ENSIBS – 3ème année, spécialité cybersécurité (parcours CyberLog FISE) – Programmation fonctionnelle

TD/TP 1d : programmation fonctionnelle, OCaml

TRÈS IMPORTANT : vérification des programmes

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

Partie 1 : fonctions

Section 1.1 (exception « failwith »)

L'utilisation de « `failwith` » arrête l'évaluation de l'expression en déclenchant ce que nous appelons 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 *)
```

Exercice 1.1.1

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.

Section 1.2 (définitions locales dans des fonctions)

Nous pouvons 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>
```

Exercice 1.2.1 (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 :

1. prend en argument un entier `x` strictement positif,
2. compare `x` à un entier `y` tiré au hasard compris entre `0` (inclus) et `10` (inclus) , et
3. renvoie la valeur de vérité de `x=y`.

Il faudra déclencher une exception dans le cas où `x` n'est pas positif.

Exercice 1.2.2 (un calcul)

Écrivez une fonction `f` qui a la signature : « `float -> float` » et qui calcule $f(x) = e^{cos^2 x+2} - 10$ où le calcul de $\cos(x)$ sera exprimé par une définition locale.

Section 1.3 (un problème de représentation)

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

Exercice 1.3.1

La correspondance ci-dessous donne le nombre de solutions réelles d'une équation du second degré $a \cdot x^2 + b \cdot x + c = 0$. Ecrire 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 ($\Delta = b^2 - 4 \cdot a \cdot c$) devra être calculée localement (définir une variable locale discriminant dans la fonction « `nombre_sol` »).

Exercice 1.3.2

Nous disposons d'un tableau des tarifs d'affranchissements d'une lettre de moins de 100g pour les envois en France.
Remarque : 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 codées 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.

Partie 2 : stratégies d'évaluation

Exercice 2.1

Soit la fonction « `d1` » définie par « `let d1 x = print_string "On est dans d1 ! \n"; 2*x ;;` ». Expliquez ce que fait la fonction « `d1` ». Expliquez également son type.

Exercice 2.2

Définissez la fonction « `d` » par « `let d b x = b && ((d1 x) <= 10);;` », où la fonction « `d1` » est définie dans l'exercice 2.1.

- Expliquez ce que fait la fonction « `d` ». Expliquez également son type.
- Évaluez :
  - `d true 4;;`
  - `d true 6;;`
  - `d false 4;;`

  Observez ce qui est affiché par l'interpréteur dans chacun des cas, qu'en déduisez-vous sur le mécanisme d'évaluation d'un && (ET logique) ?

Exercice 2.3

Définissez la fonction « `e` » par « `let e x y = y;;` ».

- Expliquez ce que fait la fonction « `e` ». Expliquez également son type.
- À quoi vous attendez-vous en évaluant « `e (d1 1) 3 ;;` » », où la fonction « `d1` » est définie dans l'exercice 2.1. ? Vérifiez.