Advanced Progamming Languages	Lecture 19
CS 6110 Spring 2015	**Mon. March 9, 2015**

Lecture 19

Discuss answers to Midterm Exam

Most students correctly solved problems 2 and 3. The variration in grades came from Problem 1. Here are selected answers:

(a) Use simple ordinary mathematics, e.g. $\sum_{i=0}^{\infty} i$. Claiming that a definition like $f(x) = 2 * f(x+1)$ is "ordinary math" is a strecth.

(b) Functions in mathematics are ordinarily typed as in $f(x) = exp(x)$ for $x \in \mathbb{N}$, $exp(x) \in \mathbb{N}$. The semantics is usually in terms of sets, e.g. $\mathbb{N}$ is a set and f will be a single valued relation, a set of ordered pairs. It does not make sense in set theory that $f(f)$ because f can't be a member of the set of ordered pairs. For $f(x) = tt$, the Boolean constant function where $\mathbb{B} = \{tt, ff\}$, we see that $f(f)$ is impossible.

(c) Self application makes sense in computer science because functions are computation *rules* or algorithms. They can be given by combinators such as **S, K**. Self application is sensible for rules or algorithms as the **Y** combinator shows.

(d) In Church style typing, the variables have types as part of their syntax, as in $\lambda x^{\alpha}.x$.

(e) Typed terms terminate – everyone got this!

(f) The evaluation of the numerical argument must be call-by-value and the fix operator must be call-by-name.

(g) The **S, K** combinators form a basis, they can define all other combinators and thus compute all computable functions, even partial ones.

(h) A subrecursive language, such as the primitive recursive or elementary recursive function definition formalism, compute a subtype of the Herbrand Gödel recursive functions; CoqPL is an example.

(i) Everyone correctly answered this!

(j) When used to define an evaluator for λ-terms, the λ-calculus is being used as a *meta language*, and this is made clear in the syntax, e.g. writing $\underline{\lambda}(x.b)$ for the object language terms, those being evaluated in the meta language evaluator.