Exercice 1 : Vrai ou Faux

1) Si l'écriture binaire d'un entier naturel se termine par n zéros alors cet entier est divisible par 2^n.
2) En base 8, on utilise des chiffres 1 à 8.
3) Avec n bits, les entiers représentables à l'aide de leur écriture binaire sont inférieurs à 2^n.
4) Soit m le mot binaire sur n bits associés à un entier naturel A en base 10. Pour convertir $\frac{A}{2}$ il suffit de supprimer un bit à la fin de m.
5) L'entier 170 s'écrit AA en hexadécimal :

Base 10	Base 16	Base 2
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

6) Si les entiers non signés sont codés sur des mots de 4 bits alors 1011+0101=0000
7) Avec des mots de 4 bits, on peut représenter tous les entiers relatifs de -8 à +8 compris
8) On code les entiers relatifs en complément A2 sur 5 bits. L'entier 10 est codé par 01010 et l'entier -10 par 11010.
9) Si des entiers non signés sont codés sur 8 bits alors il y a un nécessairement dépassement de capacité si on calcule la somme de 2 nombres dont le MSB est 1
10) Si des entiers signés sont codés en complément A2 alors si la somme de deux nombres commençant par zéro est un nombre commençant par 1, alors il y a une erreur.

1) Vrai Si $A = c_{n+1}2^{n+1} + 0 \times 2^n + \cdots + 0 \times 2^0 = c_{n+1}2^{n+1} = c_{n+1} \times 2^n \times 2$
   Et donc $\frac{A}{2^n} = 2c_{n+1} = 2$
2) Faux, en base 8, les chiffres utilisés sont compris entre 0 et 7 inclus.
3) Vrai, le plus entier est $2^n - 1$
4) Vrai, nous avons vu que la conversion en base 2 consistait en une suite de division par 2 jusqu'à obtenir un quotient nul. En partant de A/2, on enlève une division et onc le MSB de A
5) Vrai : $(170)_{10} = (1010\ 1010)_2 = (AA)_{16}$
6) Vrai, on trouve 10000 qui devient 0000 on a donc un overflow ou erreur de dépassement de capacité
7) Non, -8 à +7 compris
8) Non, il ne suffit pas de changer le MSB, le complément A2 est
   $(01010) \rightarrow (10101) \rightarrow (10110)$
9) Vrai : 1…..+1….. conduit forcément à une retenue
10) Vrai car si deux nombres sont négatifs alors leur somme l'est également

Exercice 2 :

Soient des nombres écrits en binaire :

1) Calculer la somme de 100110 + 001101 en posant l'addition
2) Traduire le calcul en décimal

$$\begin{vmatrix} 100110 \\ 001101 \\ 110011 \end{vmatrix}$$

$$(100110)_2 = 2 + 4 + 32 = 38$$

$$(001101)_2 = 1 + 4 + 8 = 13$$

$$(110011)_2 = 1 + 2 + 16 + 32 = 51$$

Exercice 3 :

Quelle affirmation est exacte ?

1) Un nombre occupe 8 fois moins de place en mémoire s'il est représenté en octet plutôt que par des bits
2) Un nombre écrit en hexadécimal comporte 8 fois moins de chiffres que s'il est écrit en binaire
3) Un nombre pair a une écriture binaire qui se termine par un 0

1) Non, il reste tout autant de bits.
2) Non, 4 fois
3) Le MSB s'obtient avec le reste de la 1e division par 2, donc si pair ce reste est bien nul

Exercice 4 :

Quelle la valeur en binaire de 1001×111 ?

			1	0	0	1
		×		1	1	1
			1	0	0	1
		1	0	0	1	
	1	0	0	1		
1	1	1	1	1	1	

Exercice 5 :

On utilise des mots de 5 bits pour coder des entiers relatifs en complément A2. Comment est codé -2 ?

$$-2 \rightarrow (00010) \rightarrow (11101) \rightarrow (11110)$$

Exercice 6 :

