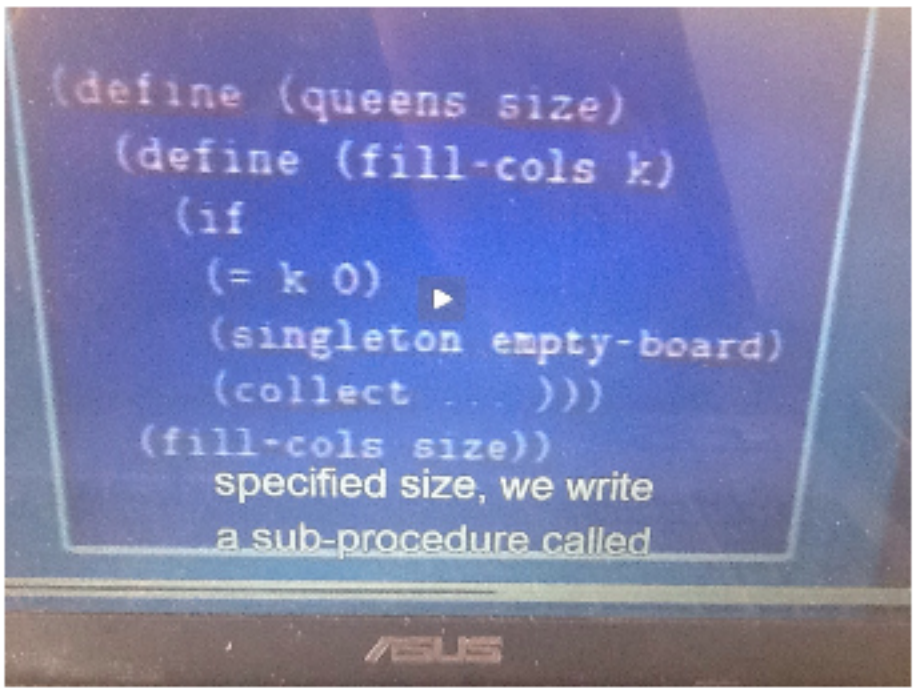
(Safe? <Row> <column> <rest-of-the-positions>)

Given this how to organize the program? Backtracking search

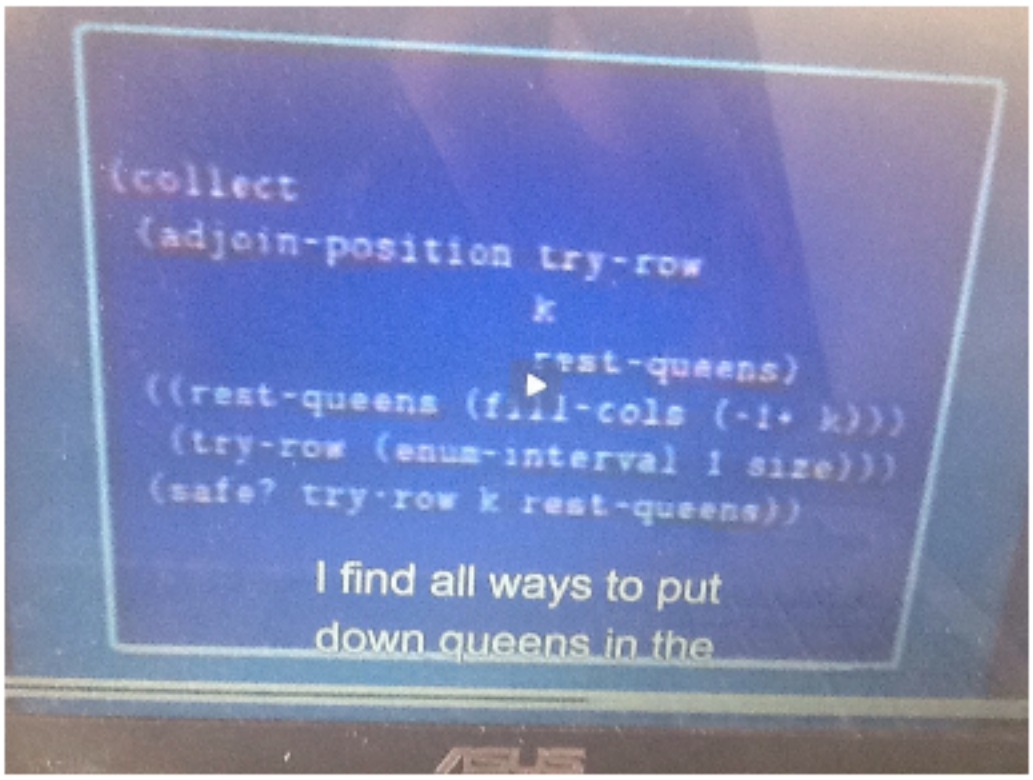
Next branch level is the next row
and we see if we can put in there

