CSC151.02 2013F, Class 28: Preconditions, Revisited

Overview

- Preliminaries.
  - Admin.
  - Finding the largest value in a list.
  - Thinking about helper recursion.
- Verifying preconditions.
- The `error` procedure.
- Husk and Kernel programming.
- Lab.

Admin

- Continue partners from Monday
- Today's writeup: Exercise 3h and 4. (Yes, only part h of #3)
- What's Friday's quiz on? Recursion!
- Upcoming EC opportunities
  - Learning from Alumni, 2:15 Thursday, 3821, Eryn O'Neil '07
  - CS Extras, 4:30 Thursday, Max Mindock
  - Open mike night, Thursday in Bobs, watch Yazan play!
  - CS Table, Noon Friday, TBD
  - Football, 1 pm, Saturday
  - Men's soccer, 1:30 pm, Saturday
  - ...
- Other upcoming stuff
  - Spare class, 1:15 pm Thursday
  - Mentor session 7:30 pm Thursday
  - ...

Finding the Largest Element

```
(define largest
  (lambda (lst)
    (if (null? (cdr lst))
        (car lst)
        (max (car lst) (largest (cdr lst))))))
```

Helper recursion; Keep track of intermediate result (typcially "best solution so far" or "partial computation") and remaining values vs.

```
(define largest
  (lambda (lst)
    (largest-helper (car lst) (cdr lst))))

(define largest-helper
  (lambda (largest-so-far remaining)
    (if (null? remaining)
        largest-so-far
        (largest-helper (max largest-so-far (car remaining))
                        (cdr remaining)))))
```

vs.

```
(define largest-helper
  (lambda (largest-so-far remaining)
    (if (null? remaining)
        largest-so-far
        (if (> (car remaining) largest-so-far)
            (largest-helper (car remaining) (cdr remaining))
            (largest-helper largest-so-far (cdr remaining))))))
```

vs.

```
(define largest-helper
  (lambda (largest-so-far remaining)
    (if (null? remaining)
        largest-so-far
        (largest-helper (if (> (car remaining) largest-so-far)
                            (car remaining)
                            largest-so-far)
                        (cdr remaining)))))
```

vs.

```
 (define largest-helper
   (lambda (largest-so-far remaining)
     (cond
       [(null? remaining)
        largest-so-far]
       [(> (car remaining) largest-so-far)
        (largest-helper (car remaining) (cdr remaining))]
       [else
        (largest-helper largest-so-far (cdr remaining))])))
```

Thinking About Helper Recursion

- Focus on the helper:
  - Intermediate result - Likely to be close to the expected output
  - Remaining items
  - Big question: How do we update the intermediate result?
- Sometimes good to run examples

"Count the number of color names that include the word "green""

```
(helper 0 '("red" "green" "blue" "lightgreen" "yellow" "greenish"))
(helper 0 '("green" "blue" "lightgreen" "yellow" "greenish"))
(helper 1 '("blue" "lightgreen" "yellow" "greenish"))
(helper 1 '("lightgreen" "yellow" "greenish"))
(helper 2 '("yellow" "greenish"))
```

If the car of the list includes the word green, add 1 to the counter If the car of the list does not incldue the word green, don't change the counter

```
(define helper
   (lambda (count remaining)
      (cond
        [(null remaining) count]
        [(string-contains? (car remaining) "green")
         (helper (+ 1 count) (cdr remaining))]
        [else (helper count (cdr remaining))])))
```

Verifying preconditions

- What happens if we call largest on the empty list?
  - Break, perhaps in unhelpful way
- We'd like to report an error in a helpful way

The error procedure

(define helpful-largest (lambda (lst) (if (null? lst) (error "helpful-largest expects a nonempty list") (if (null? (cdr lst)) (car lst) (max (car lst) (helpful-largest (cdr lst)))))))

Good programming style: Predict likely incorrect input and tell user about it

Husk and Kernel programming

- Each time we write a procedure, we write two versions/parts
  - One part does the real work, assuming that the parameters are correct "Colonel" -> "Kernel"
  - Another part does all of the checking
    - All succeed: Call the kernel to do the real work
    - Some fail: Report an error
- Husk and kernel programming

Lab

Samuel A. Rebelsky, rebelsky@grinnell.edu