Homework - Version A.

Exercise 1

We define the syntactic category of letters \( L = \{a, b, c\} \). By \( l \) we denote a meta-variable ranging over \( L \). We define inductively a set of words \( W \) by the following BNF:

\[
\begin{align*}
w & := l \mid w_1cw_2b \mid cwab
\end{align*}
\]

where \( w, w_1, w_2 \) are meta-variables ranging over the set of words \( W \). Of the following two statements, one is wrong and one is right. You get to tell which is which and justify your answers by either a proof (by induction) or a counter-example.

1. The length of words in \( W \) is congruent to 1 modulo 3.
2. All words in \( W \) finish with a letter \( b \).

The first statement is true. We provide a proof by structural induction.

Basic case:
The atoms of the set are letters. Every letter \( l \in L \) has length 1, which is indeed congruent to 1 mod 3.

Case of rule ‘if \( w_1, w_2 \in W \) then \( w_1cw_2b \in W \)’:
our induction hypothesis is that \( w_1 \) and \( w_2 \) verify the property we want to prove, i.e. their respective length is congruent to 1 mod 3. The length of the word \( w_1cw_2b \) is 2 plus that of \( w_1 \) and that of \( w_2 \). Hence we have a global length congruent to \((2+1+1) \mod 3 = 1 \mod 3\).

Case of rule ‘if \( w \in W \) then \( cwab \in W \)’:
our induction hypothesis is that \( w \) verifies the property we want to prove, i.e. its length is congruent to 1 mod 3. Hence the length of \( cwab \) is congruent to \((1+3) \mod 3 = 1 \mod 3\).

In conclusion, we have proven by structural induction that for every word \( w \in W \), the length of \( w \) is congruent to 1 mod 3.

The second statement is wrong: \( a \) is a letter, hence by the first rule it belongs to \( W \), and word \( a \) does not finish with \( b \).

Exercise 2

We consider the following program.

```plaintext
begin
var y := 2;
var x := 71 - y;
proc plop is x := y + x;
begin var y := 1;
proc p is call plop;
proc plop is x := y;
call p;
end
call plop;
end
```
You have been presented three different semantics for the While language with blocks and procedures: one with dynamic links for variables and procedures, another with dynamic links for variables but static links for procedures, and finally one with static links for variables and procedures.

What values are associated to $x$ and $y$ at the end of this program according to each of the three semantics you know? Justify your answer (you can either draw the tree or precise the state or the variable environment and the storage function after each ‘;’).

\[
\begin{align*}
\text{begin} & \quad D_V = \left\{ \begin{array}{l}
\text{var } y := 2; \\
\text{var } x := 71 - y; \\
\text{proc } plop \text{ is } x := y + x;
\end{array} \right. \\
D_P & \quad = \left\{ \begin{array}{l}
\text{begin } D'_V = \left\{ \begin{array}{l}
\text{var } y := 1; \\
\text{proc } p \text{ is call } plop;
\end{array} \right. \\
D'_P & \quad = \left\{ \begin{array}{l}
\text{proc } plop \text{ is } x := y;
\end{array} \right. \\
S' & \quad = \left\{ \begin{array}{l}
\text{call } p;
\end{array} \right. \\
S_1 & \quad = \left\{ \begin{array}{l}
\text{begin } D'_V = \left\{ \begin{array}{l}
\text{var } y := 1; \\
\text{proc } p \text{ is call } plop;
\end{array} \right. \\
D'_P & \quad = \left\{ \begin{array}{l}
\text{proc } plop \text{ is } x := y;
\end{array} \right. \\
S' & \quad = \left\{ \begin{array}{l}
\text{call } p;
\end{array} \right. \\
S_2 & \quad = \left\{ \begin{array}{l}
\text{call } plop;
\end{array} \right. \\
\end{array} \right. \\
\text{end} & \quad \text{end}
\end{align*}
\]
where: \( \sigma \) and \( env_P \) are initial state and environment

- \( \sigma_2 = \sigma[x \mapsto 69, y \mapsto 2] \)
- \( \sigma_6 = \sigma_2[y \mapsto 1] = \sigma[x \mapsto 69, y \mapsto 1] \)
- \( env_P^1 = env_P[plop \mapsto (x := y)] \)
- \( env_P^2 = env_P[p \mapsto (call plop)] \)
- \( \sigma_3 = \sigma_4[x \mapsto 3] = \sigma[x \mapsto 3, y \mapsto 2] \)
- \( \sigma_6 = \sigma_5[x \mapsto 1, y \mapsto 1] \)
- \( \sigma_4 = \sigma_5[DV(D_V) \mapsto \sigma_2], DV(D_V) = \{x\} \)
- \( \sigma_5 = \sigma_4[x \mapsto 3, y \mapsto 2] \)
- \( \sigma_6 = \sigma_5[y \mapsto 2] = \sigma[x \mapsto 1, y \mapsto 2] \)
- \( \sigma_1 = \sigma_3[DV(D_V) \mapsto \sigma], DV(D_V) = \{x, y\}, \text{ so } \sigma_1 = \sigma \)

