I Algorithmes récursifs

Rôle d'une fonction récursive

1. On définit la fonction `f(n)` où `n` est un entier naturel

```
def f(n) :
    if n == 0 :
        return 0
    elif n == 1 :
        return 1
    else :
        return f(n-2)
```

Quel est le rôle de cette fonction ? Prouver la correction de cette fonction et déterminer sa complexité.
2. Mêmes questions pour la fonction `g(L,x)` où `L` est une liste de réels et `x` un réel.

```
def g(L,x):
    if len(L) == 0 :
        return 0
    else :
        c = L.pop()
        return c + g(L,x)*x
```

Recherche binaire récursive

On considère une liste `L` ordonnée d'éléments. Proposer un algorithme récursif `dich_rec(L,elt)` qui trouve si l'élément `elt` est dans la liste en utilisant une méthode dichotomique.

Inversion d'une liste

Écrire une fonction récursive `inverser(L)` qui renvoie une liste composée des éléments de la liste `L`, rangés dans l'ordre inverse.

Test de primarité

Soit n un entier strictement supérieur à 1. Pour déterminer si n est premier, on propose de tester la divisibilité de n par les entiers i compris entre 2 et $n-1$. Dès qu'un tel entier divise n, ce dernier n'est pas premier. Proposer un algorithme récursif qui teste si n est premier.

Nombre d'occurrences

Écrire une fonction récursive `nombre(lettre,chaine)` qui compte le nombre d'occurrences d'un caractère `lettre` dans une chaîne de caractères `chaine`.

II Récursivité croisées

1. Quel est le rôle de chacune de ces fonctions ?

```
def fct1(n) :
    if n == 0 :
        return True
    else :
        return fct2(n-1)
def fct2(n) :
    if n == 0 :
        return False
    else :
        return fct1(n-1)
```

2. Écrire une fonction récursive `fct3` qui donne le même résultat que `fct1` sans utiliser `fct2`.