Inductive P

$(A_1 : \sigma_1)$

$\vdots$

$(A_n : \sigma_n)$

$: \alpha_1 \to \dots \to \alpha_m \to \mathrm{Type}$

$:=$

$\mid \mathrm{intro}_i : \mathrm{forall}$ $\vdots$

$(b_i : \beta_i)$

$\vdots$

$\vdots$

$(x_i : \xi_1$

$\to \cdots$

$\to \xi_k$

$\to P\ A_1 \dots A_n\ p_1 \dots p_m)$

$\vdots$,

$P\ A_1 \dots A_n\ q_1 \dots q_m$

Inductive P] the inductive family

$(A_1 : \sigma_1)$

$\vdots$

$(A_n : \sigma_n)$] Parameters (same in every constructor)

$$: \underbrace{\alpha_1 \to \dots \to \alpha_m}_{\text{indices (can be different in constructors)}} \to \text{Type}$$

$:=$

$| \underbrace{\text{intro}_i}_{\text{constructors}} : \text{forall}$

$\vdots$

$(b_i : \beta_i)$] non-recursive arguments

$\vdots$

recursive arguments (denoted by γ) [

$\vdots$

$(x_i : \xi_1 \to \dots \to \xi_k \to P\ A_1 \dots A_n\ p_1 \dots p_m)$

$\vdots$,

$P\ A_1 \dots A_n\ q_1 \dots q_m$

↳ return type

```
Inductive nat
   : Type

    :=
| Z : nat

| S :  nat

         → nat
```

$A : \text{empty list}$

Inductive nat

$: \underbrace{\text{Type}}_{\alpha : \text{empty list}}$

:=

$| \; Z : \text{nat} \left\{ \begin{array}{l} \beta : \text{empty list} \\ \gamma : \text{empty list} \end{array} \right.$

$| \; S : \text{nat} \left\{ \begin{array}{l} \beta : \text{empty list} \\ \gamma : \text{single element} \\ \xi : \text{empty list} \end{array} \right.$

→ nat

```
Inductive list
          (A: Type)
  : Type

  :=
| nil: list A

| cons: forall (a: A)
               (x: listA),
          list A
```

Inductive list
(A: Type) — A is a singleton list

$: \underbrace{\text{Type}}_{\alpha\text{ : empty list}}$

:=

| nil: list A $\left\{ \begin{array}{l} \beta\text{: empty list} \\ \gamma\text{ : empty list} \end{array} \right.$

| cons: forall (a: A) { β: singleton list of A

(x: list), $\left\{ \begin{array}{l} \gamma\text{: single element} \\ \xi\text{: empty list} \end{array} \right.$

list A

```
Inductive Vec
          (A: Type)
  : nat -> Type
  :=
| nil : Vec A 0

| cons : forall (b : A)
                (n: nat)
                (x: Vec A n),
          Vec A (S n)
```

Inductive Vec

(A: Type) — A singleton list

$: \underbrace{\text{nat}}_{} \rightarrow \text{Type}$

:= α : singleton list

| nil : Vec A 0 $\left\{ \begin{array}{l} \beta \text{ : empty list} \\ \text{no recursive arguments} \end{array} \right.$

| cons : forall (b : A) $\left.\begin{array}{l} (b : A) \\ (n: nat) \end{array}\right\}$ β : list of A and nat

(x: Vec A n), $\left\{ \begin{array}{l} \gamma \text{ has one element} \\ \xi \text{ : empty list} \end{array} \right.$

Vec A (S n)

```
Inductive Tree
    : Type


    :=
| leaf  :  Tree
| node  :  forall (x:nat -> Tree),
              Tree
```

A empty list

Inductive Tree

: $\underbrace{\text{Type}}_{\alpha \text{ empty list}}$

:=

| leaf : Tree

| node : forall $(X : \text{nat} \to \text{Tree})$, $\left\{\begin{array}{l}\beta : \text{empty}\\ \gamma : \text{single element}\\ \xi : \text{nat}\end{array}\right.$

Tree