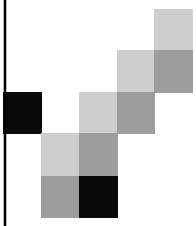


Streams…

…row, row, row your boat…

Agenda

- SUPER Brief intro to streams…
- Midterm Review
- Next Time…more streams & Scheme written in Scheme! (MCE)

Brief Intro to Streams

- Creates a sequence, but computes it only when requested!
- So the answer is outputted when program needs it, not when the program is called
- Promise!

Stream ADT

- Constructors
  - cons-stream
    (cons-stream 1 2) → (cons 1 (delay 2))
- Selectors
  - stream-car
    (stream-car (cons-stream 1 2)) → 1
  - stream-cdr
    (stream-cdr (cons-stream 1 2)) → 2

Stream Procedures…

- stream-map
  - Works similarly like map for lists, but with streams
    - Ex.
      (ss (stream-map + ones integers))
      → (2 3 4 5 6 7 8 …)
      ;; it adds the first element of each stream together, then adds the second element of both streams together etc. …
      (1+1 1+2 1+3 etc.)
- stream-append
  - Works similarly to append, but it only works on finite streams where the last element is the-empty-stream
    - Ex.
      (ss (stream-append (cons-stream 1 the-empty-stream) (cons-stream 2 the-empty-stream)))
- Interleave
  - It takes two streams and interchanges their values and produces a new stream
    - Ex.
      (ss (interleave ones twos))
      → (1 2 1 2 1 2 1 2 …)
- stream-null?
  - The-empty-stream indicates whether a stream is null.

Null streams?

- Is there a null value for a stream?
  YES! It's called **the-empty-stream**
- Below the line…
  - (define (cons-stream a b)
    (cons a (delay b)))
  - (define (stream-car stream)
    (car stream))
  - (define (stream-cdr stream)
    (force (cdr stream)))

Implicit Streams…

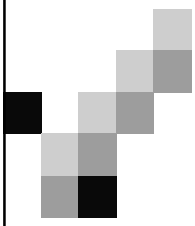
- Implicit streams
  Not using any other streams…ie
  (define ones (cons-stream 1 ones))
- **NOT implicit**
  (define integers
  (cons-stream 1
  (stream-map + ones integers)))

What's the point?

- Benefit of efficiency…no long wait for an answer…
- Deferred operations until necessary.
  - Think about being lazy and doing something only when you need to ☺

Stream Practice

- Create a stream **repeater** that takes an expression and generates this kind of stream:
  (ss (repeater 'ha)) → (ha haha hahaha …)
- Create a **binary stream**:
  (0 1 10 11 100 101 110 111 1000…)



More next time!

…time for midterm review…