1 2F-1 Bookkeeping

- **0 pts** Correct

2F-2

> contain flowers that smell the same. We will prove that the same thing holds for all subsets of size at most $n+1$. Pick an arbitrary set X such that $|X| = n+1$. Pick two distinct flowers $f, f' \in X$ and let's show that $\mathsf{smells}(f) = \mathsf{smells}(f')$. Let $Y = X - \{f\}$ and $Y' = X - \{f'\}$. Obviously Y and Y' are sets of size at most n so the induction hypothesis holds for both of them. Pick any arbitrary $x \in Y \cap Y'$. Obviously, $x \neq f$ and $x \neq f'$. We have that $\mathsf{smells}(f') = \mathsf{smells}(x)$ (from the induction hypothesis on Y) and $\mathsf{smells}(f) = \mathsf{smells}(x)$ (from the induction hypothesis on Y'). Hence $\mathsf{smells}(f) = \mathsf{smells}(f')$, which proves the inductive step, and the theorem.

I would say these three highlighted sentences are problems in the proof. Out of the three, the second is an incorrect claim and the third is nonsense given that the second is incorrect. The first and third also make claims with no reasoning or logic presented as support.

The first claim is true, but providing some logic showing that Y and Y' rather than saying "Obviously" would be more correct.

The second claim also provides no supporting logic, and in this case that hides a gap in the logic. For $n = 1$, there is no such x. Because Y and Y' would both have size 1 and no common elements (one would be $\{f\}$ and the other would be $\{f'\}$ which are chosen to be distinct). Therefore, $Y \cap Y'$ would be the empty set so no such x could be picked, no matter how arbitrary.

Finally, the third claim, which only makes sense in the context of $n > 1$, needs some supporting logic.

2F-3

Base Case:

$$\frac{\langle b, \sigma\rangle \Downarrow \mathsf{False}}{\langle \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma\rangle \Downarrow \sigma}$$

By the first evaluation rule of while, $\sigma' = \sigma$. It is known that $\sigma(x)$ is even, therefore $\sigma'(x)$ is also even.

Induction Step: With the base case being $\langle b, \sigma\rangle \Downarrow \mathsf{False}$, we shall induct on cases where $\langle b, \sigma\rangle \Downarrow \mathsf{True}$. As a BExp, b must evaluate to either True or False, so every while loop as presented will match one of these two cases.

$\langle \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma\rangle$	while redex
$\langle \mathsf{if}\ b\ \mathsf{then}\ x := x + 2\ ;\ \mathsf{while}\ b\ \mathsf{do}\ x := x + 2\ \mathsf{else\ skip}, \sigma\rangle$	assumed $b \Downarrow \mathsf{True}$ for the inductive case
$\langle \mathsf{if\ True\ then}\ x := x + 2\ ;\ \mathsf{while}\ b\ \mathsf{do}\ x := x + 2\ \mathsf{else\ skip}, \sigma\rangle$	if true redex
$\langle x := x + 2\ ;\ \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma\rangle$	assigment redex
$\langle \mathsf{skip}\ ;\ \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma[x := \sigma(x) + 2]\rangle$	skip redex
$\langle \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma[x := \sigma(x) + 2]\rangle$	

This results in a while statement of the same form as the original, except with $\sigma' = \sigma[x := \sigma(x) + 2]$. Based on the assumption that $\sigma(x)$ is even, $\sigma(x) + 2$ must also be even, so $\sigma'(x) = \sigma(x) + 2$ is even as well. For the original to evaluate to some final state σ'', it must reach the base case (equivalently, the while loop must terminate). The invariant $\sigma(x)$ *is even* holds for every state σ in the application of the inductive step, and for the base case, so therefore it must hold for the final state σ'' as well.

Peer Review ID: 65770092 — enter this when you fill out your peer evaluation via gradescope

2 2F-2 Mathematical Induction

- **0 pts** Correct

2F-2

> contain flowers that smell the same. We will prove that the same thing holds for all subsets of size at most $n+1$. Pick an arbitrary set X such that $|X| = n+1$. Pick two distinct flowers $f, f' \in X$ and let's show that $\mathsf{smells}(f) = \mathsf{smells}(f')$. Let $Y = X - \{f\}$ and $Y' = X - \{f'\}$. Obviously Y and Y' are sets of size at most n so the induction hypothesis holds for both of them. Pick any arbitrary $x \in Y \cap Y'$. Obviously, $x \neq f$ and $x \neq f'$. We have that $\mathsf{smells}(f') = \mathsf{smells}(x)$ (from the induction hypothesis on Y) and $\mathsf{smells}(f) = \mathsf{smells}(x)$ (from the induction hypothesis on Y'). Hence $\mathsf{smells}(f) = \mathsf{smells}(f')$, which proves the inductive step, and the theorem.

