7th Sept 2020

Studio 5
Data Abstraction

CS1101S AY20/21 SEM 1
Studio 03A

Chen Xihao
Year 2 Computer Science

chenxihao@u.nus.edu
@BooleanValue

Studio 5

Agenda

- Admin
- Reading Assessment 1
- Recap: Data Abstraction
- Studio sheet(s)

Admin

Studio 5
Admin

- RA1 is over
  - No matter how well you did, remember the only way is up!
  - Just 6% only
  - Focus on mid-terms
- Remember to book your mastery checks before I get busy with projects
  - Best to finish by this week

Studio 5

Preparing for Exams

- Visualise what’s going on
- Be familiar with expansions and substitution model
  - Learn to execute programmes without SourceAcademy
  - Write your programmes on paper / text editor first
- Don’t skip steps or take shortcuts!
  - DO NOT “EYEBALL” THE GIVEN PROGRAMME!!!

Recap: Pairs

Studio 5

Recap - Pairs

- Definition: Simple data structure
  - 2 values: head, tail
  - Recall: Quest - Functional Expressionism Q2

Studio 5

Recap - Pairs

- Simple test:
  - `const some_pair = pair(0, 1);`
  - `head(some_pair);    // returns ??`
  - `tail(some_pair);    // returns ??`

Studio 5

Recap - Pairs

- Simple test:
  - `const some_pair = pair(0, 1);`
  - `head(some_pair);    // returns 0`
  - `tail(some_pair);    // returns 1`

Studio 5

Recap - Pairs

- Nested pairs
  - Pairs that contain pairs as head, tail, or both
- `const some_nested_pair = pair(pair(0, 1), pair(2, 3));`
  - How to get the value 2?
    - `head(tail(some_nested_pair)) === 2   // returns ??`
    - `tail(head(some_nested_pair)) === 2   // returns ??`

Studio 5

Recap - Pairs

- Nested pairs
  - Pairs that contain pairs as head, tail, or both
- `const some_nested_pair = pair(pair(0, 1), pair(2, 3));`
  - How to get the value 2?
    - `head(tail(some_nested_pair)) === 2   // returns true!`
    - `tail(head(some_nested_pair)) === 2   // returns false!`

Studio 5

Recap - Pairs

Studio 5

Recap - Pairs

- `const some_nested_pair = pair(pair(0, 1), pair(2, 3));`
  - `head(tail(some_nested_pair)) -> 2`
- Intuitively, get the tail, then the head
- When writing a programme, we need to reverse this order!
- Why?

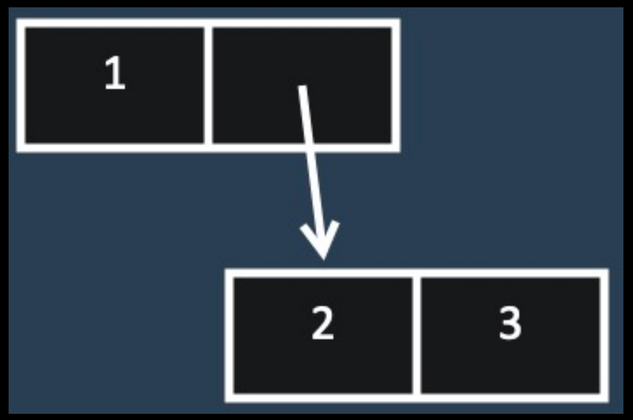
Studio 5

Recap - Pairs

- Visualisation: box and pointer diagrams

`pair(1, 2)`

`pair(1, pair(2, 3))`



Studio 5

Recap - Pairs

- Box and pointers will be tested!
- Please learn how to draw
- Ezpz marks

Recap: Lists

Studio 5

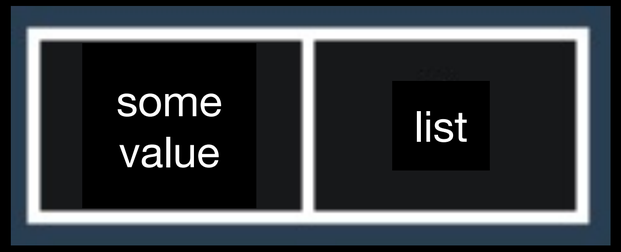
Recap - Lists

- Definition: A list is either
  - null, or
  - A pair whose tail is a list

Studio 5

Recap - Lists

- Case 1: null
  - aka. empty list
- Case 2: tail is a list
  - recursive data structure

