26th Oct 2020

Studio 11
Streams and Laziness

CS1101S AY20/21 SEM 1
Studio 03A

Chen Xihao
Year 2 Computer Science

chenxihao@u.nus.edu
@BooleanValue

Studio 11

Agenda

- Admin
- Recap:
  - Streams
- Studio sheets
- RA2 Questions (if we have time)

Admin

Preparation for Practical

- Week 13, 11th Nov, Wednesday
- How to practice?
  - Do your paths and missions under timed conditions
  - Redo your mid-term questions, but type them out now
  - But, don’t start programming straightaway! Take time to think of the solution first.
  - Get your syntax right!

Admin

Preparation for Practical

- Process:
    1. Read
    2. Think
    3. Play
    4. Programme

Admin

RA2 Q2

(2) What is the result (in *list notation*) of evaluating the following Source program?

```
const A = list(list(2), list(4));
const B = map(x => x, A);
set_head(head(B), 3);
set_head(tail(B), 5);
A;
```

A. `list(list(2), list(4))`

B. `list(list(2), list(5))`

C. `list(list(3), list(4))` **(answer)**

D. `list(list(3), list(5))`

E. `list(list(3), 5)`

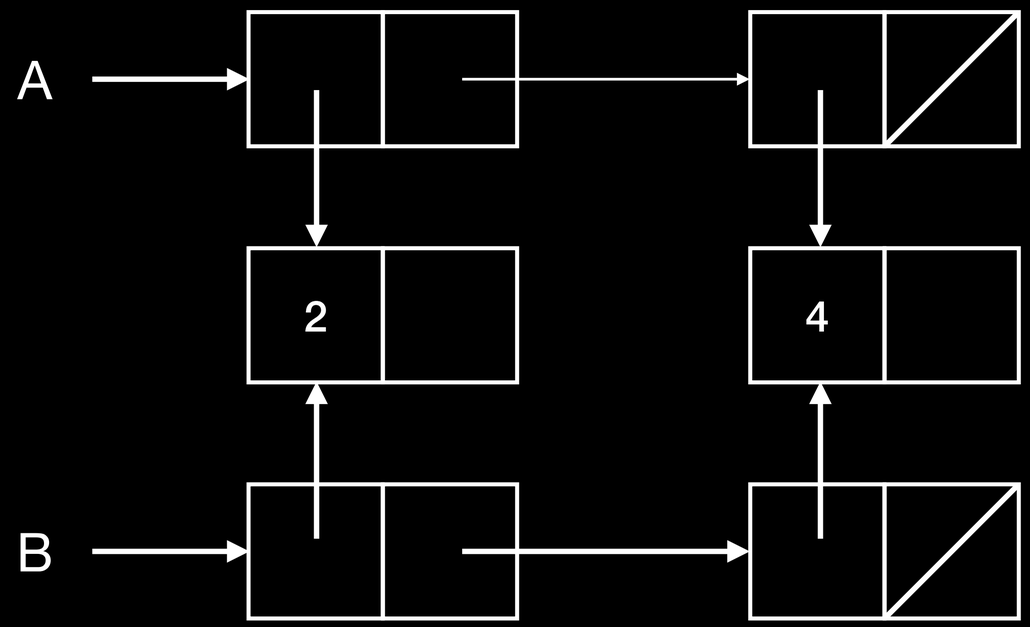
F. `list(3, list(5))`

G. `list(3, 5)`

Admin

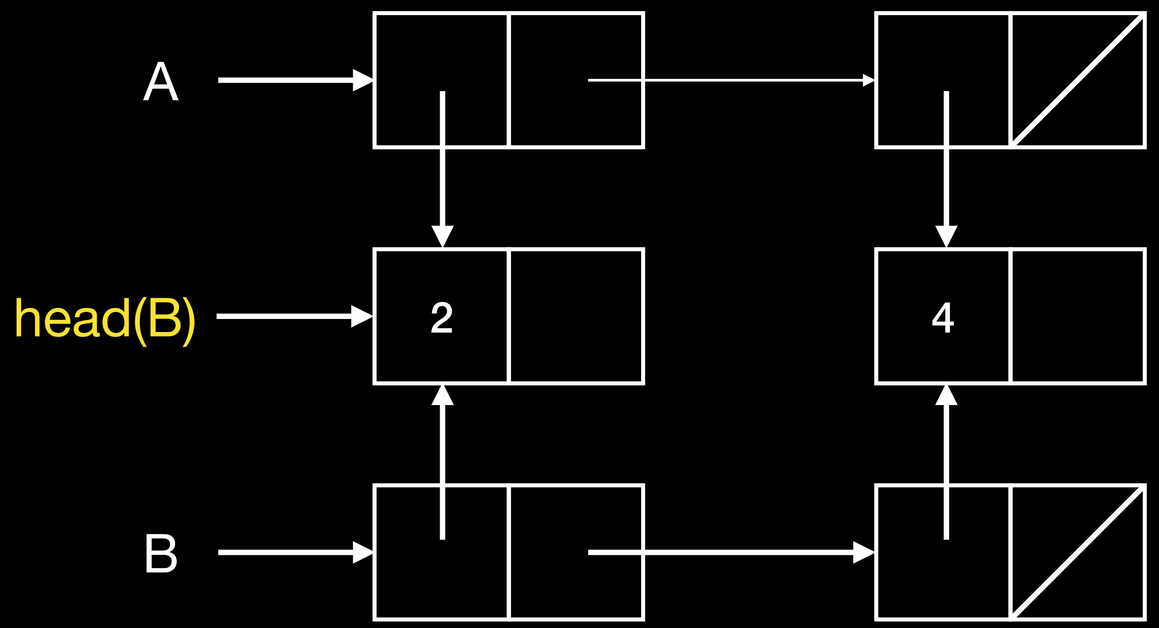
RA2 Q2

```
const A = list( list(2), list(4) );

const B = map(x => x, A)
```



Admin

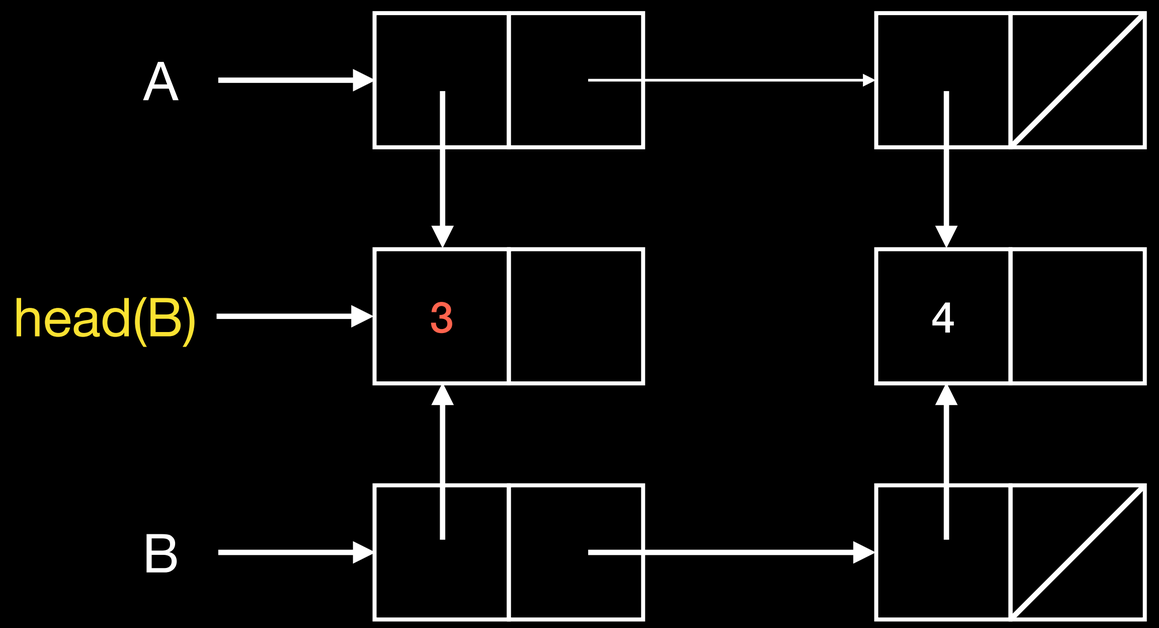
RA2 Q2




```
const A = list( list(2), list(4) );

const B = map(x => x, A)

set_head(head(B), 3);
```

