Exercise 4F-2. VCGen for Let

The bug in the verification condition for let is because we are not allocating a new location in the let expression.

Fixed rules

To address the shadowing of variables we will replace all new bindings in let expressions with fresh variables.

$$\begin{array}{lcl} VC(c_1; c_2, B) & = & VC(c_1, VC(c_2, B)) \\ VC(x := e, B) & = & [e/x]B \\ VC(\texttt{let } x = e \texttt{ in } c, B) & = & [e/\alpha]VC([\alpha/x]c, B) \text{ where } \alpha \text{ is a fresh variable} \end{array}$$

Exercise 4F-3. VCGen Mistakes

We will demonstrate the bug in VCGen for let with the following demonstration of unsoundness.

Let c be the following command:

```
(let x = 2 in skip); y := x * 2
```

Let B be a post-condition $y = 4$ and let the state $\sigma = \{x \mapsto 0, y \mapsto 0\}$.

Now we will calculate the verification condition of c with regards to B.

`VC((let x = 2 in skip); y := x * 2, y = 4)`	
$=$ `VC((let x = 2 in skip), VC(y := x * 2, y = 4))`	(Definition of VC on sequencing)
$=$ `VC(let x = 2 in skip, 4 = x * 2)`	(Definition of VC on assigment)
$=$ `VC(skip, 4 = 4)`	(Definition of VC on Let)
$=$ `4 = 4`	(Definition of VC on skip)
$=$ `true`	

Since $VC(c, B)$ is true, $\sigma \vDash VC(c, B)$ is vacuous.

Now to evaluate c using IMP's operational semantic we get

$$\langle \texttt{(let x = 2 in skip); y := x * 2}, \{x \mapsto 0, y \mapsto 0\} \rangle$$
$$\Downarrow \{x \mapsto 0, y \mapsto 0\}$$

The full derivation is elided as it follows the normal large step rules.

Given $\sigma' = \{x \mapsto 0, y \mapsto 0\}$ it follows that $\sigma' \nvDash y = 4$. Therefore, the verification condition is unsound.

Exercise 4F-4. Axiomatic Do-While

$$\frac{\vdash \{A\}\ c\ \{C\} \qquad \vdash \{C\}\ \texttt{while}\ b\ \texttt{do}\ c\ \{B\}}{\vdash \{A\}\ \texttt{do}\ c\ \texttt{while}\ b\ \{B\}}$$

The rule is sound and complete since our existing rules for commands and `while` are also sound and complete. We also have the rule of consequence which can extend/relax pre/post-conditions as appropriate to prove true statements in our system.