CS3:
Introduction to Symbolic Programming

Lecture 4:
"Difference Between Dates"
and
data abstraction

Spring 2007

Nate Titterton
nate@berkeley.edu

Schedule

3	Jan 29-Feb 3	Lecture: Conditionals, Case Studies Reading: "DbD" case study Lab: Explore "Difference between Dates"
4	Feb 5-9	Lecture: Data abstraction in DbD: extending to short dates Lab: More Difference between dates Miniproject 1
5	Feb 12-16	Lecture: Introduction to Recursion Lab: Recursion
6	Feb 19-23	Lecture: <*holiday*> Lab: Recursion II
7	Feb 26 - Mar 2	Lecture: *Midterm 1* Lab: Advanced recursion

How useful has the case study been?

Any questions about last weeks materials?

- (SchemeHandler…)

This week

- A few exercises on Tue/Wed

- Mini-project #1: You are to write `cetury-day-span`
  - Extend the `day-span` program to correctly handle dates in (possibly) 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

- Break out self-contained tasks into helper procedures: they should be easy to name.
- If you can get your main procedure to read like English, you are doing well.

• Test, and test some more.

- Remember to put test cases above each helper procedure.

• Reuse code that you have already written

• Add comments!

- Above each procedure, at least.
- Within some `cond` cases, additionally.

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.

Tests: predicates that tell you which type of data you have

- `word?`, `sentence?`

`card-greater?` (from fa05 midterm 1)

Write `card-greater?` The procedure takes two cards and returns true if and only if the first card is bigger than second.

Cards are represented by a two-character word, where the first character represents the rank (`a`, `k`, `q`, `j`, `0`, `9`, `8`, `7`, `6`, `5`, `4`, `3`, and `2`), and the second character represents the suit (`s`, `h`, `d`, and `c`). For instance, `2h` is the two of hearts, `qc` is queen of clubs, `0s` is the 10 of spades, etc. For this problem, consider *all* spades to rank higher than hearts, which all rank higher than diamonds, which all rank higher than clubs.

```
(card-greater? 'ac '3d) → #f
(card-greater? 'kh 'qh) → #t
(card-greater? '4s '4s) → #f
```

Comment all your procedures. Assume you have a working version of `outranks?`, as you wrote in lab, to use. (Remember, `outranks?` takes two ranks and returns true if the first is higher than the second.)

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).

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?