# Proof Assistants

TD 3- Recursive functions

## 1 Recursion

### 1.1 Fibonacci

1- Write the function computing the Fibonacci sequence, following its definition $F(0) = 0$, $F(1) = 1$ and $F(n+2) = F(n) + F(n+1)$. Compute $F(4)$.
2- We want to write a scheme mimicking the recursive calls of Fibonacci. For this purpose, we introduce a parameterized type $P$ such that $P(n)$ is the type of the value returned for the input $n$. We also assume we are given the value at 0, the value at 1 and the value at $n+2$ computed from the values at $n$ and $n+1$:

```
Section FibPrinciple.
Variable P : nat -> Type.
Variable P0: P 0.
Variable P1: P 1.
Variable Pr : forall n, P n -> P (S n) -> P (S (S n)).
```

Write a function `Pfib1` of type $\forall n, P(n)$.
3- We now want to write a more efficient scheme, that makes only one recursive call. The idea is to compute $F(n)$ and $F(n+1)$ at the same time. Using the same parameters as in the previous question, write a function
On va maintenant écrire un schéma plus efficace, qui ne fait qu'un seul appel récursif. L'idée est de calculer en même temps $F(n)$ et $F(n+1)$. En utilisant les mêmes paramètres que pour la question précédente, écrire une fonction `Pfib2` of type $\forall n, P(n) * P(n+1)$.
4- Close the section:

```
End FibPrinciple.
```

Constants `Pfib1` and `Pfib2` now have 4 extra arguments corresponding to `P`, `P0`, `P1` and `Pr`. Instantiate these 2 schemes in order to compute the Fibonacci sequence, and compute $F(4)$ with both functions.

### 1.2 Lists

We are now considering lists of elements of type $A$.

```
Require Import List.
Parameter A : Type.
```

1- Write a function `split` that splits a list in 2 listes of similar lengths. Taking Fibonacci as a model, write an induction scheme associated to `split`.
2- Prove that each list returned by `split` is of length less or equal to the input list. We also prove that if the input list has length greater or equal than 1, then the returned lists have a length strictly smaller then the input.
3- Let `le:A->A->bool` be an order ond $A$. Write a function taking as argument a liste and an integer $n$, and that reurns the liste using the Quicksort algorithm, not going further than depth $n$. Prove that this function does not depend on $n$ beyond some bound.

# 2 Partial functions

```
Variables (t : A) (F : nat -> nat -> A -> A).
```

We wish to define function $f$ such that $f(n, n) = t$ and $f(n, m) = F(n, m, f(n, m - 1))$ for $n < m$. Show the following lemma, the resulting proof should be in any case a subproof of that of $n \leq m$:

```
Lemma le_pred : forall n m, n <= m -> n <> m -> n <= pred m.
```

Write the function $f$ having as a third argument the proof that $n$ and $m$ belong to the domain.