Problem 1 Consider the language of natural numbers, whose abstract syntax is given by the following CFG:

$$
\begin{array}{rcll}
e \in \mathit{expr} & ::= & \mathit{value} & \text{value} \\
& | & \mathit{expr} + \mathit{expr} & \text{addition} \\
& | & \mathit{expr} \times \mathit{expr} & \text{multiplication}
\end{array}
$$

$$
\begin{array}{rcll}
v \in \mathit{value} & ::= & 0 & \text{zero} \\
& | & S(\mathit{value}) & \text{successor}
\end{array}
$$

And the operational semantics is given by the following rules:

$$
\frac{}{v \Rightarrow v}\ \textsc{Val} \qquad
\frac{e_1 \Rightarrow 0 \quad e_2 \Rightarrow v_2}{e_1 + e_2 \Rightarrow v_2}\ \textsc{Add0} \qquad
\frac{e_1 \Rightarrow S(v_1) \quad v_1 + e_2 \Rightarrow v_2}{e_1 + e_2 \Rightarrow S(v_2)}\ \textsc{AddS}
$$

$$
\frac{e_1 \Rightarrow 0 \quad e_2 \Rightarrow v_2}{e_1 \times e_2 \Rightarrow \cancel{v_2}\ 0}\ \textsc{Mul0} \qquad
\frac{e_1 \Rightarrow S(v_1) \quad v_1 \times e_2 \Rightarrow v_2 \quad v_2 + e_2 \Rightarrow v_3}{e_1 \times e_2 \Rightarrow v_3}\ \textsc{MulS}
$$

Problem 1-A Show that there exists some value v such that $S(0) + S(0) \Rightarrow v$.

Problem 1-B Show that there exists some value v such that $S(0) \times S(0) \Rightarrow v$.

1-A

$$
\dfrac{\dfrac{}{S(0) \Rightarrow S(0)}\,\text{Val} \qquad \dfrac{\dfrac{}{0 \Rightarrow 0}\,\text{Val} \quad \dfrac{}{S(0) \Rightarrow S(0)}\,\text{Val}}{0 + S(0) \Rightarrow S(0)}\,\text{Add0}}{S(0) + S(0) \Rightarrow S(S(0))}\,\text{AddS}
$$

1-B

$$
\dfrac{\dfrac{}{S(0) \Rightarrow S(0)}\,\text{Val} \qquad \dfrac{\dfrac{}{0 \Rightarrow 0}\,\text{Val} \quad \dfrac{}{S(0) \Rightarrow S(0)}\,\text{Val}}{0 \times S(0) \Rightarrow 0}\,\text{Mul0} \qquad \dfrac{\dfrac{}{0 \Rightarrow 0}\,\text{Val} \quad \dfrac{}{S(0) \Rightarrow S(0)}\,\text{Val}}{0 + S(0) \Rightarrow S(0)}\,\text{Add0}}{S(0) \times S(0) \Rightarrow}\,\text{MulS}
$$

Problem 2 Consider the CoinPython language. The abstract syntax is given by the following CFG:

$p \in prog$	$::=$	pass	No-op (do nothing)
	$\mid$	raise	Exception
	$\mid$	$\mathsf{print}(m)$	Printing a message
	$\mid$	$p_1;\ p_2$	Sequencing
	$\mid$	$\mathsf{if}\ (\star)\ \{p_1\}\ \mathsf{else}\ \{p_2\}$	Non-deterministic branching
	$\mid$	$\mathsf{while}\ (\star)\ \{p\}$	Non-deterministic loop

Let the set of messages be the set {☺, ☹}. Let $s_1 + s_2$ denotes the concatenation of strings s_1 and s_2. The operational semantics of CoinPython is defined by the following set of rules:

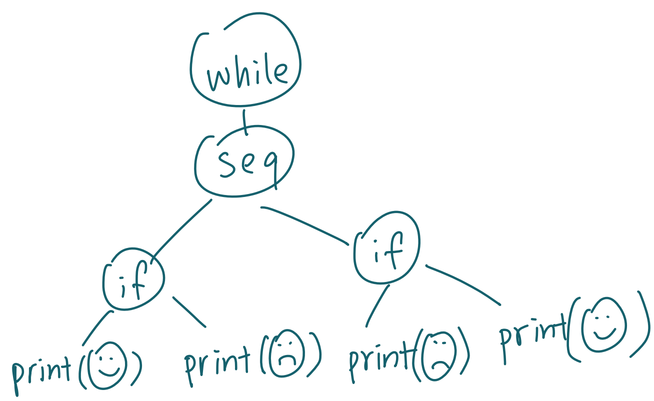
$$\frac{}{\mathsf{pass} \Rightarrow \text{“”}}\ \textsc{Pass} \qquad \text{(No rule for raise)} \qquad \frac{}{\mathsf{print}(m) \Rightarrow m}\ \textsc{Print} \qquad \frac{p_1 \Rightarrow s_1 \quad p_2 \Rightarrow s_2 \quad (s_1 + s_2 = s_3)}{p_1;\ p_2 \Rightarrow s_3}\ \textsc{Seq}$$

