Données 1 : **Binaire, octet et stockage**

I – Problème réel : magie magie

II – La base 10 (le décimal)

2.1 - Codification des nombres

La **case de poids faible** est **la plus à droite**. Elle code la valeur _____ qu'on peut écrire _____

A chaque déplacement à gauche, la case code _____ fois plus.

DOC.1

	Millier	Centaine	Dizaine	**Unité**
La case code	1000	100	10	**1**
La case code	10^3	10^2	10^1	**10^0**
Nombre N	**5**	**2**	**4**	**3**
On obtient				

Au total, le nombre n vaut : N =

2.1 - Codification des nombres

La **case de poids faible** est **la plus à droite**. Elle code la valeur **1** qu'on peut écrire **10^0**

A chaque déplacement à gauche, la case code **10** fois plus.

DOC.1

	Millier	Centaine	Dizaine	**Unité**
La case code	1000	100	10	**1**
La case code	10^3	10^2	10^1	**10^0**
Nombre N	**5**	**2**	**4**	**3**
On obtient	**5000**	**200**	**40**	**3**

Au total, le nombre n vaut : N = 5000+200+40+3 = 5243

2.2 - Que peut-on mettre dans les cases en décimal ?

Les **10** **chiffres** de la base 10 sont : **0-1-2-3-4-5-6-7-8-9**

On représente **un nombre** comme un **ensemble de chiffres**.

2.3 – Poids de case en puissance de 10 (voir le tableau ci-dessus)

- Unité $10^0 = 1$
- Dix (da pour déca) $10^1 = 10$
- Cent (h pour hecto) $10^2 = 10*10 = 100$
- Mille (**k** pour kilo) $10^3 = 10*10*10 = 1000$
- Million (**M** pour Mega) $10^6 = 10*10*10*10*10*10 = 1\ 000\ 000$
- Milliard (**G** pour Giga) $10^9 = 1\ 000\ 000\ 000$
- 1000 milliards (**T** pour Tera) $10^{12} = 1\ 000\ 000\ 000\ 000$

-

2.4 – Augmentation d'une unité (incrémentation)

Si le chiffre à un successeur, on le remplace par celui-ci

0 ⇨ 1 ⇨ 2 ⇨ 3 ⇨ 4 ⇨ 5 ⇨ 6 ⇨ 7 ⇨ 8 ⇨ 9

Sinon (pour 9) :

- on revient au premier chiffre (0) dans cette case ET
- on rajoute une retenue de +1 à gauche.

Exercice de cours A : Trouver les nombres suivants les exemples ci-dessous (en justifiant la réponse !)

Après **8** ? Après **9** ? Après **19** ? Après **13** ? Après **399** ?

2.5 – Nombre de cas dénombrables en base 10 (système décimal)

Avec X cases :

Valeurs différentes : $\mathbf{10^x}$

Valeur minimale : **0**

Valeur maximale : $\mathbf{10^x - 1}$

Exercice de cours B : Trouver le nombre de valeurs différentes disponibles et la valeur maximale avec 3 chiffres puis 6 chiffres en base 10.

2.5 – Nombre de cas dénombrables en base 10 (système décimal)

Avec X cases :

Valeurs différentes : **10^x**

Valeur minimale : **0**

Valeur maximale : **$10^x - 1$**

Exercice de cours B : Trouver le nombre de valeurs différentes disponibles et la valeur maximale avec 3 chiffres puis 6 chiffres en base 10.

3 chiffres → Nombre de cas = 10^3 = 1000
Maximum 999

6 chiffres → Nombre de cas = 10^6 = 1 000 000
Maximum 999 999

III – La base 2 (le binaire)

3.1 - Que peut-on mettre dans les cases en binaire ?

Chaque case se nomme un **BIT**, mot qui provient de la contraction de **BINARY DIGIT**.

Les ***2*** **chiffres** de la base 2 sont : **0 - 1**

3.2 - Codification des nombres

Le **bit de poids faible** est **le plus à droite**. Elle code la valeur _____ qu'on peut écrire _____

A chaque déplacement à gauche, la case code _____ fois plus.

DOC.2

				Unité
Le bit code				**1**
Le bit code	2^3	2^2	2^1	**2^0**
Nombre N	**1**	**0**	**1**	**1**
On obtient				

Au total, le nombre n vaut : N = _____

3.2 - Codification des nombres

Le **bit de poids faible** est **le plus à droite**. Elle code la valeur **1** qu'on peut écrire $\mathbf{2^0}$

A chaque déplacement à gauche, la case code **2** fois plus.

DOC.2

				Unité
Le bit code	**8**	**4**	**2**	**1**
Le bit code	2^3	2^2	2^1	$\mathbf{2^0}$
Nombre N	**1**	**0**	**1**	**1**
On obtient	8	0	2	1

Au total, le nombre n vaut : N = **8 + 0 + 2 + 1 = 11**

Exercice 01 : Exprimer les **nombres binaires** suivant en base 10 :

N_A = 1 0 0 0 **N_B = 0 0 1 0**
N_C = 0 1 1 0 **N_D = 1 1 1 0**

3.3 – Notation des bases :

On indique la base par un indice situé en bas à droite du nombre.
Exemple : **$N = 1011_2 = 11_{10}$** ou **$N = (1011)_2 = (11)_{10}$**

3.4 – Incrémenter en binaire (rajouter 1)

- Un 0 devient son successeur 1
- Un 1 étant le dernier chiffre en base 2, on repasse à 0 en rajoutant une retenue de 1 dans la case de gauche.

