Exercise 5F-2. VCGen Do-While [8 points]. Choose exactly *one* of the two options below. (If you are not certain, pick the first. The answers end up being equivalent, but the first may be easier to grasp for some students and the second easier to grasp for others.)

- Give the (backward) verification condition formula for the command $\mathsf{do}_{Inv}\ c\ \mathsf{while}\ b$ with respect to a post-condition P. The invariant *Inv* is true before each evaluation of the predicate b. Your answer may not be defined in terms of VC(while...).

- Give the (backward) verification condition formula for the command $\mathsf{do}_{Inv1,Inv2}\ c\ \mathsf{while}\ b$ with respect to a post-condition P. The invariant *Inv1* is true before c is first executed. The invariant *Inv2* is true before each evaluation of the loop predicate b. Your answer may not be defined in terms of VC(while...).

The following is the (backward) verification formula for the command $\mathsf{do}_{Inv1,Inv2}\ c\ \mathsf{while}\ b$:

$\mathrm{VC}(\mathsf{do}_{Inv1,Inv2}\ c\ \mathsf{while}\ b,\ \mathrm{P}) =$
$\mathsf{Inv}_1 \wedge \mathrm{VC}(c, \mathsf{Inv}_1\ \wedge\ \mathsf{Inv}_2) \wedge (\forall x_1 ... x_n . \mathsf{Inv}_2 \implies (\mathsf{b} \implies \mathsf{VC}(\mathsf{c}, \mathsf{Inv}_2)\ \wedge\ \neg \mathsf{b} \implies \mathsf{P}))$

Where $x_1 ... x_n$ are iterations of the do-while loop.

Inv_1 must hold before and after only the *initial* execution of c.

Inv_2 must be true immediately after the initial execution of c and prior to the entry of the loop and prior to every iteration of the loop.

The postcondition P *must* hold once the do-while command finishes—that is, the loop guard, b, is no longer true prior to an iteration, thus exiting the loop and completing the command.

Exercise 5F-3. VCGen Mistakes [20 points]. Consider the following three alternate while Hoare rules (named lannister, stark, and targaryen):

$$\frac{\vdash \{X\}\ c\ \{b \implies X \ \wedge\ \neg b \implies Y\}}{\vdash \{b \implies X \ \wedge\ \neg b \implies Y\}\ \textsf{while}\ b\ \textsf{do}\ c\ \{Y\}}\ \textsf{lannister} \qquad \frac{\vdash \{X \ \wedge\ b\}\ c\ \{X\}}{\vdash \{X\}\ \textsf{while}\ b\ \textsf{do}\ c\ \{X\}}\ \textsf{stark}$$

$$\frac{\vdash \{X\}\ c\ \{X\}}{\vdash \{X\}\ \textsf{while}\ b\ \textsf{do}\ c\ \{X \ \wedge\ \neg b\}}\ \textsf{targaryen}$$

All three rules are sound but incomplete. Choose **two** incomplete rules. For each chosen rule provide the following:

1. the name of the rule and
2. A and
3. B and
4. σ and
5. σ' and
6. c such that
7. $\langle c, \sigma \rangle \Downarrow \sigma'$ and
8. $\sigma \models A$ and
9. $\sigma' \models B$ but
10. it is not possible to prove $\vdash \{A\}\ c\ \{B\}$.

Flavor text: Incompleteness in an axiomatic semantics or type system is typically not as dire as unsoundness. An incomplete system cannot prove all possible properties or handle all possible programs. Many research results that claim to work for the C language, for example, are actually incomplete because they do not address `setjmp`/`longjmp` or bitfields. (Many of them are also unsound because they do not correctly model unsafe casts, pointer arithmetic, or integer overflow.)

1st Provision:

1. Consider the rule targaryen
2. let $A = \{x \leq 0\}$
3. let $B = \{x \geq 1\}$
4. let $\sigma = [x = 0]$

5. let $\sigma' = [x = 1]$
6. let $c = \mathsf{while}\ x < 1\ \mathsf{do}\ x := x + 1$
7. $$\frac{\langle x < 1, [x = 0]\rangle \Downarrow true \qquad D_1 = \langle x := x + 1; c, [x = 0]\rangle \Downarrow [x = 1]}{\langle c, [x = 0]\rangle \Downarrow [x = 1]}\ \mathsf{while_{true}}$$

$$\frac{\langle x := x + 1, [x = 0]\rangle \Downarrow [x = 1] \qquad D_2 = \langle c, [x = 1]\rangle \Downarrow [x = 1]}{D_1}\ \mathsf{while_{seq}}$$

$$\frac{\langle x < 1, [x = 1]\rangle \Downarrow false}{D_2}\ \mathsf{while_{false}}$$

8. $[x = 0] \vDash \{x \leq 0\}$
9. $[x = 1] \vDash \{x \geq 1\}$
10. We cannot prove this with targaryen as for $A \neq B$. We'd need to resort to something like $X = x \leq 1$, i.e. we've defined A to be a stronger, valid precondition than the targaryen rule can handle.

2nd Provision:

1. Consider the rule stark
2. let $A = \{x = 2\}$
3. let $B = \{x = 2\}$
4. let $\sigma = [x = 2]$
5. let $\sigma' = [x = 2]$
6. let $c = \mathsf{while}\ x < 2\ \mathsf{do}\ \mathtt{skip}$
7. $$\frac{\langle x < 2, [x = 2]\rangle \Downarrow false}{\langle c, [x = 2]\rangle \Downarrow [x = 2]}\ \mathsf{while_{false}}$$
8. $[x = 2] \vDash \{x = 2\}$
9. $[x = 2] \vDash \{x = 2\}$
10. The stark rule fails to prove this, as it requires the loop-guard to be true on entrance, according to its premise's precondition. In this provided example, the loop guard $x < 2$ is indeed false, thus the precondition $\{X \wedge b\}$ fails to hold.