CS3: Introduction to Symbolic Programming

Lecture 3: Review of Conditionals
Case Studies

Fall 2007 Nate Titterton
nate@berkeley.edu
Announcements

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

• Tue/Wed is a Catch-up day.
  - Use this day to catch up! That is, go back over the last two weeks and fill in places you missed

• We are still waiting on readers for homework grading…
|   | Jan 22-26   | Lecture: Introduction, Review, Conditionals  
Reading: *Simply Scheme*, ch. 3-6  
Lab: Conditionals |
|---|-------------|-----------------------------------------------------------------------------------------|
| 3 | Jan 29-Feb 3| Lecture: Conditionals, Case Studies  
Reading: "Difference between Dates" case study, in the reader (first version)  
Lab: Explore "Difference between Dates" |
| 4 | Feb 5-9     | Lecture: Data abstraction in DbD  
Lab: 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 |
1. Conditionals
   - `cond and if`
   - These are special forms, and don't follow the standard rules of evaluation

2. Booleans
   - `truth (#t, or anything) and non-truth (#f)`

4. Logical operators
   - `and, or, not`
1. Writing conditionals using only \texttt{and/or or if/cond}.

3. Organizing a series of conditionals

5. Predicates
   - procedures that return \texttt{#t} or \texttt{#f}
   - by convention, their names end with a "?"
1. Testing

- There is much more to programming than writing code. *Testing* is crucial, and an emphasis of this course
  - Analysis
  - Debugging
  - Maintenance.
  - "Design"

- Testing is an art (there is no one right way)
  - boundary cases, helper procedures, etc.
1. **Helper procedures**

- Choosing when to write helper procedures is an ... art. There is no one right way.

- This is an important skill in programming, and one you will need to focus on.
• Abstraction helps make programs understandable by simplifying them.
  - By letting the programmer or maintainer ignore details about a task at hand

  - Helper functions, when done correctly, do this
A video resource

- http://wla.berkeley.edu
  Weiner lecture archives

- The "course" is an earlier CS3
  - Different emphasis; early lectures may work better than later ones
  - Very different lab experience
  - Same book
What does “understand a program” mean?
Case Studies

• Reading!? 

• A case study:
  - starts with a problem statement
  - ends with a solution
  - in between, a …story… (narrative)
  - How a program comes to be

• You will write “day-span”, which calculates the number of days between two dates in a year
You need to read this

• The lab will cover the case study through a variety of activities.
  - This will culminate in the first “mini-project”, extending day-span to work with different years.

• We just may base exam questions on it

• It will make you a better programmer! 4 out of 5 educational researchers say so.
Some important points

• There is a large "dead-end" in this text
  - Like occur in many programming projects
  - Good "style" helps minimize the impacts of these

• There is (often) a difference between good algorithms and between human thinking
Extra Slides

(check for code)
Write an answer procedure.

Write a procedure named answer that, given a sentence that represents a question, returns a simple answer to that question. (A question's last word ends with a question mark.) If the argument sentence is not a question, answer should merely return the argument unchanged.

- Given ( am i ...? ), answer should return ( you are ...).
- Given ( are you ...? ), answer should return ( i am ...).
- Given ( some-other-word i ... ? ), answer should return ( you some-other-word ...).
- Given ( some-other-word you ... ? ), answer should return ( i some-other-word ...).
- Given any other question, answer should return the result of replacing the question mark by a period.