Exercise 4F-2. VCGen for Let [6 points]. In class we gave the following rules for the (backward) verification condition generation of assignment and let:

$$\begin{aligned} \mathrm{VC}(c_1; c_2, B) &= \mathrm{VC}(c_1, \mathrm{VC}(c_2, B)) \\ \mathrm{VC}(x := e, B) &= [e/x]\ B \\ \mathrm{VC}(\mathsf{let}\ x = e\ \mathsf{in}\ c, B) &= [e/x]\ \mathrm{VC}(c, B) \end{aligned}$$

That rule for let has a bug. Give a correct rule for let.

Solution:

$$\mathrm{VC}(\mathsf{let}\ x = e\ \mathsf{in}\ c, B) \ = [x/x']([e/x]\ \mathrm{VC}(c, [x'/x]B)),\ \text{where } x' \text{ is a fresh variable for } B.$$

Exercise 4F-3. VCGen Mistakes [6 points]. Given $\{A\}c\{B\}$ we desire that $A \implies \mathrm{VC}(c, B) \implies \mathrm{WP}(c, B)$. We say that our VC rules are *sound* if $\models \{\mathrm{VC}(c, B)\}\ c\ \{B\}$. Demonstrate the unsoundness of the buggy let rule by giving the following six things:

1. a command c and
2. a post-condition B and
3. a state σ such that
4. $\sigma \models \mathrm{VC}(c, B)$ and
5. $\langle c, \sigma \rangle \Downarrow \sigma'$ but
6. $\sigma' \not\models B$.

Solution: let command c be $x := 1; \mathsf{let}\ x = 3\ \mathsf{in}\ y := x$, the post-condition B be $x == 3$. According to the original buggy rule, we have

$$\begin{aligned} \mathrm{VC}(c, B) &:= \mathrm{VC}(x := 1, [3/x]\mathrm{VC}(y := x, (x == 3))) \\ &= [1/x][3/x][x/y](x == 3) \\ &= (3 == 3) \\ &= \mathsf{true} \end{aligned}$$

Given a state $\sigma : x \mapsto 1$, we have $\sigma \models \mathsf{true}$. Following the operational semantics for the let rule I have in HW1: $\dfrac{\langle e, \sigma \rangle \Downarrow n \quad \langle x, \sigma \rangle \Downarrow v \quad \langle c, \sigma[x := n] \rangle \Downarrow \sigma'}{\langle \text{let } x = e \text{ in } c, \sigma \rangle \Downarrow \sigma'[x := v]}$ we have $\langle c, \sigma \rangle \Downarrow \sigma'$ with $\sigma' : x \mapsto 1, y \mapsto 3$. However, we have $\sigma' \not\models (x == 3)$.

Exercise 4F-4. Axiomatic Do-While [6 points]. Write a sound and complete Hoare rule for do c while b. This statement has the standard semantics (e.g., c is executed at least once, before b is tested).

Solution: the command do c while b is equivalent to c; while b do c. Therefore, we can combine the Hoare rules for $c_1; c_2$ and while b do c and get the following rule for do-while.

$$\frac{\vdash \{A\}\ c\ \{C\} \quad \vdash C \wedge b \implies A \quad \vdash C \wedge \neg b \implies B}{\{A\}\ \mathsf{do}\ c\ \mathsf{while}\ b\ \{B\}}$$

The original proof system is already sound and complete. We translate do-while into equivalent commands using the standard semantics and we define its Hoare rule based on existing rules and standard semantics. Therefore, the provided do-while rule is also sound and complete.