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" `:-)`)

Solution: See above. f' and f are not in Y or Y' respectively, so the inductive hypothesis does not include them.

Exercise 2F-3. While Induction [10 points]. Prove by induction the following statement about the operational semantics:

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. Make sure you state what you induct on, what the base case is and what the inductive cases are. Show representative cases among the latter. Do not do a proof by mathematical induction!

Solution: Pick an arbitrary σ, σ' and $D :: \langle \texttt{while}\ b\ \texttt{do}\ x := x + 2, \sigma\rangle \Downarrow \sigma'$, such that $\sigma(x)$ is even. We will induct on the structure of the Derivation D.

Base Case: The the last rule used in D was the one for not repeating a loop, where $\langle b, \sigma\rangle \Downarrow false$.

$$D :: \frac{D_1 :: \langle b, \sigma\rangle \Downarrow false}{\langle \texttt{while}\ b\ \texttt{do}\ x := x + 2, \sigma\rangle \Downarrow \sigma}$$

. then $\sigma' = \sigma$, so $\sigma'(x) = \sigma(x)$, then since $\sigma(x)$ is even so is $\sigma'(x)$.

Inductive Step: The last rule used in D was the one for repeating a loop, where $\langle b, \sigma\rangle \Downarrow true$

$$D :: \frac{D_1 :: \langle b, \sigma\rangle \Downarrow true \quad D_2 :: \langle x := x + 2; \texttt{while}\ b\ \texttt{do}\ x := x + 2, \sigma\rangle \Downarrow \sigma'}{\langle \texttt{while}\ b\ \texttt{do}\ x := x + 2, \sigma\rangle \Downarrow \sigma'}$$

Peer Review ID: 305526074 — enter this when you fill out your peer evaluation via gradescope

We can further break down D_2

$$D :: \dfrac{D_1 :: \langle b, \sigma\rangle \Downarrow \mathit{true} \quad D_2 :: \dfrac{D_{21} :: \langle x := x+2, \sigma\rangle \Downarrow \sigma'_1 \quad D_{22} :: \langle \texttt{while}\ b\ \texttt{do}\ x := x+2, \sigma'_1\rangle \Downarrow \sigma'}{\langle x := x+2; \texttt{while}\ b\ \texttt{do}\ x := x+2, \sigma\rangle \Downarrow \sigma'}}{\langle \texttt{while}\ b\ \texttt{do}\ x := x+2, \sigma\rangle \Downarrow \sigma'}$$

Since we've assumed that $\sigma(x)$ is even, we know that $\sigma'_1(x)$ is even. Then, according to the induction hypothesis on D_{22}, $\sigma'(x)$ is even.

Exercise 2F-4. Language Features, Large-Step [12 points]. We extend IMP with a notion of integer-valued *exceptions* (or *run-time errors*), as in Java, ML or C#.

Give the large-step operational semantics inference rules (using our new judgment) for the three new commands presented here. You should present six (6) new rules total.

Solution:

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

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

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

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

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

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

Exercise 2F-4. Language Features, Analysis [6 points]. Argue for or against the claim that it would be more natural to describe "IMP with exceptions" using small-step contextual semantics. You may use "simpler" or "more elegant" instead of "more natural" if you prefer. Do not exceed two paragraphs (one should be sufficient). Both your ideas and also the clarity with which they are expressed (i.e., your English prose) matter.

Solution: It would not be more natural to describe "IMP with exceptions" using small-step contextual semantics. The elegance of small-step contextual semantics in part comes from how at each step the reduction, reducing the redex shrinks the overall context (with the exception of the reduction rule for while), and simplifies the remaining program. If we tried to introduce reduction rules for the new commands as defined above, the reduction rules would not actually "reduce" the program but grow

them instead. To ”reduce” both try/catch and after/finally, we would need to introduce an if statement on the result of c_1, making the program bulkier. Morevoer, we would also have to define a way of checking ”if c_1 terminated with an exception”, fruther3 muddling our small-step semantics.

Submission. Turn in the formal component of the assignment as a single PDF document via the `gradescope` website. Your name and Michigan email address must appear on the first page of your PDF submission but may not appear anywhere else. Turn in the coding component of the assignment via the `autograder.io` website.