Exercice de cours C : Trouver les nombres binaires suivants les exemples ci-dessous (en justifiant la réponse !)
Après **0** ? Après **1** ? Après **10** ? Après **11** ? Après **1011** ?

Exercice 02 : Compléter la séquence suivante. Indiquer la valeur obtenue en base 10 dans la colonne de droite.

En base 2			En base 10
0	0	0	0
0	0	1	1
0	1	0	
1	1	1	7

Exercice 02 : Compléter la séquence suivante. Indiquer la valeur obtenue en base 10 dans la colonne de droite.

En base 2			En base 10
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	1	1	7

Exercice 02 : Compléter la séquence suivante. Indiquer la valeur obtenue en base 10 dans la colonne de droite.

En base 2			En base 10
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	1	1	7

Exercice 02 : Compléter la séquence suivante. Indiquer la valeur obtenue en base 10 dans la colonne de droite.

En base 2			En base 10
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

3.5 – Nombre de cas dénombrables

Avec X cases : Valeurs différentes **: 2^x** Valeur minimale **: 0**
Valeur maximale : **2^x – 1**

Exercice de cours D : Trouver le nombre de valeurs différentes disponibles et la valeur maximale avec 4 bits puis 8 bits puis 16 bits.

3.5 – Nombre de cas dénombrables

Avec X cases : Valeurs différentes **: 2^x** Valeur minimale **: 0**
Valeur maximale : **2^x – 1**

Exercice de cours D : Trouver le nombre de valeurs différentes disponibles et la valeur maximale avec 4 bits puis 8 bits puis 16 bits.

4 bits → Nombre de cas 2^4 = 16 cas
Valeur maximale = 15

8 bits → Nombre de cas 2^8 = 256 cas
Valeur maximale = 255

16 bits → Nombre de cas 2^{16} = 65536 cas
Valeur maximale = 65535

IV – Octet

4.1 - Octet : binaire vers décimal

Un octet est un ensemble de 8 bits successifs.

Puisqu'il contient 8 bits, un octet possède **2^8 (soit 256) valeurs différentes**, de **0 et 255**.

Sous forme binaire, un octet est donc compris entre **0000 0000** $_2$ et **1111 1111** $_2$.

Bit de poids fort — Bit de poids faible

DOC.3

128					4	2	1
2^7					2^2	2^1	2^0
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

On peut alors écrire (**1111 1111**) $_2$ **=**

IV – Octet

4.1 - Octet : binaire vers décimal

Un octet est un ensemble de 8 bits successifs.

Puisqu'il contient 8 bits, un octet possède **2^8 (soit 256) valeurs différentes**, de **0 et 255**.

Sous forme binaire, un octet est donc compris entre **0000 0000** $_2$ et **1111 1111** $_2$.

Bit de poids fort — Bit de poids faible

DOC.3

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

On peut alors écrire

(**1111 1111**) $_2$ = **128+64+32+16+8+4+2+1 = 255**

Exercice 03 : Justifier la valeur décimale correspondant à ces octets sous forme binaire :

- Octet A : `0100 1000`
- Octet B : `1000 0011`
- Octet C : `1111 1111`

4.2 Activation progressive du bit de poids fort

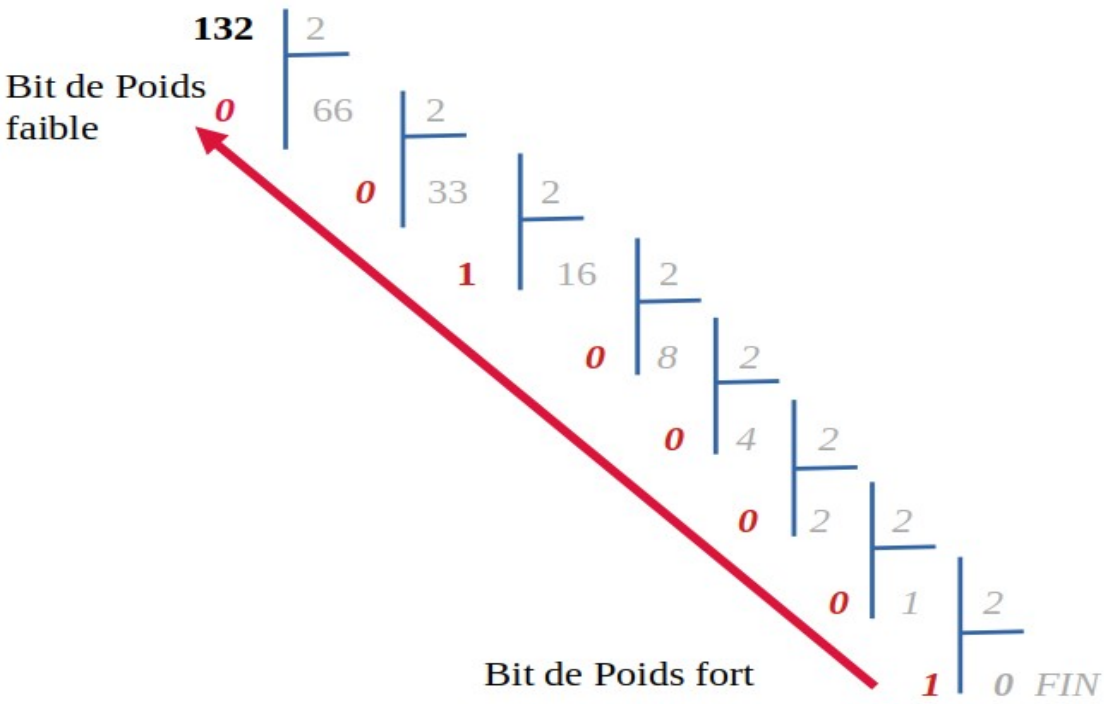
On active progressivement les bits en commençant systématiquement par le bit le plus grand nécessaire pour encoder correctement le nombre entier.

