Announcements

- Assignment 05 is due on Thursday, February 15.
- The midterm is on Monday, February 26, from 7:00 to 8:50 PM, and covers material up to and including Module 06, Slide 50.
- There is currently a Piazza poll to decide when the Midterm Review Session will be.

Goals of this tutorial

You should be able to...

- Use list abbreviations and quoted list notation for lists.
- Work with lists of structures.
- Recurse on natural numbers.
- Work with fixed-length lists inside another list.
- Understand how insertion sort works.
Review: List Abbreviations

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

The expression

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

can be abbreviated to

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

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

is equivalent to \[(\text{list } 1 \ \text{'}a\text{'} \ 32 \ \text{"hello"}).\]

Review: List Abbreviations

\text{cons} \text{ and list} have different results and different purposes.

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

We use \text{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 \text{cons} is evaluated during the running of the program).

Review: Quoting Lists

If lists built using \text{list} consist of just symbols, strings, and numbers, they may be further abbreviated using quotes.

\[(\text{cons } \text{'}\text{red}\text{'} \ (\text{cons } \text{'}\text{blue}\text{'} \ (\text{cons } \text{'}\text{green}\text{'} \text{empty}))))\]

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

\[(\text{list } 5 \ 4 \ 3 \ 2)\]

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

The same goes for strings: \[(\text{list } \text{"hi" } \text{"bye"})\]

can be written as \text{'}\text{("hi" "bye")}\text{'}.

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

(list 1 'blue (list 2 3))

What is the equivalent cons statement?
A (cons 1 (cons 'blue (cons (cons 2 (cons 3 empty)) empty)))
B (cons 1 'blue (cons 2 3 empty) empty)
C (cons 1 (cons 'blue (cons 2 (cons 3 empty))))
D (cons 1 (cons 'blue (cons (cons 2 empty) empty)))
E (cons 1 (cons 'blue (cons (cons 2 (cons 3 empty)) empty) empty) empty)

Clicker Question: Nested Lists
(cons (cons 5 empty)
   (cons 3 (cons (cons 2 (cons 5 empty))
      (cons 4 empty))))

Which of the following lists is equivalent to the one above?
A (list 5 3 2 5 4)
B (list (list 5) (list 3 2 5) 4)
C (list (list 5) 3 (list 2 5) 4)
D (list (list 5) (list 3) (list 2) (list 5) (list 4))

Clicker Question: 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))))
Lists of Structures: Books
Consider the following structure and data definition:

```
(define-struct book (title author pages))
;; A Book is a (make-book Str Str Nat)
;; requires: pages > 0
title and author are non-empty
```

Write a template, listof-book-template, for a list of Book structures.

Group problem: max-pages
Using your template, write a function max-pages, which consumes a positive natural number max-len, and a list of Books. The function produces a list of strings containing the titles of the Books in the consumed list which have a page length less than or equal to max-len. The titles in the produced list should appear in the same relative order as the consumed list. For example:

```
(max-pages 500 (list (make-book "Calculus 3" "John Smith" 374)
  (make-book "Linear Algebra 2" "Jane Doe" 687)
  (make-book "Statistics 1" "Johnny Roe" 68)))
⇒ (list "Calculus 3" "Statistics 1")
```

Group Problem: powers-of-k-alt
Recall the following template for counting up with Natural Numbers:

```
(define (upto-b-template n b)
  (cond [(= n b) (. . . b . . . )]
    [else (. . . n .
      . . . (upto-b-template (add1 n) b) . . . )])
```

Using the above template, write a function powers-of-k-alt that consumes natural numbers b and k, and produces a list of length b containing 

\[ k^1, -k^2, k^3, -k^4, \ldots, \pm k^b \], where the even powers of k are negated.

For example, (powers-of-k-alt 6 3) produces (list -9 -27 81 243 729).
You may find it helpful to make powers-of-k-alt a wrapper function.
Group Problem: Fixed Length Lists

Consider the following data definition:

;; A Pair is a (list (anyof Num Sym) Num)
;; requires: the first element of a Pair is less than or equal to the second
;; element, if they are both numbers

Write a function valid-listof-pairs?, that consumes a list, where each element is an arbitrary list of length 2, and produces true if each element in the list can be considered a Pair, according to the above data definition, and false otherwise.

