1. Terms, formulas, hybrid programs, oh my! For each of the following, determine if the expression is a (syntactically) well-formed $\mathcal{DL}$ term, a well-formed $\mathcal{DL}$ formula, a well-formed hybrid program, or none of the above (i.e., it is not well-formed). In the case that the expression is none of the above, give a short explanation why.

   (a) $z := x^5 - 1$
   (b) $? (x \cdot y \cdot z > \frac{3}{4})$
   (c) $x$
   (d) $42 + 6 \cdot 9$
   (e) $[g := 42]$
   (f) $z + 1 := x^5$
   (g) $x := y + 1 \cup; x = y'$

2. Operator precedence. Adopting a set of operator precedence rules helps reduce the number of parentheses (or braces) needed when writing down an expression. However, it is essential that you are familiar with these rules to avoid hours of debugging/misunderstanding in your labs/theory assignments!

For convenience, here is a cheatsheet for the operator precedence rules in $\mathcal{DL}$. Refer back here whenever you are not sure about how to parse a given expression.

- In the theory assignments and the textbook, parentheses ($\cdot$) are used to disambiguate terms, formulas and programs. For clarity, we will always write braces around differential equations, like this: $\{ x' = v, v' = a \}$ and $\{ t' = 1 \& t \leq T \}$.

In KeYmaera X (and in your lab assignments), parentheses ($\cdot$) are used for terms and formulas, but braces ($\cdot$) are used to group programs.

- Unary operators always bind stronger than binary operators. This includes the first-order and modal quantifiers. Examples:
  - $\forall x P \land Q \equiv (\forall x P) \land Q$ (similarly for $\exists x$),
  - $[\alpha] P \land Q \equiv ([\alpha] P) \land Q$ (similarly for $\langle \alpha \rangle$),
  - $\neg P \land Q \equiv (\neg P) \land Q$,
  - $\alpha; \beta^* \equiv \alpha; (\beta^*)$.

- The arithmetic operators have their usual precedence from mathematics.
• The binary logical connective \( \land \) binds stronger than \( \lor \), which in turn binds stronger than \( \rightarrow, \leftrightarrow \). To avoid confusion, there is no default binding precedence between \( \rightarrow \) and \( \leftrightarrow \). Explicit disambiguating parentheses are required when these appear in sequence. Examples:

- \( P \land Q \lor R \equiv (P \land Q) \lor R \)
- \( P \rightarrow Q \leftrightarrow R \) is considered illegal, and must be disambiguating either as \((P \rightarrow Q) \leftrightarrow R \) or \( P \rightarrow (Q \leftrightarrow R) \).

• Hybrid program operator \( ; \) binds tighter than \( \cup \). Example:

- \( \alpha; \beta \cup \gamma \equiv \{\alpha; \beta\} \cup \gamma \)

• All arithmetic operators +, −, · associate to the left. All logical and program operators associate to the right. In particular, implication (\( \rightarrow \)) associates to the right. Examples:

- \( a - b - c \equiv (a - b) - c \)
- \( P \rightarrow Q \rightarrow R \equiv P \rightarrow (Q \rightarrow R) \).
- \( \alpha; \beta; \gamma \equiv \alpha; (\beta; \gamma) \).

Although many of these operators satisfy an associativity law (e.g., \( a + (b + c) = (a + b) + c \)), it is important to know their default associativity because that is also how KeYmaera X parses expressions.

For this question, you will practice applying the above precedence rules. For each formula/program below, add parentheses/braces indicating the correct binding for the connectives.

