RICM2 - 2004/05
Langage et Programmation 2

Programmation fonctionnelle

Exercice 1: Maximum de deux entiers
Décrire une fonction qui détermine le maximum de deux entiers.

Spécification
    max2 : deux entiers → un entier

Réalisation
    max2(a,b)
        si a>b alors a sinon b

Caml
```
let (max2 : int*int -> int) = function
    (a,b) -> if a>b then a else b
```

Exercice 2: Factorielle
Décrire une fonction qui à un entier strictement positif n associe le produit des n premiers entiers (noté habituellement n !).

Spécification
    fac : un entier $> 0 \rightarrow$ un entier > 0

Relations de récurrence
    fac(1)=1
    fac(n)=fac(n-1)*n

Version 1

    Réalisation
        fac1(n)
            si n=1 alors 1 sinon fac1(n-1)*n

    Caml
```
let rec (fac1 : int -> int) = function
```

n -> if n=1 then 1 else n*fac1(n-1)

Version 2

Réalisation

fac2(n)
  fac_aux(n,1)

fac_aux(n,a)
  si n=1 alors a
  sinon fac_aux(n-1,n*a)

Caml

```
let rec (fac_aux : int*int -> int) = function
    (n,a) -> if n=1 then a else fac_aux(n-1,n*a)

let (fac2 : int -> int) = function
    n -> fac_aux(n,1)
```

Passage de la version 1 à la version 2

P(n)=$\forall$ a $\in$ **N**, fac_aux(n,a)=a*n !
Montrer $\forall$ n $\in$ **N**, P(n)

n=0 :
fac_aux(0,a)=a et a*0 !=a
donc fac_aux(0,a)=a*0 ! (1)

n+1 :
On suppose P(n)
fac_aux(n+1,a)=fac_aux(a*(n+1))*n
donc fac_aux(n+1,a)=fac_aux(n,a*(n+1))
or $\forall$ a $\in$ **N**, fac_aux(n,a)=a*n !
donc fac_aux(n+1,a)=(a*(n+1))*n !
donc fac_aux(n+1,a)=a*((n+1)*n !)
donc fac_aux(n+1,a)=a*(n+1) ! (2)

Par récurrence on déduit de (1) et (2) $\forall$ n $\in$ **N**, P(n)

Exercice 3: PGCD

Décrire une fonction qui à deux entiers strictement positifs a et b associe le PGCD de a et de b.

Spécification

pgcd : un entier > 0 , un entier $> 0 \rightarrow$ un entier > 0

Relations de récurrence

pgcd(a,a)=a

pgcd(a,b)=si a>b alors pgcd(a-b,b) sinon pgcd(a,b-a)

Réalisation

```
pgcd(a,b)
    si a=b alors a
    sinon
        si a>b alors pgcd(a-b,b)
        sinon pgcd(a,b-a)
```

Caml

```
let rec (pgcd : int*int -> int) = function
    (a,b)-> if a=b then a
            else
                if (a>b) then pgcd(a-b,b)
                else pgcd(a,b-a)
```

Exercice 4: Suite de Fibonacci

Décrire une fonction qui à un entier positif associe la suite de Fibonacci de rang n.

La suite

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... jusqu'à l'infini.

Spécification

fib : un entier $> 0 \rightarrow$ un entier > 0

Relations de récurrence

fib(0)=0

fib(1)=1

fib(n)=fib(n-1)+fib(n-2), pour n>1

Version 1

Réalisation

```
fib(n)
    si n=0 alors 0
    sinon
        si n=1 alors 1
        sinon fib(n-1)+fib(n-2)
```

Caml

```
let rec (fib : int -> int) = function
    n -> if n=0 then 0
         else
             if n=1 then 1
             else fib(n-1)+fib(n-2)
```

Version 2

Réalisation

```
fib(n)
    fib_aux(n,0,1)

fib_aux(n,a,b)
    si n=0 alors a
    sinon fib_aux(n-1,b,a+b)
```

Caml

```
let rec (fib_aux : int*int*int -> int) = function
    (n,a,b) -> if n=0 then a
               else fib_aux(n-1,b,a+b)

let (fib : int -> int) = function
    n -> fib_aux(n,0,1)
```