TP 1 - Premiers pas -

Exercice 1. Au *top-level*...

Le but de cet exercice est de vous familiariser avec l'interpréteur `ocaml` et les types de base. Vous allez pouvoir entrer des **expressions** que le top-level évaluera.

1. Lancer le top-level et évaluer l'entier 33. Recommencer. Pouvez-vous utiliser les flèches directionnelles ?
2. Quitter le top-level et le relancer avec la commande `rlwrap`. Refaire la question précédente. Quelles sont les informations que vous donne le top-level ?
3. Déterminer le type des expressions suivantes (lorsque ce sont des expressions caml). Vérifier en utilisant le top-level et commenter.

   `2 2.0 2,0 2;0 a 'a' "a" true () [] [1] [1,true] [1;true]`

   Pour retrouver les types de base en caml, vous pouvez visiter la page `http://caml.inria.fr/pub/docs/manual-ocaml-4.00/manual010.html`.
4. Lorsque c'est possible, donner des exemples d'expressions ayant les types suivants. Commenter.
   (a) `int * float`
   (b) `(int, float)`
   (c) `string list`
   (d) `bool list * string`
   (e) `'a * int`
5. Lorsque c'est possible, faire les calculs suivants. Commenter.
   (a) `1+2`
   (b) `1.1+2.2`
   (c) `1.1+2`
   (d) `2/3`
   (e) `7 mod 2`
   (f) `7. mod 3.`
   (g) 2^3
   (h) `2=3`
   (i) `'a'='b'`
   (j) `"a"='a'`
   (k) `"a"='a'`
   (l) `not 1=0`
   (m) `not (1=0)`
6. Utiliser le top-level pour tester si les opérateurs booléens ET et OU sont séquentiels.
7. Utiliser le top-level pour mettre en évidence la différence entre `=` et `==`, ainsi que le polymorphisme des opérateurs de comparaison.

Exercice 2. Liaisons

Dans le paradigme fonctionnel, la notion de variable (et donc d'affectation) n'existe pas. Pour nommer des expressions, on utilise le mécanisme de **liaison**. Attention, une liaison locale est une **expression** alors qu'un liaison globale ne l'est pas.

Pour une liaison globale, on utilise la syntaxe suivante :

```
let nom = expr ;;
```

Pour une liaison locale, on utilise la syntaxe suivante :

```
let nom = expr1 in expr2;;
```

Dans les deux cas, le mot-clé `and` permet de faire des liaisons simultanées.

1. Calculer le volume d'un cylindre de rayon r et de hauteur h. Vous utiliserez une liaison globale pour π et des liaisons locales pour r et h.
2. Peut-on écrire les morceaux de code suivants ? **Réfléchissez avant de tester !**

```
let a=1 in a+2;;
a+3;;

let b=5;;
let b=5.5;;

let c=1;;
let d=c;;
c+d;;

let e=1 let f=e;;

let g=1 and h=g;;

let a=1 in let b=a;;

let a=1 in let b=a in b;;
```

3. Qu'affiche le code suivant ? **Réfléchissez avant de tester !**

```
let a=1;;
let a=1.2 in a;;
a;;

let a=1
  in let a=2 and b=a
    in a+b;;
```

Exercice 3. Alternative

La structure de contrôle conditionnelle (ou alternative) en ocaml a la syntaxe suivante :

```
if expr1 then expr2 else expr3
```

Attention, il s'agit d'une expression, elle a donc un type et une valeur (c'est l'équivalent du if ternaire du C). L'expression `expr1` est de type *bool* et les expressions `expr2` et `expr3` doivent être du même type (qui est également le type de l'expression complète).

1. Calculer le maximum de `a` et `b`.
2. Calculer le minimum de `a`, `b` et `c`.
3. Tester si l'entier `a` est pair. Si oui, donner sa valeur, sinon "odd".
4. Que pensez vous du code suivant ? **Réfléchissez avant de tester !**

```
if a<10 then let b="small" else let b="large";;
```

5. Étant donné un entier `a` et en utilisant une conditionnelle, donner le code permettant d'obtenir $b = \lceil a/2 \rceil$.
6. Étant donnés trois entiers `a`, `b` et `c`, calculer l'expression suivante sans faire deux fois le même test ou le même calcul :

$$\begin{cases} min(a,b)^2 + 1 & \text{si } c \text{ est divisible par 3} \\ min(a,b)^2 & \text{sinon.} \end{cases}$$

Exercice 4. Fonctions

En caml, vous disposez d'un certain nombre de librairies prédéfinies (modules). Vous trouverez la documentation du module `Pervasives` qui est chargé par défaut :
`http://caml.inria.fr/pub/docs/manual-ocaml/libref/Pervasives.html`
et les autres librairies standard :
`http://caml.inria.fr/pub/docs/manual-ocaml/stdlib.html`.
Pour *appliquer* une fonction, il suffit de donner son expression suivie des arguments **sans parenthèses**. Par exemple :

```
int_of_float 3.4;;
mod_float 3.4 2.0;;
(+) 1 2;;
String.make 5 'A';;
(fun x -> x+1) 4;;
```

Enfin pour définir ses propres fonctions, on peut utiliser (entre autres) la syntaxe suivante :

```
let nom p1 ... pn = expr
```

1. Écrire et tester une fonction `average` qui calcule la moyenne de trois entiers. Cette fonction peut-elle être utilisée avec des flottants ?
2. Écrire une fonction `implies` qui prend deux expressions booléennes `a` et `b` et renvoie vrai si `a` implique `b` au sens logique.
3. Écrire une fonction `inv` qui prend un couple et renvoie le couple inversé. Faire deux versions : l'une en utilisant `fst` et `snd` et l'autre sans.
4. Écrire la même fonction, mais uniquement pour un couple d'entiers.
5. Écrire la fonction sans argument `f_one` qui renvoie la constante 1. Vérifiez qu'elle a bien le type attendu.
6. Qu'affiche le code suivant (pour chaque ligne) ? **Réfléchissez avant de tester !**

```
let m = 3;;
let f x = x;;
let g x = x+m;;
f 4;;
g 4;;
let m = 5;;
g 4;;
f m;;
```