Programmation applicative – L2
TD 2 : Définitions de fonction, suite – Correction

G.Artignan {artignan@lirmm.fr} M.Lafourcade {lafourcade@lirmm.fr}
S.Daudé {sylvain.daude@univ-montp2.fr} A.Chateau {chateau@lirmm.fr}
B.Paiva Lima da Silva {bplsilva@gmail.com} C.Dony {dony@lirmm.fr}

1 Expressions et fonctions non récursives

Exercice 1 Triangles. *Écrire une fonction* `median` *qui, étant donnés trois nombres, donne le nombre médian. Écrire une fonction* `triangle?` *qui permet de savoir si trois nombres peuvent être les longueurs des côtés d'un triangle. De même, écrire les fonctions* `equilateral?`, `isocele?` *et* `rectangle?`.

```
(define median
  (lambda (x y z)
    (if (< x y)
        (if (< y z)
            y
            (if (< x z) z x))
        (if (< x z)
            x
            (if (< y z) z y)))))

(define triangle?
  (lambda (a b c)
    (and (<= a (+ b c)) (<= b (+ a c)) (<= c (+ a b)))))

(define equilateral?
  (lambda (a b c)
    (and (triangle a b c) (= a b c))))

(define isocele?
  (lambda (a b c)
    (and (triangle? a b c) (or (= a b) (= a c) (= b c)))))

(define rectangle?
  (lambda (a b c)
    (and (triangle? a b c)
         (or (= (* a a) (+ (* b b) (* c c)))
             (= (* b b) (+ (* a a) (* c c)))
             (= (* c c) (+ (* b b) (* a a)))))))
```

Exercice 2 Caractères. *Ecrire une fonction qui prend un paramètre en entrée, teste si c'est un caractère minuscule plus petit que* `#\m` *(avec la fonction* `char<?`*) et le cas échéant renvoie le même caractère en majuscule. On aura besoin des fonctions suivantes sur les caractères :* `char-alphabetic?` *qui prend en paramètre un caractère et renvoie vrai si c'est une lettre, faux sinon ;* `char-lower-case?` *qui prend en paramètre un caractère et renvoie vrai si c'est une minuscule, faux sinon, et enfin* `char-upcase` *qui prend en paramètre un caractère et renvoie le même caractère mais en majuscule, s'il existe.*

```
(define test
  (lambda (c)
    (if (and (char-alphabetic? c) (char-lower-case? c) (char<=? c #\m))
        (char-upcase c)
        (display "Impossible de réaliser le traitement"))))
```

Exercice 3 Chaîne de caractères. *Ecrire une fonction qui prend deux paramètres en entrée, teste si le premier paramètre est une chaîne de caractères, et si c'en est une, donne la lettre de numéro correspondant au deuxième paramètre. Rappel : on aura besoin des fonctions* `string?` *qui prend un paramètre et renvoie vrai si c'est une chaîne de caractères, faux sinon,* `string-length` *qui renvoie la longueur d'une chaîne de caractères passée en paramètre, et* `string-ref` *qui prend en paramètre une chaîne de caractères et un indice n, et renvoie le caractère d'indice n dans la chaîne.*

```
(define test-chaine
  (lambda (s n)
    (if (and (string? s) (< n (string-length s)))
        (string-ref s n)
        (display "Impossible de réaliser le traitement"))))
```

Exercice 4 Heures/Minutes/Secondes. *Écrire une fonction* `hms-vers-s` *qui permet de convertir un temps donné en heures, minutes, secondes, en temps exprimé en secondes.*

Écrire une fonction `hms?<` *qui prend 2 temps donnés sous la forme heures-minutes-secondes et qui donne la valeur* `#t` *si et seulement si le premier temps est strictement inférieur au second. On donnera deux variantes de cette fonction : une qui utilise la fonction précédente, et une qui ne l'utilise pas.*

```
(define hms-vers-s
  (lambda (h m s)
    (+ s (* 60 m) (* 3600 h))))

(define hms<?
  (lambda (h1 m1 s1 h2 m2 s2)
    (< (hms-vers-s h1 m1 s1) (hms-vers-s h2 m2 s2))))

(define hms2<?
  (lambda (h1 m1 s1 h2 m2 s2)
    (cond ((< h1 h2) #t)
          ((and (= h1 h2) (< m1 m2)) #t)
          ((and (= h1 h2) (= m1 m2) (< s1 s2)) #t)
          (else #f))))
```

