Option PF2, L2-Info&Math, Automne 2016

« Programmation Avancée en Scheme »

LES VECTEURS

Exercice 10.1 a) Le langage Scheme propose la fonction de conversion de type `(vector->list v)`. Comment la programmeriez-vous si elle n'existait pas ?…

b) De même il existe une primitive `(list->vector L)`. Montrez que la définition ci-dessous est correcte mais non satisfaisante [quelle est sa complexité ?], et proposez-en une meilleure :

```
(define ($list->vector L)
  (define n (length L))
  (define v (make-vector n))
  (for ([i (in-range n)])
    (vector-set! v i (list-ref L i)))
  v))
```

c) Bizarrement, la norme Scheme n'offre pas l'analogue de la fonctionnelle `map` sur les vecteurs. Programmez une fonction `($vector-map f v)` qui retourne une copie transformée du vecteur *[N.B.* `vector-map` *et* `vector-map!` *existent en Racket]*.

d) Programmez une fonction `(vector-swap! v i j)` prenant un vecteur `v` ainsi que deux indices `i` et `j` de ce vecteur, et échangeant *sur place* les composantes d'indice `i` et `j`. Cette fonction n'aura aucun résultat !

```
> (define v (vector 10 20 30 40 50))
> v
#(10 20 30 40 50)
> (vector-swap! v 1 3)     ; aucun résultat, seulement un effet de bord !
> v
#(10 40 30 20 50)
```

Exercice 10.2 *Les matrices [cours page 14].*

En utilisant le type abstrait « matrice » vu en cours, programmez la multiplication matricielle `(mat* A B)`. On déclenchera une erreur si les formats des matrices n'en autorisent pas la multiplication ! On rappelle la formule du terme $M_{i,j}$ du produit :

$$M_{i,j} = \sum_k A_{i,k} \times B_{k,j}$$

Exercice 10.3 *Le drapeau hollandais [cours 10 page 17].*

a) Programmez une fonction `(random-drapeau n)` retournant un vecteur de longueur `n` composée uniquement des symboles `'bleu`, `'blanc` et `'rouge`. On utilisera `build-vector`.

```
> (define D (random-drapeau 7))
> D
(rouge blanc blanc bleu rouge bleu blanc)
```

b) Codez en Scheme l'algorithme de Dijkstra pour trier un drapeau.

```
(tri-drapeau D) → (bleu bleu blanc blanc blanc rouge rouge)
```

LES STRINGS

Exercice 8.1 a) En utilisant `build-string`, définissez en une ligne une chaîne de caractères contenant l'alphabet minuscule :

```
> (define ALPHA (build-string ...))
> ALPHA
"abcdefghijklmnopqrstuvwxyz"
```

b) Programmez une fonction sans résultat `(maj! str)` prenant une chaîne de catactères `str` et transformant les lettres minuscules de `str` en majuscules. On n'utilisera pas les primitives Scheme traitant des problèmes de *maj/min* comme `char-lower-case?` ou `char-upcase`, et on passera directement par les codes des caractères !

```
> (define str (string-copy "Le Joli mois de Novembre !"))
> (maj! str)    ; aucun résultat !
```

```
> str
"LE JOLI MOIS DE NOVEMBRE !"
```

Exercice 8.2 En utilisant `let/ec` pour simuler le mécanisme `return de Python/Java/C`, traduisez en Scheme quasiment mot à mot la fonction ci-dessous en remplaçant la liste `L` par une *string*. Programmez donc `(position c str)`.

```
def position(x,L) :                  # indice de la première occurrence de x dans la liste L
    i = 0
    while True :                     # boucle infinie avec échappements !
        if i >= len(L) : return -1
        if L[i] == x : return i
        i = i + 1
```

Exercice 8.3 En utilisant des **expressions régulières**, écrivez :

a) La fonction `(nb-blancs-en-tete str)` prenant une chaîne `str` et retournant le nombre d'espaces en tête :

`(nb-blancs-en-tete "_____une__chaîne_") → 5`

b) La fonction `(premier-mot str)` retournant le premier « mot » [suite de lettres] d'une phrase `str`, ou bien `#f` :

`(premier-mot "12345Mot!78") → "Mot"`

c) La fonction `(today)` retournant la date d'aujourd'hui sous forme de liste d'entiers :

```
(today) → (19 3 2009)   ; le 19 mars 2009
```

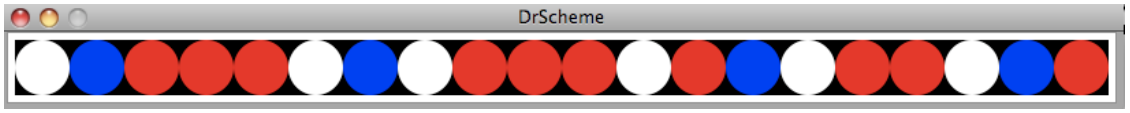
Vous utiliserez la formule `(date->string (seconds->date (current-seconds)))` qui retourne la date sous la forme d'une chaîne de caractères, puis vous traiterez cette chaîne avec des *regexp* pour construire le résultat (19 3 2009). Utilisez une A-liste pour traduire le mois en un nombre entier…

N.B. Il y a une solution plus rapide consistant à ouvrir les champs de la structure `date`*, mais l'exercice a pour objet le traitement des regexp… Il y a une solution encore plus rapide consistant à demander la date en format indian, mais vous ne le ferez pas, ce serait de la triche ! Oh, faites-le si ça vous amuse…*

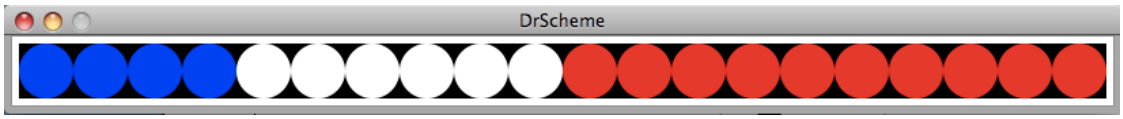
EXERCICES COMPLEMENTAIRES

Exercice 10.4 Illustrez l'algorithme du drapeau hollandais par une **animation**. Dans un canvas horizontal, on voit les cases (contenant des boules de couleur) s'échanger jusqu'à obtention du drapeau trié…

AVANT



APRES



Exercice 8.4 Programmez une fonction `(scheme->python A AL)` prenant un arbre binaire d'expression `A` contenant des variables dont les valeurs sont dans la A-liste `AL`. Cette fonction sans résultat va afficher à l'écran le texte d'une fonction Python dont le nom sera obtenu par `gensym`, et retournant la valeur approchée de l'arbre `A`. Il s'agit donc d'un nano traducteur de Scheme vers Python. Mais vous le faire pour Java ou C si vous préférez ! Exemple :

```
> (scheme->python '(+ (* (+ x 1) 2) y) '((x 3) (y -1/4)))
def func3291() :     # Look, Ma : my first compiler !
    x = 3
    y = -0.25
    return (((x + 1) * 2) + y)
```

N.B. L'écriture d'un traducteur de Python vers Scheme est plus compliqué ! Le projet GNU de la Free Software Foundation a adopté Scheme comme langage de base. Voir `http://www.gnu.org/software/guile/guile.html`

Exercice 8.4 Programmez une fonction `(spell word)` prenant une chaîne de caractères `word` représentant un mot *en anglais* et retournant un son l'épelant lettre à lettre. Utilisez par exemple `http://soundbible.com/2009-A-Z-Vocalized.html`