I would say these three highlighted sentences are problems in the proof. Out of the three, the second is an incorrect claim and the third is nonsense given that the second is incorrect. The first and third also make claims with no reasoning or logic presented as support.

The first claim is true, but providing some logic showing that Y and Y' rather than saying "Obviously" would be more correct.

The second claim also provides no supporting logic, and in this case that hides a gap in the logic. For $n = 1$, there is no such x. Because Y and Y' would both have size 1 and no common elements (one would be $\{f\}$ and the other would be $\{f'\}$ which are chosen to be distinct). Therefore, $Y \cap Y'$ would be the empty set so no such x could be picked, no matter how arbitrary.

Finally, the third claim, which only makes sense in the context of $n > 1$, needs some supporting logic.

2F-3

Base Case:

$$\frac{\langle b, \sigma\rangle \Downarrow \mathsf{False}}{\langle \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma\rangle \Downarrow \sigma}$$

By the first evaluation rule of while, $\sigma' = \sigma$. It is known that $\sigma(x)$ is even, therefore $\sigma'(x)$ is also even.

Induction Step: With the base case being $\langle b, \sigma\rangle \Downarrow \mathsf{False}$, we shall induct on cases where $\langle b, \sigma\rangle \Downarrow \mathsf{True}$. As a BExp, b must evaluate to either True or False, so every while loop as presented will match one of these two cases.

$\langle \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma\rangle$	while redex
$\langle \mathsf{if}\ b\ \mathsf{then}\ x := x + 2\ ;\ \mathsf{while}\ b\ \mathsf{do}\ x := x + 2\ \mathsf{else\ skip}, \sigma\rangle$	assumed $b \Downarrow \mathsf{True}$ for the inductive case
$\langle \mathsf{if\ True\ then}\ x := x + 2\ ;\ \mathsf{while}\ b\ \mathsf{do}\ x := x + 2\ \mathsf{else\ skip}, \sigma\rangle$	if true redex
$\langle x := x + 2\ ;\ \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma\rangle$	assigment redex
$\langle \mathsf{skip}\ ;\ \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma[x := \sigma(x) + 2]\rangle$	skip redex
$\langle \mathsf{while}\ b\ \mathsf{do}\ x := x + 2, \sigma[x := \sigma(x) + 2]\rangle$	

This results in a while statement of the same form as the original, except with $\sigma' = \sigma[x := \sigma(x) + 2]$. Based on the assumption that $\sigma(x)$ is even, $\sigma(x) + 2$ must also be even, so $\sigma'(x) = \sigma(x) + 2$ is even as well. For the original to evaluate to some final state σ'', it must reach the base case (equivalently, the while loop must terminate). The invariant $\sigma(x)$ *is even* holds for every state σ in the application of the inductive step, and for the base case, so therefore it must hold for the final state σ'' as well.

Peer Review ID: 65770092 — enter this when you fill out your peer evaluation via gradescope

3 2F-3 While Induction

- **0 pts** Correct

2F-4

Inference rules for $\mathsf{throw}\ e$

$$\frac{e \Downarrow n}{\langle \mathsf{throw}\ e, \sigma\rangle \Downarrow \sigma\ \mathsf{exc}\ n}$$

Inference rules for $\mathsf{try}\ c_1\ \mathsf{catch}\ x\ c_2$

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma'}{\langle \mathsf{try}\ c_1\ \mathsf{catch}\ x\ c_2, \sigma\rangle \Downarrow \sigma'} \qquad \frac{\langle c_1, \sigma\rangle \Downarrow \sigma'\ \mathsf{exc}\ n \qquad \langle c_2, \sigma'[x := n]\rangle \Downarrow t}{\langle \mathsf{try}\ c_1\ \mathsf{catch}\ x\ c_2, \sigma\rangle \Downarrow t}$$

