3AIT - SCE - SCT

Titouan FREVILLE

June 29, 2016

1 Exercice 1

1.1 Question 1.1

Let t= 2*3 in — t = 6 – old t
Let t=1, v=t+2 — t = 1, v = old t + 2 = 8
in t-v; — t - v = - 7
The final result have to be -7.

1.2 Question 1.2

1.3 Question 1.3

In lisp, tree are List of List under a lot of levels ... here, we could represent the tree with :
(setq tree '(
/ (* (12) (* (* (6) (8)) (4))) (* (/ (8) (/ (10) (5)) (45)) (* (3) (3)))
))
And a way to have it using car and cdr only is :
(car(car(cdr(car(cdr(cdr(car(cdr(car(cdr(cdr(tree)))))))))))

1.4 Questin 1.4

Functional programming is base on mathematical recursivity concept. Sequences work well with funcionnal programming cause they are recursive structure we can easily iterate through recursive functions.

1.5 Question 1.5

If I want to get ATOM 25 in with this functon, I'll place it here :
(() (()) ((() ())) (((() (25) ()))) ((((() () () ())))))

1.6 Question 1.6

Adventage of imperative function are quite low in functionnal programming. We don't really need imperative structure as we can always find a back way to do it recursively. But if you need to manipulate pointers and data without losses, imperative function are always welcomme. Another advantage could be that it ease the transition between full recursive programm and standart learned imperative or object programms.

2 Funciton Programming

If the PIVOT Point is the element where to cut in the list :

```
    Let CUT a function -¿ (list1, list2)
    CUT (list, pivot) = Match list with
    Vide list : (Vide,Vide)
    NonVide list : Premier list == pivot and cons (Premier list,l2)
    Else Let (l1,l2) = CUT(Suite list, pivot)
    If the PIVOT Point is the position where to cut in the list :
    Let CUT a function -¿ (list1, list2)
    CUT (list, pivot) = Match list with
    Vide list : (Vide,Vide)
    NonVide list : 0 == pivot and cons (Premier list,l2)
    Else Let (l1,l2) = CUT(Suite list, pivot-1)
```

3 Exercice 3

```
(defun NumberOfLetterInUniqueList (l letter)
(cond
((null l) 0)
((eq (first l) letter)(+ 1 (NumberOfLetterInUniqueList (cdr l) letter)))
(t (NumberOfLetterInUniqueList (cdr l) letter))
)
)

(defun NumberOfLetter (l letter)
(cond
((null l)())
(t (cons (NumberOfLetterInUniqueList (car l) letter) (NumberOfLetter (cdr
l) letter)))
)
)

(setq l1 '((I n) (t h i s) (t h e r e) (a r e) (m a y) (b e) (t h e) (e)))
```

(NumberOfLetter l1 ’e)

4 Exercice 4

```
def Ask(res):
print ” Hy, Could you enter the list of note you want to play with ?”
choice = raw_input().lower()
if choice in [’do’, ’re’, ’mi’, ’fa
’,’sol’,’la’ ,’si’]:
return Ask([choice] + res)
else:
return res
    def CountTheLa_rec(notes):
if notes == []:
return 0
elif notes[0] == ’la’:
res = 1 + CountTheLa_rec(notes[1:])
return res
else:
return CountTheLa_rec(notes[1:])

    def CountTheLa():
notes = Ask([])
return CountTheLa_rec(notes)
```

I choose to make 3 function instead of one so I have a recursive function that will ask the user to enter a note until he under an unvalid entry, and a function that recursively count the number of la in a list. The function to CountTheLa just call Ask on an empty list, and when the user finished, call the Compter on the acquier list. Function are not case sensitive with the lower() call, and thay are not optmized as I don’t try to organize the provided elements. Another solution whould be to have a single function witch ask for user entries and then add 1 to a compter if it’s a la, exit if the entry is not a note.