DCC888 – Type Systems [1]

Name: _____ ID: _____

1. In our language of boolean and arithmetic expressions we have four composite terms: *if* t_1 *then* t_2 *else* t_3, *succ* t_1, *pred* t_1 and *iszero* t_1. Each composite term has one or three subterms. For instance, *succ* t_1 has the subterm t_1, and *if* t_1 *then* t_2 *else* t_3 had three subterms, t_1, t_2 and t_3. In this exercise, you must show that a subterm of a well-typed composite term is also well-typed.

2. In our small-step evaluation rules we have a rule for evaluating the predecessor of zero that is a bit non-intuitive:

$$\textit{pred}\ 0 \rightarrow 0$$

   Were we to remove this rule, which property of safety, e.g., *progress* or *preservation*, would we break, and why?

[1] These exercises have been taken from the book *Types and Programming Languages* by Benjamin Pierce.

3. The preservation theorem, e.g., "if $t : T$ and $t \to t'$, then $t' : T$", is often called *subject reduction*. The intuition is that a typing statement $t : T$ can be thought of as a sentence, "t has type T." The term t is the subject of this sentence, and the subject reduction property then says that the truth of the sentence is preserved under reduction of the subject. It is reasonable to wonder if the opposite property – subject *expansion* – also holds. Is it always the case that, if $t \to t'$ and $t' : T$, then $t : T$? If you believe it so, prove it; otherwise, give a counterexample.

4. We read the *Progress Theorem* as follows: suppose t is a well-typed term, e.g., $t : T$ for some T. Then either t is a value, or else there exists some t' with $t \to t'$. If we were to have the following evaluation rules in our language:

   - $\mathit{true} \to \mathit{true}$
   - $\mathit{false} \to \mathit{false}$
   - $n \to n$, where n is a number

   Then how could we re-state our progress property? What would be the implications of having these rules in our language?