UV PROGRAMMATION DÉCLARATIVE
Examen de programmation fonctionnelle
Durée 2h, documents autorisés, livres interdits

1. Donner le résultat de chacune des expressions ci-dessous comme si vous étiez un interprète scheme, en expliquant vos réponses.

```
?- (let* ((a 0)
          (b 1)
          (c (+ a 1)))
     (let ((c a)
           (y c))
         (* c y)))
?- (define g (let* ((x 0)
                   (y 1)
                   (f (lambda (x) (* y x))))
              f))
?- (define x 2)
?- (define y 2)
?- (g 1)
?- (define (incrementer! x)
      (set! x (1+ x)))
?- (define x 0)
?- (incrementer! x)
?- x
```

2. Quels sont les types de l'argument et du résultat de la fonction *mystere*. Deviner quelle est l'action de cette fonction sur son argument et quelle est la relation le liant avec le résultat.

```
(define (mystere ll)
  (cond ((null? ll) 0)
        ((list? (car ll)) (+ (mystere (car ll))
                              (mystere (cdr ll))))
        (else (+ (car ll)
                  (mystere (cdr ll))))))
```

3. On se propose dans cet exercice de développer des opérations sur les vecteurs et les matrices réels. On représente un vecteur par une liste de nombres réels et une matrice par une liste dont chaque élément est une ligne, représentée elle-même par une liste de réels. On écrira les fonctions suivantes en utilisant de façon pertinente les fonctionnelles sur les listes de la bibliothèque scheme et en évitant de développer inutilement la récursivité.

– le produit scalaire :

– Entrée $v_1 = (a_1, a_2, \dots, a_n)$ et $v_2 = (b_1, b_2, \dots, b_n)$, deux vecteurs

- Sortie $r = \sum_{i=1}^n a_i * b_i$
 - Le produit d'une matrice par un vecteur :
 - Entrée $m = ((a_{11}, a_{12}, \dots, a_{1n}), (a_{21}, a_{22}, \dots, a_{2n}) \dots (a_{m1}, \dots, a_{mn}))$ et $v = (x_1, x_2, \dots, x_n)$
 - Sortie $y = (y_1, y_2, \dots, y_m)$, où $y_i = \sum_{k=1}^n a_{ik} * x_k$ pour $1 \leq i \leq m$
 - La transposition d'une matrice :
 - Entrée $m = ((a_{11}, a_{12}, \dots, a_{1n}), (a_{21}, a_{22}, \dots, a_{2n}) \dots (a_{m1}, \dots, a_{mn}))$
 - Sortie $m' = ((a_{11}, a_{21}, \dots, a_{n1}), (a_{12}, a_{22}, \dots, a_{n2}) \dots (a_{1m}, \dots, a_{nm}))$
 - Le produit de deux matrices :
 - Entrée
 - $m1 = ((a_{11}, a_{12}, \dots, a_{1n}), (a_{21}, a_{22}, \dots, a_{2n}) \dots (a_{m1}, \dots, a_{mn}))$,
 - $m2 = ((b_{11}, b_{12}, \dots, b_{1m}), (b_{21}, b_{22}, \dots, b_{2m}) \dots (b_{n1}, \dots, b_{nm}))$
 - Sortie $m' = ((c_{11}, c_{21}, \dots, c_{m1}), (c_{12}, c_{22}, \dots, c_{m2}) \dots (c_{1m}, \dots, c_{mm}))$
avec $c_{ij} = \sum_{k=1}^n a_{ik} * b_{kj}$ pour $1 \leq i \leq m$ et $1 \leq j \leq m$
4. On veut développer un module représentant un point à deux dimensions par une fermeture. Un module contient deux informations qui sont l'abscisse et l'ordonnée et fournit les fonctions d'accès à ces deux champs, ainsi qu'une fonction calculant la distance entre deux points. Voici comment devrait fonctionner ce module :

```
guile> (define p1 (make-point 1 0))
guile> (define p2 (make-point 0 1))
guile> (p1 'abscisse)
1
guile> (p1 'ordonnee)
0
guile> (p1 'distance p2)
1.4142135623731
```