Somehow it is too much complicated , why? Because we are trying too much in time. I if we can stop worrying about time...

Let's imagine I have the answers in the kth column. Then for the next row we put queen in all rows and then filter it the ones that are safe, let's look at it:



And here is the collect in the last procedure:



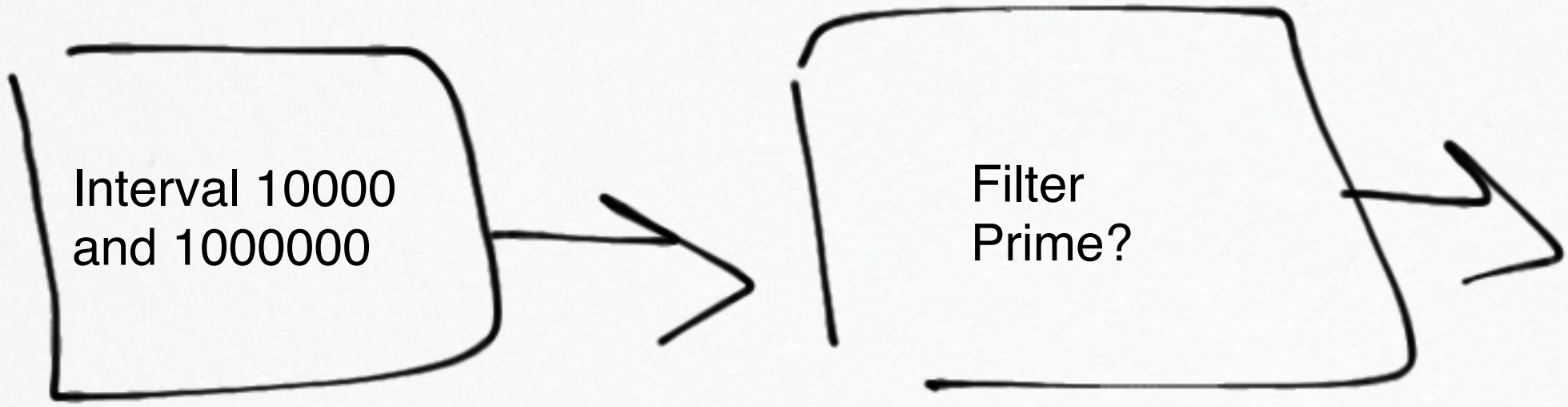
Part3: section 3.4

Now you should question this methods efficiency, there got to be a catch!

... FIND THE SECOND PRIME BETWEEN 10,000 and 1000,000

```
(Head
  (Tail (filter
          Prime?
          (ENUM-interval 10000 1000000))
```

Second element

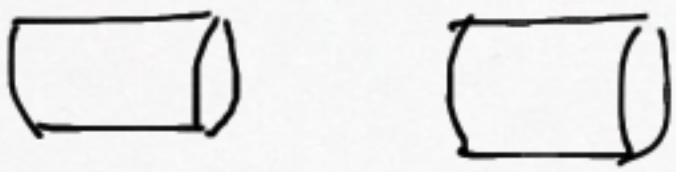


Pretty ridiculous! Not room in the machine, the power of this programming style is this weakness!

