Practice Problems – Operational Semantics

1. Recall the language IMP from class:

\[
\begin{align*}
  a & ::= \ n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \cdot a_1 \\
  b & ::= \ bv \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \land b_1 \mid b_0 \lor b_1 \\
  c & ::= \ \text{skip} \mid X := a \mid \text{if } b \text{ then } c_0 \text{ else } c_1 \mid \text{while } b \text{ do } c \\
  bv & ::= \ \text{true} \mid \text{false}
\end{align*}
\]

Suppose we extend the language with a C-style for loop:

\[
c ::= \cdots \mid \text{for}(c_0; b; c_1) \ c_2
\]

Write down big-step operational semantics for “for.” You may not use “while” in the hypotheses of your “for” rules. Hint: the “skip” command may come in handy.

2. Here is the lambda calculus, extended with integers, and its semantics:

\[
\begin{align*}
e & ::= \ v \mid x \mid e \ e \\
v & ::= \ n \mid \lambda x.e
\end{align*}
\]

\[
\begin{array}{l}
\text{Beta} \\
(\lambda x.e) v \rightarrow e[ x \mapsto v ]
\end{array}
\quad
\begin{array}{l}
\text{Left} \\
e_1 \rightarrow e'_1 \\
e_1 e_2 \rightarrow e'_1 e_2
\end{array}
\quad
\begin{array}{l}
\text{Right} \\
e_2 \rightarrow e'_2 \\
v e_2 \rightarrow v e'_2
\end{array}
\]

Draw derivations showing that the following reductions hold:

(a) \((\lambda x.42) \ 13 \rightarrow 42\)

(b) \((\lambda x. (\lambda y.y)) \ (\lambda z.z) \rightarrow (\lambda y.y) \ (\lambda z.z)\)

(c) \((\lambda x. ((\lambda x.\lambda y.x) \ 42)) \rightarrow (\lambda x.\lambda y.42)\)

3. Write down big-step semantics for lambda calculus that are equivalent to the rules above (for terminating programs).

4. Draw a derivation of the following in your big-step semantics: \((\lambda x. (\lambda x.\lambda y.x) \ 42) \rightarrow (\lambda y.42)\)