```
module 2022Cats6 where

open import Level
open import Relation.Binary.PropositionalEquality as Eq
open Eq.≡-Reasoning

record Cat {ℓ₀ ℓ₁ : Level} : Set (suc (ℓ₀ ⊔ ℓ₁)) where
  infix 10 _▫_
  field
    Objs  : Set ℓ₀
    Hom   : Objs → Objs → Set ℓ₁
    id    : {D : Objs} → (Hom D D)
    _▫_   : {C D E : Objs} → (g : Hom D E) → (f : Hom C D) → (Hom C E)
    idL   : {D E : Objs} → (g : Hom D E) → (g ≡ g ▫ id)
    idR   : {D E : Objs} → (g : Hom D E) → (g ≡ id ▫ g)
    assoc : {C D E F : Objs}
            → (f : Hom C D)
            → (g : Hom D E)
            → (m : Hom E F)
            → (m ▫ (g ▫ f) ≡ (m ▫ g) ▫ f)
  --
  f≡f▫id : {D E : Objs} → {f : Hom D E} → f ≡ f ▫ id
  f≡f▫id {D} {E} {f} = idL f
  f≡id▫f : {D E : Objs} → {f : Hom D E} → f ≡ id ▫ f
  f≡id▫f {D} {E} {f} = idR f
  f▫id≡f : {D E : Objs} → {f : Hom D E} → f ▫ id ≡ f
  f▫id≡f {D} {E} {f} = sym (idL f)
  id▫f≡f : {D E : Objs} → {f : Hom D E} → id ▫ f ≡ f
  id▫f≡f {D} {E} {f} = sym (idR f)
  f▫[g▫h]≡[f▫g]▫h : {C D E F : Objs}
                    → {f : Hom C D} → {g : Hom D E} → {h : Hom E F}
                    → h ▫ (g ▫ f) ≡ (h ▫ g) ▫ f
  f▫[g▫h]≡[f▫g]▫h {C} {D} {E} {F} {f} {g} {h} = assoc f g h
  [f▫g]▫h≡f▫[g▫h] : {C D E F : Objs}
                    → {f : Hom C D} → {g : Hom D E} → {h : Hom E F}
                    → (h ▫ g) ▫ f ≡ h ▫ (g ▫ f)
  [f▫g]▫h≡f▫[g▫h] {C} {D} {E} {F} {f} {g} {h} = sym (assoc f g h)
```

```
record Functor {ℓ₀ ℓ₁ ℓ₂ ℓ₃ : Level}
               (catB : Cat {ℓ₀} {ℓ₁})
               (catA : Cat {ℓ₂} {ℓ₃})
         : Set (suc (ℓ₀ ⊔ ℓ₁ ⊔ ℓ₂ ⊔ ℓ₃)) where
  module A = Cat catA
  module B = Cat catB
  field
    ac0     : B.Objs → A.Objs
    ac1     : {C D : B.Objs} → (B.Hom C D) → (A.Hom (ac0 C) (ac0 D))
    respids : {C : B.Objs} → (ac1 (B.id {C}) ≡ A.id {ac0 C})
    respcomp : {C D E : B.Objs} → {f : B.Hom C D} → {g : B.Hom D E}
               → (ac1 (g B.▫ f) ≡ ((ac1 g) A.▫ (ac1 f)))
  --
  id[FC]≡F[idC] : {C : B.Objs} → ac1 (B.id {C}) ≡ A.id {ac0 C}
  id[FC]≡F[idC] {C} = respids
  F[idC]≡id[FC] : {C : B.Objs} → A.id {ac0 C} ≡ ac1 (B.id {C})
  F[idC]≡id[FC] {C} = sym respids
  F[g▫h]≡Fg▫Fh : {C D E : B.Objs} → {f : B.Hom C D} → {g : B.Hom D E}
                  → (ac1 (g B.▫ f) ≡ ((ac1 g) A.▫ (ac1 f)))
  F[g▫h]≡Fg▫Fh {C} {D} {E} {f} {g} = respcomp
  Fg▫Fh≡F[g▫h] : {C D E : B.Objs} → {f : B.Hom C D} → {g : B.Hom D E}
                  → ((ac1 g) A.▫ (ac1 f)) ≡ (ac1 (g B.▫ f))
  Fg▫Fh≡F[g▫h] {C} {D} {E} {f} {g} = sym respcomp
  --
  -- How to use it:
  -- R.ac0 C = RC
  -- R.ac1 {C} {D} f = R f
  -- R.respids {C} : R(id C) ≡ id(RC)
  -- R.respcomp {C} {D} {E} {f} {g} : R(g∘f) ≡ Rg∘Rf
```

```
catSet : Cat {suc zero} {zero}
Cat.Objs  catSet       = Set
Cat.Hom   catSet A B   = (A → B)
Cat.id    catSet {A}   = (λ a → a)
Cat._▫_   catSet g f   = (λ a → g (f a))
Cat.idL   catSet f     = refl
Cat.idR   catSet f     = refl
Cat.assoc catSet f g h = refl

catSetH : Cat {suc zero} {zero}
Cat.Objs  catSetH       = Set
Cat.Hom   catSetH A B   = (A → B)
Cat.id    catSetH {A}   = (λ a → a)
Cat._▫_   catSetH g f   = (λ a → g (f a))
Cat.idL   catSetH f     = refl
Cat.idR   catSetH f     = refl
Cat.assoc catSetH f g h = refl
```