```agda
module lib where

infix 4 _≤_ _<_ _≡_
infixl 6 _+_ _∸_ _−_
infixl 7 _*_

data ℕ : Set where
  zero : ℕ
  suc : ℕ → ℕ
{-# BUILTIN NATURAL ℕ #-}

data ℤ : Set where
  pos : ℕ → ℤ
  negsuc : ℕ → ℤ
{-# BUILTIN INTEGER          ℤ #-}
{-# BUILTIN INTEGERPOS       pos #-}
{-# BUILTIN INTEGERNEGSUC negsuc #-}

_+_ : ℕ → ℕ → ℕ
zero + n = n
suc m + n = suc (m + n)
{-# BUILTIN NATPLUS _+_ #-}

-- Monus (a∸b = max{a-b, 0})
_∸_ : ℕ → ℕ → ℕ
m ∸ zero = m
zero ∸ suc n = zero
suc m ∸ suc n = m ∸ n
{-# BUILTIN NATMINUS _∸_ #-}

_*_ : ℕ → ℕ → ℕ
zero * n = zero
suc m * n = n + m * n
{-# BUILTIN NATTIMES _*_ #-}

-- Relations of natural numbers
data _≡_ {a} {A : Set a} (x : A) : A → Set a where
  refl : x ≡ x
{-# BUILTIN EQUALITY _≡_ #-}

cong : ∀ {A B : Set} (f : A → B) {x y : A} → x ≡ y → f x ≡ f y
cong f refl = refl

sym : ∀ {A : Set} {x y : A} → x ≡ y → y ≡ x
sym refl = refl

sub : ∀ {A : Set} {x y : A} (P : A → Set) → x ≡ y → P x → P y
sub P refl px = px

trans : ∀ {A : Set} {x y z : A} → x ≡ y → y ≡ z → x ≡ z
trans refl refl = refl

-- n∸n≡0 : ∀ {n : ℕ} → n ∸ n ≡ zero
-- n∸n≡0 {zero} = refl
-- n∸n≡0 {suc n} = n∸n≡0 {n}

data _≤_ : ℕ → ℕ → Set where
  z≤n : ∀ {n : ℕ} → zero ≤ n
  s≤s : ∀ {m n : ℕ} → m ≤ n → suc m ≤ suc n

inv-s≤s : ∀ {m n : ℕ} → suc m ≤ suc n → m ≤ n
inv-s≤s (s≤s m≤n) = m≤n

≤-refl : ∀ {n : ℕ} → n ≤ n
≤-refl {zero} = z≤n
≤-refl {suc n} = s≤s ≤-refl

≤-trans : ∀ {m n p : ℕ} → m ≤ n → n ≤ p → m ≤ p
≤-trans z≤n _ = z≤n
≤-trans (s≤s m≤n) (s≤s n≤p) = s≤s (≤-trans m≤n n≤p)

n≤suc-n : ∀ {n : ℕ} → n ≤ suc n
n≤suc-n {zero} = z≤n
n≤suc-n {suc n} = s≤s n≤suc-n

+→≤ : ∀ {m n : ℕ} → m ≤ m + n
+→≤ {zero} {n} = z≤n
+→≤ {suc m} {n} = s≤s (+→≤ {m} {n})

data Order : ℕ → ℕ → Set where
  leq : ∀ {m n : ℕ} → m ≤ n → Order m n
  geq : ∀ {m n : ℕ} → n ≤ m → Order m n

≤-compare : ∀ {m n : ℕ} → Order m n
≤-compare {zero} {n} = leq z≤n
≤-compare {suc m} {zero} = geq z≤n
≤-compare {suc m} {suc n} with ≤-compare {m} {n}
... | leq m≤n = leq (s≤s m≤n)
... | geq n≤m = geq (s≤s n≤m)

-- ∸-≤ : ∀ {m n} → m ∸ n ≤ m
-- ∸-≤ {m} {zero} = ≤-refl
-- ∸-≤ {zero} {suc n} = z≤n
-- ∸-≤ {suc m} {suc n} = ≤-trans (∸-≤ {m} {n}) n≤suc_n

data _<_ : ℕ → ℕ → Set where
  z<s : ∀ {n : ℕ} → zero < suc n
  s<s : ∀ {m n : ℕ} → m < n → suc m < suc n

<→s≤ : ∀ {m n : ℕ} → m < n → suc m ≤ n
<→s≤ (z<s) = s≤s z≤n
<→s≤ (s<s m<n) = s≤s (<→s≤ m<n)

<→≤ : ∀ {m n : ℕ} → m < n → m ≤ n
<→≤ m<n = ≤-trans n≤suc-n (<→s≤ m<n)

