Question 2

Find the flaw in the following inductive proof.

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.

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

(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 <u>the same</u>

The base case statement is ambiguous. Each singleton set contain one flower thus there is only one element, which is the "same" to itself.

(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 <u>any arbitrary $x \in Y \cap Y'$</u>.

The intersection may be empty, i.e., $Y \cap Y' = \emptyset$, when for example, $n = 2$, i.e., $X = \{f, f'\}$.

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. $\square$

Summary: The inductive step from $n = 1$ to $n = 2$ is not valid.

Question 3

Proof. Let σ be an arbitrary state such that $\sigma(x)$ is even. We rename the command `add2` := "$x := x + 2$" to save space.

We will do structural induction on the derivation/construction of $\langle \texttt{while}\ b\ \texttt{do}\ \texttt{add2},\ \sigma\rangle \Downarrow \sigma'$, (and implicitly on the Boolean expression b), based on inversion and determinism.

For any Boolean expression b, by the derivation rules, we must have either $\langle b,\ \sigma\rangle \Downarrow \texttt{false}$ or $\langle b,\ \sigma\rangle \Downarrow \texttt{true}$.

Base case: When $\langle b,\ \sigma\rangle \Downarrow \texttt{false}$, we have

$$\frac{D' :: \langle b,\ \sigma\rangle \Downarrow \texttt{false}}{D :: \langle \texttt{while}\ b\ \texttt{do}\ \texttt{add2},\ \sigma\rangle \Downarrow \sigma}\ \texttt{base}$$

Since $\sigma(x)$ is even, the returned state is just $\sigma'(x) = \sigma(x)$ which is also even.

Induction: When $\langle b,\ \sigma\rangle \Downarrow \texttt{true}$, and $\langle \texttt{add2}, \sigma\rangle \Downarrow \sigma''$. The inductive hypothesis is: If $\sigma''(x)$ is even, and $\langle \texttt{while}\ b\ \texttt{do}\ \texttt{add2},\ \sigma''\rangle \Downarrow \sigma'$, then $\sigma'(x)$ is even. We then have the following inference by inversion of while-true and seq:

$$\dfrac{D_0 :: \langle b,\ \sigma\rangle \Downarrow \texttt{true} \qquad \dfrac{\dfrac{}{D_1 :: \langle \texttt{add2}, \sigma\rangle \Downarrow \sigma''}\ \texttt{res} \qquad \dfrac{}{D_2 :: \langle \texttt{while}\ b\ \texttt{do}\ \texttt{add2},\ \sigma''\rangle \Downarrow \sigma'}\ \texttt{IH}}{D' :: \langle \texttt{add2};\ \texttt{while}\ b\ \texttt{do}\ \texttt{add2},\ \sigma\rangle \Downarrow \sigma'}\ \texttt{seq}}{D :: \langle \texttt{while}\ b\ \texttt{do}\ \texttt{add2},\ \sigma\rangle \Downarrow \sigma'}\ \texttt{while-true}$$

The `res` rule implies that $\sigma''(x) = \sigma(x) + 2$ by definition. Since $\sigma(x) \in \mathbb{Z}$ is even, then $\sigma''(x) \in \mathbb{Z}$ is also even by integer arithmetic. Then by the inductive hypothesis, we get $\sigma'(x)$ is even.

Therefore by induction, we have proved the property for all possible derivations of the while loop $\langle \texttt{while}\ b\ \texttt{do}\ \texttt{add2},\ \sigma\rangle \Downarrow \sigma'$, for all Boolean expressions b. □

Question 4

The rules for the new commands are as follows, with one for `throw`, two for `try-catch`, and three for `after-finally`:

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

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

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

$$\frac{\langle c_1,\ \sigma\rangle \Downarrow \sigma' \qquad \langle c_2,\ \sigma'\rangle \Downarrow t}{\langle \texttt{after}\ c_1\ \texttt{finally}\ c_2,\ \sigma\rangle \Downarrow t}\ \texttt{after-finally-normal}$$

$$\frac{\langle c_1,\ \sigma\rangle \Downarrow \sigma'\ \texttt{exc}\ e_1 \qquad \langle c_2,\ \sigma'\rangle \Downarrow \sigma''}{\langle \texttt{after}\ c_1\ \texttt{finally}\ c_2,\ \sigma\rangle \Downarrow \sigma''\ \texttt{exc}\ e_1}\ \texttt{after-finally-exc-normal}$$

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

Question 5

I feel the large-step operational semantics for the exception would be generally simpler and more elegant than the small-step contextual semantics, as they are essentially "large-step". Each of the reference rules above could correspond to multiple steps in the contextual semantics reduction. In particular, the contextual semantics may also have to keep track of the extra exceptional argument in each step of the execution, which could make it less succinct.