Advanced Programming Languages	**Problem Set 3**
CS 6110 Spring 2015	**Due: Weds. March 25, 2015**

Problem Set 3

Exercises

1. (a) We have used primitive recursion with simple types,

   e.g. $add : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$

   $$\begin{cases} add\ 0\ y = y \\ add\ S(x)\ y = S(add\ x\ y) \end{cases}$$

   Prove that *add* and *mult* as defined before are total functions on $\mathbb{N}$, e.g. have type $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$. We could also define the type $\mathbb{N} \times \mathbb{N}$ of ordered pairs of numbers, $< n, m >$. In this case we could assign the type $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

   (b) We can define *higher-order* primitive recursion as follows:

   $R\ a\ b\ 0 \ = \ a$
   $R\ a\ b\ S(n) \ = \ b\ n\ (R\ a\ b\ n)$

   Where $a \in \alpha,\ b \in \mathbb{N} \to \alpha \to \alpha,\ 0 \in \mathbb{N},\ S : \mathbb{N} \to \mathbb{N}$

   $R :\ \alpha \to (N \to \alpha \to \alpha) \to (\mathbb{N} \to \alpha)$

   Define $\sum\limits_{i=0}^{n} f(i)$ with higher-order primitive recursions.

   Give the types. Prove that functions defined by higher order primitive recursion from (total) computable functions are total. Take α to be $\mathbb{N}$ for the proof.

2. Prove that $x * y = y * x$ using Goodstein's rules from Lecture 22.

3. Sketch how to state Euclid's Theorem in primitive recursive arithmetic that there are an unbounded number of primes.

4. This problem explores the possibility of adding partial types and a fix construct to simply typed lambda calculus (STLC). First read a formal presentation of STLC at : `http://www.cis.upenn.edu/~bcpierce/sf/current/Stlc.html` In particular, pay attention to Coq definitions of *ty*, *tm*, *step* and *has_type* which respectively define the syntax of types, syntax of terms and the one step evaluation relation and the typing relation.

   To add partial types, we add new clauses to each of the above definitions. The new definitions can be found at: `http://www.cs.cornell.edu/~aa755/CS6110/StlcPart.html` In particular, the last 1,1,2,2 clauses respectively are new in the definitions of *ty*, *tm*, *step* and *has_type*.

   The new rules for *step*, the one step evaluation relation characterize how the new fix construct computes in one step. These can also be understood as follows:

$$\frac{f \mapsto f'}{fix\ f \mapsto fix\ f'}\ \textsf{ST\_Fix}$$

$$\frac{}{fix\ f \mapsto f\ (fix\ f)}\ \textsf{ST\_FixUnfold}$$

   The rules for *has_type*, the typing relation characterize the members of partial types. These can also be understood as follows:

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash t : \overline{T}}\ \textsf{T\_TotalPart}$$

$$\frac{\Gamma \vdash f : T \to T}{\Gamma \vdash fix\ f : \overline{T}}\ \textsf{T\_Fix}$$

   The above file also has examples that use the above rules to prove that $true$ and $fix\ (\lambda t : Bool.t)$ are members of the partial type $\overline{Bool}$.

   You have to prove that the progress theorem of STLC still holds after extending STLC with partial types and fix in the above way. The theorem has been stated without proof at the end of the above file. You might find it useful to look at the proof of this theorem for the original STLC: `http://www.cis.upenn.edu/~bcpierce/sf/current/StlcProp.html#lab682`