INTELLIGENCE ARTIFICIELLE

SUPINFO International University

Sign of Success

3AIT – TP CORRECTION

Intelligence Artificielle

L'analyse d'un monde guidé

Programmation Fonctionnelle
Lisp
Python

Version 1.0
Last update: 30/04/2018
Use: Students/Staff
Author: Cyril Alexandre Pachon

SOMMAIRE

1 PREAMBULE : LES CONSIGNES GENERALES

Votre rendu se fera sous la forme d'un **dossier compressé** (**.zip**) nommé **[3AIT]-IDOpenCampus-NomDuCampus-Nom-Prénom-TP**. Votre dossier peut contenir **uniquement** des documents sous la forme **.pdf**, **.py** et **.lsp**.

Pour cet examen, **vous pouvez utiliser les supports de cours (.ppt, LABS)**. L'utilisation d'**internet est interdite**. Les outils **autorisés** sont les interpréteurs **Python** et **Lisp, pas d'autres outils**. Si votre surveillant(e) constate une tricherie, votre épreuve sera annulée et votre relevé de notes portera la mention de « **cheater** » pour cet examen.

2 LA PROGRAMMATION FONCTIONNELLE (6 POINTS)

Question1.1 (1 point): Avec les sélecteurs ou les constructeurs, écrire la fonction récursive nommée **NB** qui compte le nombre d'atomes dans une liste d'atomes. Spécifiez tous les éléments de vos réalisations.

SOLUTION

NB = La fonction qui donne le nombre d'atome contenu dans la liste. L est le paramètre liste passé en argument

NB(L) → 0 si pas d'élément le nombre d'élément sinon **(0.25 point)**

NB([]) : 0 **(0.25 point)**

NB(e o D) = 1 + NB(D) **(0.5 point)**

Vous disposez de la liste suivante L = ((1 2 3) (4 5 6) (7 8 9))

Question1.2 (3 points): Avec les sélecteurs ou les constructeurs, écrire une fonction récursive nommée **VF** qui vérifie si une liste contient des sous-listes (uniquement de profondeur 1, comme la liste L) contenant le même nombre d'atomes dans chaque sous-liste (comme la liste L). Spécifiez tous les éléments de vos réalisations.

SOLUTION

VF = La fonction qui vérifie si le nombre d'éléments dans chaque sous listes sont égaux. L est l'argument entré constitué de sous liste.

VF (L) → VRAI si les sous-listes ont le même nombre d'élément FAUX sinon

e représente une sous-liste.

L et D représentent des listes de listes.

VF([]) : FAUX **(0.25 point)** La liste est vide n'a donc pas de sous-listes.

VF([e]) : FAUX **(0.25 point)** La liste comporte l'unique sous liste.

VF(e1 o e2 o D) : soit NB(e1) = VF(e2 o D) **(0.5 point)**

Dans D = [] : NB(e2) **(0.5 point)** Le point d'arrêt est l'envoie de la dernière somme de sous liste.

D != [] : NB(e2) et puis VF(D) **(0.5 point)**

Question1.3 (2 points) : Avec les sélecteurs ou les constructeurs, écrire une fonction récursive nommée **VLA** qui vérifie si le nombre d'atomes dans chaque sous-liste de L est égal au nombre de sous-liste de L. Spécifiez tous les éléments de vos réalisations.

SOLUTION

VLA = La fonction qui vérifie si la liste passée en argument représente une matrice carrée

VLA(L) → VRAI si L est une matrice carrée FAUX sinon. **(0.5 point)**

VLA(L) : selon L
Vide(L) : VRAI **(0.25 point)**
nonVide(L) : selon NB(L), L **(0.25 point)**
NB(L) = VF(L) : VRAI **(0.5 point)**
NB(L) ! = VF(L) : FAUX **(0.5 point)**

3 LE PROBLEME LISP (8 POINTS)

Un Système Expert contient l'expression suivante :

```
(and (defun E()
(lambda (A B)
(cond
  ((< A 20) (expt B 10))
  (T  "Confirmez-moi le résultat obtenu ?"))))
  (funcall (E) *read-base* *print-base*))
```

Question 2.1 (1 point) : Donnez le résultat de l'expression.

SOLUTION

```
CL-USER 1 > (and (defun E()
(lambda (A B)
(cond
  ((< A 20) (expt B 10))
  (T  "Confirmez-moi le résultat obtenu ?"))))
  (funcall (E) *read-base* *print-base*))
```

10000000000

Question2.2 (2 points) : Expliquez ligne à ligne ce que fait l'expression donnée. Comment le résultat est-il produit ?

SOLUTION

And → opérateur logique pour faire un ET entre la définition de la fonction E et l'interprétation funcall

Lambda → déclaration temporaire des variable A et B qui auront comme valeurs *read-base* et *print-base*

read-base → fait la lecture de la base arithmétique de l'interpréteur

(< *read-base* 20) → fait le test de la base avec la valeur 20

(< *read-base* 20) → rend faux si la base est plus petite que 20

La condition est VRAI → l'action (expt *print-base* 10) est exécutée

print-base → fait l'écriture de la base arithmétique de l'interpréteur

(expt *print-base* 10) → fait la puissance 10 de la base

La condition (< *read-base* 20) est FAUSSE alors T VRAI ➔ "Confirmez-moi le résultat obtenu ?" s'exécute à l'écran

Question2.3 (1 point) : Faites une seule modification dans l'expression donnée pour que la chaine de caractères soit interprétée et affichée. Il n'est pas demandé de refaire l'expression, mais juste de modifier un des éléments de l'expression pour obtenir le résultat : "Confirmez-moi le résultat obtenu ?". Expliquer votre choix.