Exercice 5 Tonneau dans une voiture. *Une voiture ne peut contenir que des tonneaux de forme cylindrique tels que la hauteur est inférieur à 1m, le rayon inférieur à 80cm et le volume inférieur à* $1m^3$. *Écrire une fonction* `voiture?` *qui, étant données 2 nombres représentant respectivement la hauteur et le rayon du tonneau en mètre, permet de savoir si le tonneau tient dans une voiture.*

```
(define voiture?
  (lambda (h r)
    (and (<= h 1) (<= r 0.8) (<= (* pi r r h) 1))))
```

Exercice 6 Viticulture. *Une entreprise viticole expédie une quantité* `q` *de vin de prix unitaire* `prix`. *Si la commande est de 600 Euros, ou plus, le port est gratuit, sinon il est facturé à 10% de la commande (dans le cas où la commande est inférieure à 600 Euros et que la commande plus le port est supérieur à 600 Euros, le prix est ramené à 600 Euros). Écrire une fonction* `commande` *permettant de calculer la somme à facturer.*

```
(define commande
  (lambda (q prix)
    (cond ((<= 600 (* q prix)) (* q prix))
          ((and (> 600 (* q prix)) (<= 600 (* q prix 1.1))) 600)
          (else (* q prix 1.1)))))
```

2 Exercice supplémentaire : Calcul de la date de Pâques

Le dimanche de Pâques se situe chaque année entre le 22 mars et le 25 avril, c'est le 1^{er} dimanche après la première pleine lune de l'équinoxe de printemps. La date de Pâques est calculée à partir du nombre d'or d'une année et de l'âge de la Lune appelé l'Epacte. Cet algorithme assez compliqué, caractéristique des religieux, renvoie néanmoins le numéro du jour à partir du 1^{er} mars qui correspond à la date de Pâques. Pour une `annee` exprimée en 4 chiffres posons :

- a = annee modulo 19
- b = annee modulo 4
- c = annee modulo 7
- d = (19a + 24) modulo 30
- e = (2b+4c+6d+5) modulo 7
- Le numéro du jour est alors : 22+d+e

Exercice 7 *Écrivez les fonctions suivantes :*

1. `numjour` *qui prend en argument* `annee` *et renvoie le numéro du jour à partir du* 1^{er} *mars.*
2. `traduit` *qui prend en argument le résultat de la fonction* `numjour` *et imprime la date du dimanche de Pâques :* $\Rightarrow$ `jour mois`.
3. *Enfin,* `paques` *qui imprime la date de Pâques pour une année donnée en argument.*

Remarque — On dispose en Scheme de la fonction `modulo` qui renvoie le reste d'une division. Ainsi pour $n_1 = n_2 n_3 + n_4$ la fonction `(modulo n1 n2)` retourne n_4 et la fonction `(quotient n1 n2)` retourne n_3.

```
(define (numjour annee)
    (+ 22 (modulo (+ (* 19 (modulo annee 19)) 24) 30)
          (modulo (+ (* 2 (modulo annee 4))
                     (* 4 (modulo annee 7))
                     (* 6 (modulo (+ (* 19 (modulo annee 19)) 24) 30)) 5) 7)))
```

Ok dès qu'on aura vu `let` ce sera plus facile!!

```
 (define (numjour-let annee)
    (let* ((a (modulo annee 19))
           (b (modulo annee 4))
           (c (modulo annee 7))
           (d (modulo (+ (* 19 a) 24) 30))
           (e (modulo (+ (* 2 b) (* 4 c) (* 6 d) 5) 7)))
        (+ 22 d e)))

(define (traduit jour)
    (if (< jour  31)
        (begin (display jour) (display" mars "))
        (begin (display (- jour 31)) (display " avril "))))

(define (paques annee)
    (begin (traduit (numjour annee)) (display annee)))
```