Assignment Description
Do the following by yourself.

Consider the following types.

$$\tau ::= \bot \mid T \mid \tau_1 \to \tau_2 \mid \mu t.\tau \mid t$$

Assume that all types under consideration start out with no free variables and have only “uniquified” type variables, and that alpha-conversion for types has already been defined.

Using the final set of rules we discussed in class for defining joins and meets:

a) Choose two nontrivial types, $\tau_1$ and $\tau_2$, whose join, $\tau_3$, is also nontrivial. Here nontrivial means that recursion is involved, but not just simple recursive types like $\mu t.t$, $\mu t.T$, or $\mu t.\bot$. Please also don’t choose the example types used in class, or simple extensions thereof, such as $\mu t.(\mu t'.(t' \to \bot))$ (recall that in class we used $\mu t'.(t' \to \bot)$). The idea is to try out the join rules on your own, using some new types that will really test the rules.

b) Show the derivation proving that $\tau_3$ is the join of $\tau_1$ and $\tau_2$.

c) Show the derivations that $\tau_1$ and $\tau_2$ are subtypes of $\tau_3$.

d) Show the derivation calculating that some type $\tau_4$ is the meet of $\tau_1$ and $\tau_2$.

e) Show the derivations that $\tau_1$ and $\tau_2$ are supertypes of $\tau_4$.

Your derivations can take the same shortcuts we used when writing derivations in class. Please explain any problems you run into.

Note: Even with this seemingly simple type system, the subtyping relation is not anti-symmetric, meaning that it’s just a preorder (satisfying reflexivity and transitivity), not a partial order (satisfying reflexivity, transitivity, and anti-symmetry). There exist non-identical types $\tau_1$ and $\tau_2$ that are subtypes of each other. In such cases we say that $\tau_1$ and $\tau_2$ are equivalent. For example, $\tau_1 = \mu t.(t \to \bot)$ and $\tau_2 = \mu t'.((\mu t'.(t' \to \bot)) \to \bot)$ are equivalent, so it’s OK that joining $\tau_1$ with itself produces $\tau_2$.

For +20% extra credit, you could prove or disprove that the join and meet rules are correct, i.e., they really do return joins and meets for types.