TP 1 d'option informatique - MPSI - 29/01/2021

- Toutes les fonctions doivent être testées, même lorsque ce n'est pas précisé.

Exercice 1. Exemples d'expressions en Ocaml

1. Sur une feuille, donner le type des expressions suivantes. Lesquelles sont équivalentes ? Vérifier avec Caml.

   `(1,2,3,4)` `1,2,3,4` `(1,2),3,4` `(1,2),(3,4)`

2. Pour chaque type ci-dessous, donner un exemple d'expression de ce type. Vérifier à l'aide d'Ocaml :

   `int*float*string` `(int*string)*(float*string)`
   `(int*(float*string))*int` `(char*int)*(string*float)`

3. Sur une feuille, prévoir la réponse d'Ocaml lors de l'évaluation des expressions arithmétiques suivantes. Le vérifier sur machine.

```
42;;
1-23;;
4 *
  (4+4);;
8/4;;
8/3;;
```

```
1.414;;
2 * 1.414;;
2. *. 1.414;;
2 * (* commentaire *) 3;;
(* Commentaire *) 0+0;;
```

4. Même question pour des expressions utilisant des chaînes de caractère :

```
"Hello World !";;
"Bonjour", 0;;
"Bonjour, 0";;
"Bonjour le Monde !\n";;
print_string "Bonjour le Monde !\n";;
```

5. Même question pour des expressions utilisant des booléens :

```
true;;
true && false;;
1+2+3 = 6;;
if 0 = 1 then true else 0;;
if 0 = 1 then "true" else "false";;
```

6. Même question en définissant des variables locales et globales :

```
let a = 2;;
a;;
let b = a + 2 * a;;
b;;
let a = a + a + a + a;;
a;;
b;;
```

```
let c = 2 in a * b * c;;
c;;
let a = -4 in a * 2;;
a;;
let a,b = 2,3 in a*b;;
a,b;;
```

```
let x =
  let x = 2 and y = 7 in
  let x = x + y and y = x - y in
  x - y;;
let y = 2 in x - y*y;;
x;;
y;;
```

Exercice 2. Exemples de fonctions en Ocaml

1. Sur une feuille, donner le type de chacune des fonctions suivantes ainsi que le résultat des appels à ces fonctions. Vérifier votre réponse avec Ocaml.

```
let carre n = n*n;;
carre 5;;
let mult n m = n*m;;
mult 7 9;;
let triple = mult 3;;
triple 4;;
```

```
let mult2 (n,m) = n*m;;
mult2 (7,9);;
let triple2 m = mult2 (3,m);;
```

2. Même question avec :

```
let f x = x = 0 = true;;
f 0;;
f 1;;
let f x = x = (0 = true);;
```

```
let inv_arg f x y = f y x;;
let g x y = x,y;;
inv_arg g 0 true;;
```

3. Sans forcer le typage, trouver une fonction dont le type est :

`int -> float`, `int*int -> bool`,
`int -> int -> bool`, `int -> int -> int`,
`'a -> 'a -> 'a`, `('a -> 'b -> 'c) -> ('a -> 'b) -> 'a -> 'c`.

Exercice 3. Fonctions élémentaires

1. Écrire une fonction `signe_int: int -> string` qui prend en entrée un entier n, qui renvoie la chaîne de caractères `"positif"` lorsque n est strictement positif, `"negatif"` lorsque n est strictement négatif et affiche un message d'erreur lorsque n vaut 0.
2. De même, écrire une fonction `signe_float: float -> string`. En particulier, testez l'évaluation de `signe_float 0.5` et de `signe_float 0.`.
3. À l'aide d'un `if then else`, écrire deux fonctions qui calculent la valeur absolue. L'une pour les entiers et l'autre pour les réels.

