FONCTIONNELLES

Exercice 1 : Utilisation de schémas de quantification logique.

1°) Ecrire une fonction `tous-egaux` qui a comme argument une liste `L` non vide, et qui retourne `#t` si tous les éléments de `L` sont égaux et `#f` sinon.

a) en utilisant un schéma universel.
b) en utilisant un schéma existentiel.

2°) Ecrire une fonction `tous-diff` qui a comme argument une liste `L` qui contient au moins deux éléments, et qui retourne `#t` si les éléments de `L` sont tous différents deux à deux et `#f` sinon.

a) en utilisant un schéma universel.
b) en utilisant un schéma existentiel.

Exercice 2 : Schémas de récurrence simple et double.

1°) Ecrire un schéma de récurrence simple permettant d'évaluer le n^ième^ terme d'une suite (u_n) définie par :

$$u_n = f(u_{n-1}, n) \text{ et } u_{n_0} = B.$$

Appliquer ce schéma aux cas de la somme des n premiers entiers non nuls, de la somme des carrés des n premiers entiers non nuls, de la factorielle, de la suite de Fibonacci (en posant $U_n = (u_n, u_{n+1})$, $n_0 = 0$ et $U_0 = B =$ `'(1 1)`).

2°) En déduire un schéma de récurrence double permettant d'évaluer le n^ième^ terme d'une suite (u_n) définie par :

$$u_n = f(u_{n-1}, u_{n-2}, n),\ u_{n_0} = B_0,\ u_{n_0+1} = B_1.$$

Appliquer à la suite de Fibonacci.

Exercice 3 : Relation d'équivalence; Ensemble quotient.

On représente un ensemble fini E avec une liste `E`, et une relation R dans E par une fonction anonyme `R` à deux arguments formels `x` et `y` telle que l'évaluation de cette fonction anonyme pour deux arguments effectifs `x1` et `y1` retourne `#t` si `x1` R `y1` et retourne `#f` sinon.

Exemple :

```
(let ((E '(0 1 2 3 4 5 6))
      (R (lambda (x y) (= (modulo (- x y) 3) 0))) )
   ...
```

1°) Ecrire la fonction `reflexive` qui a deux arguments `R` et `E`, et qui retourne `t` si `R` est réflexive et `#f` sinon.

2°) Ecrire a fonction `symetrique` qui a deux arguments `R` et `E`, et qui retourne `t` si `R` est symétrique et `#f` sinon.

3°) Ecrire la fonction `transitive` qui a deux arguments `R` et `E`, et qui retourne `t` si `R` est transitive et `#f` sinon.

4°) Ecrire la fonction `quotient` qui a deux arguments `R` et `E`, et qui, si `R` est une relation d'équivalence, retourne l'ensemble quotient de `E` par `R` sous forme d'une liste contenant les classes d'équivalence de `R`, chacune représentée par une sous-liste.

Exemple :

Avec l'exemple ci-dessus `(quotient E R)` retourne `((0 3 6) (1 4) (2 5))`.

Exercice 4 : Produits cartésiens

Ecrire la fonction `PC` qui a deux arguments, l'ensemble `E` et un entier `n` et qui retourne une liste contenant, sous forme de sous-listes, l'ensemble des `n`-uplets de `E`.

Exemple :

`(PC '(0 1) 3)` retourne :

`((0 0 0) (0 0 1) (0 1 0) (0 1 1) (1 0 0) (1 0 1) (1 1 0) (1 1 1))`.

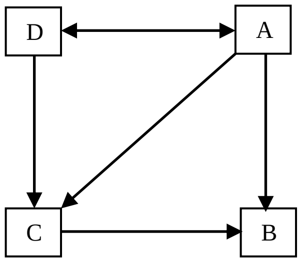
Exercice 5 : Chemins dans un graphe orienté

On représente un graphe orienté par une liste `G` de couples dont le premier élément est un sommet du graphe, et le second la liste des successeurs de ce sommet.

Exemple :

```
(let ((G '((D (A C)) (A (D C B)) (C (B)) (B ()))))
     ...
```

Dans l'exemple ci-dessus `G` représente le graphe orienté représenté ci-contre :



Ecrire la fonction chemins qui a trois paramètre, `dep`, `but`, et `G` et qui retourne la liste de tous les chemins de `G` (représentés par des sous listes) partant de `dep` et arrivant à `but`.

Exercice 6 : Algèbre linéaire

On représente une matrice carrée $n \times n$ par une liste de n sous-listes contenant chacune n valeurs numériques, chaque sous-liste représentant une ligne de la matrice.
On représente un vecteur colonne de taille n par une liste de n valeurs numériques.

1°) Ecrire une fonction `trace` ayant comme argument une matrice `M` et telle que l'évaluation de l'expression `(trace M)` retourne la trace de la matrice `M`, c'est-à-dire la somme de ses éléments diagonaux.

