TD n°6

Algorithmique : de la dérécursivation

Rem : Il s'agit dans les exercices qui suivent de trouver une version récursive terminale de fonctions déjà étudiées dans les TD précédents. Il faut s'efforcer de suivre le modèle vu en cours avec la fonction multiplication en écrivant la fonction qui calcule effectivement que l'on appellera `xxx_it` *encapsulée dans une fonction lanceuse que l'on appelera* `xxx_term`

Exemple pour la fonction multiplication :

```
(define (mul_term…..)          ; fonction "lanceuse"
     (define (mul_it…….)       ; fonction récursive terminale qui calcule
         (…
           …)
    (mul_it …….. )             ; appel de la fonction qui calcule
```

I 1) Réécrire en DrScheme les versions de `mul` et `mul_term`, `mul_it` vues en cours
2) en utilisant l'outil de trace , faire tracer `mul` et `mul_it` (*pour cela, il faut temporairement « décapsuler » cette fonction de la fonction* `mul_term`) et analyser comment se construit le résultat dans les deux cas.

II De la copie …. En s'appuyant sur le modèle de la fonction multiplication vue en cours, écrire :

1) la fonction `puis_term, et puis_it` *:calcul de la puissance*
2) en utilisant l'outil de trace , faire tracer `puis` et `puis_it` *-voir remarque dans* ***I 2)*** - et analyser comment se construit le résultat dans les deux cas.
3) **Vive l'économie de pensée (suite !)**

En remarquant que les fonctions `mul_it` et `puis_it` ont exactement la même forme. Ecrire une fonction `travaille_it` permettant de passer en argument notamment la nature de l'opération à faire sur l'argument subissant l'opération essentielle.

On pourra alors définir : `(define (mul_term2 x y) (travaille_it ……..))`
et `(define (puis_term2 x y) (travaille_it ……..))`

Idée : Bien réfléchir aux arguments de travaille_it

III Du bégaiement … Ecrire les versions récursives terminales de :

1) la fonction `mul_dicho : la multiplication dichotomique`
2) a fonction `puis_dicho : la puissance dichotomique`
3) **Vive l'économie de pensée (encore et toujours !)**

En remarquant que les fonctions `mul_dicho_it` et `puis_dicho_it` ont exactement la même forme. Ecrire une fonction `travaille_dicho_it` permettant de passer en argument notamment la nature de l'opération à faire sur l'argument subissant l'opération essentielle.

On pourra alors définir : `(define (mul_dicho_term2 x y)(travaille_dicho_it ……)`
et `(deine (puis_dicho_term2 x y) (travaille_dicho_it ……)`

Idée : Bien réfléchir aux arguments de travaille_dicho_it

IV La fonction de Fibonacci

C'est une fonction très utile dans de nombreux domaines. Elle est définie de la façon suivante :
Fib(n) = Fib (n-1) + Fib (n-2) avec Fib (0) = 0 et Fib (1)

1) écrire `Fib` une version récursive non terminale de Fib
2) Faire tracer `Fib` et déterminer le nombre d 'appels récursifs pour n = 10
3) Ecrire `Fib_term` une version récursive terminale de cette fonction
4) Faire tracer `Fib_it` et déterminer le nombre d 'appels récursifs pour n = 10

Conclusions ?