Midterm

- Stats:
  - Mean: 53.7, StdDev: 16.5
  - Median: 51, 3Q: 61, Max: 90
- Many excellent answers on Q3 (incl 3.3)
- Q4 was the hardest (recitation today)
- Don’t sweat the absolute score
- Still have Midterm 2 and Final (40%)
- Will be returned in recitation today

Announcements

- Project 1 (two-week assignment)
  - Due Thu Oct 15
  - Checkpoint Due Thu Oct 8
    - Finish attempt at Part 1; Pass/Fail - intended to help you figure out the questions to ask.
    - Partners
    - Sizeable project but not a lot of code needed (more than HW4)
    - Hints
      - Part 1: Implement directly (pattern match, recursion)
      - Part 2: Use List library higher-order functions

Review: Higher-Order Functions

Induction and Correctness of Code

Correctness of Code

fun factorial 0 = 1
| factorial n = n * factorial (n - 1)

- What does it mean for this code to be correct?
Correctness of Code

fun factorial 0 = 1
| factorial n = n * factorial (n - 1)

• What does it mean for this code to be correct?
  - Pre-condition: n ≥ 0
  - Post-condition: factorial n = n!

• Theorem: If n ≥ 0, then factorial n = n!
• Theorem: If n ≥ 0 and factorial n = v, then v = n!

• What's the difference?

Proof:

1. Show base case = P(0) holds
2. Assume P(m) holds (m ≥ 0), show that P(m+1) holds

Exercise

fun exp (x,0) = 1
| exp (x,n) = x * exp (x, n - 1)

Theorem: If n ≥ 0, then exp (x,n) = x^n
Proof:
How about with datatypes?

datatype 'a list = nil | :: of 'a * 'a list

fun append (nil, k) = k
| append (h :: t, k) = h :: append (t, k)

Theorem: For values l : 'a list and k : 'a list,
append (l, k) \Downarrow v (for some value v)

Proof:

1. Structural Induction

Example: append is associative

fun @ (nil, k) = k
| @ (h :: t, k) = h :: (t @ k)

Theorem: For values l_1, l_2, l_3 of type 'a list, (l_1 @ l_2) @ l_3 \equiv l_1 @ (l_2 @ l_3)
Exercise: rev' computes rev

fun rev nil = nil | rev (h :: t) = rev t @ h
fun rev' (nil, acc) = acc
| rev' (h :: t, acc) = rev' (t, h :: acc)

Theorem: For values l, k : ty list, (rev l) @ k ≅ rev' (l, k)

For Next Time

• Next Time:
  - Wrap-up functional programming with an advanced topic: Lambda Calculus

• Reading
• Online discussion forum
  - ≥1 substantive question, comment, or answer each week
• Project checkpoint