E.S.I 2009/2010 – EMD2 – MCP 4SI(Q) – Durée 1h30 – Doc. interdit

Barème : (5x4)

1) Donner le nombre de radicaux (redex) dans la lambda expression suivante, et trouver sa forme normale en appliquant l'ordre normal des réductions (appel par nom) :
(λy. (λx. (λy. x y) 4) (λx. * x y)) 2
Rappel: un radical (ou redex) est une expression réductible

2) Donner un programme Lisp, permettant de calculer le nombre d'étapes nécessaire pour atteindre l'élément 1 dans une suite de Syracuse.

Une suite de Syracuse commençant par l'entier a, est définie comme suit:

$U_0 = a$;

$U_n = U_{n-1}$ div 2 si U_n est pair – avec $n > 0$

$U_n = 3U_{n-1} + 1$ si U_n est impair – avec $n > 0$

On pourra utiliser le prédicat (pair x) qui teste si x est pair.
Exemples:

- pour a = 3, on aura 7 étapes pour atteindre la valeur 1
(U_0=3, U_1=10, U_2=5, U_3=16, U_4=8, U_5=4, U_6=2, U_7=1)
- pour a = 4, on aura 2 étapes pour atteindre la valeur 1
(U_0=4, U_1=2, U_2=1)

3) Donner un programme Prolog qui permet d'insérer un entier V dans une liste ordonnée d'entiers:
Ins(V, L1, L2) sera vrai si l'insertion de V dans la liste ordonnée L1 donne la liste ordonnée L2.
Par exemples:

Ins(3, [1,6,9], L) est vrai pour L=[1,3,6,9]

Ins(5, [1,5], [1, 5, 5]) est vrai

4) En programmation logique pure, on peut définir une liste de symboles comme étant un terme structuré défini par l'équation à point fixe suivante:
Liste = nil / f(symbole, Liste)
où le premier argument du symbole fonctionnel f, désigne le premier élément de la liste alors que son deuxième argument désigne le reste de la liste.
Par exemple, la liste [a,b,c] sera représentée par le terme f(a,f(b,f(c,nil)))
Donner un programme Prolog purement logique, qui réalise la concaténation de 2 listes.

5) Donner un programme purement logique pour réaliser la multiplication de 2 entiers naturels.
On utilisera la définition suivante pour les entiers naturels:
Entier = zero / s(Entier)
Par exemple l'entier 3 sera représenté par le terme s(s(s(zero)))

E.S.I 2009/2010 – Corrigé EMD2 – MCP 4SI(Q)

1) Lambda calcul

- Il existe 3 radicaux (ou redex) dans l'expression : (λy. (λx. (λy. x y) 4) (λx. * x y)) 2
  - le premier lambda y avec son argument 2
  - le premier lambda x avec son argument (λx. * x y)
  - le deuxième lambda y avec son argument 4

- Forme normale:

(λy. (λx. (λy. x y) 4) (λx. * x y)) 2 → (λx. (λy. x y) 4) (λx. * x 2) → (λy. (λx. * x 2) y) 4
→ (λx. * x 2) 4 → * 4 2 → 8

2) Programme Lisp:

```
(DE   NbEtape ( a )
        (IF   (> a 1)
              (IF (pair a)      (+ 1 (NbEtape (/ a 2)))
                                (+ 1 (NbEtape (+ (* 3 a) 1)))
              )
              /* sinon du IF (> a 1)  */
              0
        )
)
```

3) Prolog : Insertion dans une liste ordonnée

```
Ins(X, [], [X]).
Ins(X, [Y|L], [X|[Y|L]]) :-
              X <= Y.
Ins(X, [Y|L1], [Y|L2]) :-
              X > Y,  Ins(X,L1,L2).
```

4) Programmation purement logique : concaténation de listes

```
concat( nil, L, L ).
concat( f(X,L1), L2, f(X,L3) )  :- concat(L1, L2, L3).
```

5) Programmation purement logique : Multiplication d'entiers

```
/* addition d'entiers */
plus(X,zero,X).
plus (X, s(Y), s(Z) ) :-  plus( X, Y, Z ).

/* multiplication d'entiers */
mult( X, zero, zero ).
mult( X, s(zero), X ).
mult( X, s(Y), Z ) :-
              mult(X,Y,A),  plus(A,X,Z).
```