Exercise 1

One can extend the Imp language with a command “repeat c until b”. The operational semantics of this loop is given by the following derivation rules which should be added to the set of standard Imp rules.

\[
\begin{align*}
\langle c, \sigma \rangle &\rightarrow \sigma' \quad \langle b, \sigma' \rangle &\rightarrow \text{true} \\
\langle \text{repeat } c \text{ until } b, \sigma \rangle &\rightarrow \sigma'
\end{align*}
\]

\[
\begin{align*}
\langle c, \sigma \rangle &\rightarrow \sigma'' \quad \langle b, \sigma'' \rangle &\rightarrow \text{false} \\
\langle \text{repeat } c \text{ until } b, \sigma'' \rangle &\rightarrow \sigma'
\end{align*}
\]

Let \( \sigma = \sigma[3 \mapsto 5, 4 \mapsto 4] \). Let

\[
p \equiv \text{repeat if not}(y < x) \text{ then } y := y - 1 \text{ else } x := x - y \text{ until } x = y
\]

Find \( \sigma_* \) such that \( \langle p, \sigma \rangle \rightarrow \sigma_* \) can be derived. Draw a complete derivation tree. What is the height of that tree?

Exercise 2

Prove that for any \( b \in \text{Bexp}, c \in \text{Com} \) and any states \( \sigma, \sigma_* \) if

\[
\langle \text{repeat } c \text{ until } b, \sigma \rangle \rightarrow \sigma_ *
\]

can be derived then

\[
\langle c; \text{while not } b \text{ do } c, \sigma \rangle \rightarrow \sigma_ *
\]

can also be derived.

Exercise 3

Using the operational semantics of Imp prove that for any program \( c \in \text{Com} \) which does not contain any assignment statements if \( \langle c, \sigma \rangle \rightarrow \sigma' \) then \( \sigma' = \sigma \).

Exercise 4

Imagine introducing the break command into Imp, whose intended behaviour is to immediately abort execution of the smallest enclosing while-loop (if any) and to return control to the following commands (if any).

Thus for example, using this new command, the command \( \text{while } b \text{ do } c \) is equivalent to

\[
\text{while true do if } b \text{ then } c \text{ else break}
\]

The syntax of the extended language is

\[
c := \text{skip | break | } x := a | a_0 ; c_1 | \text{if } b \text{ then } a_0 \text{ else } c_1 | \text{while } b \text{ do } c.
\]

Give a complete operational semantic definition for this extended language.
Note: You only need to give rules for executing commands—infrence rules for arithmetic and Boolean expressions are the same as in the original Imp language.

Exercise 5

Modify the ML implementation of the operational semantics to include the language extension with break above. Submit a printout of the ML file with your modifications marked using a pen. Include a printout of an ML run where you demonstrate the changes by executing some simple example programs of your own invention.