1 Hereditary Substitution [30 points]

The operation of hereditary substitution is like ordinary (capture-avoiding) substitution, but it does some \(\beta\)-reduction as it goes. The notation is \([t/x]^\phi t'\), where \(t\) is the term to substitute for variable \(x\) in term \(t'\), and \(\phi\) is a type which limits the \(\beta\)-reduction which is allowed to happen. The definition is as follows (where I write \(b\) for a sole base type):

\[
\begin{align*}
[t/x]^{T}_x &= t \\
[t/x]^{T}_y &= y, \text{ if } y \in \text{Var}, y \neq x \\
[t/x]^{\lambda y.t'} &= \lambda y.([t/x]^{T'} t') \\
[t/x]^{t_1 t_2} &= ([t/x]^{t_1}) ([t/x]^{t_2}) \\
[t/x]^{T \rightarrow T'} (t_1 t_2) &= \begin{cases} 
\text{let } s_1 = ([t/x]^{T \rightarrow T'} t_1) \text{ in} \\
\text{let } s_2 = ([t/x]^{T \rightarrow T'} t_2) \text{ in} \\
\text{if } s_1 \equiv \lambda y.s'_1 \text{ for some } y, s'_1, \text{ then} \\
\frac{s_2/y}{T} s'_1 \\
\text{else} \\
(s_1 s_2)
\end{cases}
\end{align*}
\]

1. Using the above definition, compute the values of the following [5 points each]:

   a. \([\lambda x.(z (x x))]/y^b(y y)\)
   b. \([\lambda x.(z (x x))]/y^{b-b}(y y)\)
   c. \([\lambda x.(z (x x))]/y^{(b-b)-(b)}(y y)\)

2. Let us call a measure of the inputs to hereditary substitution \textit{sometimes decreasing} iff there is at least one equation in the definition above where that measure decreases from the left hand side to some recursive call on the right hand side. List two measures that are sometimes decreasing. [5 points]

3. Prove that hereditary substitution always terminates, by defining a measure which is always decreasing from each left hand side above to each recursive call on the corresponding right hand side. [10 points]

2 Constraint-based typing [25 points]

1. Using the constraint-generating typing rules, compute the set of constraints for the term \(\lambda n. \lambda s. \lambda z. (s (n s z))\). Show the typing derivation that generates the constraints. [10 points].

2. Using the constraint-solving rules (i.e., the unification rules), compute a solved form for those constraints (hint: they are solvable). Show the derivation you use. [10 points]

3. Apply the substitution determined by the solved form of the constraints to the type (with meta-variables) computed by the derivation from part (1). [5 points]
3 Reducibility [15 points]

1. Argue that \((\lambda x.x)(\lambda y.y)) \in \[ b \rightarrow b \] [5 points].

2. Give an example of a term which is in \([ b \rightarrow b ]\) but not \([ b \rightarrow b \rightarrow b ]\). Give an argument for why your term is in the first set but not the second [10 points].

4 System F and Church-Encoded Data [30 points]

1. Give a Church-encoding in System F of the datatype of polymorphic binary trees – including a definition of \((tree A)\) parametric in the type \(A\) – with these term constructors [10 points]:
   \[
   \text{leaf} : A \rightarrow (tree A) \\
   \text{node} : A \rightarrow (tree A) \rightarrow (tree A) \rightarrow (tree A)
   \]

2. Write down the term in System F corresponding to the tree with 1 at the root, left subtree a leaf with 0, and right subtree a leaf with 0, where 1 and 0 are the Church-encoded versions of the corresponding natural numbers [10 points].

3. Write down a term in System F which will compute the list (using the Church-encoded version of lists, which we saw in class) of elements stored in the tree, in prefix order [10 points].