Corrigé examen calcul formel variante

1) Evaluation : manipulation des listes

Sexpression	Résultat d'évaluation
a >(cons 3 (cons 1 (cons 0 '(2 5 9))))	>(3 1 0 2 5 9)
b >(car (cdr (cdr '(a b (d e) ((f g) (h i))))))	> (D E)
c >(atom (car (cdr (car '(((a b) (c d e)) g h)))))	> NIL

2) Evaluation : manipulation des tableaux

Sexpression	Résultat d'évaluation
a > (defparameter *a* (make-array '(2 3 2) :adjustable t :initial-element 0))	> *a* > #3A(((0 0) (0 0) (0 0)) ((0 0) (0 0) (0 0)))
b > (adjust-array *a* '(2 3 3) :initial-element 4)	>*a* > #3A (((0 0 4) (0 0 4) (0 0 4)) ((0 0 4) (0 0 4) (0 0 4)))
c > (defparameter *a* (make-array '(2 3):initial-contents '((2 5 8) (9 3 1))))	>*a* > #2A ((2 5 8) (9 3 1))
d > (defparameter *b* (make-array '(3): displaced-to *a* : displaced-index-offset 2))	>*b* > #(8 9 3)
e > (setf (aref *b* 2) 0)	>*b* > #(8 9 0)

3) Ecrire les functions en Lisp :

a) calcul de la somme des carrés des factorielles des n premiers entiers impairs : $S(n) = 1!^2 + 3!^2 + \ldots\ldots(2n-1)!^2$ Exemple : > (S 2) > 37 Remarque: si vous utilisez la fonction factorielle il faut la définir auparavant	> (defun fac(n) (if (= n 0) 1 (* n (fac (- n 1))))) >(defun S(n) (if (= n 0) 0 (+ (* (fac (- (* n 2) 1)) (fac (- (* n 2) 1))) (S (- n 1)))))
b) recherche du premier nombre impair dans une liste Exemple : >(premimp '(12 82 7 12 25)) >7 >(premimp '(4 12 6 8)) >NIL Indication : vous pouvez utiliser la fonction prédéfinie (mod x y) qui donne le reste de la division de x par y >(mod 13 5) >3	> (Defun premimp (l) (cond ((null l) nil) ((atom (car l))(if (= (mod (car l) 2) 0) (premimp (cdr l)) (car l))) (t (or (premimp (car l)) (premimp (cdr l))))))

Corrigé examen calcul formel variante

1) Evaluation : manipulation des listes

Sexpression	Résultat d'évaluation
a >(cons 3 (cons 1 (cons 0 '(2 5 9))))	> (3 1 0 2 5 9)
b >(car (cdr (cdr '(a b (d e) ((f g) (h i))))))	> (D E)
c >(atom (car (cdr (car '(((a b) (c d e)) g h)))))	>NIL

2) Evaluation : manipulation des tableaux

Sexpression	Résultat d'évaluation
a > (defparameter *a* (make-array '(3 2 2) :adjustable t :initial-element 0))	> *a* > #3A (((0 0) (0 0)) ((0 0) (0 0)) ((0 0) (0 0)))
b > (adjust-array *a* '(3 3 3) :initial-element 8)	>*a* > #3A (((0 0 8)(0 0 8)(8 8 8)) ((0 0 8)(0 0 8)(8 8 8)) ((0 0 8)(0 0 8)(8 8 8)))
c > (defparameter *a* (make-array '(3 2):initial-contents '((2 5) (8 9) (3 1))))	>*a* > #2A ((2 5)(8 9)(3 1))
d > (defparameter *b* (make-array '(4): displaced-to *a* : displaced-index-offset 1))	>*b* > # (5 8 9 3)
e > (setf (aref *b* 3) 7)	>*b* > # (5 8 9 7)

3) Ecrire les functions en Lisp :

a) calcul de la somme des carrés des factorielles des n premiers entiers pairs : $S(n) = 2!^2 + 4!^2 + \ldots\ldots 2n!^2$ Exemple : > (S 2) > 580 Remarque : si vous utilisez la fonction factorielle il faut la définir auparavant	> (defun fac(n) (if (= n 0) 1 (* n (fac (- n 1))))) >(defun S(n) (if (= n 0) 0 (+ (* (fac (* n 2)) (fac (* n 2))) (S (- n 1)))))
b) recherche du premier nombre pair dans une liste Exemple : >(prempair '(11 82 7 12 25)) >82 >(prempair '(5 11 7 9)) >NIL Indication : vous pouvez utiliser la fonction prédéfinie (mod x y) qui donne le reste de la division de x par y. >(mod 13 5) >3	> (Defun prempair (l) (cond ((null l) nil) ((atom (car l))(if (> (mod (car l) 2) 0) (prempair (cdr l)) (car l))) (t (or (prempair (car l)) (prempair (cdr l))))))