Recall that we have a relational model of $\lambda 2$ with the following properties.

**Theorem 1** (Fundamental property of logical relations). $\Delta; \Gamma \vdash e : \tau$ implies $\Delta; \Gamma \models e \approx_{\log} e : \tau$

which we prove using the following compatibility lemmas:

**Lemma 2** (Compatibility lemmas).

\[
\begin{align*}
\text{log\_var} & \quad \Gamma(x) = \tau \\
\Delta; \Gamma & \models x \approx_{\log} x : \tau \\
\text{log\_true} & \quad \Delta; \Gamma \models \text{true} \approx_{\log} \text{true} : \text{bool} \\
\text{log\_false} & \quad \Delta; \Gamma \models \text{false} \approx_{\log} \text{false} : \text{bool} \\
\text{log\_if} & \quad \Delta; \Gamma \models e_0 \approx_{\log} e'_0 : \text{bool} \quad \Delta; \Gamma \models e_1 \approx_{\log} e'_1 : \tau \\
& \quad \Delta; \Gamma \models \text{if } e_0 \text{ then } e_1 \text{ else } e_2 \approx_{\log} \text{if } e'_0 \text{ then } e'_1 \text{ else } e'_2 : \tau \\
\text{log\_app} & \quad \Delta; \Gamma \models e_0 \approx_{\log} e'_0 : \tau \rightarrow \tau' \quad \Delta; \Gamma \models e_1 \approx_{\log} e'_1 : \tau' \\
& \quad \Delta; \Gamma \models (x : \sigma), \Gamma \models e \approx_{\log} e' : \tau \\
\text{log\_lam} & \quad \Delta; \Gamma \models \lambda x.e \approx_{\log} \lambda x.e' : \sigma \rightarrow \tau \\
\text{log\_tapp} & \quad \Delta; \Gamma \models e \approx_{\log} e' : \forall \alpha.\tau(\alpha) \quad \Delta \vdash \sigma \\
& \quad \Delta; \Gamma \models e[\sigma] \approx_{\log} e'[\sigma] : \tau(\sigma) \\
\text{log\_lam} & \quad \alpha, \Delta; \Gamma \models e \approx_{\log} e' : \tau(\alpha) \\
& \quad \Delta; \Gamma \models \Lambda \alpha.e \approx_{\log} \Lambda \alpha.e' : \forall \alpha.\tau(\alpha)
\end{align*}
\]

We wish to show that when two terms are logically related, they are equivalent as programs. We formalise this with the help of **contextual equivalence**.
Definition 3 (Program contexts). Program contexts are given as the following inductive definition/grammar where $e, x, \tau$ are types/grammars for expressions, variables, and types, respectively.

$$\mathcal{C} ::= \square \mid \text{if } e \text{ then } e \text{ else } e \mid \text{if } e \text{ then } e \text{ else } e \mid \text{if } e \text{ then } e \text{ else } e \mid \lambda x : \tau . \mathcal{C} \mid e \mid e \mid e \mid e \mid \Delta \alpha \mathcal{C} \mid \mathcal{C}[\tau]$$

If $\mathcal{C}$ is a program context and $e$ is an expression, we write $\mathcal{C}[e]$ for the (variable-binding) substitution of $e$ for $\square$ in $\mathcal{C}$.

We will only consider typed contexts. We write $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \tau')$ if whenever $\Delta; \Gamma \vdash e : \tau$, it is the case that $\Delta'; \Gamma' \vdash \mathcal{C}[e] : \tau'$.

Using typed program contexts we can define the notion of contextual equivalence, which formalizes the notions of program equivalence.

Definition 4 (Context equivalence). We say that two (possibly open) expressions are contextually equivalent (denoted as $\Delta; \Gamma \vdash e \approx_{ctx} e'$), if they have the same “observable behavior” under any program context; that is

$$\Delta; \Gamma \vdash e \approx_{ctx} e' : \tau \iff \forall (\mathcal{C} : \Delta; \Gamma \vdash \tau \Rightarrow \cdot \vdash \tau'). (\forall v. \mathcal{C}[e] \downarrow v \iff \mathcal{C}[e'] \downarrow v)$$

Note that we only quantify over the typed context with the return type bool. In general, we would like to quantify over all the typed contexts $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \cdot \vdash \tau')$ where $\tau'$ is a base type (e.g., integer, boolean, unit type ...), but the only base type in our system is bool. If we allow $\mathcal{C}$ to be quantified over arbitrary program contexts, then the notion of contextual equivalence will be too fine. Consider, for instance, a context $\mathcal{C} = \lambda x : \text{bool}. \square$. This context has a type $\mathcal{C} : (\cdot; (x : \text{bool}) \vdash \tau \Rightarrow \cdot \vdash \text{bool} \Rightarrow \text{bool} \Rightarrow \tau)$. If we were to allow contexts of such type in definition 3, then the notion of contextual equivalence will collapse to syntactic equality: $\forall v. (\lambda x : \text{bool}. \square)[e] \downarrow v \iff (\lambda x : \text{bool}. \square)[e'] \downarrow v$ holds iff $(\lambda x : \text{bool}. e) = (\lambda x : \text{bool}. e')$ iff $e = e'$.