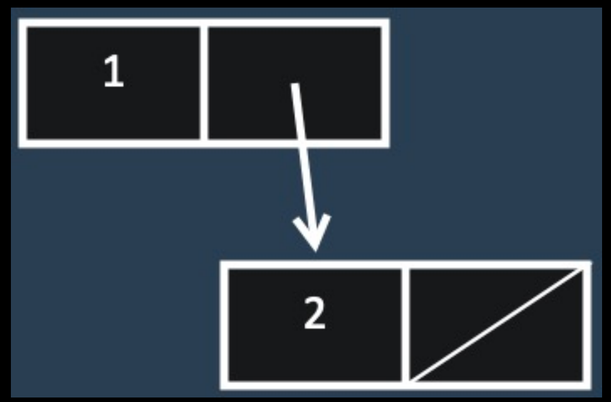


  - (note: this is not 100% accurate)

Studio 5

Recap - Lists

- null represented using a diagonal line
  - (direction of line doesn’t matter)
- `pair(1, pair(2, null));`



Studio 5

Recap - Lists

- Quiz: is this a list?
  - `pair(1, null);`

- Answer: yes!
  - Tail: null, which is a list!
- Actually this is “equivalent” to `list(1)`

Studio 5

Recap - Lists

- “Equality” can be hard to define
- However, we can say that these are “structurally equal”

Studio 5

Recap - Lists

- Recall: definition of a list
  - A list is either null or a pair whose tail is a list
  - A list is either null or a pair whose tail is (either null or a pair whose tail is a list)
  - A list is either null or a pair whose tail is (either null or a pair whose tail is a (either null or a pair whose tail is a list))
  - … you get the idea!

Studio 5

Recap - Lists

- Length of a list:
  - The length of the empty list is 0,
  - and the length of a non-empty list is one more than the length of its tail
    - (another recursive definition!)

```
function length(xs) {
    return is_null(xs)
        ? 0
        : 1 + length(tail(xs));
}
```

Studio 5

Recap - Lists

- Quiz: find the length of the following
  - `const a = list(1, 2, 3);   // recall the `list` function from lecture`
  - `const b = list(a);`
  - `const c = list(a, list(a, a));`
  - `const d = list(null, null);`

```
function length(xs) {
    return is_null(xs)
        ? 0
        : 1 + length(tail(xs));
}
```

Studio 5

Recap - Lists

- Generalisation:
  - RECURSIVE data structures
  - Just nested pairs
  - Empty list represented with `null` value

Recap: Data Structures

Studio 5

Recap - Data Structures

- What are DSes?
  - Stuff to help us organise stuff (duh!)
  - More formally: data organisations, managements and storage formats that enable efficient access and modification (abstraction)
- In this module:
  - pairs
  - lists
  - trees (implemented using pairs)

Studio 5

Recap - Data Structures

- Assume:
  - `const a = pair(1, pair(2, 3));`
  - `const b = list(4, 5, 6);`
- Quiz (pair work): draw the following using box and pointer diagrams!
  - `const w = pair(a, b);`
  - `const x = list(a, pair(a, b));`
  - `const y = pair(a, list(a, b));`
  - `const z = pair(null, list(null, pair(null, null)));`

Studio 5

Recap - Data Structures

- We choose data structures for efficiency for access and modification
  - e.g. arrays:
    - access in O(1) time, insertion in O(n) time
  - e.g. stacks:
    - the only built-in DS in modern computers
    - only can view the first element in a stack (like a stack of cards)
    - complexity? find our yourself :D

Studio 5

Recap - Data Structures

- Choice of data structures will be important
  - Depends on purpose and features of priority
- Important focus area for CS1101S (hint)

Any questions?

End of Recap