Admin

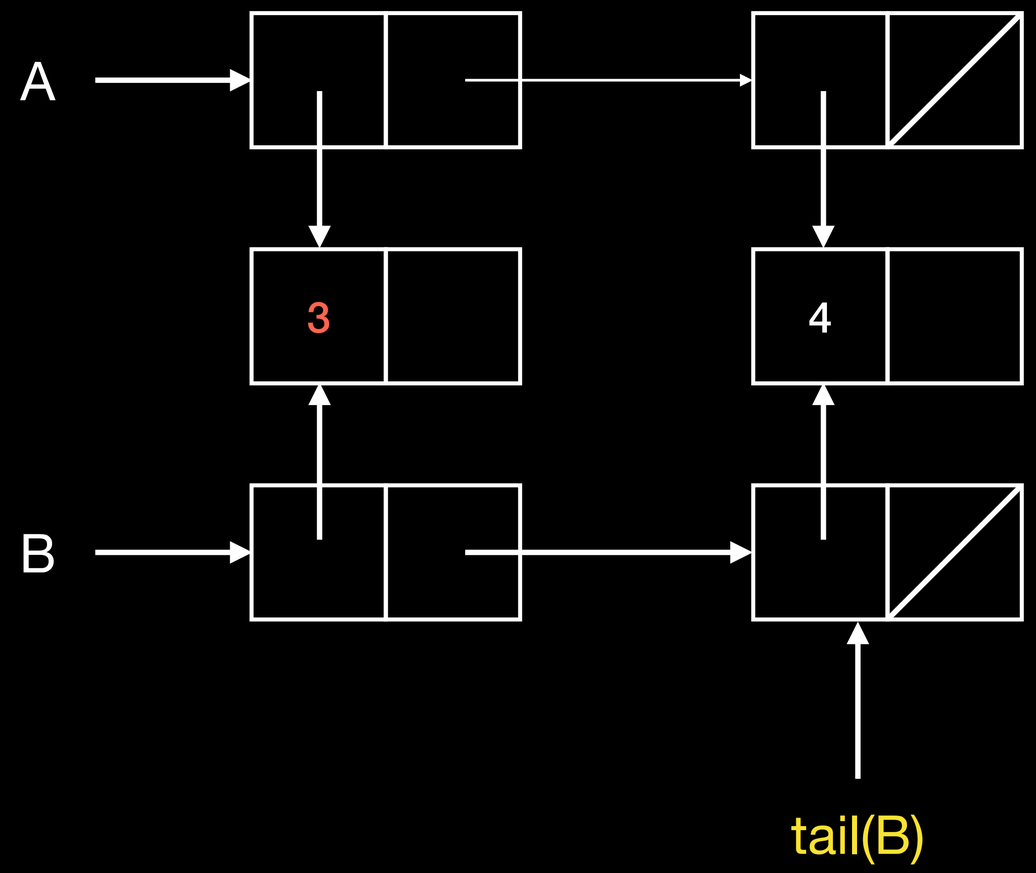
RA2 Q2




```
const A = list( list(2), list(4) );

const B = map(x => x, A)

set_head(head(B), 3);
```

Admin

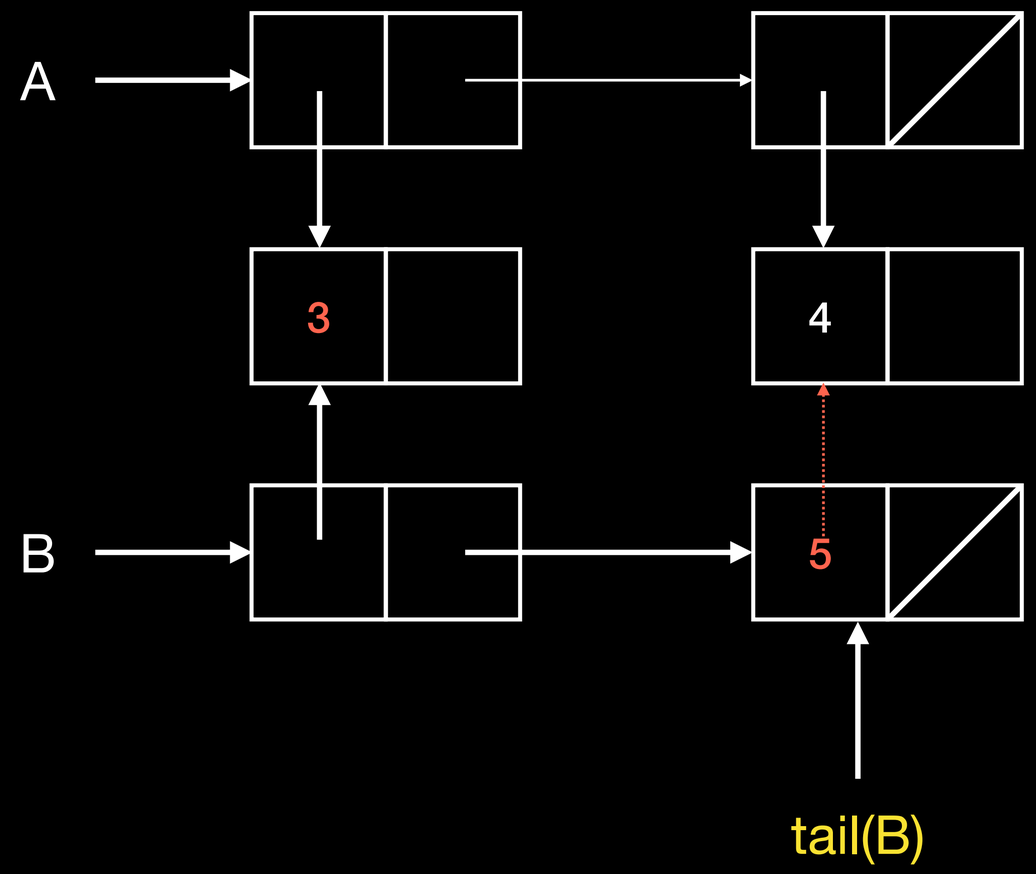
RA2 Q2




```
const A = list( list(2), list(4) );

const B = map(x => x, A)

set_head(head(B), 3);

set_head(tail(B), 5);
```