SOLUTION

Il suffit de modifier l'opérateur de test < en > par exemple.

```
(and (defun E()
(lambda (A B)
(cond
  ((> A 20) (expt B 10))
  (T  "Confirmez-moi le résultat obtenu ?"))))
  (funcall (E) *read-base* *print-base*))
```

Une autre réponse possible serait de modifier la valeur

Question 2.4 (1,5 points) : Ecrire une fonction Lisp nommée **F** sans argument qui comporte notamment l'expression initiale pour correspondre aux résultats suivants :

(funcall (F) 0) → 0
(funcall (F) 5) → 9765625
(funcall (F) 3) → 59049
(funcall (F) 19) → 6131066257801
(funcall (F) 20) → " Confirmez-moi le résultat obtenu?"
"Oui il s'agit d'un 20"
(funcall (F) 25) → " Confirmez-moi le résultat obtenu?"
"Oui il s'agit d'un 25"

SOLUTION

Nous savons depuis la première question que *read-base* à une valeur numérique par exemple 10.
Il suffit de ne pas utiliser *read-base* et *print-base* mais de les remplacer par un paramètre.
Comme la fonction fct est sans argument nous pouvons utiliser un lambda pour manipuler la variable:

```
(defun F()
 (lambda(x)
  (cond ((< x 20) (expt x 10))
      (T (and (print "Confirmez-moi le résultat obtenu ?") (format () "Oui il s'agit : ~A " x)))
  )
 )
)
```

ou faire une sous fonction : La notation est la même si l'étudiant passe par une fonction au lieu d'un lambda

```
(defun fct()
 ( defun nouvellefct(x)
  (cond ((< x 20) (expt x 10))
      (T (and (print "Confirmez-moi le résultat obtenu ?") (format () "Oui il s'agit : ~A " x)))
  )
```

```
  )
 )
```

Ne pas oublier que nous sommes en interprétation est que seule la dernière interprétation serait prise en compte donc il nous faut des print et/ou format pour avoir un résultat.

Question2.5 (1,5 points) : La fonction Lisp **expt** a été donnée dans l'expression initiale. Il faut maintenant la faire. Ecrire une fonction récursive lisp nommée **expt2** qui produit la même interprétation qu'**expt**. Définissez tous les paramètres que vous allez utiliser.

SOLUTION

expt calcule la puissance d'un nombre
La fonction expt2 a deux paramètres a et b
a est le nombre élevé à la puissance de b

```
(defun expt2(a b)
 ( cond
   ((equal b 0) 1)
   (T (* a (expt2 a (- b 1))))
 )
)
```

Question 2.6 (1 point) : Modifiez la fonction **F** en incluant maintenant **expt2**. Votre fonction **F** doit être vérifiée. Il vous est demandé de proposer un ensemble de tests pour vérifier son bon fonctionnement.

SOLUTION

Il suffit de remplacer expt par expt2

```
(defun F()
 (lambda(x)
  (cond ((< x 20) (expt2 x 10))
      (T (and (print "Confirmez-moi le résultat obtenu ?") (format () "Oui il s'agit : ~A " x))))))
```

Pour faire les tests il suffit de simuler de nouveau la fonction fct :

```
CL-USER 1 > (funcall (F) 1)
CL-USER 2 > (funcall (F) 2)
CL-USER 3 > (funcall (F) 20)
.....
```

4 LE SYTEME PYTHON (6 POINTS)

Vous disposez de la fonction suivante :

```
F = lambda L : L.reverse()
```

Question 3.1 (1 point): Que fait la fonction **F**, comment éditer le résultat ?

SOLUTION

La fonction F inverse les valeurs d'une liste

```
>>> L = ['1', '2']
>>> F(L)
>>> L
['2','1']
```

Vous disposez des fonctions python suivantes :

car = lambda liste: liste[0]
cdr = lambda liste: liste[1:]
membre = lambda a, liste : a in liste

Question 3.2 (2 points): Avec les fonctions données, et en gardant le principe de la programmation fonctionnelle, écrire une fonction en python nommée **EG** d'arité 2 qui élimine toutes les valeurs à gauche d'une valeur donnée.

>>> EG('a', ['d','c','b']) → ['d', 'c', 'b']
>>> EG('a', ['d','c','a','b']) → ['a', 'b']
>>> EG('1', ['d','a','b']) → ['d', 'a', 'b']

SOLUTION

```
EG = lambda X,L : L if (X == car(L) or not(membre(X,L))) else EG(X, cdr(L))
```

Ou , mais 0,5 point de moins même en cas de bon résultat.

```
def EG (X,L):
    if(membre(X,L) == False):
        print(L)
    else:
        if(X == car(L)):
            print(L)
        else:
            EG(X,cdr(L))
```

Question 3.3 (3 points) Vous avez les résultats d'une suite **S** :

```
>>> S([])
1
>>> S(['1'])
1
>>> S(['1','2'])
2
>>> S(['1','2','3'])
2
>>> S(['1','2','3','4'])
24
>>> S(['1','2','3','4','5'])
24
>>> S(['1','2','3','4','5','6'])
720
>>> S(['1','2','3','4','5','6','7'])
720
>>> S(['1','2','3','4','5','6','7','8'])
40320
>>> S(['1','2','3','4','5','6','7','8','9'])
40320
>>> S(['1','2','3','4','5','6','7','8','9','10'])
```

3628800

En utilisant les fonctions car et cdr, écrire en python la fonction **S** qui réalise la suite.

SOLUTION

```
S = lambda L : 1 if (L == [] or cdr(L) == []) else int(car(L))*int(car(cdr(L)))*int(S(cdr(cdr(L))))
```