CSC151.02 2014S, Class 30: Preconditions, Revisited

Overview

- Preliminaries.
  - Admin.
  - Questions.
- Topics.
  - Verifying preconditions.
  - The error procedure.
  - Husk and Kernel programming.
- Lab.

Preliminaries

Admin

- Pick new partners. Work with someone you have not worked with before.
- Extra review session tonight at 9 pm in tragic CS commons
- Extra credit:
  - ?

Upcoming Work

- Reading for tomorrow: Naming Local Procedures
- NO LAB WRITEUP TODAY
  - Was CSC 151 Writeup 20: Local Procedure Bindings
- Exam due THURSDAY night/Friday class.

Exam Questions

Verifying preconditions

- When we design procedures, we make assumptions about the inputs (and, maybe the state of the world).
  - (+ val0 val1 ... valn)
    - parameters must be real numbers
    - Error message when the precondition is not met
  - (largest values)
    - parameter must be a nonempty list of real numbers
What should happen if the assumptions (preconditions) don’t hold?

- If the preconditions are not met, the procedure must report an error
- If the preconditions are not met, the procedure can do whatever it wants
  - "It’s the user’s fault." It must be a design by lawyers from Micro...
  - "You might get something cool."
  - It’s expensive to check preconditions.

```
(define square (lambda (x) (* x x))

(define square (lambda (x) (if (not (number? x)) (report-an-error "Excuse me, sir/maam/sam, but you seem to be confused about the appropriate input to this procedure.") (* x x))))
```

Sam’s conclusion: It doesn’t matter which you do, as long as you document it.

The error procedure

(error "Message" additional values)

Husk and Kernel programming

- When you’re iterating a list, you probably don’t want to check the same preconditions at every step.
- Protect the part that does the work (the kernel) with one outer precondition check (the husk)