info401 : Programmation fonctionnelle
TD 1 : premières expressions en Caml

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/
wiki : http://www.lama.univ-savoie.fr/wiki

Exercice 1 : expressions simples, types simples

Question 1. Tartempion lance l'interprète Caml et tape les lignes suivantes.

```
# 401 / 10 ;;
# 10 + 1.5 ;;
#   "Bonjour, "
  ^ "je m'appelle Bob."  ;;
# "Je m'appelle" ^ (2*"to") ;;
# let pi = 3.14159 ;;
# let x = 3.0 *. pi /. 2.0 ;;
# if (x < 2*pi) then x
                else (x-2*pi) ;;
# let n = 42
  ( if (n > length("anticonstitutionalisement"))
      then 3+length("Bob")
      else 0 )
  + int_of_float (2. *. pi)  ;;
# if ( if n<0 then true else (pi > 22.0 /. 7.0) )
    then "Toto"
    else "Tata" ;;
```

Quelles sont les expressions valides, et quelles sont leurs types et leurs valeurs ?

Pour les expressions invalide, comment les corrigeriez-vous ?

Rappels : nous avons vu plusieurs utilisation du mot clé `let` en cours :

- une définition simple : `let` *var* `=` *expr*
- une définition multiple : `let` var_1 `=` $expr_1$ `and` var_2 `=` $expr_2$
- une définition récursive : `let rec` *var* `=` *expr*
- une définition locale : `let` *var* `=` *expr* `in` *expr*′

Une définition locale peut elle aussi être simple, multiple ou récursive...

Chaque définition peut modifier l'environnement en rajoutant des liaisons entre des noms de variables et des valeurs...

Question 2. Quelles sont les différences entre les expressions Caml suivantes, et que se passe-t'il quand on les donne à l'interprète Caml :

```
# 2 + 1 + 3 ; ;
# let n = 2 + 1 + 3 ; ;
# let rec n = 2 + 1 + 3 ; ;
```

Question 3. Décrivez l'environnement après chaque ligne :

```
# let a = 1
  let b = 2
  let c = 3 ;;
```

```
# let b = a+2 ;;
# let c = a+b+c ;;
# let a = 0 and d = 1
  in  a + b + d ;;
# let e = d + 1 ;;
# let l = length ;;
# let un_nombre = let e = 2.718 in
                  let l = 42 in
                  e + (float_of_int l) ;;
# "Salut" ^ " !" ;;
# #quit ;;
```

Question 4. En voulant définir deux valeurs, Tartempion écrit la chose suivante dans l'interprète Caml :

```
# let a=42 and b=a+1 ; ;
```

Que se passe-t'il, et pourquoi ?

Comment Tartempion aurait-il du s'y prendre ?

Question 5. Caml possède une fonction `max` de type `int -> int -> int` qui permet de trouver la plus grande de deux valeurs.

- Écrivez une fonction `maxPaire` dont le type est `int*int -> int` sans utiliser les opérateurs de comparaison.
- Écrivez une fonction `max3valeurs` dont le type est `int->int->int->int`

Exercice 2 : un peu de récursion

Question 1. Tartempion essaie de définir la suite (modélisée comme une fonction des naturels vers les naturels) :

$$u_1 = 0 \qquad u_{n+1} = 1 + u_n$$

Il commence par

```
# let rec u n =
    if (n < 2)
      then 0
      else u (n-1) ;;
```

mais après quelques tests, il se rend compte qu'il a oublié la partie "$1 + \ldots$". Il fait un deuxième essai :

```
# let u n =
    if (n<2)
      then 0
      else 1 + u (n-1) ;;
```

Que se passe-t'il ?

Question 2. Ecrivez une fonction `fact` pour calculer : $!n = 1 \times 2 \times \ldots \times n$.

Vous devrez écrire votre fonction de manière récursive, sans utiliser de boucle `for` comme vous le feriez en Pascal.

Question 3. Écrivez une petite fonction `somme` qui calcule la somme $1 + 2^2 + 3^2 + 4^2 + \ldots + n^2$.

Question 4. Vous avez besoin de calculer les sommes suivantes :

- $\sum_{i=1}^{n} i^3$,
- $\sum_{i=1}^{n} i^i$,
- $\sum_{i=1}^{n} (3i^2 + 2i)$.

Plutôt que de réécrire la fonction précédente trois fois avec quelques modifications, que proposez-vous ?

Écrivez la fonction correspondante, et donnez son type.