CS 135 Winter 2017

Tutorial 4: More lists and recursions
Goals of this tutorial

You should be able to...

• use list abbreviations and quote notation for lists.

• understand how to write custom string processing functions

• understand and use the principle of insertion sort to write custom sorting functions.

• use the properties of sorted lists to improve efficiency.

• understand and process two-dimensional data represented by nested lists.
Review: List Abbreviation

List abbreviations are available in language level Beginning Student With List Abbreviations, and all subsequent levels.

The expression

\((\text{cons}\ exp1\ (\text{cons}\ exp2\ (\ldots\ (\text{cons}\ expn\ \text{empty})\ldots)))\)

can be abbreviated as

\((\text{list}\ exp1\ exp2\ \ldots\ expn)\)

Example: \((\text{cons}\ 1\ (\text{cons}\ \text{'a}\ (\text{cons}\ 32\ (\text{cons}\ "hello"\ \text{empty}))))\)

is equivalent to \((\text{list}\ 1\ \text{'a}\ 32\ "hello")\)
Review: List Abbreviation

`cons` and `list` have different results and different purposes.

We use `list` to construct a list of fixed size (whose length is known when we write the program).

We use `cons` to construct a list from one new element (the first) and a list of arbitrary size (whose length is known only when the second argument to `cons` is evaluated during the running of the program).
Review: Quoting Lists

If lists built using list consist of just symbols, strings, and numbers, the list abbreviation can be further abbreviated using the quote notation we used for symbols.

\((\text{cons } \text{'red } (\text{cons } \text{'blue } (\text{cons } \text{'green empty})))\) can be written \(\text{'(red blue green)}\).

\((\text{list } 5 4 3 2)\) can be written \(\text{'(5 4 3 2)}\), because quoted numbers evaluate to numbers; that is, \(\text{'1}\) is the same as \(1\).

Now we can write empty as \((\text{list})\) or \(\text{'()}\).
Clicker Question - List Translation

Given this list:

\[(\text{list } 1 \ 'blue \ (\text{list } 2 \ 3))\]

What is the equivalent cons statement?

A \((\text{cons } 1 \ (\text{cons } \ 'blue \ (\text{cons } (\text{cons } 2 \ (\text{cons } 3 \ \text{empty})) \ \text{empty}))))\)

B \((\text{cons } 1 \ 'blue \ (\text{cons } 2 \ 3 \ \text{empty}) \ \text{empty})\)

C \((\text{cons } 1 \ (\text{cons } \ 'blue \ (\text{cons } 2 \ (\text{cons } 3 \ \text{empty}))))\)

D \((\text{cons } 1 \ (\text{cons } \ 'blue \ (\text{cons } 2 \ 3)))\)

E \((\text{cons } 1 \ (\text{cons } \ 'blue \ (\text{cons } (\text{cons } (\text{cons } 2 \ (\text{cons } 3 \ \text{empty})) \ \text{empty}) \ \text{empty}) \ \text{empty}))\)
Clicker Question - Nested Lists

(cons (cons 5 (cons 4 empty))
  (cons (cons 3 empty)
    (cons (cons 2 (cons 5 empty))
      (cons 5 (cons 4 empty))))))

Which of the following lists is equivalent to the one above?

A  (list 5 4 3 2 5 5 4)
B  (list (list 5 4 3) (list 2 5) 5 4)
C  (list (list 5 4) (list 3) (list 2 5) 5 4)
D  (list (list 5) (list 4) (list 3) (list 2) (list 5) (list 5) (list 4))
E  (list (list 5 4) (list 3) (list 2 5) (list 5 4))

CS 135 Fall 2016  5: More lists and recursions
Clicker Question - Parsing Nested Lists

(define lonum (list (list 5) (list 4 3) (list 2) 1))

Which of the following would produce a value of 3?

A  (rest (first (rest lonum)))
B  (first (rest (rest lonum)))
C  (first (rest (rest (rest lonum))))
D  (rest (rest (first (rest lonum))))
E  (first (rest (first (rest lonum))))
Group Problem - string-vowels

Write a function string-vowels which consumes a string and produces a string of vowels in the same order as they appear in original string.

∗ You may not use any built-in string functions other than string→ list and list→ string.
Insertion Sort Trace

We will perform a condensed trace of an insertion sort:

```
(define (sort lon)
  (cond [(empty? lon) empty]
        [else (insert (first lon) (sort (rest lon)))]))

(define (insert n slon)
  (cond [(empty? slon) (cons n empty)]
        [(<= n (first slon)) (cons n slon)]
        [else (cons (first slon) (insert n (rest slon)))]))

(sort (list 5 3 9 2 5 7 1 4))
```
Write a structure definition, and a data definition for a structure called `book` for the above data. Also write a template for a list of `book` structures. Note that the date is given as a Natural number representing the year of publication.
Group Problem - sort-book

Given a list of Book, write a function sort-book that sorts the list into chronological order (oldest to newest). Include a purpose, contract, and examples. You may want to look at the insertion sort example in the slides.
Group Problem - sbooks-published

Based on the template written earlier, write a function called sbooks-published that consumes a number representing a year and a list of Books which has already been date-sorted. sbooks-published will produce a list of Books that contains all the books published in that year.

To improve efficiency, you should avoid searching through the whole list when possible.
Group Problem - Remove Firsts (Optional)

Write a function called `remove-firsts` that consumes a `(listof (listof Any))`. It should produce a new `(listof (listof Any))` with the first item removed from each of the sublists. Note that if a list in the `(listof (listof Any))` is empty, then no item should be removed from it. Do not use helper functions. Include the contract, examples and tests.

For testing you can include the following test:

```scheme
(check-expect (remove-firsts '((1 2 3) (2 3) (1) ()))
               '((2 3) (3) () ())
```