Chapter 1

Lista_5

```
module Lista_5 (
    NFData(rnf), deepseq, ($!!), subseqM, ipermM, spermM,
    List(Nil, Cons), SimpleList(SimpleList, fromSimpleList),
    ListView(nil, cons, toList, viewList),
    CList(CNil, CSingle, (:++:)), DList(DList, fromDList), dappend
  ) where
```

```
class NFData a where
```

Methods

```
rnf ::  a -> ()
```

```
instance Num a => NFData a
instance NFData a => NFData [a]
instance (NFData a, NFData b) => NFData (a, b)
```

```
deepseq ::  NFData a => a -> b -> b
```

```
($!!)  ::  NFData a => (a -> b) -> a -> b
```

```
subseqM ::  MonadPlus m => [a] -> m [a]
```

```
ipermM ::  MonadPlus m => [a] -> m [a]
```

```
spermM :: MonadPlus m => [a] -> m [a]
```

```
data List t a
```

Constructors

```
= Cons a (t a)
| Nil
```

```
newtype SimpleList a
```

Constructors

```
= SimpleList
      { fromSimpleList :: List SimpleList a
      }
```

```
class ListView t where
```

Methods

```
viewList :: t a -> List t a

toList :: t a -> [a]

cons :: a -> t a -> t a

nil :: t a
```

```
instance ListView DList
instance ListView CList
```

```
data CList a
```

Constructors

```
= (CList a) :++: (CList a)
| CSingle a
| CNil
```

```
instance Monad CList
instance Functor CList
instance Applicative CList
instance Foldable CList
instance Traversable CList
instance MonadPlus CList
instance Alternative CList
instance ListView CList
instance Show a => Show (CList a)
```

```
newtype DList a
    Constructors
    =  DList
          { fromDList ::  [a] -> [a]
          }
```

```
instance Monad DList
instance Functor DList
instance Applicative DList
instance Foldable DList
instance Traversable DList
instance MonadPlus DList
instance Alternative DList
instance ListView DList
```

```
dappend ::  DList a -> DList a -> DList a
```