TD 1 Programmation fonctionnelle

1 Environnement de travail

1. Installez ocaml sur votre machine personelle en suivant les instructions :
   — sous Linux : `https://webusers.i3s.unice.fr/~elozes/enseignement/PF/install-linux.html`
   — sous Mac OS : `https://webusers.i3s.unice.fr/~elozes/enseignement/PF/install-mac-os.html`
   — sous Windows : `https://webusers.i3s.unice.fr/~elozes/enseignement/PF/install-windows.html`
2. Logiciels conseillés
   — `https://try.ocamlpro.com/` : un toplevel en ligne, rien à installer pour démarrer
   — la distribution officielle ocaml (`https://www.ocaml.org/`) : le compilateur, le toplevel, et d'autres outils standards
   — `https://github.com/ocaml-community/utop` : un toplevel alternatif à ocaml
3. Comment lancer un toplevel ?
   (a) depuis un navigateur web, sans même avoir installé OCaml, grâce à TryOCaml https ://-try.ocamlpro.com.
   (b) depuis un terminal, il suffit de taper la commande ocaml.
   (c) en installant un toplevel non-officiel mais plus populaire : i) ocaml-top fait l'indentation automatique et la coloration syntaxique ii) utop offre la complétion automatique
   (d) Quelques remarques sur le toplevel
      — Avant d'afficher la valeur calculée, le toplevel affiche son type (int, string, float, ...)
      — On peut écrire sur plusieurs lignes, c'est ;; qui marque la fin de la saisie
      — La commande `#quit` interrompe le toplevel
   (e) A la place du toplevel, vous pouvez compiler votre fichier :

```
$ ocamlopt -o execfile file.ml
$ ./execfile
```

2 Types et fonctions

Déterminez, si possible, le type des fonctions suivantes :

```
1. let f1 x = if x < 0 then -x else x
2. let f2 x = x / 0
3. let f3 x = (x + 1, x +. 1.0)
4. let f4 x y z = if x then y + 1 else z
5. let f5 x y z = if x then y else z
6. let f6 x y = x y
7. let rec f7 x = f7 x
8. let rec f8 x = if true then f8 x else 0
```

Solution.

```
let f1 x = if x < 0 then -x else x;; (* int -> int *)

let f2 x = x /. 0.;; (* float -> float *)
le type existe mais ...

let f3 x = (x + 1, x +. 1.0);; (* impossible *)
x ne peut être à la fois un entier et un flottant

let f4 x y z = if x then y + 1 else z;; (* bool -> int -> int -> int *)
l'addition impose le type de y et du résultat
z doit avoir le même type que l'autre branche

let f5 x y z = if x then y else z;; (* bool -> 'a -> 'a -> 'a,
même cas que précédement avec l'addition en moins,
il n'y a plus de contraintes de type sur y

let f6 x y = x y;; (* ('a -> 'b) -> 'a -> 'b, *)
le type de l'application de fonction

let rec f7 x = f7 x;; (* 'a -> 'b, *)
on a aucune information à exploiter, cette fonction est une boucle infinie

let rec f8 x = if true then f8 x else 0;; (* 'a -> int*)
même si la deuxième branche ne peut être atteinte,
elle impose son type à la première branche
```

□

3 Fonctions simples

Ecrivez les fonctions :

1. `max_of_three x y z` le maximum de 3 arguments

   Solution.

   ```
   let max_of_three x y z =
     if x>y then
       if x > z then x
       else z
     else if y > z then y
     else z;;
   ```

   □

2. `is_even x` vérifie si l'argument est paire

   Solution.

   ```
   let is_even x =
     if x mod 2 = 0 then true
     else false
   ```

   □

3. `is_leap_year y` vérifie si une année est bissextile. Idée :

```
if (year is not divisible by 4)
    then (it is a common year)
    else if (year is not divisible by 100)
             then (it is a leap year)
             else if (year is not divisible by 400)
                      then (it is a common year)
                      else (it is a leap year)
```

Solution.

```
let is_leap_year y =
  if y mod 4 <> 0 then false
  else if y mod 100 <> 0 then true
  else if y mod 400 <>0 then false
  else true
```

□

4 Fonctions Récursives

Ecrivez les fonctions :

1. `sum_range a b` La somme de tous les entiers compris entre a et b. Exemple : `sum_range 1 5` doit être égal à $1 + 2 + 3 + 4 + 5 = 15$.

Solution.

```
let rec sum_range a b =
  if a > b then
    0  (* base case: empty range *)
  else
    sum_range a (b-1) + b
```

□

2. `num_digits x` Le nombre de chiffres dans l'entier x.

Solution.

```
let rec num_digits n =
  if abs n < 10 then
    1
  else
    num_digits (n/10) + 1
```

□

3. `nth_digit i x` i-ième chiffre (décimal) de l'entier x. Supposons que les chiffres soient indexés de droite à gauche, en commençant par i=1. La fonction doit renvoyer 0 pour les valeurs non valides de i. Examples :

```
nth_digit 1 7025 = 5
nth_digit 5 7025 = 0
```

Solution.

```
let rec nth_digit i n =
  if i = 0 then 0
  else if i=1 then
    if abs n < 10 then abs n
    else 0
  else nth_digit (i-1) (n/10)
```

□

Exercice à la maison : comment changer le code si les chiffres sont indexés de gauche à droite ?

4. `binom n k`

Implementez le coefficient binomial C(n,k), de manière récursive en utilisant l'identité de Pascal :

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

Cette fonction est définie uniquement pour $0 \leq k \leq n$. Les cas de bases sont les valeurs de $k = n$ et $k = 0$ où la fonction renvoie 1.

Solution.

```
let rec binom n k =
  if k < 0 || k > n then 0
  else if k = 0 || k = n then 1
  else binom (n-1) (k-1) + binom (n-1) k
```

□