We shall now consider a more realistic programming language, one where we can assign values to variables and execute control constructs such as "**if**" and "**while**". The syntax for this simple imperative language, called IMP, is as follows:

expressions	$e \in \mathsf{Expr} = \mathsf{AExp} \cup \mathsf{BExp}$	
arithmetic expressions	$a \in \mathsf{AExp}$	$a ::= x \mid n \mid a_1 + a_2$
boolean expressions	$b \in \mathsf{BExp}$	$b ::= \textbf{true} \mid \textbf{false} \mid a_1 < a_2$
commands	$c \in \mathsf{Com}$	$c ::= \textbf{skip} \mid x := a \mid c_1; c_2$ $\mid \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2$ $\mid \textbf{while } b \textbf{ do } c$

We'll first give a small-step operational model for IMP. The configurations in this language are of the form $\langle c, s\rangle$, $\langle b, s\rangle$, and $\langle a, s\rangle$, where s is a store. And the final configurations are $\langle \textbf{skip}, s\rangle$, $\langle \textbf{true}, s\rangle$, $\langle \textbf{false}, s\rangle$, and $\langle n, s\rangle$. We need to define the one-step evaluation relations for commands and expressions: $\langle c, s\rangle \rightarrow \langle c', s'\rangle, \langle b, s\rangle \rightarrow \langle b', s'\rangle, \langle a, s\rangle \rightarrow \langle a', s'\rangle$.

The evaluation rules for arithmetic and boolean expressions are similar to the ones we've seen before. For commands, the rules are:

$$\frac{\langle e, s\rangle \rightarrow \langle e', s\rangle}{\langle x := e, s\rangle \rightarrow \langle x := e', s\rangle} \qquad \frac{}{\langle x := n, s\rangle \rightarrow s[x \mapsto n]}$$

$$\frac{\langle c_1, s\rangle \rightarrow \langle c_1', s'\rangle}{\langle c_1; c_2, s\rangle \rightarrow \langle c_1'; c_2, s\rangle} \qquad \frac{}{\langle \textbf{skip}; c, s\rangle \rightarrow \langle c, s\rangle}$$

For **if** commands, we gradually reduce the test until we get either **true** or **false**; then, we execute the appropriate branch:

$$\frac{\langle b, s\rangle \rightarrow \langle b', s\rangle}{\langle \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2, s\rangle \rightarrow \langle \textbf{if } b' \textbf{ then } c_1 \textbf{ else } c_2, s\rangle} \qquad \frac{}{\langle \textbf{if true then } c_1 \textbf{ else } c_2, s\rangle \rightarrow \langle c_1, s\rangle}$$

$$\frac{}{\langle \textbf{if false then } c_1 \textbf{ else } c_2, s\rangle \rightarrow \langle c_2, s\rangle}$$

For **while** loops, the above strategy doesn't work (why?). Instead, we can use the following rule:

$$\frac{}{\langle \textbf{while } b \textbf{ do } c, s\rangle \rightarrow \langle \textbf{if } (b) \textbf{ then } (c; \textbf{while } b \textbf{ do } c) \textbf{ else skip}, s\rangle}$$

We can now take a concrete program and see how it executes under the above rules. Consider we start with state $s = \{x = 0\}$ and we execute the program:

$$x := 3; \textbf{while } (x < 4) \textbf{ do } x := x + 5$$

The execution works as follows:

$$
\begin{aligned}
&\langle x := 3; \textbf{while } (x < 4) \textbf{ do } x := x + 5,\ s\rangle \rightarrow \\
\rightarrow&\langle \textbf{skip}; \textbf{while } (x < 4) \textbf{ do } x := x + 5,\ s'\rangle && \text{(where } s' = s[x \mapsto 3]\text{)} \\
\rightarrow&\langle \textbf{while } (x < 4) \textbf{ do } x := x + 5,\ s'\rangle \\
\rightarrow&\langle \textbf{if } (x < 4) \textbf{ then } (x := x + 5; W) \textbf{ else skip},\ s'\rangle \\
\rightarrow&\langle \textbf{if } (3 < 4) \textbf{ then } (x := x + 5; W) \textbf{ else skip},\ s'\rangle \\
\rightarrow&\langle \textbf{if true then } (x := x + 5; W) \textbf{ else skip},\ s'\rangle \\
\rightarrow&\langle x := x + 5; \textbf{while } (x < 4) \textbf{ do } x := x + 5,\ s'\rangle \\
\rightarrow&\langle x := 3 + 5; \textbf{while } (x < 4) \textbf{ do } x := x + 5,\ s'\rangle \\
\rightarrow&\langle x := 8; \textbf{while } (x < 4) \textbf{ do } x := x + 5,\ s'\rangle \\
\rightarrow&\langle \textbf{while } (x < 4) \textbf{ do } x := x + 5,\ s''\rangle && \text{(where } s'' = s'[x \mapsto 8]\text{)} \\
\rightarrow&\langle \textbf{if } (x < 4) \textbf{ then } (x := x + 5; W) \textbf{ else skip},\ s''\rangle \\
\rightarrow&\langle \textbf{if } (8\langle 4) \textbf{ then } (x := x + 5; W) \textbf{ else skip},\ s''\rangle \\
\rightarrow&\langle \textbf{if false then } (x := x + 5; W) \textbf{ else skip},\ s''\rangle \\
\rightarrow&\langle \textbf{skip},\ s''\rangle
\end{aligned}
$$

(where W is an abbreviation for the **while** loop **while** $(x < 4)$ **do** $x := x + 5$).