Les bases de Rocq.

1 Les définitions par induction : `Inductive`.

En Rocq (anciennement Coq), on peut définir des ensembles par induction. Pour cela, on utilise le mot `Inductive`.

Par exemple, pour définir un type de liste d'entiers, on utilise le code ci-dessous.

```
Inductive nlist : Set:=
| Nil : nlist
| Cons : nat -> nlist -> nlist.
```

Code 1 | *Définition du type* nlist *en Rocq*

En Rocq, au lieu de définir la fonction `Cons` comme une « fonction » de la forme `Cons` : nat ∗ nlist → nlist, on la *curryfie* en une « fonction » de la forme `Cons` : nat → nlist → nlist. Les types définis par les deux versions sont isomorphes.

Pour définir une relation, on utilise aussi le mot clé `Inductive` :

```
Inductive le : nat -> nat -> Prop :=
| le_refl : forall n, le n n
| le_S : forall n k, le n k -> le (S n) (S k).
```

Code 2 | *Définition de la relation* le

Aux types définis par induction, on associe un principe d'induction (qu'on voir avec `Print le_ind.` ou `Print nlist_ind.`). Ce principe d'induction permet de démontrer une propriété $\mathscr{P}$ sur un ensemble/une relation définie par induction.

2 Quelques preuves avec Rocq.

On décide de prouver le lemme suivant avec Rocq.

> **"Lemme" 1.** Soit ℓ une liste triée, et soient a et b deux entiers tels que $a \leq b$. Alors la liste $a :: b :: \ell$ est triée.

Pour cela, on écrit en Rocq :

```
Lemma exemple_triee :
  forall l, triée l →
    forall a b, le a b →
      triée (Cons a (Cons b l)).
```

Il ne reste plus qu'à prouver ce lemme. On commence la démonstration par introduire les variables et hypothèses : les variables `l`, `a`, `b`, et les hypothèses (`H1`) : triée `l`, et (`H2`) : le `a` `b`. On commence par introduire la liste `l` et l'hypothèse `H1` et on s'occupera des autres un peu après.

```
Proof.
  intros l H1.
```

On décide de réaliser une preuve par induction sur la relation triée, qui est en hypothèse (`H1`).

```
  induction H1.
```

Dans le cas d'une preuve par induction sur triée, on a *trois* cas.

- ▷ *Cas 1.* On se trouve dans le cas `l` = `Nil`. Pas trop de problèmes pour prouver que `[a;b]` est triée avec l'hypothèse `a` ≤ `b`. On introduit les variables et hypothèses `a`, `b` et `H2`.

  ```
    - intros a b H2.
  ```

  À ce moment de la preuve, l'objectif est de montrer :

  $$\text{triée } \texttt{Cons}(\texttt{a}, \texttt{Cons}(\texttt{b}, \texttt{Nil})).$$

  Pour cela, on utilise deux fois les propriétés de la relation triée :

```
apply t_cons.
apply t_singl.
```

Notre objectif a changé, on doit maintenant démontrer `le a b`. C'est une de nos hypothèses, on peut donc utiliser :

```
assumption.
```

Ceci termine le cas 1.

- *Cas 2.* On se trouve dans le cas `l` = `[k]`. On doit de démontrer que la liste `[a;b;k]` est triée. On a l'hypothèse `a` ≤ `b`, mais aucune hypothèse de la forme `b` ≤ `k`. On est un peu coincé pour ce cas...

(Un jour je finirai d'écrire cette partie... Malheureusement, ce n'est pas aujourd'hui...)