Common.hs

```
module Common where

  -- Comandos interactivos o de archivos
  data Stmt i = Def String i           --  Declarar un nuevo identificador x, let x = t
            | Eval i                  --  Evaluar el término
    deriving (Show)

  instance Functor Stmt where
    fmap f (Def s i) = Def s (f i)
    fmap f (Eval i)  = Eval (f i)

  -- Tipos de los nombres
  data Name
     =  Global  String
     |  Quote   Int
    deriving (Show, Eq)

  -- EntornosLVar String
  type NameEnv v t = [(Name, (v, t))]

  -- Tipo de los tipos
  data Type = Base
           | Fun Type Type
           | TUnit
           | TPar Type Type
           | TNat
           deriving (Show, Eq)

  -- Términos con nombres
  data LamTerm  =  LVar String
              |  Abs String Type LamTerm
              |  App LamTerm LamTerm

              |  LetIn String LamTerm LamTerm
              |  Asc LamTerm Type
              |  LUnit
              |  Par LamTerm LamTerm
              |  Pri LamTerm
              |  Seg LamTerm
              |  Cero
              |  Succ LamTerm
              |  Rec LamTerm LamTerm LamTerm
              deriving (Show, Eq)


  -- Términos localmente sin nombres
  data Term  = Bound Int
            | Free Name
            | Term :@: Term
            | Lam Type Term
```

```haskell
           | Let Term Term
           | As Term Type
           | Unit
           | Pair Term Term
           | Fst Term
           | Snd Term
           | Zero
           | Suc Term
           | R Term Term Term
        deriving (Show, Eq)

-- Valores
data Value = VLam Type Term
           | VUnit
           | VPar Value Value
           | NV Nv

data Nv = V0
        | VSuc Nv




-- Contextos del tipado
type Context = [Type]
```