2°) Ecrire une fonction `transp` qui a comme argument une matrice `M` et qui retourne la matrice `M` transposée.
Exemple :
`(transp '((11 21 31) (12 22 32) (13 23 33)))` retourne :
`((11 12 13) (21 22 23) (31 32 33))`.

3°) Ecrire une fonction `MV` qui a comme argument une matrice `M` et un vecteur `V` et qui retourne le vecteur produit de `M` par `V`.

4°) Ecrire une fonction `AL` qui a comme argument une matrice `M` et qui retourne l'application linéaire qui lui correspond.

Exercice 7 : Suite des itérés d'une fonction

1°) Ecrire une fonction `compose` qui a comme arguments deux fonctions d'une seule variable, `f` et `g`, et qui retourne la fonction `fog` : $x \mapsto$ f (g (x)).
2°) Ecrire une fonction `trace` qui a comme arguments une fonction d'une seule variable, `f`, et un entier `n`, et qui retourne la liste de fonctions (Id f f^2 ... f^n), où `Id` désigne la fonction identité et f^n désigne la composition de `f` (n-1) fois par elle-même.
3°) Ecrire une fonction `applique` qui a comme arguments une liste de fonctions d'une seule variable, `Lf`, et une variable `x`, et qui retourne la liste des applications des fonctions de `Lf` à la variable `x`.
Par exemple, l'évaluation de `(applique (trace (lambda (x) (* x x)) 3) 2)` doit retourner la liste `(2 4 16 256)`.

Exercice 8 : Ensemble des parties d'un ensemble

Ecrire une fonction `P` ayant comme argument une liste `E` représentant un ensemble E et telle que l'évaluation de l'expression `(P E)` retourne une liste de sous-listes représentant l'ensemble des parties de E, chaque sous-liste représentant une partie de E.
Par exemple, l'évaluation de `(P '(1 2 3))` doit retourner

`(() (1) (2) (3) (1 2) (1 3) (2 3) (1 2 3))`

ou toute autre liste contenant les mêmes éléments dans un ordre différent.

Exercice 9 : Partition d'un ensemble en deux parties

Etant donné un ensemble E, on appelle partition de E tout ensemble P_E de parties de E tel que:

$$\forall A \in P_E, A \neq \emptyset ,$$
$$\forall (A , B) \in (P_E)^2, A \cap B = \emptyset ,$$
$$\bigcup_{A \in P_E} A = E.$$

On note P_E^2 l'ensemble des partitions de E en deux parties.
Cet ensemble est forcément vide si E est de cardinalité inférieure à 2.
Pour $E = \{3,4\}$, on a $P_E^2 = \{ \{\{3\},\{4\}\}\}$.
Pour $E = \{2,3,4\}$, on a $P_E^2 = \{ \{\{2,3\},\{4\}\},\{\{3\},\{2,4\}\},\{\{2\},\{3,4\}\}\}$.
Pour $E = \{1,2,3,4\}$, on a $P_E^2 = \{ \{\{1,2,3\},\{4\}\},\{\{1,3\},\{2,4\}\},\{\{1,2\},\{3,4\}\},$
$\{\{2,3\},\{1,4\}\},\{\{3\},\{1,2,4\}\},\{\{2\},\{1,3,4\}\},$
$\{\{1\},\{2,3,4\}\}\}$.
Ecrire une fonction `P2` qui a comme argument une liste `E`, représentant un ensemble E, et telle que l'évaluation de l'expression `(P2 E)` retourne la liste représentant l'ensemble des partitions en deux parties de l'ensemble E.

Exercice 10 : Ensemble des fonctions stabilisant une partie d'un ensemble

1°) a) Ecrire une fonction `fct` ayant comme arguments deux listes `d` et `a` de même longueur, l, et telle que l'évaluation de l'expression `(f d a)` retourne une fonction qui appliquée à tout élément placé en $n^{\text{ième}}$ position dans `d` (avec $1 \leq n \leq l$) retourne l'élément placé en $n^{\text{ième}}$ position dans `a`.
Par exemple l'évaluation de `((fct '(a b c) '(x y z)) 'b)` doit retourner `y`.
b) On suppose que l'on dispose d'une fonction `PC` ayant comme arguments une liste `E` représentant un ensemble, et un entier `n`, et telle que l'évaluation de l'expression `(PC E n)` retourne la liste de tous les `n`-uplets de l'ensemble `E`.
Ecrire, en utilisant les fonctions `fct` et `PC`, une fonction `LF` ayant comme argument deux listes non vides représentant deux ensembles non vides `E` et `F`, et telle que l'évaluation de l'expression `(LF E F)` retourne la liste de toutes les applications de `E` dans `F` (une application de `E` dans `F` est une fonction de `E` dans `F` pour laquelle chaque élément de `E` a une image).
Indication : si `E` est un ensemble de cardinalité n, toute fonction de `E` dans `F` est caractérisée par un n-uplet d'éléments de `F`, chaque terme en $k^{\text{ième}}$ position d'un tel n-uplet étant l'image du $k^{\text{ième}}$ élément de `E`.

