1 4F-1 Bookkeeping

- **0 pts** Correct

2 4F-2

Since Imp expression evaluation produces no side-effects, a let command should be equivalent to simply substituting the expression for every reference to the bound variable in the command, provided that this is done in a scope-preserving way. So, assuming we have properly defined our substitution relation on commands, as described on the Piazza:

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

3 4F-3

Let command c be $\textsf{let } x = 2 \textsf{ in skip}$, postcondition B be that $x = 2$, and state σ be $\{x : 1\}$.

Then $\text{VC}(c, B) = \text{VC}(\textsf{let } x = 2 \textsf{ in skip}, x = 2) = [2/x]\ \text{VC}(\textsf{skip}, x = 2) = [2/x]\ (x = 2) = (2 = 2) = \mathit{true}$, so by the provided semantics of assertions, we have that $\sigma \models \text{VC}(c, B)$ (for any sigma, including ours in particular).

But by our operational semantics for let statements from a previous assignment, we know that let bindings are local, so $\langle \textsf{let } x = 2 \textsf{ in skip}, \{x : 1\}\rangle \Downarrow \sigma' = \{x : 1\}$. But, again by the provided semantics of assertions, we know $\sigma' \models x = 2$ iff $\langle x, \sigma'\rangle \Downarrow = \langle 2, \sigma'\rangle \Downarrow$. Since the former is 1 and the latter is 2, we have that $\sigma' \not\models B$. So the provided rule is unsound.

4 4F-4

My approach here is to note that the command $\textsf{do } c \textsf{ while } b$ is operationally equivalent to the sequenced command c ; $\textsf{while } b \textsf{ do } c$. Thus I obtain a Hoare rule for the latter by composing the provided Hoare rules for sequencing and while loops (second version):

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

2 4F-2 VCGen for Let

- **0 pts** Correct

2 4F-2

Since Imp expression evaluation produces no side-effects, a let command should be equivalent to simply substituting the expression for every reference to the bound variable in the command, provided that this is done in a scope-preserving way. So, assuming we have properly defined our substitution relation on commands, as described on the Piazza:

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

3 4F-3

Let command c be $\textsf{let } x = 2 \textsf{ in skip}$, postcondition B be that $x = 2$, and state σ be $\{x : 1\}$.

Then $\text{VC}(c, B) = \text{VC}(\textsf{let } x = 2 \textsf{ in skip}, x = 2) = [2/x]\ \text{VC}(\textsf{skip}, x = 2) = [2/x]\ (x = 2) = (2 = 2) = \mathit{true}$, so by the provided semantics of assertions, we have that $\sigma \models \text{VC}(c, B)$ (for any sigma, including ours in particular).

But by our operational semantics for let statements from a previous assignment, we know that let bindings are local, so $\langle \textsf{let } x = 2 \textsf{ in skip}, \{x : 1\}\rangle \Downarrow \sigma' = \{x : 1\}$. But, again by the provided semantics of assertions, we know $\sigma' \models x = 2$ iff $\langle x, \sigma'\rangle \Downarrow = \langle 2, \sigma'\rangle \Downarrow$. Since the former is 1 and the latter is 2, we have that $\sigma' \not\models B$. So the provided rule is unsound.

4 4F-4

My approach here is to note that the command $\textsf{do } c \textsf{ while } b$ is operationally equivalent to the sequenced command c ; $\textsf{while } b \textsf{ do } c$. Thus I obtain a Hoare rule for the latter by composing the provided Hoare rules for sequencing and while loops (second version):

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

3 4F-3 VCGen Mistakes

- **0 pts** Correct

2 4F-2

Since Imp expression evaluation produces no side-effects, a let command should be equivalent to simply substituting the expression for every reference to the bound variable in the command, provided that this is done in a scope-preserving way. So, assuming we have properly defined our substitution relation on commands, as described on the Piazza:

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

3 4F-3

Let command c be $\textsf{let } x = 2 \textsf{ in skip}$, postcondition B be that $x = 2$, and state σ be $\{x : 1\}$.

Then $\text{VC}(c, B) = \text{VC}(\textsf{let } x = 2 \textsf{ in skip}, x = 2) = [2/x]\ \text{VC}(\textsf{skip}, x = 2) = [2/x]\ (x = 2) = (2 = 2) = \mathit{true}$, so by the provided semantics of assertions, we have that $\sigma \models \text{VC}(c, B)$ (for any sigma, including ours in particular).

But by our operational semantics for let statements from a previous assignment, we know that let bindings are local, so $\langle \textsf{let } x = 2 \textsf{ in skip}, \{x : 1\}\rangle \Downarrow \sigma' = \{x : 1\}$. But, again by the provided semantics of assertions, we know $\sigma' \models x = 2$ iff $\langle x, \sigma'\rangle \Downarrow = \langle 2, \sigma'\rangle \Downarrow$. Since the former is 1 and the latter is 2, we have that $\sigma' \not\models B$. So the provided rule is unsound.

4 4F-4

My approach here is to note that the command $\textsf{do } c \textsf{ while } b$ is operationally equivalent to the sequenced command c ; $\textsf{while } b \textsf{ do } c$. Thus I obtain a Hoare rule for the latter by composing the provided Hoare rules for sequencing and while loops (second version):

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

4 4F-4 Axiomatic Do-While

- **0 pts** Correct