Problem 0.2 from [2] Consider the program SB given below:

```
out x: integer where x = 0
l0 : [[l1 : while x ≥ 0 do l2 : x := x + 1] or
l3 : await x > 0]
l4 :
```

b. Show that this program has a terminating computation as well as no terminating. Solution The transitions of the program are \( \{l_0^{m}, l_0^{n}, l_2, l_3\} \). A terminating computation:

\[
\sigma = \langle \pi : \{l_0\}, x : 0 \rangle \xrightarrow{l_0} \langle \pi : \{l_2\}, x : 0 \rangle \xrightarrow{l_2} \langle \pi : \{l_0\}, x : 1 \rangle \xrightarrow{l_0} \langle \pi : \{l_3\}, x : 1 \rangle \xrightarrow{l_2} \langle \pi : \{l_4\}, x : 1 \rangle
\]

A no terminating computation is:

\[
\sigma' = \langle \pi : \{l_0\}, x : 0 \rangle \xrightarrow{l_0} \langle \pi : \{l_2\}, x : 0 \rangle \xrightarrow{l_2} \langle \pi : \{l_0\}, x : 1 \rangle \xrightarrow{l_0} \langle \pi : \{l_2\}, x : 1 \rangle \xrightarrow{l_2} \langle \pi : \{l_0\}, x : 2 \rangle \ldots
\]

The computation \( \sigma' \) is proper computation as:
- even though the transition \( l_2 \) is offered infinitely often it is not in continuity as it is not enabled in states with \( \pi : \{l_2\} \). Thus the justice requirement is satisfied.
- Regarding the compassion requirement no compassionate statements appear in the computation, thus this requirements is satisfied trivially.

Problem 0.1 from [2] Consider the program ANY-NAT presented below. Argue that the program always terminate. Also, show that for every natural number \( n \geq 0 \) there exists a computation such that \( y = n \) on termination.

```
out y: integer where y = 0
local x: boolean where x = T
P1 :: [l0 : while x do
l1 : y := y + 1
l2 :
] || P2 :: [m0 : x := F
m1 :
```

Solution Consider the only possible no terminating computation:

\[
\sigma = \langle \pi : \{l_0, m_0\}, y : 0, x : T \rangle \xrightarrow{l_0} \langle \pi : \{l_1, m_0\}, y : 0, x : T \rangle \xrightarrow{l_1} \langle \pi : \{l_0, m_0\}, y : 1, x : T \rangle \xrightarrow{l_0} \langle \pi : \{l_1, m_0\}, y : 1, x : T \rangle \xrightarrow{l_1} \ldots
\]

In this sequence the statement (transition) \( m_0 \) is continuously enabled but it is not taken, thus this sequence is not just on \( m_0 \). Therefore, it is not a computation.

Assume \( n \geq 0 \) is a natural number. Then the following computation terminates with \( y = n \).

\[
\sigma_0 = \langle \pi : \{l_0, m_0\}, y : 0, x : T \rangle \xrightarrow{m_0} \langle \pi : \{l_1, m_1\}, y : 0, x : F \rangle \xrightarrow{l_0} \langle \pi : \{l_2, m_1\}, y : 0, x : F \rangle
\]

\[
\sigma_n = \langle \pi : \{l_0, m_0\}, y : 0, x : T \rangle \xrightarrow{l_0} \langle \pi : \{l_1, m_0\}, y : 0, x : T \rangle \xrightarrow{l_1} \langle \pi : \{l_0, m_0\}, y : 1, x : T \rangle \xrightarrow{l_0} \langle \pi : \{l_1, m_0\}, y : 1, x : T \rangle \xrightarrow{l_1} \langle \pi : \{l_0, m_0\}, y : 2, x : T \rangle \xrightarrow{l_0} \langle \pi : \{l_1, m_0\}, y : 2, x : T \rangle \xrightarrow{l_1} \langle \pi : \{l_0, m_0\}, y : 3, x : T \rangle \xrightarrow{l_0} \langle \pi : \{l_1, m_0\}, y : 3, x : T \rangle \xrightarrow{l_1} \langle \pi : \{l_0, m_0\}, y : 4, x : T \rangle \xrightarrow{l_0} \ldots \xrightarrow{l_0} \langle \pi : \{l_0, m_0\}, y : n, x : T \rangle \xrightarrow{m_0} \langle \pi : \{l_0, m_1\}, y : n, x : F \rangle
\]

b. Construct a program with a single process that exhibits a similar behaviour; that is, all of its computations terminate and, for each natural number \( n \), there exists a computation producing \( n \).

Solution
out y : integer where y = 0
out z : integer where z = 1
local x : boolean where x = T

P₀ :: l₀ :

\[
\left[
\begin{array}{c}
l₁₀ : \text{while } y \geq 0 \text{ do} \\
l₂ : \text{[when } x = T \text{ do } l₃ : y := y + 1] \\
l₄ : \text{release}(z) \\
\end{array}
\right]
\]

or

\[
\left[
\begin{array}{c}
l₁₀ : \text{request}(z) \\
m₂ : x := F \\
\end{array}
\right]
\]

A computation that terminates with \( y = n \) can be found in a similar way as above.
The program does not have a no terminating computations, as any infinite sequence of transitions does not satisfy compassion requirement.

NOTE: The program can be simplified as discussed during the instructions.

Problem 0.4 from [2] (mutual exclusion)

Solution

TRY-MUX1:
- Mutual exclusion NO!
- Accessibility NO!
- Communal accessibility YES!

TRY-MUX2:
- Mutual exclusion NO!
- Accessibility NO!
- Communal accessibility NO!

TURN:
- Mutual exclusion YES!
- Accessibility YES!
- Communal accessibility YES!