We want to prove the following theorem:

Theorem 5. $\Delta; \Gamma \models e \approx_{log} e' : \tau$ implies $\Delta; \Gamma \vdash e \approx_{ctx} e' : \tau$

Proving this theorem by induction on the structure of the context won’t work. We will need an auxiliary lemma.

Lemma 6. Let $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \tau')$ and $\Delta; \Gamma \models e \approx_{log} e' : \tau$. Then $\Delta'; \Gamma' \models \mathcal{C}[e] \approx_{log} \mathcal{C}[e'] : \tau'$.

To see that Lemma 6 implies theorem 5 let $\Delta; \Gamma \models e \approx_{log} e' : \tau$ and let $\mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \cdot \vdash \text{bool})$. From lemma 6 you get $\cdot \vdash \mathcal{C}[e] \approx_{log} \mathcal{C}[e'] : \text{bool}$. By picking empty substitutions, one can deduce that both $\mathcal{C}[e]$ and $\mathcal{C}[e']$ terminate to the same value.
Proof of Lemma. We prove this by structural induction on \( \mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \tau') \).

Case (1): \( \mathcal{C} = \Box \). This is trivial.

Case (2): \( \mathcal{C} = \text{if } C' \text{ then } p \text{ else } p' \). Since \( \mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \tau') \), it must be the case that \( C' : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \text{bool}) \) and \( \Delta'; \Gamma' \vdash p, p' : \tau' \). By the induction hypothesis we have \( \Delta'; \Gamma' \vdash C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \text{bool} \). Then we use the fundamental property of logical relations for \( p \) and \( p' \), and the compatibility lemma \( \text{log_if} \):

\[
\Delta'; \Gamma' \models C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \text{bool} \quad \text{fund} \quad \Delta'; \Gamma' \models p \approx_{\text{log}} p' : \tau' \quad \text{fund} \quad \Delta'; \Gamma' \models p' \approx_{\text{log}} p' : \tau'
\]

Case (3): \( \mathcal{C} = \text{if } c \text{ then } p \text{ else } C' \). Similar to case (2).

Case (4): \( \mathcal{C} = \text{if } \text{then } C' \). It then must be the case that \( \Delta'; \Gamma' \vdash c : \text{bool} \), and \( \Delta'; \Gamma' \vdash p : \tau' \), and \( C' : (\Delta; \Gamma \vdash \Delta'; \Gamma' \vdash \tau') \). By the induction hypothesis we have \( \Delta'; \Gamma' \models C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \tau' \). We get the desired result by using the \( \text{log_if} \) compatibility lemma and the fundamental property.

Case (5): \( \mathcal{C} = \lambda x. \sigma. C' \). Because \( \mathcal{C} : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \tau') \), it must be the case that \( \tau' = \sigma \rightarrow \sigma' \) and \( C' : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; (x : \sigma), \Gamma' \vdash \sigma') \). Then, by the induction hypothesis, \( \Delta'; (x : \sigma), \Gamma' \models C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \sigma' \). We get the desired result by the compatibility lemma.

\[
\Delta'; (x : \sigma), \Gamma' \models C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \sigma' \\
\Delta'; \Gamma' \models \lambda x. C'[\epsilon] : \sigma \rightarrow \sigma' = \tau'
\]

Case (6): \( \mathcal{C} = C' \ t \). In that case \( C' : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \sigma \rightarrow \tau') \) and \( \Delta'; \Gamma' \vdash t : \sigma \) for some type \( \sigma \). By the induction hypothesis we have \( \Delta'; \Gamma' \models C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \sigma \rightarrow \tau' \). Then we use the compatibility lemma

\[
\Delta'; \Gamma' \models C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \sigma \rightarrow \tau' \\
\Delta'; \Gamma' \models t \approx_{\text{log}} t : \sigma \\
\Delta'; \Gamma' \models C'[\epsilon'] \approx_{\text{log}} C'[\epsilon'] : t : \tau'
\]

Case (7): \( \mathcal{C} = C' \ C' \). Similar to Case (6).

Case (8): \( \mathcal{C} = \Lambda \alpha. C' \). Then \( \tau' = \forall \alpha \sigma \) for some type \( \sigma \) and \( C' : (\Delta; \Gamma \vdash \alpha, \Delta'; \Gamma' \vdash \sigma) \). By the induction hypothesis it is the case \( \alpha, \Delta'; \Gamma' \models C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \sigma \). We get the necessary result by the compatibility lemma

\[
\alpha, \Delta'; \Gamma' \models C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \sigma \\
\Delta'; \Gamma' \models \Lambda \alpha. C'[\epsilon'] \approx_{\text{log}} \Lambda \alpha. C'[\epsilon'] : \forall \alpha. \sigma = \tau'
\]

Case (9): \( \mathcal{C} = C'[\sigma] \). Then \( \tau' = \phi(\sigma) \) and \( C' : (\Delta; \Gamma \vdash \tau \Rightarrow \Delta'; \Gamma' \vdash \forall \alpha. \phi(\alpha)) \). By the induction hypothesis \( \Delta', \Gamma' \models C'[\epsilon] \approx_{\text{log}} C'[\epsilon'] : \forall \alpha. \phi(\alpha) \). We obtain the result by applying the \( \text{log_if} \) compatibility lemma.