Université Paris 6
Licence d'informatique

Module de Programmation
ANNÉE 2000-2001

Travaux Dirigés n° 11

Exercice 1 On considère la déclaration :

```
let rec it = function f -> function a -> function n ->
  if n=0 then a else (f (it f a (n-1)));;
```

1 On rappelle que `map : ('a -> 'b) -> 'a list -> 'b list`, quel sont le type et la valeur dans un environment E de `it (map (fun x -> x + 3))` ? Que calcule cette fonction ?

2 On peut définir l'addition de n à m comme l'opération d'ajouter n fois 1 à m. En utilisant la fonction `it`, définissez une fonction `add` qui calcule la somme d'un entier `n` à un entier `m`.

3 De même, la multiplication de n par m peut s'écrire comme l'addition de m, n fois à lui-même. En supposant définie la fonction d'addition `add`, définissez la fonction de multiplication.

4 En utilisant le même principe, définissez la fonction d'exponentiation m^n.

5 Écrivez une fonction qui rajoute un caractère devant une chaîne de caractères (*cf* bibliothèque `string`). Utilisez alors la fonction `it` pour définir une fonction `mk_string`, de type `int -> string` créant une chaîne de longueur spécifiée par l'argument et ne contenant que le caractère `' '` (par exemple `(mk_string 3)` vaudra `"   "`).

Exercice 2 Quels sont le type et la valeur dans un environment E de la fonction :

```
let rec rec_schema = function (b,d,f,a,n) ->
  if (b n) then a
  else f (rec_schema (b,d,f,a,(d n))) ;;
```

En déduire la fonction `it` de l'exercice précédent.

Exercice 3 On considère la déclaration suivante :

```
let rec foo = function l ->
  if (l = [])
  then []
  else
    let r = (foo (tl l)) in
      if (r = [])
      then
         [hd l]
      else
        (hd r) :: (foo ((hd l) :: (foo (tl r))))
```

En notant $E_{\texttt{foo}}$ l'environnement de définition de `foo`, détaillez l'évaluation de `foo`([1 ;2]). Que calcule `foo` ? Combien d'appels récursifs ferait `foo`([1 ;2 ;3]) ? Réécrivez la fonction `foo` en utilisant des constructions `match ... with ...`.

Exercice 4 Soit le programme suivant :

```
type 'a untyp =  Vid | N of 'a * ('a list) ;;

let rec une_fct = function h -> function un_elt -> match un_elt with
 | Vid -> []
 | N(s, []) -> [h(s)]
 | N (s, t :: r) -> h(t) :: (une_fct  h (N(s,r)) ) ;;

une_fct (function x -> x+3) (N(5, [10;11]));;
```

1 Que vaut `(une_fct (function x -> x+3) (N(5, [10 ;11])) )` ? Définissez le calcul effectué par `une_fct`, à l'aide d'une famille d'équations.

2 Définissez une fonction récursive terminale qui retourne le même résultat que la fonction `une_fct`.

Exercice 5 On considère les déclarations suivantes :

```
exception pas_grave of int ;;
exception grave of string ;;

let pseudo_id = function x ->
    if (x < 0)
    then raise(pas_grave(-x))
    else if (x = 0)
         then raise(grave "Argh !!!")
         else x ;;
```

Quelles sont les valeurs de `pseudo_id(-4)`, `pseudo_id(0)` et `pseudo_id(2)` ? On ajoute la déclaration :

```
let f = let val = 10
        in
        function x ->
          try (let val = 100 in pseudo_id(x)-val) with
          |  pas_grave n -> if (n < 0)
                            then raise(grave "impossible ?")
                            else n + val
          | grave "Argh !!!" -> raise(pas_grave(10))
          | grave s -> raise(grave "possible ?") ;;
```

Que donnent les évaluations `f(-100)`, `(0)`, `f(100)`, `f(grave "Ouf !!!")` et `f(raise(grave "Ouf !!!"))` ?