<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Lecture</th>
<th>Reading</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Feb 12-16</td>
<td>Lecture: Introduction to Recursion</td>
<td>Reading: Simply Scheme, Chapter 11</td>
<td>Lab: Recursion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DbD case study, recursive version</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Feb 19-23</td>
<td>Lecture: <em>&lt;holiday&gt;</em></td>
<td>Reading: &quot;Roman Numerals&quot; case study</td>
<td>Lab: More recursion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Special Lecture: Recursion, Midterm</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Tuesday, Feb 20, 9:30-10:30, 306 Soda)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Feb 26 - Mar 2</td>
<td>Lab: More recursion</td>
<td>Lecture: <em>Midterm 1</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Lab: Advanced recursion</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Mar 5 – 9</td>
<td>Lecture: Recursion</td>
<td></td>
<td>Lab: Miniproject #2</td>
</tr>
</tbody>
</table>
Announcements

• Nate's office hours:
  - Wednesday, 2-4, in 329 Soda

• Reading for this week
  - Simply Scheme, chapter 11
  - Difference between Dates, Recursive version
  - (These will be on the midterm)

• More reading next week…

• The last day to drop is Feb 16th
Midterm 1: Feb 26\textsuperscript{th} (in two weeks)

• Location: ???
  - Will send out via email, ucwise announcements

• Time: In the lecture slot, plus 20 minutes
  - (5:10-6:30)

• Open book, open notes.
  - Nothing that can compute, though

• Everything we’ve covered, including the coming TWO weeks on recursion.
  - (But not the "roman numerals" case study)
Special midterm issues

• Special Lecture
  - Tuesday, Feb 20\textsuperscript{th}, 9:30-10:30, 306 Soda

• TA-led review session: Date and Loc TBD
  - Probably Sat, Feb 24, 2-4pm, 430 Soda

• There are two practice exams in your reader
  - The first is shorter than yours will be, the second is the right length
  - Do these as if you were taking a Midterm: i.e., in one sitting, without STk, etc.

• Check the announcements for solutions, and more practice exams.
Any questions about the miniproject?
An algorithmic technique where a function, in order to accomplish a task, calls itself with some part of the task.
Using recursive procedures

• Everyone thinks it's hard!
  - (well, it is... aha!-hard, not complicated-hard)

• Using repetition and loops to find answers

• The first technique (in this class) to handle arbitrary length inputs.
  - There are other techniques, easier for some problems.
All recursion procedures need…

1. Base Case(s)
   • Where the problem is simple enough to be solved directly

2. Recursive Cases (s)
   1. Divide the Problem
      • into one or more smaller problems
   2. Invoke the function
      • Have it call itself recursively on each smaller part
   3. Combine the solutions
      • Combine each subpart into a solution for the whole
Problem: *find the first even number in a sentence of numbers*

\[
\text{(define (find-first-even sent)}
\]

\[
\text{(if (even? (first sent)) (find-first-even (bf sent)))}
\]

\[
\begin{align*}
\text{base case: return} &\quad ; \text{that even number} \\
\text{recursively (base case)} &\quad ; \text{recurse on the rest of sent}
\end{align*}
\]
(define (count sent)

  (if (empty? (bf sent))

      1 ;base case: return 1

      (+ 1
         (count (bf sent))) ;recurse on the ; rest of sent

  ))

Count the number of words in a sentence
Base cases can be tricky

- By checking whether the (\textbf{bf} sent) is empty, rather than sent, we won't choose the recursive case correctly on that last element!
  - Or, we need two base cases, one each for the last element being odd or even.
- Better: let the recursive cases handle \textit{all} the elements

Your book describes this well
Count the number of words in a sentence

(define (count sent)
  (if (empty? (bf sent)) 0
    (+ 1 (count (bf sent)))) ;base case: return 0
)

(+ 1
  (count (bf sent)) ;recurse on the ; rest of sent
))
> (count '(a b c))

\[ \begin{align*}
\text{sent} &= (a b c) \\
& \Rightarrow (+ 1 (+ 1 (+ 1 0))) \\
& \Rightarrow 3
\end{align*} \]
Problem: find all the even numbers in a sentence of numbers

```
(define (find-evens sent)
  (cond ((empty? sent) ; base case
      '())
        ((odd? (first sent)) ; rec case 1: odd
      (find-evens (bf sent)))
        (else ; rec case 2: even
      (se (first sent)
      (find-evens (bf sent)))
    ))
)```
\textbf{> (find-evens ' (2 3 4 5 6))}

\[ \text{sent} = (2 3 4 5 6) \]

\[ (\text{se} \ 2) \]

\[ \text{sent} = (3 4 5 6) \]

\[ (\text{se} \ 4) \]

\[ \text{sent} = (4 5 6) \]

\[ (\text{se} \ 6) \]

\[ \text{sent} = (6) \]

\[ (\text{se} \ 6) \]

\[ \text{sent} = () \]

\[ (\text{se} \ 6) \]

\[ () \]

\[ (\text{se} \ 2 \ (\text{se} \ 4 \ (\text{se} \ 6 \ (())))) \]

\[ (\text{se} \ 2 \ (\text{se} \ 4 \ (\text{se} \ 6 \ (())))) \]

\[ \text{sent} = (2 4 6) \]