$$\frac{p_1 \Rightarrow s_1}{\mathsf{if}\ (\star)\ \{p_1\}\ \mathsf{else}\ \{p_2\} \Rightarrow s_1}\ \textsc{IfTrue} \qquad \frac{p_2 \Rightarrow s_2}{\mathsf{if}\ (\star)\ \{p_1\}\ \mathsf{else}\ \{p_2\} \Rightarrow \cancel{s_1}\ s_2}\ \textsc{IfFalse}$$

$$\frac{}{\mathsf{while}\ (\star)\ \{p\} \Rightarrow \text{“”}}\ \textsc{WhileFalse} \qquad \frac{p \Rightarrow s_1 \quad \mathsf{while}\ (\star)\ \{p\} \Rightarrow s_2 \quad (s_1 + s_2 = s_3)}{\mathsf{while}\ (\star)\ \{p\} \Rightarrow s_3}\ \textsc{WhileTrue}$$

Let P be the program (in concrete syntax):

```
while (*):
  if (*):
    print(☺)
  else (*):
    print(☹)
  if (*):
    print(☹)
  else (*):
    print(☺)
```



Problem 2-A Draw the AST for P.

Problem 2-B Show that $P \Rightarrow$ ☺☹☹☺.

Problem 2-C How would you describe the set of all strings that *may* be printed by P?

——— Print
p(☺) ⇒ ☺

——— If True
p(☹) ⇒ ☹

——— If True
if (*) {p(☺)} else {p(☹)} ⇒ ☺

——— If False
if (*) {p(☹)} else {p(☺)} ⇒ ☹

(☺ + ☹ = ☺☹)
——— Seq
if (*) {p(☺)} else {p(☹)};
if (*) {p(☹)} else {p(☺)} ⇒ ☺☹

⋰
———
while ⋯ ⇒ ☹☺

(☺☹ + ☹☺ =
☺☹☹☺)

——— While True
while (*) { if (*) {p(☺)} else {p(☹)};
if (*) {p(☹)} else {p(☺)} } ⇒ ☺☹☹☺

Problem 3 Let Q be the Python program (in concrete syntax):

```
i = 0
you_won_one_million_usd = None
while found is None:
  if collatz(i):
    print(☹)
    print(☺)
  else:
    you_won_one_million_usd = i
    print(☺)
  i += 1
```

$\Rightarrow$ $\hat{Q}$

```
while (*):
  if (*):
    print(☹)
    print(☺)
  else:
    print(☺)
```

Assume that `collatz(i)` returns `True` if and only if the Collatz sequence starting at `i` converges to 1. Recall that the Collatz sequence is obtained by repeatedly applying the following two rules: a) if the number is even, divide it by two, and b) if the number is odd, triple it and add one. The Collatz conjecture – which remains an open problem – states that any Collatz sequence eventually converges to 1. The first person to solve the conjecture wins 120 million JPY (approx 0.8 million USD).

Problem 3-A Abstract Q into a CoinPython program $\hat{Q}$ (by keeping only those features that are present in the CoinPython language). Then draw the AST for $\hat{Q}$.

Problem 3-B Does $\hat{Q} \Rightarrow$ ☺☹ hold? → No! If so, draw the derivation tree. Otherwise, → Try all possibilities. draw the partial tree and indicate the exact place(s) where the proof(s) gets stuck. If $\hat{Q} \Rightarrow$ ☺☹ does/doesn't hold, what does it tell you about the printing behavior of Q?

→ $Q \not\Rightarrow$ ☺☹. If $Q \Rightarrow s$, then $\hat{Q} \Rightarrow s$. So if $\hat{Q} \not\Rightarrow s$, then $Q \not\Rightarrow s$.

Problem 3-C Does $\hat{Q} \Rightarrow$ ☺☹☺☺ hold? → yes If so, draw the derivation tree. Otherwise, draw the partial tree and indicate the exact place(s) where the proof(s) gets stuck. If $\hat{Q} \Rightarrow$ ☺☹ does/doesn't hold, what does it tell you about the printing behavior of Q?

Problem 3-D How would you describe the set of all strings that *may* be printed by Q? → every ☹ must be followed by a ☺

Problem 4 (bonus) Can you write a CoinPython program that prints all and only those strings such that every ☹ is matched by exactly one ☺ to its right?

No. CoinPython = regular expressions!