```
instance (υ : kind → kind) ⇒ Monad (freer υ)
  { lift a   := pure a
  ; map  f m := cases m
                 { pure    a    ⇒ pure (f a)
                 | impure y q ⇒ impure y (map f ∘ q) }
  ; bind m k := cases m
                 { pure    x    ⇒ k x
                 | impure y q ⇒ impure y (x ⇒ bind (q x) k) } }.

type nondeterministic-base : kind → kind := list.
type nondeterministic      : kind → kind := freer nondeterministic-base.

term sample-base (α:kind) (l : list α) : nondeterministic-base α := l.
term sample      (α:kind) (l : list α) : freer-lift (sample-base l).

effect nondeterministic
 lifts list
  { sample l := l }.

type exceptional-base (ε α : kind) : kind := ε ⊕ α.
type exceptional      (ε α : kind) : kind := freer (exceptional-base ε) α.

term throw-base (ε α : kind) (e:ε) : exceptional-base ε α := left e.
term throw-base (ε α : kind) (e:ε) : exceptional      ε α := freer-lift (throw-base e).

term valid-base (ε α : kind) (a:α) : exceptional-base ε α := right a.
term valid      (ε α : kind) (a:α) : exceptional      ε α := freer-lift (valid-base a).
```