DS 1 - Informatique - CPBX 1

Durée : 1h30

27 novembre 2013

Exercice 1

1) Quel est le codage binaire (entier non signé), sur 10 bits, de 142 ?
2) Quel est le plus grand entier et le plus petit entier que l'on puisse coder sur 10 bits, à l'aide du codage binaire (entier non signé) ?
3) Quel est le codage en complément à 2 (entier signé), sur 10 bits de 142 et de -142 ?
4) Quel est le plus grand entier et le plus petit entier que l'on puisse coder sur 10 bits, en utilisant le complément à 2 ?

Vous justifierez vos réponses.

Exercice 2

On définit l'hyperfactorielle de n par

$$H(n) = \prod_{i=1}^{n} k^k = 1^1.2^2.3^3 \dots (n-1)^{n-1}.n^n$$

Proposer une fonction $hyperfactorielle(n)$ qui prend en paramètre un entier n et qui calcule l'hyperfactorielle de n.

Exercice 3

Soit U_n la suite définie par :

$$\begin{cases} U_n = U_{n-1} + 5.U_{n-2} + 3.U_{n-3} \\ U_0 = 1 \\ U_1 = 0 \\ U_2 = -1 \end{cases}$$

1) Écrire un programme `suite(n)` qui renvoie le terme U_n de la suite précédente.
2) Écrire un programme `somme_suite(n)` qui renvoie la somme des n premiers termes (de 0 à $n-1$).

Exercice 4

Écrire un programme $flou(t)$ qui prend en paramètre un tableau t et qui modifie t de sorte que, pour tout entier k, la nouvelle valeur de $t[k]$ soit égale à la moyenne entre les anciennes valeurs de $t[k-1]$ et $t[k+1]$. Par convention, on fixe $t[-1] = t[n] = 0$ où n est la taille de t.

Par exemple, si $t = [6, 0, 3, 1, 1]$, alors, après exécution de $flou(t)$, t vaudra $[0.0, 4.5, 0.5, 2.0, 0.5]$.

Exercice 5

Un nombre parfait est un nombre tel que la somme de ses diviseurs stricts est égal à lui même.

Par exemple, 6 est un nombre parfait car les diviseurs stricts de 6 sont 1, 2 et 3, et leur somme $1 + 2 + 3$ est égale à 6.

1) Parmi les nombres suivants, lesquels sont parfaits : 5, 6, 12, 28, 42 ;
2) Écrire une fonction `est_parfait(n)` qui renvoie `True` si n est parfait et `False` sinon.
3) Proposez une fonction `liste_nombre_parfait(n)` qui prend en paramètre un entier n non nul et qui renvoie une liste de tous les nombres parfaits entre 1 et n inclus.
4) Proposez une fonction `prochain_nombre_parfait(n)` qui renvoie le plus petit nombre parfait strictement supérieur à n.

Solution : exercice 1

1. $142 = 2^7 + 2^3 + 2^2 + 2^1 = \overline{00010001110}^2$
2. Un entier non signé codé sur $n = 10$ bits est compris entre 0 et $2^n - 1 = 1023$.
3. $complement_2(142) = binaire(142) = 00010001110$

   $complement_2(-142) = complement(binaire(142)) + 1 = 11101110010$
4. Un entier signé codé sur $n = 10$ bits est compris entre $-2^{n-1} = -512$ et $2^{n-1} - 1 = 511$.

Solution : exercice 2

```
def puissance( n,m ):
    res = 1
    for i in range( m ):
        res = res * n
    return res

def hyperfactorielle(n):
    res = 1
    for i in range( 1, n+1 ):
        res=res*puissance(i,i)
    return res
```

Solution : exercice 3

```
def suite(n):
    un3 = 1
    un2 = 0
    un1 = -1
    if n==0 : return un3
    if n==1 : return un2
    if n==2 : return un1
    for i in range(3, n+1):
        un = un1 + 5*un2 + 3*un3
        un3 = un2
        un2 = un1
        un1 = un
    return un

def somme_suite(n):
    res = 0
    for i in range( n ):
        res = res + suite(i)
    return res
```

Solution : exercice 5

```
def est_parfait(n):
    somme = 0
    for entier in range( 1, n ):
        if n % entier == 0 :
            somme = somme + entier
    if somme == n :
        return True
    return False

def liste_nombre_parfait(n):
    result = []
    for i in range(1,n+1) :
        if est_parfait(i) :
            result.append( i )
    return result

def prochain_nombre_parfait(n):
    next = n+1
    while not( est_parfait(next) ) :
        next = next + 1
    return next
```

Solution : exercice 4

```
def flou(t):
    if len(t) == 1 :
        t[0] = 0
    if len(t) >= 2 :
        ancien = [ t[i] for i in range(len(t)) ]
        t[0] = ancien[1] / 2.0
        t[ len(t)-1 ] = ancien[ len(t)-2 ]/2.0
        for i in range( 1, len(t)-1 ):
            t[i] = ( ancien[i-1] + ancien[i+1] )/2.0
```