Admin

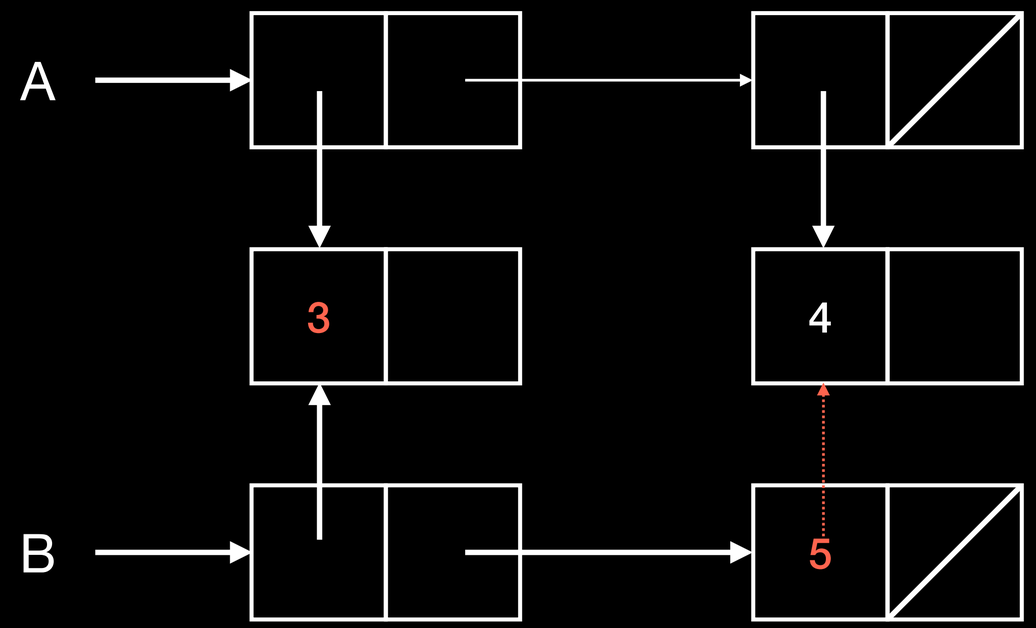
RA2 Q2




```
const A = list( list(2), list(4) );

const B = map(x => x, A)

set_head(head(B), 3);

set_head(tail(B), 5);
```

Admin

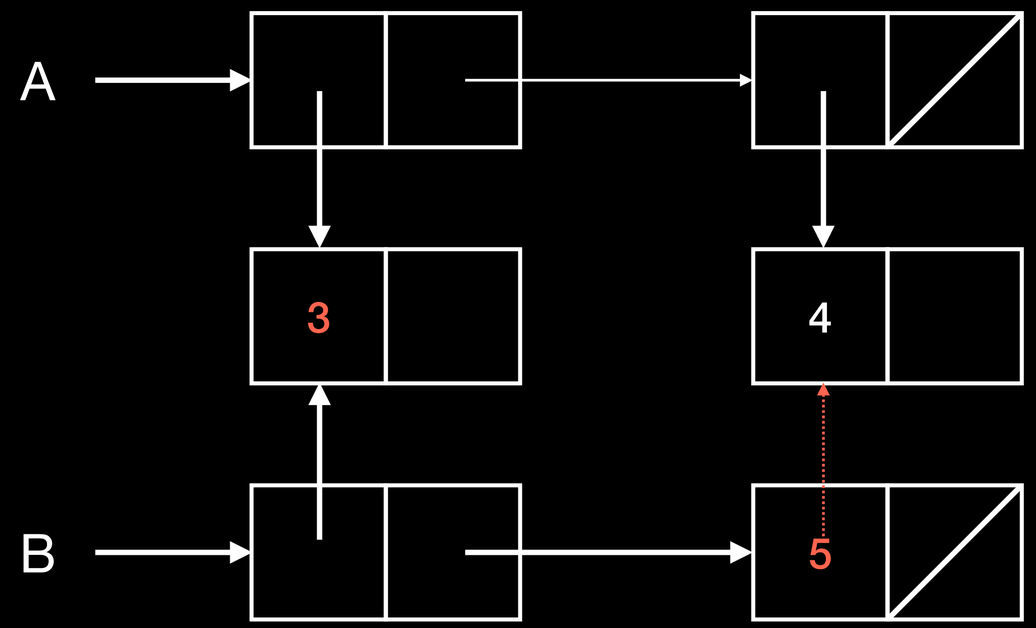
RA2 Q2




```
const A = list( list(2), list(4) );

const B = map(x => x, A)

set_head(head(B), 3);

set_head(tail(B), 5);

A; // list(list(3), list(4))
```

