ExerciceType Bac **Épreuve écrite**
NSI *Terminale* Année .....

Exercice 2

Récursivité

Étant donné un entier naturel N, la décomposition : $N = a_n b^n + a_{n-1} b^{n-1} + ... + a_1 b^1 + a_0 b^0$ donne l'écriture en base b de N : $\overline{N}^b = \overline{a_n a_{n-1} ... a_1 a_0}^b$ avec $0 \leq a_i < b$

Exemple : On utilise une succession de divisions euclidienne pour déterminer les coefficients a_i

7 7	1 6		4	1 6	
1 3	4	et	4	0	l'écriture en hexadécimal de 77 est donc 4D

On rappelle que si $b > 10$:
les nombres 10,11, *etc. sont représentés par les lettres de l'alphabet A, B, C...*

Dans l'autre sens le nombre $\overline{1001101}^2 = 1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0$ s'écrit 77 en décimal.

? Question 1:

1. Quel est le plus grand entier que l'on peut représenter en binaire sur 1 octet?
2. Quel est en décimal le nombre représenté en hexadécimal par $\overline{FF}^{16}$?

? Question 2:

1. Que contient la variable `r` après exécution de ce code:

```
r = 77%16
```

2. Que contient la variable `q` après exécution de ce code:

```
q = 77//16
```

Voici une fonction qui est sensée donner l'écriture en base 10 d'un entier naturel écrit en base b
La liste `signes` permet d'accéder au symbole 'i' se trouvant à l'indice i
L'instruction `signes.index("A")` renvoie l'indice du symbole "A" soit 10.

```
signes=["0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F"]

def bTodec(mot,b):
# b est un entier
    assert (b>1 and b<17) ,"b doit être compris entre 2 et 16"
    assert (type(mot) == str), "Le nombre n doit être une chaîne de caractères"
    resul = 0
    p = len(mot)
    for i in range(p):
        resul = resul + signes.index(mot[i])*b**(p-1-i)
    return resul
```

? Question 3:

Recopier et compléter le programme Python ci-dessus sur votre copie en remplaçant les??? par les bonnes instructions.

? Question 4:

En remarquant par exemple que:
`bTodec('10011',2)`$=1\times 2^4+$ `bTodec('0011',2)`
L'objectif est d'écrire une version récursive de la fonction `bTodec`.

1. Quelle est la condition d'arrêt et que doit elle renvoyer?
2. Écrire en Python une version récursive de la fonction `bTodec`.

On pourra utiliser du slicing appliqué aux chaînes de caractères.

```
# slicing
mot = '110001100'
print(mot[1:])
print(mot[2:])
print(mot[3:])
```

Affichage des résultats :

```
10001100
0001100
001100
```

```
def bTodecr(mot,b):
    p=len(mot)
    if p<2:
        return signes.index[mot[0]]
    else:
        return signes.index(mot[0])*b**(p-1) + bTodec(mot[1:],b)
```