Due date: Sept. 22nd in class.

1. (1 points) What does \((\lambda x. x (\lambda y. x y)) (\lambda x y x)\) reduce to?

2. (3 points) Write a lambda-calculus term that represents the Fibonacci function, defined by

\[
\text{Fib}(n) = \begin{cases} 
1 & \text{when } n=0 \\
1 & \text{when } n=1 \\
\text{Fib}(n-2) + \text{Fib}(n-1) & \text{when } n > 1
\end{cases}
\]

Apply your term to 4 and evaluate it to a normal form.

3. Two styles of operational semantics are in common use. The one we studied in class and used in Fig.3-2 of the textbook is called the small-step style, because the definition of evaluation relation shows how individual steps of computation are used to rewrite a term, bit by bit, until it eventually becomes a value. On top of this, we define a multi-step evaluation relation that allows us to talk about terms evaluating (in many steps) to values. An alternative style, called big-step semantics (or sometimes natural semantics), directly formulates the notion of “this term evaluates to that final value,” written \(t \Downarrow v\). The big-step evaluation rules for the language of boolean and arithmetic expressions looks like this:

\[
\begin{array}{c}
v \Downarrow v \\
\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow v_2 \\
t_1 \Downarrow \text{true} \\
t_1 \Downarrow \text{false} \\
succ t_1 \Downarrow \text{succ } n v_1 \\
pred t_1 \Downarrow 0 \\
\text{iszero } t_1 \Downarrow \text{true} \\
\text{iszero } t_1 \Downarrow \text{false} \\
t_1 \Downarrow 0 \\
t_1 \Downarrow \text{succ } n v_1 \\
t_1 \Downarrow \text{succ } n v_1 \\
t_1 \Downarrow \text{succ } n v_1 \end{array}
\]
(a) (3 points) Show that the small-step semantics in Fig.3-2 and the big-step semantics for the language coincide, i.e., $t \rightarrow^* v$ iff $t \Downarrow v$.

(b) (3 points) With the big-step semantics, how should the intuitive property of type safety be formalized? Write a formal theorem for it and also prove it.