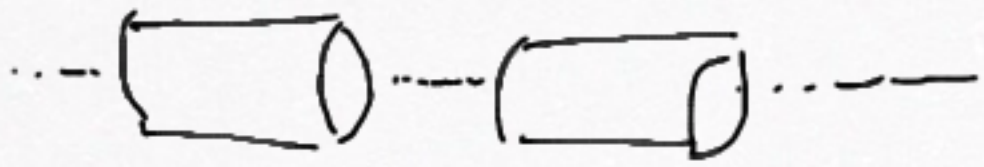
Turns out, we can do something for it to run efficiently ,
the key is streams are not lists!!

Lists:

(List)



Streams:



Stream is like a data structure make itself on demand,

Basic idea comes from the very basic idea that there is not firm distinction between data and program. We already have everything because we have functions as the first order citizens.

(Cons-stream x y) is abbreviation for (cons x (delay y))

(Head s) is (car s)

(Tail s) is (force CDR s))

Delay is promise to compute y

Therefore the (E 10000 1000000) is
(cons 10000 (delay (E 10001 1000000)))

Filter is not going to generate something more than you want.

All this magic is into the delay function,

(DELAY <expr>) is abbrev for (lambda () <expr>) , defining!!!

And force is running the procedure:

(Force p) is (P)

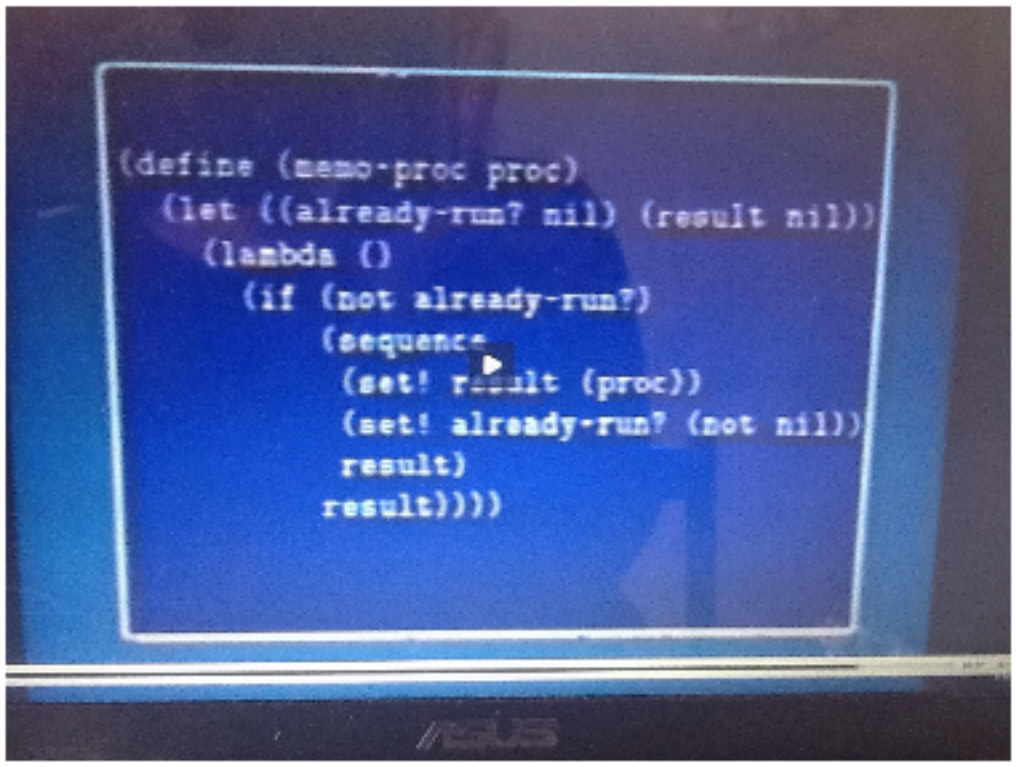
Delay decoupled the natural process of events in our programs. Therefore giving delay the power of rearranging the order of the events.

For accessing the elects of the streams we use:

(tail (tail (tail .....))) ← Inefficiet!!

There is a little hack for changing I. The way delay works:

Memo-proc : takes the procedure with no arguments and transforms it into a procedure that have to only to its computing once. Dies the computing, Remember the results and after that only calls the results as the cache .



This hack called memorization!

What we have done so far, is we have build a data structure with sort of is a procedure for controlling the iteration in it.