Exercice 4. Ordre lexicographique

Soit $a_1, b_1, a_2, b_2 \in \mathbb{Z}$. Le couple $(a_1, b_1) \in \mathbb{Z}^2$ est inférieur ou égal au couple $(a_2, b_2) \in \mathbb{Z}^2$ pour l'ordre lexicographique lorsque $a_1 < a_2$ ou bien lorsque $a_1 = a_2$ et $b_1 \leqslant b_2$.
Écrire une fonction `est_plus_petit: int * int -> int * int -> bool` qui renvoie `true` si et seulement si le premier couple est inférieur ou égal au deuxième couple pour l'ordre lexicographique.

Exercice 5. Racines d'un polynôme du second degré

Écrire une fonction qui prend en entrée trois réels a, b et c, et renvoie les deux réels racines du polynôme $aX^2 + bX + c$. Faire les tests ci-dessous :

```
solution 1. 2. 1.;;     solution 1. 2. 1.1;;     solution 1. (-1.) (-1.);;
solution 0. 0. 0.;;     solution 0. 0. 1.;;      solution 0. 1. (-5.);;
```

Exercice 6. Fonctions avec fonctions en paramètre

1. Écrire une fonction `fonction1: (float -> float) -> float` qui associe à une fonction f la valeur :
$$\frac{f(5) + f(-5)}{2}$$
Testez votre fonction.
2. Écrire une fonction `fonction2: (float -> float) -> float -> float` qui prend en entrée une fonction f et un flottant x et renvoie $f(x)^2$.
3. Soit `fonction3` la fonction qui prend en entrée une fonction f`: float -> float` et qui renvoie f^2. Quel est le type de `fonction3` ? Écrire `fonction3`.
4. Soit `fonction4` la fonction qui prend en entrée une fonction f`: float -> float` et qui renvoie la fonction $x \mapsto f(2x)$. Quel est le type de `fonction4` ? Écrire `fonction4`.
5. Écrire une fonction `somme` qui prend en entrée deux fonctions `f: int -> int` et `g: int -> int`, et renvoie la somme de ces deux fonctions. Tester votre fonction.
6. Écrire une fonction `produit` qui prend en entrée deux fonctions `f: int -> int` et `g: int -> int`, et renvoie le produit de ces deux fonctions.
7. Écrire une fonction `produit_ext: (int -> int) -> int -> int -> int` qui prend en entrée une fonction f ainsi qu'un entier a, et renvoie la fonction $a \times f$.
8. Écrire une fonction `derive: (float -> float) -> float -> float -> float` qui prend en entrée une fonction f et un réel ϵ et renvoie la fonction :
$$x \mapsto \frac{f(x + \epsilon) - f(x)}{\epsilon}.$$
9. Sur feuille, donner une approximation de la valeur de l'expression suivante puis vérifier avec Ocaml.

```
let f x = x*.x*.x +. 2.*.x +. 1. in
    derive f (10.**(-5.)) 2.;;
```

Exercice 7. Déterminer le type d'une fonction

On considère les trois fonctions suivantes :

```
let fonction1 a b c = a (b c);;
let fonction2 a b c = (a b) c;;
let fonction3 a b c = a b c;;
```

1. Donner le type de ces trois fonctions puis vérifier avec Caml.
2. Trouver des valeurs pour `a`, `b` et `c` auxquelles on peut appliquer `fonction1`.
3. Trouver des valeurs pour `a`, `b` et `c` auxquelles on peut appliquer `fonction2`.
4. Trouver des valeurs pour `a`, `b` et `c` auxquelles on peut appliquer `fonction3`.

Exercice 8. Maximum de trois éléments

En Caml, la fonction `max: 'a -> 'a -> 'a` renvoie le maximum de ses deux arguments. On souhaite écrire une fonction `max3: 'a -> 'a -> 'a -> 'a` qui renvoie le maximum de ses trois arguments.

1. Sur une feuille de papier puis en vérifiant avec Caml, donner et expliquer le type de la fonction suivante :

```
let max3_fail x y z = max max x y z;;
```

2. Écrire une fonction `max3` correcte. Testez votre fonction.
3. Sur une feuille de papier puis en vérifiant avec Caml, donner et expliquer le type de la fonction suivante :

```
let max3_fail_bis a b c = max a (max b) c;;
```