Assignment 4: The Highs and Lows of Information Flow
15-316 Software Foundations of Security and Privacy

1. **Flow through abort (25 points).**

   In lecture, we defined non-interference in terms of a language that contains assignment, composition, conditional statements, and while loops.

   $$\forall \omega_1, \omega_2 . \omega_1 \approx_{\Gamma,\mathtt{L}} \omega_2 \wedge \langle \omega_1, \alpha \rangle \Downarrow \omega_1' \wedge \langle \omega_2, \alpha \rangle \Downarrow \omega_2' \rightarrow \omega_1' \approx_{\Gamma,\mathtt{L}} \omega_2' \tag{1}$$

   This definition depends on the relation $\approx_{\mathtt{L}}$, which says that two states are "low equivalent" whenever their low-variables are the same.

   $$\omega_1 \approx_{\mathtt{L}} \omega_2 \text{ if and only if } \forall x. \Gamma(x) = \mathtt{L} \rightarrow \omega_1(x) = \omega_2(x) \tag{2}$$

   This question will develop an extention to this notion of noninterference that accounts for $\mathtt{assert}(P)$ commands.

   If our threat model allows an attacker to detect whether a trace of this program aborts, then the attacker can learn information about the value of x by observing whether the final state is Λ or not.

   Part 1 (5 points). Show how the following program leaks information labeled H to an observer who can see whether the final state is Λ, as well as the initial and final values of L variables. You should assume that the policy is $\Gamma(x) = \mathtt{H}, \Gamma(y) = \mathtt{L}$.

   $$\mathtt{if}(y \neq 0)\,\{x := 2\}\,\mathtt{else}\,\{\mathtt{assert}(x = 2)\}$$

   Your solution should provide two initial L-equivalent states, and explain how the observer learns information about the H variables of the initial states from their observations.

Part 2 (5 points). Modify Equation 1 above to arrive at a formal definition of "abort-sensitive non-interference", which characterizes programs that do not leak information about `H` variables through the `L` variables in final states, or through the program's termination status (i.e., whether the final state is Λ). *Hint: the most straightforward way to complete this part may be to change the low-equivalence relation* $\approx_{\mathtt{L}}$ *to account for the error state* Λ*.*

Part 3 (5 points). Provide a big-step semantics for the $\mathtt{assert}(Q)$ command; your semantics should match the trace semantics for `assert` given in prior lectures, in the sense that:

$$\langle \omega, \mathtt{assert}(Q) \rangle \Downarrow \nu \text{ if and only if } (\omega, \nu) \in [\![\mathtt{assert}(Q)]\!]$$

Part 4 (10 points). Design a typing rule for `assert`(Q) commands, and prove its soundness. In other words, prove that if $\Gamma \vdash$ `assert`(Q), then `assert`(Q) satisfies your definition of failure-sensitive non-interference under Γ.