Exercise 2F-2. Mathematical Induction [5 points]. Find the flaw in the following inductive proof that "All flowers smell the same". Please indicate exactly which sentences are wrong in the proof via highlighting or underlining.

Proof: Let F be the set of all flowers and let $\mathsf{smells}(f)$ be the smell of the flower $f \in F$. (The range of smells is not so important, but we'll assume that it admits equality.) We'll also assume that F is countable. Let the property $P(n)$ mean that all subsets of F of size at most n contain flowers that smell the same.

$$P(n) \stackrel{\text{def}}{=} \forall X \in \mathcal{P}(F).\ |X| \leq n \implies (\forall f, f' \in X.\ \mathsf{smells}(f) = \mathsf{smells}(f'))$$

(the notation $|X|$ denotes the number of elements of X)

One way to formulate the statement to prove is $\forall n \geq 1.P(n)$. We'll prove this by induction on n, as follows:

Base Case: $n = 1$. Obviously all singleton sets of flowers contain flowers that smell the same (by the definition of $P(n)$).

Induction Step: Let n be arbitrary and assume that all subsets of F of size at most n 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.

(One indication that the proof might be wrong is the large number of occurrences of the word "obviously" `:-)`)

This proof is incorrect because it assumes that $Y \cap Y'$ is not empty. For example, if we assume the set $X = \{f, f'\}$, then $Y \cap Y'$ is empty and there is no x we can choose from $Y \cap Y'$. Therefore, the rest of the proof is invalid because we do not have an x that satisfies $x \neq f$ and $x \neq f'$.

Questions assigned to the following page: 3 and 4

Exercise 2F-3. While Induction [10 points].
Using structural induction, we can prove that for any BExp b and any initial state σ such that $\sigma(x)$ is even, if

$$\langle \texttt{while } b \texttt{ do } x := x + 2, \sigma \rangle \Downarrow \sigma'$$

then $\sigma'(x)$ is even.

Base Case: b is false. In this case, σ remains unchanged. Therefore, $\sigma(x) = \sigma'(x)$. Since $\sigma(x)$ is originally even, $\sigma'(x)$ is even as well.

Inductive Step: b is true. If b is true, then the while loop runs at least once. Let us define a σ'' representing the state after the loop has been run once. In this case, $\sigma''(x) = \sigma(x) + 2$. As our inductive hypothesis, let us assume that if $\sigma''(x)$ is even and

$$\langle \texttt{while } b \texttt{ do } x := x + 2, \sigma'' \rangle \Downarrow \sigma'$$

then $\sigma'(x)$ is even. Since $\sigma''(x)$ is even and $\sigma(x) = \sigma''(x) - 2$, we know that $\sigma(x)$ must be even as well. Based on our inductive hypothesis, we can therefore conclude that

$$\langle \texttt{while } b \texttt{ do } x := x + 2, \sigma \rangle \Downarrow \sigma'$$

results in $\sigma'(x)$ being even. We have proved the inductive step, and the statement.

Exercise 2F-4. Language Features, Large-Step [12 points].
The throw e command evaluates the argument e and raises an exception with the evaluated n:

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

The try command requires two rules. If c_1 terminates normally, the try command also terminates normally:

$$\frac{\langle c_1, \sigma \rangle \Downarrow \sigma'}{\langle \text{try } c_1 \text{ catch } x \ c_2, \sigma \rangle \Downarrow \sigma'}$$

On the other hand, if c_1 raises an exception with value e, x is assigned the value e and c_2 is executed:

$$\frac{\langle c_1, \sigma \rangle \Downarrow \sigma' \texttt{ exc } e \quad \langle c_2, \sigma[x := e] \rangle \Downarrow t}{\langle \text{try } c_1 \text{ catch } x \ c_2, \sigma \rangle \Downarrow t}$$

Questions assigned to the following page: 4 and 5

The finally command requires three rules. If c_1 terminates normally, the command terminates by executing c_2:

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma' \quad \langle c_2, \sigma'\rangle \Downarrow t}{\langle \text{after } c_1 \text{ finally } c_2, \sigma\rangle \Downarrow t}$$

If c_1 raises an exception e_1 and c_2 terminates normally, the command terminates by throwing an exception with e_1:

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma' \ \texttt{exc}\ e_1 \quad \langle c_2, \sigma'\rangle \Downarrow \sigma''}{\langle \text{after } c_1 \text{ finally } c_2, \sigma\rangle \Downarrow \sigma'' \ \texttt{exc}\ e_1}$$

Finally, if c_2 also throws an exception e_2, the command terminates by throwing an exception with e_2:

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma' \ \texttt{exc}\ e_1 \quad \langle c_2, \sigma'\rangle \Downarrow \sigma'' \ \texttt{exc}\ e_2}{\langle \text{after } c_1 \text{ finally } c_2, \sigma\rangle \Downarrow \sigma'' \ \texttt{exc}\ e_2}$$

Exercise 2F-4. Language Features, Analysis [6 points].
When describing "IMP with exceptions", it would be more suitable to use small-step contextual semantics. With large-step contextual semantics, the evaluation of expressions omits intermediate steps and simply determines if an exception is thrown. As a result, it is unsuitable for identifying specific steps in which exceptions may occur, particularly in instances involving loops or conditions. Small-step semantics are more elegant as they model the execution steps sequentially and can explicitly identify locations in which exceptions stem from. From a real-life standpoint, the capabilities of small-step semantics make it much more appropriate method for capturing how and when exceptions occur.