Tableau de conversion pour FLOAT

n	2^n										
13	8192										
12	4096										
11	2048										
10	1024										
9	512										
8	256										
7	128										
6	64										
5	32										
4	16										
3	8										
2	4										
1	2										
0	1										
-1	0.5										
-2	0.25										
-3	0.125										
-4	0.0625										
-5	0.03125										
-6	0.015625										
-7	0.0078125										
-8	0.00390625										
-9	0.00195312										
-10	0.000976562										
-11	0.000488281										

n	10	9	8	7	6	5	4	3	2	1	0
2^n	1024	512	256	128	64	32	16	8	4	2	1

```python
def algo_malcolm() :
    x = 1.0
    y = x + 1.0
    i = 0
    while y-x == 1.0 :
        x = 2.0 * x
        y = x + 1.0
        i = i + 1
    return i
```

```python
def colonne_tableau(x, n) :
    """
    permet de remplir la colonne du tableau de
        conversion pour FLOAT
    on s'arrête à n = -11
    on impose x <= 8192 (2**13)
    """
    if x > 8192 :
        print("nombre trop grand")
        exit()
    if n == -12 :
        return ''
    else :
        if x-2**n < 0 :
            return '0' + colonne_tableau(x,n-1)
        else :
            return '1' + colonne_tableau(x-2**n,n-1)
```

Exercice 1

Pour chaque nombre binaire donné, déterminer la valeur décimale (en base 10) correspondante.

La série de bit est découpée en `signe` (1 bit) / `exposant` (11 bits) / `mantisse` (52 bits).

Parmi les valeurs données, il y a la masse d'un électron (en kg), le diamètre de la Voie Lactée (en m), le nombre d'Avogadro (en mol^{-1}) : saurez-vous les retrouver ?

a) `0 / 01111111100 / 0011011101001011110001101010011111101111100111011011`
b) `0 / 10000001001 / 0100001110010000000000000000000000000000000000000000`
c) `0 / 10000001100 / 0011100001111001000000000000000000000000000000000000`
d) `0 / 10000000000 / 0000001111010111000010100011110101110000101000111101`
e) `0 / 10001001101 / 1111110111101001111100010000101010001101001101100001`
f) `0 / 10001000100 / 1011000110101110010011010110111000101110111101010000`
g) `0 / 01110011011 / 0010011110011010100111001000000001110011110110001011`

Exercice 2

On fait l'inverse : on convertit un nombre réel (flottant) en nombre binaire.

On donnera les chiffres de la mantisse jusqu'à 2^{-11} compris.

a) 299792458.0 $m.s^{-1}$: la vitesse de la lumière dans le vide
b) 1.380658×10^{-23} $J.K^{-1}$: la constante de Boltzmann
c) 22.414 $L.mol^{-1}$: le volume molaire
d) 52.917×10^{-12} m : le rayon de Bohr de l'atome d'hydrogène
e) -96485.31 C : la charge électrique d'une mole d'électron
f) 1.67×10^{-27} kg : la masse d'un nucléon
g) 6.63×10^{-34} J.s : la constante de Planck

Exercice 3

a) Quelle est la représentation binaire de la plus grande valeur possible pour un float 64 bits ?
b) Idem pour la plus petite valeur positive ?

Exercice 4 sur machine

a) Exécuter l'algorithme de Malcolm sur votre machine. Pourquoi s'arrête-t-il ?
b) Tester ce que renvoie `0.1+0.1+0.1-0.3 == 0`.
c) Est-ce que la fonction `colonne_tableau` permet bien de remplir la colonne du tableau de conversion Float ?

Correction des exercices

Exercice 1

a) 0.152
b) 1294.25
c) 9999.125
d) 2.03
e) 6.02×10^{23} : nombre d'Avogadro (en mol^{-1})
f) 1.0×10^{21} : diamètre de la Voie Lactée (en m)
g) 9.109×10^{-31} : masse de l'électron (en kg)

Exercice 2

a) `0 / 10000011011 / 0001110111100111100001001010000000000000000000000000000`
b) `0 / 01110110011 / 0000101100001110110111110110110010010001110000011011`
c) `0 / 10000000011 / 0110011010011111101111100111011011001000101101000100`
d) `0 / 01111011100 / 1101000101110110011111011010000100110000000110101011`
e) `1 / 10000001111 / 0111100011100101010011110101110000101000111101011100`
f) `0 / 01110100110 / 0000100010011111001111111000000111101011111011110100`
g) `0 / 01110010000 / 1011100010100011100110101100100101010000000110100101`

Exercice 3

a) on a vu que l'exposant le plus qu'on pouvait prendre était 1023. Donc on code sur 11 bits 1023+1023=2046, ce qui donne `11111111110`.
On veut le plus grand nombre, donc un nombre positif : le bit de signe sera donc `0`.
La mantisse sera la plus grand possible : 52 bits à 1 : `111111... 1`.
`0 / 11111111110 / 1111111111111111111111111111111111111111111111111111`
Ce qui correspond à la valeur 1.7977×10^{308}.
b) Pour trouver la plus petite valeur, il faut prendre l'exposant le plus petit possible, soit -1022. Donc son code sur 11 bits sera -1022+1023=1, soit `00000000001`.
On cherche la plus petite valeur positive, donc le bit de signe sera `0`.
La mantisse sera la plus petite possible donc composée de 52 bits à 0.
`0 / 00000000001 / 0000000000000000000000000000000000000000000000000000`
Ce qui correspond à la valeur 2.2251×10^{-308}.

Remarque : la norme `IEEE 754` précise les points suivants

`+0.0 : 0 / 00000000000 / 0000000000000000000000000000000000000000000000000000`

`-0.0 : 1 / 00000000000 / 0000000000000000000000000000000000000000000000000000`

$+\infty$: `0 / 11111111111 / 0000000000000000000000000000000000000000000000000000`

$-\infty$: `1 / 11111111111 / 0000000000000000000000000000000000000000000000000000`

`nan : 0 ou 1 / 11111111111 / mantisse autre que 52 bits à 0`