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))
Insertion Sort Trace
(sort (list 5 3 9 2 5 7 1 4))
=> (insert 5 (sort (list 3 9 2 5 7 1 4)))

```plaintext
5 3 9 2 5 7 1 4
```

Insertion Sort Trace
(sort (list 5 3 9 2 5 7 1 4))
=> (insert 5 (sort (list 3 9 2 5 7 1 4)))
=> (insert 5 (insert 3 (sort (list 9 2 5 7 1 4))))

```plaintext
5 3 9 2 5 7 1 4
```

Insertion Sort Trace
(sort (list 5 3 9 2 5 7 1 4))
=> (insert 5 (sort (list 3 9 2 5 7 1 4)))
=> (insert 5 (insert 3 (sort (list 9 2 5 7 1 4))))
=> (insert 5 (insert 3 (insert 9 (sort (list 2 5 7 1 4)))))

```plaintext
5 3 9 2 5 7 1 4
```
Insertion Sort Trace

==> (insert 5 (insert 3 (sort (list 9 2 5 7 1 4))))
==> (insert 5 (insert 3 (insert 9 (sort (list 2 5 7 1 4)))))
==> (insert 5 (insert 3 (insert 9 (insert 2 (sort (list 5 7 1 4))))))

5 3 9 2 5 7 1 4
Insertion Sort Trace

= > (insert 5 (insert 3 (insert 9 (insert 2
(insert 5 (insert 7 (sort (list 1 4))))))))
= > (insert 5 (insert 3 (insert 9 (insert 2
(insert 5 (insert 7 (insert 1 (sort (list 4)))))))))

5 3 9 2 5 7 1 4
Insertion Sort Trace

```scheme
=> (insert 5 (insert 3 (insert 9 (insert 2
(insert 5 (insert 7 (insert 1 (insert 4 empty))))))))
=> (insert 5 (insert 3 (insert 9 (insert 2
(insert 5 (insert 7 (insert 1 (list 4)))))))
```

```
5 3 9 2 5 7 1
1 4
```

Insertion Sort Trace

```scheme
=> (insert 5 (insert 3 (insert 9 (insert 2
(insert 5 (insert 7 (list 1 4))))))
=> (insert 5 (insert 3 (insert 9 (insert 2
(insert 5 (list 1 4)))))
```

```
5 3 9 2 5 7
1 4
```

Insertion Sort Trace

```scheme
=> (insert 5 (insert 3 (insert 9 (insert 2
(insert 5 (insert 7 (list 1 4)))))
=> (insert 5 (insert 3 (insert 9 (insert 2
(insert 5 (list 1 4 7))))))
```

```
5 3 9 2 5
1 4 7
```
Insertion Sort Trace

$= \Rightarrow (\text{insert } 5 \ (\text{insert } 3 \ (\text{insert } 9 \ (\text{insert } 2 \ (\text{insert } 5 \ (\text{list } 1 \ 4 \ 7)))))))$

$= \Rightarrow (\text{insert } 5 \ (\text{insert } 3 \ (\text{insert } 9 \ (\text{insert } 2 \ (\text{list } 1 \ 4 \ 5 \ 7)))))$

5 3 9 2

1 4 5 7

Insertion Sort Trace

$= \Rightarrow (\text{insert } 5 \ (\text{insert } 3 \ (\text{insert } 9 \ (\text{list } 1 \ 2 \ 4 \ 5 \ 7))))$

$= \Rightarrow (\text{insert } 5 \ (\text{insert } 3 \ (\text{list } 1 \ 2 \ 4 \ 5 \ 7 \ 9)))$

5 3

1 2 4 5 7 9

Insertion Sort Trace

$= \Rightarrow (\text{insert } 5 \ (\text{insert } 3 \ (\text{insert } 9 \ (\text{list } 1 \ 2 \ 4 \ 5 \ 7))))$

$= \Rightarrow (\text{insert } 5 \ (\text{insert } 3 \ (\text{list } 1 \ 2 \ 4 \ 5 \ 7 \ 9)))$

5 3

1 2 4 5 7 9
Insertion Sort Trace

\[
\Rightarrow (\text{insert } 5 (\text{insert } 3 (\text{list } 1 2 4 5 7 9)))
\]
\[
\Rightarrow (\text{insert } 5 (\text{list } 1 2 3 4 5 7 9))
\]

Insertion Sort Trace

\[
\Rightarrow (\text{insert } 5 (\text{list } 1 2 3 4 5 7 9))
\]
\[
\Rightarrow (\text{list } 1 2 3 4 5 5 7 9)
\]