Inference rules for $\mathsf{after}\ c_1\ \mathsf{finally}\ c_2$

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma' \qquad \langle c_2, \sigma'\rangle \Downarrow \sigma''}{\langle \mathsf{after}\ c_1\ \mathsf{finally}\ c_2, \sigma\rangle \Downarrow \sigma''} \qquad \frac{\langle c_1, \sigma\rangle \Downarrow \sigma'\ \mathsf{exc}\ n \qquad \langle c_2, \sigma'\rangle \Downarrow \sigma''}{\langle \mathsf{after}\ c_1\ \mathsf{finally}\ c_2, \sigma\rangle \Downarrow \sigma''\ \mathsf{exc}\ n}$$

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma'\ \mathsf{exc}\ n_1 \qquad \langle c_2, \sigma'\rangle \Downarrow \sigma''\ \mathsf{exc}\ n_2}{\langle \mathsf{after}\ c_1\ \mathsf{finally}\ c_2, \sigma\rangle \Downarrow \sigma''\ \mathsf{exc}\ n_2}$$

2F-5

In my opinion, large scale semantics are a more natural way of describing IMP with exceptions. I think it's hard to put into words for me exactly why, and a large part of that preference is due to being more familiar with that form after taking 490. I think the best reason I can express that is specific to IMP with exceptions is the idea of locality. With small-step semantics, context is expressed using contexts and holes, and although this works well enough for expressions, I think it becomes slightly harder to reason about where exceptions are occurring than in large-step semantics. With large step semantics, it is clearer and more intuitive to me that the commands are discarded when an exception occurs. More specifically, I see it as easier to express which commands are being discarded, as well as when and where in semantic terms.

Peer Review ID: 65770092 — enter this when you fill out your peer evaluation via gradescope

4 2F-4 Language Features, Large Step

- **0 pts** Correct

2F-4

Inference rules for $\mathsf{throw}\ e$

$$\frac{e \Downarrow n}{\langle \mathsf{throw}\ e, \sigma\rangle \Downarrow \sigma\ \mathsf{exc}\ n}$$

Inference rules for $\mathsf{try}\ c_1\ \mathsf{catch}\ x\ c_2$

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma'}{\langle \mathsf{try}\ c_1\ \mathsf{catch}\ x\ c_2, \sigma\rangle \Downarrow \sigma'} \qquad \frac{\langle c_1, \sigma\rangle \Downarrow \sigma'\ \mathsf{exc}\ n \qquad \langle c_2, \sigma'[x := n]\rangle \Downarrow t}{\langle \mathsf{try}\ c_1\ \mathsf{catch}\ x\ c_2, \sigma\rangle \Downarrow t}$$

Inference rules for $\mathsf{after}\ c_1\ \mathsf{finally}\ c_2$

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma' \qquad \langle c_2, \sigma'\rangle \Downarrow \sigma''}{\langle \mathsf{after}\ c_1\ \mathsf{finally}\ c_2, \sigma\rangle \Downarrow \sigma''} \qquad \frac{\langle c_1, \sigma\rangle \Downarrow \sigma'\ \mathsf{exc}\ n \qquad \langle c_2, \sigma'\rangle \Downarrow \sigma''}{\langle \mathsf{after}\ c_1\ \mathsf{finally}\ c_2, \sigma\rangle \Downarrow \sigma''\ \mathsf{exc}\ n}$$

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma'\ \mathsf{exc}\ n_1 \qquad \langle c_2, \sigma'\rangle \Downarrow \sigma''\ \mathsf{exc}\ n_2}{\langle \mathsf{after}\ c_1\ \mathsf{finally}\ c_2, \sigma\rangle \Downarrow \sigma''\ \mathsf{exc}\ n_2}$$

2F-5

In my opinion, large scale semantics are a more natural way of describing IMP with exceptions. I think it's hard to put into words for me exactly why, and a large part of that preference is due to being more familiar with that form after taking 490. I think the best reason I can express that is specific to IMP with exceptions is the idea of locality. With small-step semantics, context is expressed using contexts and holes, and although this works well enough for expressions, I think it becomes slightly harder to reason about where exceptions are occurring than in large-step semantics. With large step semantics, it is clearer and more intuitive to me that the commands are discarded when an exception occurs. More specifically, I see it as easier to express which commands are being discarded, as well as when and where in semantic terms.

Peer Review ID: 65770092 — enter this when you fill out your peer evaluation via gradescope

5 2F-5 Language Features, Analysis

- **0 pts** Correct