Lecture 32:

• Haskell intro (as time)

Announcements:

• HW-6 out
• Quiz 8 on Fri: Turing Machines, λ-calculus
Some of the major features of Haskell

1. A *purely* functional language
   - Only “pure” functions
   - In general, functions **do not** have *side effects* (do not modify state)
   - some nice features: e.g., memoization
   - Values (variables) are *immutable*
   - Functions (and operations) always produce **entirely new values**
   - Very different than most other PLs

2. Static typing
   - All type checking done at compile time (statically)
   - Employs *type inference* ... unobtrusive—w/out type annotations

3. “Strong” typing
   - Guarantees a program cannot contain certain type errors
   - Haskell places limits on type conversion (implicit/explicit)
4. Functions are “first-class” objects ... used like any other kind of value
   • Can take functions as parameters (and call them in the function body)
   • Can create new functions during program execution
   • Can store functions in data structures
   • Can return functions as values of other functions

5. Lazy evaluation ... vs eager evaluation
   • Defer computation until the result is needed
   • One benefit: possible performance gain (no needless computations)
     – e.g., using quicksort, can ask for first (first two, etc.) values, without sorting entire list
   • Another benefit: “infinite” data structures
     – and in particular, the ability to compute with them
     – somewhat similar to iterators (or streams)
   • Another benefit: programmer-defined control structures
     – e.g., short circuit evaluation of if-then-else
     – this means you don't need special constructs for control flow

6. Expression-oriented
   • All statements return values (e.g., even if statements!)