UNIVERSITÉ D'ORLÉANS **DEUG MIAS**
IF 3.01 - Informatique **Année 2001-2002**

Feuille d'exercices n° 7

Exercice 1

```
# let ls x = let y = sin x in
        if y>0. then log y
                 else failwith ("Argument invalide");;
val ls : float -> float = <fun>
```

Noter qu'ici, la fonction `log` ne provoque pas une erreur si son argument est négatif, elle dit que ce nombre n'existe pas (`nan` signifie « not a number »). Pour zéro, CAML renvoie $-\infty$:

```
# log 0.;;
- : float = -inf
# log (-.1.);;
- : float = nan
```

Exercice 2

```
# let divent x y = try x/y with
        Division_by_zero -> 0;;
val divent : int -> int -> int = <fun>
# divent 5 0;;
- : int = 0
# divent 57 19;;
- : int = 3
```

Exercice 3

```
# let rec map_succeed f l = match l with
        [] -> []
        | a::m -> try (f a)::(map_succeed f m) with
                _ -> (map_succeed f m);;
val map_succeed : ('a -> 'b) -> 'a list -> 'b list = <fun>
# let liste = [-2;-1;0;1;2];;
val liste : int list = [-2; -1; 0; 1; 2]
# map_succeed (function x -> 1/x) liste;;
- : int list = [0; -1; 1; 0]
```

Exercice 4

1.
```
# let rec nieme liste n = match (n,liste) with
          (_, [])           -> failwith("liste trop courte")
        | (i, _) when (i<=0) -> failwith("indice incorrect")
        | (1,e::m)          -> e
        | (i,e::m)          -> nieme m (i-1);;
val nieme : 'a list -> int -> 'a = <fun>
# nieme [1;2;3;4;5;6;7;8;9] 5;;
- : int = 5
# nieme [1;2;3;4;5;6;7;8;9] 10;;
Uncaught exception: Failure "liste trop courte".
```

2.
```
# let rec nieme liste n = print_string("Entr\'ee dans nieme");
                          print_newline();
      match (n,liste) with
          (_, [])           -> print_string "exception : liste trop courte";
                               print_newline();
                               failwith("liste trop courte")
        | (i, _) when (i<=0) -> print_string "exception : indice incorrect";
                               print_newline();
                               failwith("indice incorrect")
        | (1,e::m)          -> print_string "sortie avec succès";
                               print_newline();
                               e
        | (i,e::m)          -> print_string "appel à nieme avec n=";
                               print_int (i-1);
                               print_newline();
                               let res = nieme m (i-1) in
                               print_string "fin de nieme avec n=";
                               print_int (i-1);
                               print_newline();
                               res;;
val nieme : 'a list -> int -> 'a = <fun>
# nieme [1;2;3;4;5;6;7] 4;;
Entr\'ee dans nieme
appel à nieme avec n=3
Entr\'ee dans nieme
appel à nieme avec n=2
Entr\'ee dans nieme
appel à nieme avec n=1
Entr\'ee dans nieme
sortie avec succès
fin de nieme avec n=1
fin de nieme avec n=2
fin de nieme avec n=3
- : int = 4
```

```
# nieme [1;2;3;4;5] 6;;
Entr\'ee dans nieme
appel à nieme avec n=5
Entr\'ee dans nieme
appel à nieme avec n=4
Entr\'ee dans nieme
appel à nieme avec n=3
Entr\'ee dans nieme
appel à nieme avec n=2
Entr\'ee dans nieme
appel à nieme avec n=1
Entr\'ee dans nieme
exception : liste trop courte
Uncaught exception: Failure "liste trop courte".
```

3.
```
# exception Resultat of int;;
exception Resultat of int
# let nieme liste n =
    let rec aux liste n = print_string("Entr\'ee dans nieme");
                          print_newline();
        match (n,liste) with
          (_, [])            -> print_string "exception : liste trop courte";
                                print_newline();
                                failwith("liste trop courte")
        | (i, _) when (i<=0) -> print_string "exception : indice incorrect";
                                print_newline();
                                failwith("indice incorrect")
        | (1,e::m)           -> print_string "sortie avec succès";
                                print_newline();
                                raise (Resultat e)
        | (i,e::m)           -> print_string "appel à nieme avec n=";
                                print_int (i-1);
                                print_newline();
                                let res = aux m (i-1) in
                                print_string "fin de nieme avec n=";
                                print_int (i-1);
                                print_newline();
                                res
    in
    try aux liste n with
        Resultat r -> print_string "Réception du résultat";
                      print_newline();
                      r;;
val nieme : int list -> int -> int = <fun>
```

```
# nieme [1;2;3;4;5] 4;;
Entr\'ee dans nieme
appel à nieme avec n=3
Entr\'ee dans nieme
appel à nieme avec n=2
Entr\'ee dans nieme
appel à nieme avec n=1
Entr\'ee dans nieme
sortie avec succès
Réception du résultat
- : int = 4
# nieme [1;2;3;4;5] 6;;
Entr\'ee dans nieme
appel à nieme avec n=5
Entr\'ee dans nieme
appel à nieme avec n=4
Entr\'ee dans nieme
appel à nieme avec n=3
Entr\'ee dans nieme
appel à nieme avec n=2
Entr\'ee dans nieme
appel à nieme avec n=1
Entr\'ee dans nieme
exception : liste trop courte
Uncaught exception: Failure "liste trop courte".
```

De cette façon, on n'est pas obligé d'attendre que tous les appels de fonction intermédiaires se « terminent », on sort du programme dès que l'on a trouvé le résultat.

4.
```
# let longueur liste =
        let rec long l n = match l with
                  []   -> raise (Resultat n)
                | a::m -> long m (n+1)
        in try long liste 0 with
                  Resultat r -> r;;
val longueur : 'a list -> int = <fun>
# longueur [1;2;3;4;5;6;7;8];;
- : int = 8
```