Exercice 04 : En activant progressivement les bits de poids fort nécessaires, trouver l'encodage binaire des entiers suivants :

- Octet D : 37
- Octet E : 72
- Octet F : 123

4.3 Division successive par deux

La méthode de la division consiste à diviser le nombre par deux jusqu'à atteindre un résultat de 0.



Les restes de la division euclidienne vont fournir les valeurs des bits.

$$\mathbf{132}_{10} = \mathbf{10000100}_{2}$$

Exercice 05 : Retrouver les résultats de l'exercice 04 mais en utilisant la division successive.

V – Utilisation de la notion d’octet

Un octet possède 256 valeurs différentes. Lorsqu'on veut mémoriser une information en machine, il suffit donc d'attribuer une valeur à une information. Cela se nomme un **encodage**. Les techniques d'encodage n'ont rien de secret : elles doivent être connue et diffusée de façon à pouvoir encoder puis à décoder.

5.1 – Encodage des entiers de 0 à 255

5.2 – Encodage de caractères

Les caractères sont stockés sous forme d’une suite d’octets.

Table d’encodage 1 octet : les 128 premiers caractères (0 à 127) sont communs à la plupart des encodages sur 1 octet : il s’agit de la **norme ASCII.** Les autres sont différents selon la table.

5.3 – Encodage des images

Chaque image est décomposée en pixels (un carré qui peut faire briller plus ou moins fortement).
En RGB, un **pixel est décomposé en 3 sous-pixels** : Rouge (Red), Vert (Green) et bleu (Blue).
On définit chaque luminosité par un nombre sur un octet (de 0 à 255).

255-0-0 sert à définir la couleur

175-0-0 sert à définir la couleur

0-255-0 sert à définir la couleur

255-255-0 sert à définir la couleur

5.3 – Encodage des images

Chaque image est décomposée en pixels (un carré qui peut faire briller plus ou moins fortement).
En RGB, un **pixel est décomposé en 3 sous-pixels** : Rouge (Red), Vert (Green) et bleu (Blue).
On définit chaque luminosité par un nombre sur un octet (de 0 à 255).

255-0-0 sert à définir la couleur **rouge**

175-0-0 sert à définir la couleur

0-255-0 sert à définir la couleur

255-255-0 sert à définir la couleur

5.3 – Encodage des images

Chaque image est décomposée en pixels (un carré qui peut faire briller plus ou moins fortement).
En RGB, un **pixel est décomposé en 3 sous-pixels** : Rouge (Red), Vert (Green) et bleu (Blue).
On définit chaque luminosité par un nombre sur un octet (de 0 à 255).

255-0-0 sert à définir la couleur **rouge**

175-0-0 sert à définir la couleur **rouge foncée**

0-255-0 sert à définir la couleur

255-255-0 sert à définir la couleur

5.3 – Encodage des images

Chaque image est décomposée en pixels (un carré qui peut faire briller plus ou moins fortement).
En RGB, un **pixel est décomposé en 3 sous-pixels** : Rouge (Red), Vert (Green) et bleu (Blue).
On définit chaque luminosité par un nombre sur un octet (de 0 à 255).

255-0-0 sert à définir la couleur **rouge**

175-0-0 sert à définir la couleur **rouge foncée**

0-255-0 sert à définir la couleur **vert**

255-255-0 sert à définir la couleur

5.3 – Encodage des images

Chaque image est décomposée en pixels (un carré qui peut faire briller plus ou moins fortement).
En RGB, un **pixel est décomposé en 3 sous-pixels** : Rouge (Red), Vert (Green) et bleu (Blue).
On définit chaque luminosité par un nombre sur un octet (de 0 à 255).

255-0-0 sert à définir la couleur **rouge**

175-0-0 sert à définir la couleur **rouge foncée**

0-255-0 sert à définir la couleur **vert**

255-255-0 sert à définir la couleur **jaune**