1 Portée des variables et Fonctions

Le but de cet exercice est de manipuler la notion de portée en OCaml. Justifier successivement les résultats de la suite des expressions OCaml ci-dessous.

```
let a=1
  in let a=2 and b=a
    in a+b;;
(* justifier le resultat de l'evaluation:          *)

a;;
(* justifier le resultat de l'evaluation:          *)

let a = 42;;
(* justifier le resultat:          *)

let a = a / 2;;
(* justifier le resultat:  *)

let b x =
  let a = float_of_int a (* utilisation d'une definition locale *)
  in
  a ** x;;
(* justifier le resultat: *)

b 1.;;
(* justifier le resultat: *)

let a = -3;;
(* justifier le resultat: *)

b 2.;;
(* justifier le resultat: *)

let f1 x y = if x < 0 then x-y else x+y ;;
(* justifier le resultat: *)

 f1 a (int_of_float (b 5.));;
(* justifier le resultat: *)

f1 -4 a;;
(* justifier le resultat: *)

f1 (-4) a;;
(* justifier le resultat: *)

let f x = log (float_of_int x) /. log 2. ;;
(* justifier le resultat: *)

f f a;;(* justifier le resultat: *)
```

```
f (f a);;
(* justifier le resultat: *)
```

2 L'exception failwith

L'utilisation de `failwith` stoppe l'évaluation de l'expression en déclenchant ce que l'on appelle une *exception.* . Exemple :

```
#let a=9 and x=0 in a/x;;
Exception: Division_by_zero. (* exception déclenchée par l'interpréteur *)
#let a=9 and x=0 in
   if x=0 then failwith "erreur: division par zero"
   else a/x;;
Exception: Failure "erreur: division par zero"  (* message d'erreur *)
(* exception  déclenchée par la fonction *)
(* le message d'erreur est pris en compte par OCaml *)
```

Application
En utilisant la fonction OCaml `sqrt: float-> float`, écrire la fonction `rac: int -> float` qui prend en argument un entier strictement positif et renvoie sa racine carrée. Cette fonction devra déclencher une exception dans le cas où l'argument n'est pas positif.

3 Définitions locales dans des fonctions

On peut utiliser des définitions locales dans les fonctions par exemple la fonction `reste` :

```
# let reste x y =
let quotient = x / y in
x - y * quotient;;
val reste : int -> int -> int = <fun>
```

3.1 Un jeu

La fonction OCaml `Random.int: int -> int = <fun>` prend en argument un entier n et renvoie un entier tiré au hasard compris entre 0 (inclu) et $n-1$ (inclu). Par exemple : `Random.int 4` renverra au hasard l'une des valeurs suivantes : $0, 1, 2$ ou 3.
En utilisant `Random.int` écrire une fonction `deviner: int -> bool` qui prend en argument un entier x strictement positif et compare x à un entier y tiré au hasard compris entre 1 (inclu) et 10 (inclu) , et renvoie la valeur de vérité de $x = y$. Il faudra déclencher une exception si x n'est pas positif.

3.2 Un calcul

Écrivez une fonction f de profil `float -> float` qui calcule $f(x) = e^{cos^2 x+2} - 10$ où le calcul de $cos\ x$ sera exprimée par une définition locale.

4 Un problème de représentation

Le but des deux exercices ci-dessous est *d'implémenter* par des fonctions des correspondances données dans une représentation tabulaire.

1. La correspondance ci-dessous donne le nombre de solutions réelles d'un équation du second degré $ax^2 + bx + c = 0$. Écrire une fonction `nombre_sol: int -> int -> int -> int` qui prend en argument les coefficients a, b, c et retourne le nombre de solutions.

Discriminant	nombre de solutions
positif	2
négatif	0
nul	1

   La valeur du discriminant devra être calculée localement (définir une variable locale `dicriminant` dans la fonction`nombre_sol`).

2. On dispose d'un tableau des tarifs d'affranchissements d'une lettre de moins de 100g pour les envois en France (NB : Les tarifs indiqués peuvent ne pas correspondre aux prix réels pratiqués par la Poste).

Poids jusqu'à	ordinaire	recommandé
20g	0.50	3.33
50g	0.78	3.61
100g	1.18	4.02

   Ce tableau indique une correspondance entre l'ensemble des couples (poids,formes d'envois) et l'ensemble des tarifs. Les formes d'envois seront codés par les entiers 1 et 2 pour `ordinaire` et `recommandé` respectivement.
   Écrire une fonction `tarif: float -> int -> float` qui prend en argument un poids appelé `poids`, une forme d'envoi `forme` et renvoie le tarif correspondant. Elle devra déclencher des exceptions pour les formes ou les poids inconnus.
   Les valeurs des tarifs d'envois ordinaire et recommandé devront être calculées localement (définir 2 variables locales `t_ordinaire` et `t_recommande` dans la fonction `tarif`)

5 Expressions Booléennes

Écrire une fonction `bissextile` de profil `int -> bool` qui détermine si une année est bissextile. Votre fonction ne devra pas utiliser la structure `if ... then ... else`.
On rappelle qu'une année est bissextile si elle est multiple de 4, mais pas de 100 à moins qu'elle soit aussi multiple de 400.