TP 1

Environnement de travail

Consulter : **Mise en place de l'environnement de travail**

TRÈS IMPORTANT : vérification des programmes

Testez systématiquement tous les exemples Ocaml qui ont été donnés en TD. Testez, avec encore plus de soin, les petits programmes Ocaml que vous avez vous-même écrits. Vous devez considérer comme faux un programme que vous n'avez pas testé.

Premiers pas

Evaluer les expressions donner dans les exercices ci-dessous et expliquer le résultat de l'interpréteur Ocaml.

Exercice 1

```
#1+3*5;;

#3=5;;

#"Bonjour";;

#2345;;

#3456789009;;

# 10.6 * 3;;

#let pi = 3.14159;;

#let rayon = 10.0;;

# pi *. rayon *. rayon ;;

# let circonf = pi *. rayon *. rayon ;;

#circonf ;;

# 3 * circonf;;

#10 / 4;;

#10.0 /. 4.0;;

#let x=1 and y= 2;;

#let x=y;;
```

```
# x=y;;

#let x = 2 in let y = 2 in x*x+y;;
```

Exercice 2

```
# let carre x = x * x;;

# carre 4.0;;

#carre (4+ 7);;

#carre 4 + 7;;

# carre (carre 4);;

# let f x = x + 1;;

# f 1;;

# f (f 1);;

# let f = function x -> function y -> x + y;;

# f 1 2;;

#  let g = function x -> (f x);;

# g 1 2;;
```

Expressions bizarres

Les expressions suivantes sont quelquefois correctes, quelquefois incorrectes et souvent surprenantes ! Tapez ces expressions, **observez** les résultats, notez-les et **expliquez** le résultat. Vous corrigerez les expressions rejetées par Caml en justifiant votre correction.

```
(* expression 1 *)
# let rac_pi = sqrt(22/7);;

(* expression 2 *)
# 1/2 = 0;;

(*expression 3 *)
# let pair = function n -> 2*(n/2) = n;;

(*expression 4*)
# let a = 2.0;;
let b = 1.0;;
let c = 1.0;;
# let delta = b*.b -. 4.0*.a*.c;;
# let r1 = if delta >= 0.0 then ((sqrt delta) -. b)/.a/.2.0
                           else "pas de racine"
  and r2 = if delta >= 0.0 then ((sqrt delta) +. b)/.a/.2.0
                           else "pas de racine";;
```

```
(* expression 5 *)
# let entier = function x -> float_of_int (int_of_float x) = x;;
# let oups = function num ->
               if (entier num)
                 then int_of_float num
                 else num;;

(* expression 6 *)
# let f = function n -> 2*n;;
# let g = function n -> n+n;;
# f = g;;
# (f 2) = (g 2);;

(* expression 7 *)
# let bencaalors =   function n -> if (exp (float_of_int n) *. (log 10.0)) <>
                     (exp (float_of_int n) *. (log 10.0)) +. 1.0
                   then "tout va bien"
                   else "euh ...";;
# bencaalors 8;;
# bencaalors 10;;
# bencaalors 40;;
```

Retour sur le problème de l'écrou (cf TD1)

Écrivez une fonction `volume` de trois arguments ($base, h$ et r) qui calcule le volume de l'écrou. Vous testerez avec différents paramètres, par exemple $base = 1.0, r = 0.5, h = 2.0$, $base = 10.0, r = 1.0, h = 10.0$, puis $base = 10.0, r = 0.0, h = 10.0$ et $base = 10.0, r = 1.0, h = 0.0$.

Comment déposer un fichier sur arche

La procédure pour déposer votre compte-rendu (uniquement sous la forme d'un fichier texte d'extension .ml) est la suivante :

1. Ouvrez un navigateur web
2. Allez sur le site `http ://arche.uhp-nancy.fr`
3. En haut à droite de la page (au dessus du choix de la langue), vous avez la possibilité de vous connecter.
   Faites le.
4. Vous êtes alors renvoyés sur *le service central d'authentification de l'Université*.
   Saisissez votre login et votre mot de passe.
5. Vous arrivez sur la liste des cours disponibles.
6. Cliquez sur la rubrique *Informatique*.
7. Cliquez sur *Algoritmique et Programmation Fonctionnelle (ELP-LCMIN1U2-0)*.
8. Dans le thème TP1, cliquez sur le devoir correspondant à votre groupe.
9. Cliquez sur *Parcourir*
10. Cherchez votre fichier TP1.ml (qui doit normalement se trouver dans le répertoire TP1). Ne vous occupez pas des fichiers d'extension .ml , il s'agit de copies de sauvegarde.
11. Cliquez sur *Déposer* ce fichier.
12. Vérifiez enfin que vous avez déposé le bon fichier.