REPRÉSENTATION DES NOMBRES

Exercice 1 : base 2

1) Convertir les nombres binaires suivants vers leur équivalent décimal :

(a) $(1011)_2$ = **11** (b) $(1101\ 0011)_2$ = **211** (c) $(1101\ 1011)_2$ = **219**

2) Trouver la représentation binaire des nombres suivants :
(a) 85 = **0101 0101** (b) 126 = **0111 1110** (c) 2 000 = **0111 1101 0000**

3) Donner les représentations en base deux des nombres 1, 3, 7, 15, 31 et 63. Expliquer le résultat.

1 = **$(1)_2$** ; 3 = **$(11)_2$** ; 7 = **$(111)_2$** ; 15 = **$(1111)_2$** ; 31 = **$(1\ 1111)_2$** ; 63 = **$(11\ 1111)_2$**

$1 = 1\times 20$
$3 = 1\times 2^1 + 1\times 2^0$
$7 = 1\times 2^2 + 1\times 2^1 + 1\times 2^0$
$15 = 1\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0$
$31 = 1\times 2^4 + 1\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0$
$63 = 1\times 2^5 + 1\times 2^4 + 1\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0$

Le nombre en base 2 formé de n digits égal à 1 est égal en base 10 à :
$1 + 2^1 + 2^2 + 2^3 + \ldots\ldots + 2^{n-1} = (2^n - 1)/(2 - 1) = 2n - 1$
(somme des n premiers termes d'une suite géométrique de raison 2 et de premier terme égal à 1). *A voir plus tard en mathématique*

Et on vérifie bien que :

- $(1)_2 = 2^1 - 1 = 2$
- $(11)_2 = 2^2 - 1 = 3$
- $(111)_2 = 2^3 - 1 = 7$
- $(1111)_2 = 2^4 - 1 = 15$
- $(11111)_2 = 2^5 - 1 = 31$
- $(111111)_2 = 2^6 - 1 = 63$

4) Pour additionner des nombres binaires, on utilise la même méthode qu'avec les décimaux.

On a : $(0)_2 + (0)_2 = (0)_2$; $(1)_2 + (0)_2 = (0)_2 + (1)_2 = (1)_2$; $(1)_2 + (1)_2 = (10)_2$.

Calculer $(1011\ 1101)_2 + (10\ 1101)_2$ puis vérifier le calcul avec les équivalents décimaux.

$(1011\ 1101)_2 + (10\ 1101)_2$ = **1110 1010**

$(1011\ 1101)_2$ = 189 et $(10\ 1101)_2$ = 45 de plus 189 + 45 = **234** et 234 = **(1110 1010)**

7) Établir, à la main, un algorithme qui permet la conversion décimale d'un nombre binaire entré par l'utilisateur.

On écrit ici une fonction python qui prend en paramètre un entier naturel exprimé en base 2(chaîne de caractères) et renvoie l'écriture en base dix de ce nombre.

```
def base2vers10(b) :
    n = 0
    for bit in b:
        n = 2 * n + int(bit)
    return n
```

8) Établir, à la main, un algorithme qui permet la conversion binaire d'un nombre décimal entré par l'utilisateur.

On écrit ici une fonction python qui prend en paramètre un entier naturel exprimé en base 10 et renvoie l'écriture en base 2 de ce nombre (chaine de caractères).

```
def base10vers2(n) :
    if n == 0 :
        return " 0 "
    b = " "
    while n != 0 :
        r =  n % 2
        n = // 2
        b = str(r) + b # on ajoute le reste de la chaîne
         return b
```

Exercice 2 : base 16

1) Trouver la représentation en base 10 des nombres suivants :
   (a) $(4E2C)_{16}$ = **20012** (b) $(ABC)_{16}$ = **2748** (c) $(281EF)_{16}$ = **164335** (d) $(92)_{16}$ = **146**

2) Trouver la représentation en base 16 des nombres suivants :
   (a) 3685 = **$(E65)_{16}$** (b) 5 425 = **$(1531)_{16}$** (c) 9 379 = **$(24A3)_{16}$**

3) Convertir les nombres binaires suivants en hexadécimal :
   (a) $(1001\ 1011)_2$ = **$(9B)_{16}$** (b) $(1000\ 1001)_2$ = **$(89)_{16}$** (c) $(0011\ 1001\ 0101)_2$ = **$(395)_{16}$**

4) Convertir les nombres hexadécimaux suivants vers leur équivalent binaire :
   (a) $(F3)_{16}$ = **$(1111\ 0011)_2$** (b) $(37E)_{16}$ = **$(0011\ 0111\ 1110)_2$** (c) $(15)_{16}$ = **$(0001\ 0101)_2$**

5) Les couleurs en HTML sont définies par 3 nombres hexadécimaux représentant les tons de **R**ouge, de **V**ert et de **B**leu (selon le codage *RGB* (*Red Green Blue)*, en français : *RVB*) de la couleur choisie.

   Ainsi la syntaxe de codage d'une couleur en HTML est la suivante :

   couleur="#RRVVBB"
   RR, VV et BB représentent respectivement un nombre hexadécimal entre 00 et FF pour le Rouge, le Vert et le Bleu.

Ainsi, plus de 16 millions de couleurs sont disponibles pour colorer les pages web.

Le tableau ci-dessous représente la correspondance des valeurs en hexadécimal et en décimal pour les niveaux de gris des couleurs standards.

