TD 4

1 Bases arithmétiques

Exercice 1: Quelques conversions "de base"

Effectuer les conversions suivantes :

— Du binaire en décimal : 1010, 1101101.
— Du décimal en binaire : 63, 318.
— De l'hexadécimal en décimal : 3E , FAC.
— Du décimal en hexadécimal : 44, 1000.
— De l'hexadécimal en binaire : 101011111000.
— Du binaire en hexadecimal : 89A0.

Exercice 2: 111111111...

On considère le nombre dont l'écriture binaire est constituée de `n` '1'.

Quelle est sa valeur ? Quelle est son écriture hexadécimale ?

Exercice 3: Nombre de chiffres en base b

Écrire un programme python `nombre_chiffres(n,b)` qui renvoie le nombre de chiffres de l'entier `n` en base `b`. Ce programme ne manipulera pas de listes.

Exercice 4: Jackpot

Écrire un programme python `que_des_sept(n)` qui renvoie `True` si l'entier `n` appartient à la suite 7, 77, 777, 7777... ; `False` dans le cas contraire. Ce programme ne manipulera pas de listes.

Exercice 5: Des chiffres et peut-être des lettres

Écrire un programme python `lettre_en_hexadecimal(n)` qui renvoie `True` si l'écriture de l'entier `n` en base hexadécimale contient une lettre (i.e A, B, C, D, E ou F), `False` sinon. Ce programme ne manipulera pas de listes.

Exercice 6: Nombre de 1 en écriture binaire

Écrire un programme python `nombre_de_1(n)` qui renvoie le nombre de 1 de l'entier `n` en écriture binaire. Ce programme ne manipulera pas de listes.

Exercice 7: Fonction mystère

Simuler l'exécution du programme suivant pour `n` = `301` et `n` = `1010` en donnant les valeurs successives de `res`, `aux` et `n`.

```
def mystere(n):
    res = 0
    aux = 1
    while (n != 0):
        b = n % 10
        if b < 2:
            res = res + aux*b
            aux = aux*2
            n = n / 10
        else:
            print("Erreur")
            return None
    return res
```

Que fait le programme suivant ? Quand affiche-t-il une erreur ?

2 Opérations en écriture binaire

Dans tous les exercices suivants, on supposera que `t1` et `t2` sont deux tableaux représentant chacun l'écriture binaire d'un nombre, le bit faible étant en première position. Par exemple, l'entier 12 sera modélisé par le tableau `[0,0,1,1]`.

Les résultats des fonctions devront être également renvoyés sous cette forme.

Enfin, vous ne devez pas utiliser les fonctions qui convertissent `t1` et `t2` dans des bases natives afin de profiter des opérateurs déjà implémentés dans la base native.

Exercice 8: Addition

Écrire un programme python `addition(t1,t2)` qui calcule l'addition de `t1` et de `t2`.

Exercice 9: Soustraction

Écrire un programme python `soustraction(t1,t2)` qui calcule la soustraction de `t1` par `t2`.

Exercice 10: Reste de la division euclidienne

Écrire un programme python `reste(t1,t2)` qui calcule le reste de la division euclidienne de `t1` par `t2`.

Exercice 11: Quotient de la division euclidienne

Écrire un programme python `quotient(t1,t2)` qui calcule le quotient de la division euclidienne de `t1` par `t2`.

Exercice 12: Multiplication

Écrire un programme python `multiplication(t1,t2)` qui calcule la multiplication de `t1` et de `t2`.

Exercice 13: Racine carrée

Écrire un programme python `racine(t1)` qui calcule la racine carré de `t1` qu'on tronquera à l'unité.