DEUG 1^{ere} année
MIAS 1 et MASS 1 – Programmation fonctionnelle

TD n° 8

Thème : Évaluation de la complexité en temps et en espace (suite)

Exercice 1 :
a) Évaluer la complexité en temps et en espace de la fonction suivante :

```
(define facto
   (lambda (n)
      (if (= 0 n)
          1
          (* n (facto (- n 1))))))
```

b) Une approximation à l'ordre n du nombre e est donnée par $e_n = \sum_{k=0}^{n} \frac{1}{k!}$ qu'on peut encore écrire sous forme récursive :

$$\begin{cases} e_0 = 1 \\ e_n = \frac{1}{n!} + e_{n-1} \end{cases}$$

Écrire une fonction scheme **approx** directement calquée sur les équations ci-dessus qui calcule e_n pour tout entier n et en évaluer la complexité en temps et en espace.

Exercice 2 : Un méthode pour savoir si un entier donné n est premier est de passer en revue tout les entiers compris entre 2 et $\sqrt{2}$. Si l'un d'eux divise n alors n n'est pas premier. On peut programmer en scheme cette méthode par les fonctions suivantes :

```
; trouvdiv : Entier --> Entier
;              n    |-> renvoie le plus petit entier inferieur a
;                         racine de n qui divise n
(define trouvdiv
   (lambda (n)
      (define trouv-aux
         (lamnda (p)
            (cond ((>= p (sqrt n)) 1)
                  ((= 0 (remainder n p)) p)
                  (else (trouv-aux (+ p 1))))))
      (trouv-aux 2)))

; testprem : Entier --> Booleen
;              n    |-> vrai si n est premier
;                       faux sinon
(define testprem
   (lambda (n)
      (= (trouvdiv n 1))))
```

a) On note $T(p, n)$ le temps maximum consommé pour calculer `(trouv-aux p)`, n étant fixé, et on admet que le temps consommé pour évaluer les fonctions prédéfinies `sqrt` et `remainder` est constant. Donner les équations de récurrences permettant de calculer $T(p, n)$ quelque soit p.
En déduire l'ordre de grandeur de la complexité en temps de la fonction `testprem`.