Nom(s) de couleur	Échantillon	RGB (hex)	RGB (décimal)
black		#000	rgb(0, 0, 0)
dimgray, dimgrey		#696969	rgb(105, 105, 105)
gray, grey		#808080	rgb(128, 128, 128)
darkgray, darkgrey		#A9A9A9	rgb(169, 169, 169)
silver		#C0C0C0	rgb(192, 192, 192)
lightgray, lightgrey		#D3D3D3	rgb(211, 211, 211)
gainsboro		#DCDCDC	rgb(220, 220, 220)
whitesmoke		#F5F5F5	rgb(245, 245, 245)
white		#FFF	rgb(255, 255, 255)

(a) Dans chaque cas, convertir le code hexadécimal en RGB décimal.

(i) `#8B4513` = **`rgb(139, 69, 19)`** (ii) `#7CFC00` = **`rgb( 124, 252, 00)`**
(iii) `#C71585` = **`rgb(199, 21, 133)`**

(b) Dans chaque cas, convertir le code RGB décimal en hexadécimal.

(i) rgb(148, 0, 211) = **`#9400D3`** (ii) rgb(175, 238, 238) = **`#AFEEEE`**
(iii) rgb(250, 128, 114) = **`#FA8072`**

Exercice 3 : entiers signés (Méthode complément à deux)

1) Quels entiers relatifs peut-on représenter avec des mots de 16 bits ? De 32 bits ? De 64 bits ?

2) Trouver la représentation binaire sur 8 bits des entiers relatifs 0, -128, -127 et 127.

0 = **$(0)_2$** , -128 = **$(1000\ 0000)_2$** , -127 = **$(1000\ 0001)_2$** et 127 = **$(0111\ 111)_2$** .

3) Trouver la représentation décimale des entiers relatifs dont la représentation binaire sur 8 bits est :
(a) 0000 0011 = **$(3)_{10}$** (nombre positif)

(b) 1001 0000 = **$(-48)_{10}$** Le complément à deux de 0010000 vaut 1101111 + 1 = 110000 soit (42)

(c) 0111 0111 = **$(119)_{10}$** (nombre positif)

(d) 1000 0101 = **$(-123)_{10}$** Le complément à deux de 0000101 vaut 1111010 + 1 = 1111011 soit (123)

4) Calculer la représentation binaire sur 8 bits de l'entier relatif 16, puis de son opposé.
On a 16 = $(0001\ 0000)_2$ sur 8 bits.
L'opposé de 16 est –16. On inverse les bits soit : 1110 1111 et on ajoute 1, on obtient : 1111 0000
Donc **–16 = (1111 0000)**

Exercice 4 : Opérations

1) Ecrire la table d'addition et de multiplication binaire.

Tables d'addition en base deux nécessaires pour poser des additions en base 2.

Additions : Table de zéro en base deux	Additions : Table de un en base deux	Additions : Table de deux en base deux
0 + 0 = 0	1 + 0 = 1	10 + 0 = 10
0 + 1 = 1	1 + 1 = 10	10 + 1 = 11
0 + 10 = 10	1 + 10 = 11	10 + 10 = 100

Toutes les tables de multiplication en base 2. Tables allant de zéro à deux en base deux.
Ces tables peuvent, dans le cas de bases non-décimales comme la base 2_{10}, servir à poser des multiplications en base deux.

Multiplications Table de zéro en base deux	Multiplications Table de un en base deux	Multiplications Table de deux en base deux
0 × 0 = 0	1 × 0 = 0	10 × 0 = 0
0 × 1 = 0	1 × 1 = 1	10 × 1 = 10
0 × 10 = 0	1 × 10 = 10	10 × 10 = 100

2) a. Ecrire les nombres décimaux suivant en binaire sur 8 bits (1 octet) :
$(135)_{10}$ = **$(1000\ 0111)_2$** ; $(100)_{10}$ = **$(0110\ 0100)_2$**
b. Additionner $(135)_{10}$ et $(100)_{10}$ en binaire
1000 0111 + 0110 0100 = 1110 1011
c. Additionner $(135)_{10}$ et $(135)_{10}$ en binaire.
1000 0111 + 1000 0111 = 0000 1110 sur 8 bits .
Que constatez-vous ? Comment appelez-vous ce problème ?
Il faut un 9ième bit pour coder le résultats qui est : 1 0000 1110
On a ici un dépassement de mémoire.

3) Pour multiplier deux nombres codés en binaire, il suffit de procéder comme en base 10 et de décaler d'un rang quand on change de bit du multiplicateur.
a. Calculer $(6)_{10} \times (3)_{10}$ en binaire.
$(6)_{10}$ = $(110)_2$ et $(3)_{10}$ = $(11)_2$
En binaire : 110 × 11 = 1 0010
b. Combien de bit son nécessaire pour coder cette opération ? **5 bits**

4) Donner le résultat des opérations binaires suivantes ainsi que le nombre de bits nécessaires au codage du résultat en binaire :
a. $(1101)_2 + (111)_2$ = **$(1\ 0100)_2$, il faut 5 bits**
b. $(1101)_2 \times (111)_2$ = **$(101\ 1011)_2$, il faut 7 bits**
c. $(1111)_2 + (10)_2$ = **$(1\ 0001)_2$, il faut 5 bits**