<-trans : ∀ {m n p : ℕ} → m < n → n < p → m < p
<-trans z<s (s<s _) = z<s
<-trans (s<s m<n) (s<s n<p) = s<s (<-trans m<n n<p)

data Fin : ℕ → Set where
  fzero : ∀ {n} → Fin (suc n)
  fsuc : ∀ {n} → Fin n → Fin (suc n)

toℕ : ∀ {m} → Fin m → ℕ
toℕ fzero = zero
toℕ (fsuc i) = suc (toℕ i)

-- max-Fin : ∀ {m} → Fin (suc m)
-- max-Fin {zero} = fzero
-- max-Fin {suc m} = fsuc max-Fin

-- toℕ-max-Fin : ∀ {n} → toℕ (max-Fin {n}) ≡ n
-- toℕ-max-Fin {zero} = refl
-- toℕ-max-Fin {suc n} = cong suc toℕ-max-Fin

≤→Fin : ∀ {m n} → m ≤ n → Fin (suc n)
≤→Fin z≤n = fzero
≤→Fin (s≤s p) = fsuc (≤→Fin p)

toℕ-≤→Fin : ∀ {m n} → (m≤n : m ≤ n) → toℕ (≤→Fin m≤n) ≡ m
toℕ-≤→Fin z≤n = refl
toℕ-≤→Fin (s≤s m≤n) = cong suc (toℕ-≤→Fin m≤n)

-- max-Fin≡≤-refl→Fin : ∀ {n} →  max-Fin {n} ≡ ≤→Fin (≤-r
-- max-Fin≡≤-refl→Fin {zero} = refl
-- max-Fin≡≤-refl→Fin {suc n} = cong fsuc max-Fin≡≤-refl→F

-- Minus
-- _-_ : (m : ℕ) → (n : ℕ) → (n ≤ m) → ℕ
-- (m - n) _ = m ∸ n
-- (m - zero) _ = m
-- (suc m - suc n) p = (m - n) (inv-s≤s p)
-- _-_ : (m : ℕ) → (n : Fin (suc m)) → ℕ
-- m - n = m ∸ toℕ n
_−_ : (m : ℕ) → Fin (suc m) → ℕ
m − fzero = m
suc m − fsuc n = m − n

-- -→≤ : ∀{m : ℕ} → ∀{n : Fin (suc m)} → m - n ≤ m
-- -→≤ {m} {n} = ∸-≤ {m} {toℕ n}
−→≤ : ∀ {m} → ∀ {n : Fin (suc m)} → m − n ≤ m
−→≤ {m} {fzero} = ≤-refl
−→≤ {suc m} {fsuc n} = ≤-trans ((−→≤ {m} {n})) (n≤suc-n {m})

-- n-n≡0 : ∀{n : ℕ} → n - (max-fin {n}) ≡ 0
-- n-n≡0 {n} = subst (λ m → n ∸ m ≡ 0) (sym toℕ-max-fin) (n
-- n-n≡0 : ∀{n} → n - (max-Fin {n}) ≡ 0
-- n-n≡0 {zero} = refl
-- n-n≡0 {suc n} = n-n≡0 {n}

n−n≡0 : ∀ {n} → n − (≤→Fin (≤-refl {n})) ≡ 0
n−n≡0 {zero} = refl
n−n≡0 {suc n} = n−n≡0 {n}

-- suc (n - m) ≡ suc n - m
−-suc : ∀ {n m} → {m≤n : m ≤ n} → suc (n − ≤→Fin m≤n) ≡ suc n −
−-suc {_} {zero} {z≤n} = refl
−-suc {suc n} {suc m} {s≤s m≤n} = −-suc {n} {m} {m≤n}


n−[n−m]≡m : ∀ {m n} → (m≤n : m ≤ n) → n − (≤→Fin (−→≤ {n} {≤-
n−[n−m]≡m {zero} {n} z≤n = n−n≡0 {n}
n−[n−m]≡m {suc m} {suc n} (s≤s m≤n) = trans (sym (−-suc {n} {n − ≤

-- m ≤ n → m - p ≤ n - p
−-monoʳ-≤ : ∀ {p m n} → (p≤m : p ≤ m) → (m≤n : m ≤ n) → m − (≤-
−-monoʳ-≤ z≤n m≤n = m≤n
−-monoʳ-≤ (s≤s p≤m) (s≤s m≤n) = −-monoʳ-≤ p≤m m≤n

-- m ≡ n, p ≤ n → p ≤ m
m≡n,p≤n→p≤m : ∀ {p m n} → m ≡ n → p ≤ n → p ≤ m
m≡n,p≤n→p≤m m≡n p≤n rewrite sym m≡n = p≤n


-- suc d ≤ d' → d ≤ d' - (d' - (suc d))
suc-d≤d'→d≤d'−[d'−[suc-d]] : ∀ {d d'} → (δ₁≤δ₂ : suc d ≤ d') → d ≤ (d
suc-d≤d'→d≤d'−[d'−[suc-d]] {d} {d'} δ₁≤δ₂ = m≡n,p≤n→p≤m (n−[n−m]
```