Université A.Mira Béjaia
Département Maths MI 1ière année
Examen de Calcul formel - S2 / 15 Juin 2009

Nom : ……
Prénom : ……
Section /Groupe :…….. Emargement : ……

Exercice 1 (2 points) :

Ecrire une fonction qui prend en entrée un flottant, et rend une fonction qui calcule sa racine si le nombre est positif, et zéro sinon.

```
# let racine x = match x with
     i when i>0. -> sqrt x
     |_ -> 0.;;
     val racine : float -> float = <fun>
```

0ù bien :

```
# let racine x = if x>0. then sqrt x else 0.;;
     val racine : float -> float = <fun>
```

Exercice 2 (8 points) :

a. Donnez le type des fonctions suivantes :

```
# let f x y z = if x > 2.3 then y else z;;
val f : float -> 'a -> 'a -> 'a = <fun>
```

```
#let f (x, a, b) = if b then a else int_of_float   x;;
val f : float * int * bool -> int =<fun>
```

```
#let h (x,y)(z,t)=(x+z,y+t);;
val h : int * int -> int * int -> int * int = <fun>
```

```
# let f x g = g x +. x ;;
val f : float -> (float -> float) -> float = <fun>
```

```
#(fun (y,z) -> y ^ (string_of_int (z+1)));;
- : string * int -> string = <fun>
```

b) Soit la définition suivante :

```
#let rec  f  l = let  rec  g  e  l= match l with
     []->[]
     |x::r->if x=e then g e r else x:: g e r  in  match  l  with
          []->[]
          |a::b->a::f  (g  a  b);;
```

- Son type : `val f : 'a list -> 'a list = <fun>`
- L'interprétation suivante # f [1;5;3;5;4;6;7;8];; renvoie : - : int list = [1; 5; 3; 4; 6; 7; 8]
- f est une fonction qui prend en entrée une liste l et qui renvoie une liste qui contient une seule fois les éléments présents dans la liste l

Exercice 3(3points) :

Avec le filtrage la fonction f s'écrit

```
# let f n m = match n,m with
      (i,j) when i>j -> 1
      |(i,j) when i=j -> 0
      |_ -> -1;;
val f : 'a -> 'a -> int = <fun>
```

Exercice 4 (7 points) :

Ecrire le code des fonctions suivantes :

1. La fonction *fus_list* qui fusionne deux listes *a* et *b* déjà triées et renvoie leur union, c'est-à-dire une liste triée qui contient à la fois les éléments de *a* et ceux de *b*
2. La fonction *impair_list* qui renvoie la liste des éléments de rang impair de la liste *l*
3. La fonction *pair_list* qui renvoie la liste des éléments de rang pair de la liste *l*
4. La fonction *fus_tri* qui tri par fusion une liste *l*. Cette fonction utilise les trois premières fonctions, elle consiste à scinder la liste *l* en deux listes *a* (rang pair) et *b* (rang impair), trier les deux listes et fusionner les deux listes triées en une seule. Le processus récursif du tri s'arrête nécessairement puisque l'on sait trier une liste vide et une liste qui contient un seul élément.

NB : *Le premier élément d'une liste est l'élément de rang zéro.*

Le corrigé :

1.
```
# let rec fus_list a b = match a,b with
    |[],_ -> b
    |_,[] -> a
    |ha::ta,hb::tb -> if ha < hb then ha::(fus_list ta b)else
    hb::(fus_list a tb);;
    val fus_list : 'a list -> 'a list -> 'a list = <fun>
```

Application :

```
    # fus_list [1;5;7][2;3;4];;
    - : int list = [1;2;3;4;5;7]
```

2.
```
# let rec impair_list l = match l with
  |[] ->[]
    |[a] ->[]
      |a::b::rl -> b::(impair_list rl);;
    val impair_list : 'a list -> 'a list = <fun>
```

Application :

```
    # impair_list [1;2;3;4;5;7];;
    - : int list = [2;4;7]
```

3.
```
# let rec pair_list l = match l with
    |[]->[]
    |a::ra-> a::(impair_list ra);;
    val pair_list : 'a list -> 'a list = <fun>
```

Où encore:

```
    # let rec pair_list l = match l with
```

```
          |[]->[]
          |[a] ->[a]
          |a::ra-> a::(pair_list ra);;
     val pair_list : 'a list -> 'a list = <fun>
```

Application :

```
     # pair_list [1;2;3;4;5;7];;
- : int list = [1;3;5]
```

4.
```
# let rec fus_tri l = match l with
    |[]->[]
    |[a]->[a]
    | _ ->let p = pair_list l in
         let q = impair_list l in
         fus_list (fus_tri p) (fus_tri q);;
    val fus_tri : 'a list -> 'a list = <fun>
```

Application :

```
     # fus_tri [1;9;4;0;2;5];;
     - : int list = [0;1;2;4;5;9]
```