(a) \([y := 5]x = 3 \lor x = 5 \rightarrow x + 1 = 6\)

(b) \(\exists x x = 5 \rightarrow x + 1 = 6 \rightarrow x = 1\)

(c) \([x := 5; y := y + x \cup \{x' = v, v' = a \& v = -1 \lor v = 1 \land v = 2\}]x > 0\)

3. Evolve nondeterministically! This question will test your understanding of nondeterministic evolution.

\[
\beta \overset{\text{def}}{=} x := x_0; v := v_0; t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0\}; ?v = 0
\]

Intuitively, hybrid program \( \beta \) first sets the initial values of \( x, v \) to \( x_0, v_0 \), and the initial value of the clock variable \( t \) to 0. It then runs the differential equations (where \( a \) is a constant acceleration) subject to the evolution domain constraint \( v \geq 0 \). Finally, it tests that \( v = 0 \) at the end of the run.

(a) Assume that \( a < 0 \land v_0 \geq 0 \). At the end of a run of hybrid program \( \beta \), what is the value of \( t \) as a function of \( x_0, v_0 \), and \( a \)?
Let us modify our program a little by removing the test:
\[
\gamma \overset{\text{def}}{=} x := x_0; v := v_0; t := 0; \{x' = v, v' = a, t' = 1 \& v \geq 0\}
\]

(b) Again assuming that \( a < 0 \land v_0 \geq 0 \), what are the possible values of \( v \) at the end of a run of \( \gamma \)? What about the possible values of \( t \)?

(c) Suppose we assume instead that \( a < 0 \land v_0 \leq 0 \) (\( v_0 \) is less than or equal to zero). What are the possible values of \( v \) and \( t \) at the end of a run of \( \beta \)?

(d) Let us consider some \( \mathcal{DL} \) formulas that use the above programs \( \beta \) and \( \gamma \). For each of the following formulas, state whether the formula is valid and give a brief explanation why. (The antecedents correspond to the various sign assumptions on \( a \) and \( v_0 \) from the previous parts of this question.)

i. \( a < 0 \land v_0 \geq 0 \rightarrow [\beta]v = 0 \)
ii. \( a < 0 \land v_0 < 0 \rightarrow [\beta]v = 0 \)
iii. \( a < 0 \land v_0 < 0 \rightarrow \langle \beta \rangle v = 0 \)
iv. \( a < 0 \land v_0 \geq 0 \rightarrow [\gamma]v = 0 \)
v. \( a < 0 \land v_0 \geq 0 \rightarrow \langle \gamma \rangle v = 0 \)

**Hint:** Carefully review the semantics of differential equations with evolution domain constraints \( \{x' = f(x) \& Q\} \).

4. **Search for the truth!** Determine whether each of the following formulas is valid/satisfiable/unsatisfiable. If the formula is satisfiable, describe the set of states in which it is satisfiable. If it is unsatisfiable, briefly explain why.

(a) \( \forall x \langle \{x' = c\} \rangle x > 0 \)
(b) \([?x \geq 0; x := -x]x < 0 \)
(c) \( \langle \{z' = -c \& z > 0\}; \{z' = c \& z < 0\}\rangle z = k \)

5. **Find a program!**

(a) Write down a program \( \alpha \) that makes the following formula satisfiable, but not valid: \([\alpha]z > 5 \)
(b) Write down a program \( \alpha \) that makes the formula \( \forall x \forall y \langle \alpha \rangle x = y \) valid. The program may mention \( x \) but not \( y \).

6. **Define an operator!** As we’ve seen in class, the primitive operators of hybrid programs can be used to define more complex operators. Define an \( n \)-ary nondeterministic switch statement with a fallback program \( \beta \). This statement should run program \( \alpha_i \) if formula \( P_i \) is true, for \( 1 \leq i \leq n \). If multiple \( P_i \)’s are true, \( 1 \leq i \leq n \), then it chooses
nondeterministically between the corresponding $\alpha_i$’s. If none of the $P_i$’s is true, then it runs $\beta$. In pseudocode, this could be written as:

```
switch {
    case $P_1$ : $\alpha_1$
    case $P_2$ : $\alpha_2$
    :
    case $P_n$ : $\alpha_n$
    default : $\beta$
}
```