Terminology
Regular expression, judgement, premise, conclusion, reduction rule, inference rule, operational semantics, short-circuiting, order of evaluation, type error, stuck error, substitution, static typing, dynamic typing.

Order of Evaluation
The Scala operator ^ represents the logical connective ‘XOR’, which we saw in Homework 1. Suppose we extend Smalla with the following rules for ^, where \( \oplus \) denotes mathematical XOR:

\[
\begin{align*}
& e_2 \rightarrow e_2' \\
& e_1^\top e_2 \rightarrow e_1^\top e_2'
\end{align*}
\]

\[
\begin{align*}
& e_1 \rightarrow e_1' \\
& e_1^\top b_2 \rightarrow e_1^\top b_2'
\end{align*}
\]

\[
\begin{align*}
& b_1 \oplus b_2 = b \\
& b_1^\top b_2 \rightarrow b
\end{align*}
\]

What is the order of evaluation for \( e_1^\top e_2 \)? (Similar to homework question 2.b.i) Explain what each of these rules is saying.

Substitution
To understand how the substitute function in problem 5 works, we will look at an example of function application.

**APPLY**

\[
((x_1: \tau_1) \Rightarrow e_1)(v_2) \rightarrow [v_2/x_1]e_1
\]

The left hand side of the APPLY rule uses the notation \((x_1: \tau_1) \Rightarrow e_1\) to denote a function which takes in an argument \(x_1\) of type \(\tau_1\) and has body \(e_1\), and is applied to a value \(v_2\).

The right hand side of the APPLY rule uses the notation \([v_2/x_1]e_1\) to say that we “replace all occurrences of variable \(x_1\) with value \(v_2\) in the function body \(e_1\)”. This is exactly what the substitute function in the homework is meant to do. In fact, the arguments to substitute(v: Val, x: String, e: Expr): Expr are in the same order as \([v_2/x_1]e_1\).

**Example.** How would taking a step of reduction (including a substitution) on the following expression work?

\((x: Int \Rightarrow (x + y) > (x * x))(6)\)

Reduction Rules
To understand how the step function in problem 5 works, we will look at several of the “trickier looking” evaluation rules. We will split into several groups, and each group will get a rule. Do the following for your rule so that you can present it to the class:
• Determine what the rule means. What are the premises and conclusion saying?

• Give an expression to which we can apply the rule, and one that we cannot. Explain why we can apply the rule to the first expression but not the second.

• Show the steps in a reduction from the first expression down to a value using your “tricky” rule and some easier rules.

\[
\begin{align*}
\text{EQUALITY} & \\
v_1 \neq (x_1: \tau_1) \Rightarrow e_1 & \quad v_2 \neq (x_2: \tau_2) \Rightarrow e_2 & \quad b' = (v_1 = v_2) \\
\hline
v_1 == v_2 & \rightarrow b'
\end{align*}
\]

\[
\begin{align*}
\text{SUBSTITUTION} & \\
((x_1: \tau_1) \Rightarrow e_1)(v_2) & \rightarrow [v_2/x_1]e_1
\end{align*}
\]

\[
\begin{align*}
\text{BOP-REDUCE} & \\
e_2 & \rightarrow e'_2 & \quad \text{bop} \in \{+, -, *, <\}
\end{align*}
\]

\[
\begin{align*}
n_1 \text{ bop } e_2 & \rightarrow n_1 \text{ bop } e'_2
\end{align*}
\]

The multi-step reduction relation

Recall that “\(\rightarrow\)” denotes a single step of reduction, and “\(\rightarrow^*\)” zero or more steps of reduction. These reductions relate closely to two functions used in the problem 5: the \texttt{step} function reduces by a single step, while the \texttt{eval} function reduces many steps. In fact, to evaluate certain expressions we need the multi-step reduction. Here we define the multi-step reduction relation using the single-step relation.

We define the \textbf{multi-step reduction relation} to be the reflexive, transitive closure of the small-step rules in figure 3. That is, the rules in figure 3 together with the \texttt{reflexivity} and \texttt{transitivity} rules given below. To help build intuition, think about what happens in \texttt{eval}: if it detects that the input expression is a value then it will not call \texttt{step} at all. However, it may also call \texttt{step} many times. In some cases (see example) the multi-step relation helps us to evaluate expressions which we couldn’t reduce in a single step.

\[
\begin{align*}
\text{REFLEXIVITY} & \\
e & \rightarrow^* e
\end{align*}
\]

\[
\begin{align*}
\text{TRANSITIVITY} & \\
e & \rightarrow e' & e' & \rightarrow^* e'' \\
e & \rightarrow^* e''
\end{align*}
\]

Example. \textit{How would the new rules be used in evaluating the following expression?}
\((x : \text{Int} \Rightarrow y : \text{Int} \Rightarrow x + y)(6)(7)\)}