Lecture 4:
"Difference Between Dates"
and
data abstraction
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Lecture:</th>
<th>Lab:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Sep 4-8</td>
<td>&lt;holiday&gt;</td>
<td>Conditionals, Booleans, Testing</td>
</tr>
<tr>
<td>3</td>
<td>Sep 11-15</td>
<td>Case Studies</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reading: Difference between Dates (just the first version in the reader)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lab: Work with Difference between Dates</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Sep 18-22</td>
<td>Data abstraction in DbD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lab: More DbD; Miniproject I</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Sep 25-29</td>
<td>Introduction to Recursion</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Oct 2-6</td>
<td>Midterm 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lab: Recursion II</td>
<td></td>
</tr>
</tbody>
</table>
How useful has the case study been?
Miniproject #1: this week

• You are to write century-day-span
  - Calculate the number of days between dates in (possibly) two different years

• Consider the central lesson of the case study: there are easier and harder ways to solve problems. Choose easier.
This is your first large program

- Use helper functions

- Test, and test some more.

- Reuse code that you have already written

- Add comments!
What does “understand a program” mean?
A Big Idea: abstraction

“the process of leaving out consideration of one or more properties of a complex object or process so as to attend to others”
• Abstracting with a new function
  
  - (square x) instead of (* x x)
  - (third sent) instead of (first (bf (bf sent)))

• Abstracting a new datatype

  A datatype provides functionality necessary to store "something" important to the program

  - Selectors: to look at parts of the "something".
  - Constructor: to create a new "something".
  - Tests (sometimes): to see whether you have a "something", or a "something else"
Data abstration: words and sentences

- **Constructors**: procedures to make a piece of data
  - word
  - sentence

- **Selectors**: procedures to return parts of that data piece
  - first, butfirst, etc.
Benefits

• Why is "leaving out consideration of", or "not knowing about", a portion of the program a good thing?

• Consider two ways one can "understand a program":
  - Knowing what each function does
  - Knowing what the inputs are (can be), and what the outputs are (will be).
• Disregarding the "understanding" issue, why might it be a good idea to "modularize" your code?

(where modules are abstracted from each other)
Data abstraction in the DbD code

- How does the code separate out processing of the date-format from the logic that does the "real" work?

  - **Selectors**
    - month-name (takes a date)
    - date-in-month (takes a date)
    - ? month-number (takes a month name)

  - **Constructors? Tests?**