CSE 505 HW04 Sample Solution

1. Machines; Continuations: See `main.ml`

2. Sub-typing: See `q2.ml`

3. Polymorphic types

   (a) The typing rule for $\mathsf{let\ rec}\ \alpha_1, \ldots, \alpha_n\ \mathsf{f}\ x = e$:

   $$\text{LETRECTYPE}\quad \frac{\Delta, \alpha_1, \ldots, \alpha_n; \Gamma, \mathsf{f} : \forall\alpha_1 \ldots \forall\alpha_n.\ \alpha_1, \ldots, \alpha_n\ \tau_1 \to \tau_2, x : \tau_1 \vdash e : \tau_2 \qquad \Delta, \alpha_1, \ldots, \alpha_n \vdash \tau_1}{\Delta; \Gamma \vdash \mathsf{let\ rec}\ \alpha_1, \ldots, \alpha_n\ \mathsf{f}\ x = e : \forall\alpha_1 \ldots \forall\alpha_n.\ \alpha_1, \ldots, \alpha_n\ \tau_1 \to \tau_2}$$

   (b) The full typing derivation tree for $\mathsf{let\ rec}\ \alpha_1, \alpha_2\ \mathsf{f}\ x = \mathsf{f}[\alpha_1][\alpha_2]\ x$ is

   $$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Delta; \Gamma \vdash \mathsf{f} : \forall\alpha_1\forall\alpha_2.\alpha_1 \to \alpha_2} \quad \dfrac{\alpha_2 \in \Delta}{\Delta \vdash \alpha_2}}{\Delta; \Gamma \vdash \mathsf{f}[\alpha_1] : \forall\alpha_2.\alpha_1 \to \alpha_2} \quad \dfrac{\alpha_2 \in \Delta}{\Delta \vdash \alpha_2}}{\Delta; \Gamma \vdash \mathsf{f}[\alpha_1][\alpha_2] : \alpha_1 \to \alpha_2} \quad \dfrac{}{\Delta; \Gamma \vdash x : \alpha_1}}{\Delta; \Gamma \vdash \mathsf{f}[\alpha_1][\alpha_2]\ x : \alpha_2} \quad \dfrac{\alpha_1 \in \Delta}{\Delta \vdash \alpha_1}}{\cdot; \cdot \vdash \mathsf{let\ rec}\ \alpha_1, \alpha_2\ \mathsf{f}\ x = \mathsf{f}[\alpha_1][\alpha_2]\ x : \forall\alpha_1\forall\alpha_2.\alpha_1 \to \alpha_2}$$

   where $\Delta = \alpha_1, \alpha_2$ and $\Gamma = \mathsf{f} : \forall\alpha_1.\forall\alpha_2.\alpha_1 \to \alpha_2, x : \alpha_1$.

   (c) $\Gamma = \mathsf{raise} : \forall\alpha.\mathsf{exn} \to \alpha, \mathsf{Foo} : \mathsf{exn}$

   $$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot, \alpha_1, \alpha_2; \Gamma, x : \alpha_1 \vdash \mathsf{raise} : \forall\alpha.\mathsf{exn} \to \alpha} \quad \dfrac{\alpha_2 \in \cdot, \alpha_1, \alpha_2}{\cdot, \alpha_1, \alpha_2 \vdash \alpha_2}}{\cdot, \alpha_1, \alpha_2; \Gamma, x : \alpha_1 \vdash \mathsf{raise} : \mathsf{exn} \to \alpha_2} \quad \dfrac{}{\cdot, \alpha_1, \alpha_2; \Gamma, x : \alpha_1 \vdash \mathsf{Foo} : \mathsf{exn}}}{\cdot, \alpha_1, \alpha_2; \Gamma, x : \alpha_1 \vdash \mathsf{raise\ Foo} : \alpha_2} \quad \dfrac{\alpha_1 \in \cdot, \alpha_1, \alpha_2}{\cdot, \alpha_1, \alpha_2 \vdash \alpha_1}}{\cdot, \alpha_1, \alpha_2; \Gamma \vdash \lambda\ x.\ \mathsf{raise\ Foo} : \alpha_1 \to \alpha_2}}{\cdot, \alpha_1; \Gamma \vdash \Lambda\alpha_2.\ \lambda\ x.\ \mathsf{raise\ Foo} : \forall\alpha_2.\alpha_1 \to \alpha_2}}{\cdot; \Gamma \vdash \Lambda\alpha_1.\ \Lambda\alpha_2.\ \lambda\ x.\ \mathsf{raise\ Foo} : \forall\alpha_1\forall\alpha_2.\alpha_1 \to \alpha_2}$$

   (d) α is not free in τ; therefore, a value of type α can not be derived from the argument to the v. A typing derivation must assume that we have a function which can take something and return any type α (e.g., `raise` in part (c)) or that the function never returns a value (e.g., the infinite loop in part(b)).

4. Protocol Enforcement:

   (a) See `q4.ml`

   (b) See `q4.ml`

   (c) See `q4.ml`

   (d) Client2 cannot violate the protocol because the client does not have access to a read function until it has called `open`. Client3 cannot violate the protocol until because it cannot call read until it has a value of type α; since α is unknown, the only way to get an alpha is by calling `open` first. They are both more efficient than client one because they do not have any runtime effect while `dynamicCheck` must set a boolean when it opens a file and read that boolean before every read.

(e) The approach taken in part (a) will enforce the `close` protocol; once a file is closed, the boolean flag is set back to false and reading cannot occur. Neither the approach from part (b) nor the approach from part (c) will work. These approaches both need to *give* permissions before reading is allowed, but they cannot *revoke* permissions to ensure that reading does not occur after a close.