1) Ecrire une fonction qui prend en paramètre un entier naturel exprimé en base 10 et renvoie l'écriture en base deux de ce nombre. Le paramètre en entrée est de type *int* et la valeur en sortie est de type *str*.
2) Ecrire une fonction qui prend en paramètre un entier naturel exprimé en base deux et renvoie l'écriture en base 10 de ce nombre. Le paramètre en entrée est de type *str* et le paramètre en sortie de type *int*
3) Reprendre le programme précédent en remarquant qu'il est possible d'écrire :

$$(nbre)_{10} = b_0 + 2(b_1 + 2(b_2 + \cdots 2(b_{n-1} + 2b_n)..)$$

```
def conversion_binaire(nbre):
    chaine=""
    if nbre==0:
        return "0"
    while nbre!=0:
        nbre,r=nbre//2,nbre%2
        chaine=str(r)+chaine#pour avoir le MSB en 1e
    return chaine


def conversion_dec(mot):
    nombre=0
    for i in range(len(mot)):
        nombre=nombre+int(mot[i])*2**(len(mot)-1-i)
    return nombre
print(conversion_dec(conversion_binaire(10)))


def conversion_dec2(mot):
    nombre=int(mot[0])
    for i in range(1,len(mot)):
        nombre=nombre*2+int(mot[i])
    return nombre

print(conversion_dec2(conversion_binaire(10)))
```

Exercice 7 :

Pour coder des entiers relatifs on utilise ici 5 bits.

1) Combien de nombres peut-on coder et lesquels ?
2) Comment est codé le nombre 6 ?
3) Comment est codé le nombre -7 ?

1) On peut donc coder 32 nombres entiers compris entre $-2^4 = -16$ à $2^4 - 1 = 15$
2) $(6)_{10} = (00110)_2$
3) $(7)_{10} = (00111)_2$ donc $(-7)_{10} = (11001)_2$

Exercice 8 :

1) Combien de secondes peut-on écrire sur un mot de 32 bits signé ?
2) Est-il possible de coder le nombre de secondes qui se sont écoulées depuis le 1e janvier 1970 à 0h ?

1) Avec 32 bits signé, on peut atteindre $10^{31} - 1$
2) $N \approx 52 * 365 * 24 * 3600 < 10^{31} - 1$ donc pas de dépassement. Il faut attendre typiquement 48 ans pour avoir un problème de codage

Exercice 9 :

1) Soit un nombre entier naturel x nécessitant n bits pour être décrit en binaire. Estimer, en fonction de n, le nombre de chiffres m nécessaire pour exprimer $(x)_{10}$.
2) Commenter le résultat précédent pour $n = 10$
3) Donner un ordre de grandeur, évaluée en Go, de la place occupée par 2^{30} blocs de quatre octets chacun
4) La valeur de la capacité de mémoire vive d'un ordinateur est de 16Go, quelle est le plus grand entier numérisable ?

1) On a typiquement $x \approx 10^m \approx 2^n$ donc $m = n \log_{10} 2$
2) $m \approx 3$, on retrouve que $10^3 \approx 2^{10}$ ce qui explique la confusion acceptée entre $1ko = 1000o$ et $2^{10}o = 1024o = 1kio \approx 1ko$
3) $\frac{2^{30}*4}{10^9} = \frac{2^{10^3}*4}{10^9} \approx \frac{10^{3^3}*4}{10^9} \approx 4Go$ (capacité de stockage typique d'un DVD)
4) $n = 16 * 10^9 * 8$ représente le nombre de bits utilisables soit $n = 38531839444 \approx 10^{11}$ et donc un nombre pouvant dépasser $2^{10^{11}} \approx 10^m$ et avec $m = 10^{11}$. On pourra remarque qu'à partir de x= 10^{10^6} l'ordinateur commence à mettre beaucoup de temps à écrire ce nombre.

Exercice 10 :

1) Ecrire une fonction qui prend en paramètre un entier naturel et renvoie la liste des chiffres de son écriture binaire rangés dans l'ordre inverse. Pour $n = 11$, on obtient [1,1,0,1].
2) Ecrire une fonction qui prend en paramètre la liste des chiffres de l'écriture binaire d'un entier naturel rangés dans l'ordre inverse et renvoie cet entier naturel.

Corrigé

```
def conversion_binaire(nbre):
    liste=[]
    if nbre==0:
        return [0]
    while nbre!=0:
        nbre,r=nbre//2,nbre%2
        liste.append(r)
    return liste
print(conversion_binaire(11))
print(bin(11))

def conversion_dec(liste):
    nombre=0
    for i in range(len(liste)):
        nombre=nombre+liste[i]*2**(i)
    return nombre
print(conversion_dec(conversion_binaire(10)))


def conversion_dec2(liste):
    nombre=liste[-1]
    for i in range(len(liste)-2,-1,-1):
        nombre=nombre*2+int(liste[i])
    return nombre

print(conversion_dec2(conversion_binaire(10)))
```