G22.3110 Honors Programming Languages
Type Theory Homework
Due Monday, December 20

1. In the simply typed lambda calculus, $\lambda \rightarrow$, prove the following typing assertions using the typing axioms and inference rules.
   a) $x: \sigma, y: \sigma \rightarrow \tau \vdash yx: \tau$
   b) $x: \sigma, y: \sigma \vdash \lambda y: \sigma \rightarrow \tau. yx: (\sigma \rightarrow \tau) \rightarrow \tau$
   c) $\emptyset \vdash (\lambda x: \sigma \rightarrow \sigma. \lambda y: \tau. x) (\lambda x: \sigma. x): \tau \rightarrow \sigma \rightarrow \sigma$

2. In the predicative polymorphic calculus, $\lambda \rightarrow \Pi$, sketch the derivations of the following typing judgements.
   a) $\emptyset \vdash \lambda s: U_1. \lambda t: U_1. \lambda x: s \rightarrow t. \lambda y: s. xy: (s \rightarrow t) \rightarrow s \rightarrow t$
   b) $s: U_1, x: s, y: s \vdash \lambda z: s \rightarrow s. zxy: (s \rightarrow s \rightarrow s) \rightarrow s$
   c) $s: U_1, x: s \rightarrow s \vdash (\lambda y: s \rightarrow s. x) (\lambda x: s. x): s \rightarrow s$

3. In $\lambda \rightarrow \Pi$, the predicative polymorphic calculus extended with the ML-style let construct, show the derivation for
   $$\emptyset \vdash \text{let} f: \Pi t: U_1. t \rightarrow t = \lambda t: U_1. \lambda x: t. x \text{ in } \text{if } f \text{ true then } f \ 3 \ \text{ else } 4 \ : \ \text{nat}$$
   Be sure to specify the relevant portions of the signature $\Sigma$ of the language.

4. Write the full term (i.e. including the types) for the pre-term $(\lambda x. x \ x)$ in the impredicative calculus.

5. Write the full term for a very simple abstract data type (using abstype). Be sure to include the type information.

6. What does the term contravariance refer to in the context of $\lambda < \rightarrow$?, the simply typed lambda calculus with subtyping? Give the typing rule that induces the contravariance.

7. Given the signature of a language in $\lambda < \rightarrow$ as $\Sigma = \langle \{\text{nat, real}\}, \{\text{nat } <: \text{ int, int } <: \text{ real}\}, ... \rangle$, show the subtyping hierarchy of types of the form $(\sigma \rightarrow \tau) \rightarrow \rho$, where each of $\sigma, \tau, \text{ and } \rho$ are either $\text{nat, int, or real}$. 

8. Given the type $\text{point}$ as defined in the lecture notes, show that the type $\text{movable}$ defined by
   $$\text{type} \movable = \langle \text{move}: \text{int } \rightarrow \text{int } \rightarrow \movable \rangle$$
   is a supertype of $\text{point}$.

9. Write, using a lambda calculus with F-bounded subtype polymorphism, a polymorphic sort procedure for sorting lists of elements of any record type $\sigma$ providing the $\text{lessthan}$ method of type $\sigma \rightarrow \sigma \rightarrow \text{bool}$. However, don’t write the actual code for the sort function, just write its signature, along with any necessary type declarations. Assume $\sigma \text{list}$ is a valid type, for any type $\sigma$. 