Exercise 4F-2

We can fix this rule by defining

$$VC(\texttt{let } x := e \texttt{ in } c, B) = VC(y := x; x := e; c; x := y, B)$$

where y is a fresh variable not used elsewhere. As long as y is not re-assigned, this works as we are just re-stating the `let in` command.

I expect this is not the *right* answer. It's particularly suspicious that I'm requiring the ability to pull a fresh variable out of my hat. For the fun of it, here are a couple other rules I tried and explanations of why they don't work. We could consider

$$VC(\texttt{let } x := e \texttt{ in } c, B) = VC([e/x]c, B).$$

However, this fails whenever x is re-assigned in c (e.g., `let x := 7 in x := 5` will result in `7 := 5`). The following rule fails for the same reason the original failed: the value of x in B should have nothing to do with e.

$$VC(\texttt{let } x := e \texttt{ in } c, B) = VC(c, [e/x]B)$$

Question assigned to the following page: 3

Exercise 4F-3

Define our command c as

```
let x=3 in skip
```

our postcondition B as

```
x=3
```

and our initial state σ as

$$\{x \to 5\}$$

(that is, $\sigma(x) = 5$ and no other variables are defined).

Applying the given rule, we find that $VC(c, B)$ is

```
[3/x] x=3
```

which evaluates to

```
3=3
```

This is a tautology, so regardless of the actual value of σ, we find that $\sigma \vDash VC(c, B)$. However, c boils down to σ itself $\langle c, \sigma \rangle \Downarrow \{x \to 5\}$. And since $5 \neq 3$, we have that $\sigma \nvDash B$. Thus, this rule is unsound.

Question assigned to the following page: 4

Exercise 4F-4

We use the following rule.

$$\frac{\vdash \{A\}\ \texttt{c}\ \{B\} \quad \vdash \{B \wedge b\}\ \texttt{c}\ \{B\}}{\vdash \{A\}\ \texttt{do c while b}\ \{B \wedge \neg b\}}$$

Informally, this checks that A implies B after running c once, and then checks that B is maintained if the loop runs multiple times. Finally, it concludes that once the loop terminates, B will continue to hold and the loop guard b must be false.