In a language with array element assignment, the program
\[ x := i. \quad i := A \cdot i. \quad A \cdot i := x \]
was written with the intention to swap the values of \( i \) and \( A \cdot i \). Assume that all
variables and array elements are of type \( \text{nat} \), and that \( i \) has a value that is an index of \( A \).

(a) In variables \( x \), \( i \), and \( A \), specify that \( i \) and \( A \cdot i \) should be swapped, the rest of \( A \) should be unchanged, but \( x \) might change.
§
\[ i' = A \cdot i \quad \land \quad A' = i \mapsto i \mid A \]

(b) Find the exact precondition for which the program refines the specification of part (a).
§
\[ \forall x', i'. A' \cdot i' = A \land A' = i \mapsto i \mid A \iff (x := i. \quad i := A \cdot i. \quad A := i \mapsto x \mid A) \quad \text{expand final asm} \]
\[ \quad \quad \text{substitution law twice} \]
\[ = \forall x', i'. A' \cdot i' = A \land A' = i \mapsto i \mid A \iff x' := i \land i' := A \land A' = A \mapsto i \mid A \]
\[ = Ai= Ai \land Ai \mapsto i \mid A = A \mapsto i \mid A \quad \text{reflexivity and identity} \]
\[ = Ai \mapsto i \mid A = A \mapsto i \mid A \quad \text{case idempotent} \]
\[ = \text{if } Ai = i \text{ then } Ai \mapsto i \mid A = i \mapsto i \mid A \text{ else } Ai \mapsto i \mid A = i \mapsto i \mid A \quad \text{context, reflexive} \]
\[ = \text{if } Ai = i \text{ then } \top \text{ else } Ai \mapsto i \mid A = i \mapsto i \mid A \quad \text{One Case Law} \]
\[ = Ai = i \quad \land \quad Ai \mapsto i \mid A = i \mapsto i \mid A \quad \text{list equality} \]
\[ = Ai = i \quad \land \quad \forall j. \quad (Ai \mapsto i \mid A)j = (i \mapsto i \mid A)j \quad \text{split domain of } j \]
\[ = Ai = i \quad \land \quad \forall j. \quad j \neq i \Rightarrow (Ai \mapsto i \mid A)j = (i \mapsto i \mid A)j \quad \text{use the context } Ai \neq i \text{ in the right disjunct. Use it to simplify } (Ai \mapsto i \mid A)i. \text{ Also simplify } (i \mapsto i \mid A)i. \]
\[ = Ai = i \quad \land \quad \forall j. \quad j \neq i \Rightarrow (Ai \mapsto i \mid A)j = (i \mapsto i \mid A)j \quad \text{absorption} \]
\[ = Ai = i \]
So \( i \) and \( A \cdot i \) will be swapped if and only if they have the same value to start with, making the swap useless.

(c) Find the exact postcondition for which the program refines the specification of part (a).
§
\[ \forall x, i, A. \quad i = A \cdot i \quad \land \quad A' = i \mapsto i \mid A \iff x' := i \land i' := A \cdot i \land A' = A \mapsto i \mid A \]
context to drop first \( i := A \cdot i \); \( x \) doesn't appear; one-pt for \( i \); context to replace last \( Ai \)
\[ = \forall A. \quad A' = x' \mapsto x' \mid A \iff i' := A \cdot x' \land A' = i \mapsto x' \mid A \quad \text{case idempotent} \]
\[ = \text{if } x' \neq i' \text{ then } \forall A. \quad A' = x' \mapsto x' \mid A \iff i' := A \cdot x' \land A' = i \mapsto x' \mid A \quad \text{context: replace } i' \]
\[ = \exists A. \quad A' = x' \mapsto x' \mid A \iff i' := A \cdot x' \land A' = i \mapsto x' \mid A \quad \text{specialization} \]
\[ = x' := i' \quad \Rightarrow (\forall A. \quad i' := x' \mid A = x' := x' \mid A \iff i' := A \cdot x' \land A' = i \mapsto x' \mid A) \]
\[ = x' := i' \quad \Rightarrow (\forall A. \quad x' := Ai' \land A' = x' \mapsto x' \iff i' := A \cdot x' \land A' = i \mapsto x' \mid A) \quad \text{context} \]
\[ = x' := i' \quad \Rightarrow (\forall A. \quad i' := x' \land A' = i \mapsto x' \mid A) \quad \text{note that } x' := i' \land A' = i \mapsto x' \mid A \implies A \cdot x' = A \cdot x' \]
\[ = x' := i' \quad \Rightarrow (\forall A. \quad i' := x' \land A' = i \mapsto x' \mid A) \]
\[ = x' := i' \land i' := A \cdot x' \iff \exists A. \quad A' = i \mapsto x' \mid A) \]
\[ = x' := i' \land i' := A \cdot x' \iff A \cdot x' = A \cdot x' \]
\[ = x' := i' \land A' := i \mapsto x' \quad \land A' := i \mapsto x' \mid A) \]
If, in the end, we see \( x' := i' \) or \( A' := i \cdot x' \) or \( A \cdot i' := x' \) we know they were swapped (well, we won't see \( A \cdot i' := x' \) because of the final assignment, so really it's just the first two possibilities).