Admin

RA2 Q2




```
const A = list( list(2), list(4) );

const B = map(x => x, A)

set_head(head(B), 3);

set_head(tail(B), 5);

A; // list(list(3), list(4))
```

notice the difference between

```
  set_head(head(B), 3);

  set_head(B, 3);
```

Recap: Streams

Recap
Lazy Evaluation

- Delay evaluation of expression until needed
- How?
  - Functions
  - We can define a function
  - Body isn’t evaluated until function is called!

Recap
Stream

- What is a stream?
  - A stream is either a:
    - null, or
    - a pair whose tail is a nullary function that returns a stream
  - Nullary function?
    - A function with no parameters

Recap
Stream

- Why use streams?
  - Ability to represent an infinite set of elements
  - Only compute when needed
    - Less resource wasted!
    - And your computer doesn’t crash when defining `ones` stream
- And a lot of wishful thinking!

Recap
Stream

- Thinking framework:
  - Streams are “delayed lists”
  - Anatomy:

  - `pair(1, () => some_stream)`

Recap
Stream

- Example of a stream:
  - `const ones = pair(1, () => ones);`

Recap
Stream

- A stream of streams:

```
function stream() {
    return pair(‘ ~ ’,, stream);
}
// ‘ ~ ’, ‘ ~ ’, ‘ ~ ’, ‘ ~ ’, …
```

Recap
Stream

- Quiz time: are these streams?
  - `null;`
  - `pair(1, () => pair(2, null) );`
  - `pair(1, () => pair(2, () => null) );`
  - `pair(1, () => pair(2, () => 3) );`

Recap
Stream

- Answer:
  - `null;  // yes`
  - `pair(1, () => pair(2, null) );  // no (tail is not a nullary function)`
  - `pair(1, () => pair(2, () => null) );  // yes`
  - `pair(1, () => pair(2, () => 3) );  // no (tail is not a nullary function)`

Recap
Stream

- Some useful functions:
  - build_stream
  - eval_stream
  - stream_tail
  - stream_to_list
  - stream_length

Recap
Stream

- Take not whether implementation is lazy!
  - E.g. stream_length is NOT lazy
  - Try running `stream_length(ones)`
    - (and be prepared to force restart source academy)
- Can’t use non-lazy functions on infinite lists!

Recap

Stream - Techniques

- Defining streams recursively:
  - `const s = pair(1, pair(2, () => s));`
  - `// 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, …`
- Recall:
  - `const xs = pair(1, pair(2, xs));`
  - `// 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, …`

Recap
Stream - Techniques

- Using helper functions: streams can access names declared in the function body

```
function s(x) {
    let y = 1;
    function helper() {
        x = x + 1;  y = y + 1;
        return pair(x + y, () => helper());
    }
    return helper();
}
const stream = s(111);
```

```
function t(x) {
    return pair(x, () => t(x + 1));
}
const stream_2 = t(1);
```

Recap
Stream - Techniques

- Note: these are equivalent

```
function s() {
    return pair(1, s);
}
// evaluates s() directly
```

```
function s() {
    return pair(1, () => s());
}
// evaluates (() => s())() which reduces to s()
```

Recap
Stream - Techniques

- Play with multiple streams!

```
function add_stream(s1, s2) {
  if (is_null(s1)) { return s2; }
  else if (is_null(s2)) { return s1; }
  else {
      return pair(head(s1) + head(s2),
                  () => add_streams(stream_tail(s1), stream_tail(s2))
                );
  }
}
const fibs_stream = pair(0, () => pair(1, () => add_streams(stream_tail(fibs), fibs) ) );
```

End of Recap

Any questions?

End of File