Figure 1: Dynamic-dynamic semantics
\[
\begin{align*}
&\text{BL} < D_V, \sigma > \rightarrow V \sigma_6 \\
&\text{CALL} \quad (env_p, x := y + x, \sigma_6) \rightarrow \sigma_5 \\
&\text{ASG} \\
&\text{CALL} \quad (env_p', call plop, \sigma_6) \rightarrow \sigma_5 \\
&\text{CALL} \quad (env_p', S_1, \sigma_2) \rightarrow \sigma_4 \\
&\text{CALL} \quad (env_p', S_2, \sigma_4) \rightarrow \sigma_3 \\
&\text{CALL} \quad (env_p, begin D_V D_p S, \sigma) \rightarrow \sigma_1
\end{align*}
\]

where:
- \(\sigma\) and \(env_p\) are initial state and environment
- \(\sigma_2 = \sigma[x \mapsto 69, y \mapsto 2]\)
- \(env_p' = env_p[plop \mapsto (y := x + y, env_p)]\)
- \(\sigma_6 = \sigma_2[y \mapsto 1] = \sigma[x \mapsto 69, y \mapsto 1]\)

There are two steps to compute the proc. env. update:
- \(env_p'' = env_p'[p \mapsto (call plop, env_p')]\)
- \(env_p'' = env_p'[plop \mapsto (x := y, env_p')]\)

Note that \(env_p'' = env_p'[p \mapsto (call plop, env_p')]\), \(plop \mapsto (x := y, env_p''')\)
- \(\sigma_5 = \sigma_6[x \mapsto 70] = \sigma[x \mapsto 70, y \mapsto 1]\)
- \(\sigma_4 = \sigma_5[y \mapsto 2] = \sigma[x \mapsto 70, y \mapsto 2]\)
- \(\sigma_3 = \sigma_4[x \mapsto 72] = \sigma[x \mapsto 72, y \mapsto 2]\)
- \(\sigma_1 = \sigma\)

Figure 2: Static-dynamic semantics
\[
\begin{align*}
\text{Bl} & \quad <D_V, \sigma > \xrightarrow{V} (env^2_V, sto^2) \quad \frac{T_1}{(env^1_P, S, env^2_V, sto^2) \xrightarrow{\text{SEQ}} (env^3_V, sto^3)} \quad \frac{T_2}{(env^1_V, sto^1)}
\end{align*}
\]

with \( T_1 \):

\[
\begin{align*}
\text{Bl} & \quad <D'_V, env^2_V, sto^2 > \xrightarrow{V} (env^6_V, sto^6) \quad \frac{\text{ASG}}{(env^1_P, x := y + x, env^2_V, sto^4) \xrightarrow{\text{CALL}} (env^5_V, sto^5)} \quad \frac{\text{CALL}}{(env^1_P, call \ plop, env^6_V, sto^5) \xrightarrow{\text{CALL}} (env^5_V, sto^5)}
\end{align*}
\]

with \( T_2 \) being:

\[
\begin{align*}
\text{Bl} & \quad (env^1_P, x := y + x, env^2_V, sto^4) \xrightarrow{\text{ASG}} (env^5_V, sto^5) \quad \frac{\text{CALL}}{(env^1_P, S_1, env^2_V, sto^2) \xrightarrow{\text{CALL}} (env^4_V, sto^4)}
\end{align*}
\]

where:

\( \sigma \) and \( env_P \) are initial state and environment

\[
\begin{align*}
env^0_P = env_V[y \leftarrow 1, x \leftarrow 2] & , sto^0 = sto[1 \leftarrow 2, 2 \leftarrow 69] \\
env^0_V = env_V[x \leftarrow 1, sto^0 = sto[1 \leftarrow 2, 2 \leftarrow 69, 3 \leftarrow 1]
\end{align*}
\]

There are two steps to compute the proc. env. update:

\[
\begin{align*}
env'_p = env_P[p \leftarrow (call \ plop, env^0_P, env^0_V)] & , env'_V = env_V[y \leftarrow 1, x \leftarrow 2] \\
\end{align*}
\]

Note that \( env'_P = env_P[p \leftarrow (call \ plop, env^0_P, env^0_V)] \)

\[
\begin{align*}
env'_P = env^0_P & , sto^5 = sto[1 \leftarrow 2, 2 \leftarrow 71, 3 \leftarrow 1] \\
env^0_V = env^0_V & , sto^4 = sto[1 \leftarrow 2, 2 \leftarrow 71] \\
env^0_V = env^0_V & , sto^3 = sto[1 \leftarrow 2, 2 \leftarrow 73] \\
env^0_V = env^0_V & , sto^1 = sto.
\end{align*}
\]

Figure 3: Static-static semantics