2°) a) Soit `SR` le schéma suivant :
```
(define SR (lambda (L B C)
   (if (null? L)
       B
      (C (car L) (SR (cdr L) B C)) ) ))
```
En utilisant le schéma `SR`, écrire une fonction `filtrer` ayant comme arguments une liste `E` et un prédicat `P`, et telle que l'évaluation de l'expression `(filtrer E P)` retourne la liste des éléments de `E` satisfaisant `P`.
Par exemple, l'évaluation de `(filtrer '(1 a 2 b) integer?)` doit retourner la liste `(1 2)`.

b) Soit `qqs?` le schéma suivant :
```
(define qqs? (lambda (L P)
   (if (null? L)
       #t
      (and (P (car L)) (qqs? (cdr L) P)) ) ))
```
En utilisant le schéma `qqs?`, écrire un prédicat `stabilise?` ayant comme arguments une fonction `f` et une liste `P` représentant un ensemble, et telle que l'évaluation de l'expression `(stabilise? f P)` retourne une valeur logique vraie si `f` stabilise `P` (c'est-à-dire si l'image de tout élément de `P` par `f` appartient à `P`), et une valeur logique fausse sinon.
On pourra utiliser la fonction prédéfinie `member?` Ayant comme argument un élément `x` et une liste `L` et telle que l'évaluation de l'expression `(member? X L)` retourne une valeur logique vraie si `x` est élément de `L`, et une valeur logique fausse sinon.

c) Ecrire une fonction `LFS` ayant comme arguments une liste `E` représentant un ensemble, et une liste `P` représentant une partie de `E`, et telle que l'évaluation de l'expression `(LFS E P)` retourne la liste des fonctions de `E` dans `E` qui stabilisent `P`.

<u>Exercice 11 : Définition des entiers naturels par la représentation de Church</u>

Le principe de représentation d'un nombre entier dans la théorie du λ-calcul (représentation dite des *nombres de Church*) est le suivant : un entier $n > 1$ est représenté par une fonction qui, à toute fonction $f : x \mapsto f(x)$, associe la composition de f, $n - 1$ fois par elle même, c'est-à-dire,

$$\underset{\overset{\uparrow}{1}}{f} \circ \underset{\overset{\uparrow}{2}}{f} \circ \ \ldots \circ \underset{\overset{\uparrow}{n}}{f} \quad .$$

Par exemple l'entier 2 est représenté par la fonction *deux* : $f \mapsto deux\,(f) = f \circ f$.
On a donc : $deux\,(f) : x \mapsto [\ deux\,(f)\ ]\,(x) = [\ f \circ f\ ]\,(x) = f\,[\ f\,(x)\ ]$.

Par généralisation :
- l'entier 1 est représenté par la fonction : $f \mapsto f$;
- l'entier 0 est représenté par la fonction : $f \mapsto I$, où I représente la fonction identité.

1°) Ecrire les fonctions `zero`, `un`, et `deux`, représentant respectivement les entiers 0, 1 et 2.

2°) Ecrire la fonction `trois`, représentant l'entier 3, en utilisant la fonction `deux`.

3°) En déduire une fonction `succ` ayant comme argument la représentation de Church d'un entier n, `rn`, et telle que l'évaluation de `(succ rn)` retourne la représentation de Church de l'entier $n + 1$.

4°) Ecrire une fonction `nom` ayant comme argument la représentation de Church d'un entier n, `rn`, et une liste de symboles, `listenoms` (supposée de longueur au moins égale à $n + 1$), telle que l'évaluation de l'expression `(nom rn listenoms)` retourne le $(n + 1)^{\text{ème}}$ élément de `listenoms`.
Par exemple l'évaluation de `(nom trois '(Z I II III IV V VI))` retourne `III`.

5°) Ecrire une fonction `entier` ayant comme argument un symbole, `nom`, et une liste de symboles, `listenoms`, et retournant la représentation de Church de l'entier $n - 1$ si `nom` figure en $n^{\text{ième}}$ position dans `listenoms`, et `#f` sinon.
Par exemple l'évaluation de `(entier 'III '(Z I II III IV V VI))` retourne la fonction qui à toute fonction f associe $f \circ f \circ f$.
On pourra utiliser les fonctions `zero` et `succ` définies ci-dessus.

6°) a) Ecrire une fonction `plus` ayant comme arguments les représentations de Church de deux entiers n et m, respectivement `rn` et `rm`, et telle que l'évaluation de l'expression `(plus rn rm)` retourne la représentation de Church de l'entier $n + m$.

b) Que fait la fonction suivante ? Justifiez votre réponse.

```
(define A (lambda (rn rm)
        (lambda (f)
           (lambda (x) (((rn rm) f) x) ) ))
```