RND. Nombres entiers. Exercices

Exercice 1 : Vrai / Faux

	VRAI	FAUX
Si l'écriture binaire d'un entier naturel se termine par *n* zéros, alors cet entiers est divisible par 2^n.		
En base 8, on utilise les chiffres de 1 à 8.		
Avec *n* bits, les entiers représentables sont strictement inférieurs à 2^n.		
L'entier 170 s'écrit AA en hexadécimal (base 16).		
Avec 4 bits, on peut représenter les entiers relatifs de -8 à 8 bornes incluses.		
On code les entiers relatifs en complément à deux sur 5 bits. L'entier 10 est codé par 01010 et l'entier -10 par 10101.		

Exercice 2 : QCM
Pour chaque question, une seule réponse est correcte parmi les quatre proposées.

Question 1 : A quelle période sont nés les premiers ordinateurs ?
1. Au début du XX^e siècle.
2. Au milieu du XX^e siècle.
3. A la fin du XX^e siècle.
4. Au début du XXI^e siècle.

Question 2 : on considère le nombre 1000 écrit en base 10. Quelle affirmation est exacte ?
1. Ce nombre s'écrit AAA en hexadécimal.
2. Ce nombre s'écrit avec neuf chiffres en binaire.
3. Ce nombre s'écrit avec quatre chiffres en hexadécimal.
4. L'écriture de ce nombre en binaire se termine par 000.

Question 3 : Quelle affirmation est exacte ?
1. Un nombre occupe 8 fois moins de place en mémoire s'il est représenté par des octets 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.
4. Un nombre pair a une écriture hexadécimale qui se termine par un 0.

Question 4 : Quelle est la valeur en binaire de 1001 x 111 ?
1. 111111.
2. 101010.
3. 100111.
4. 111001.

Question 5 : Si on utilise 5 bits pour coder les entiers relatifs en complément à 2, comment est codé le nombre -2 ?
1. 10010.
2. 01111.
3. 10110.
4. 11110.

<u>Exercice 3</u> : Effectuer en binaire les additions des nombres entiers positifs (écrits en binaire) suivants :

a) 11010 + 100001
b) 11100010 + 10011
c) 10100011 + 11100111
d) 11111 + 11111

<u>Exercice 4</u> : **Déterminer** les nombres négatifs suivants :

a) 10001101
b) 11100010
c) 11111111

<u>Exercice 5</u> : Additionner à l'aide du complément à 2^n.

Le codage en complément à 2^n est utilisé car il permet d'effectuer simplement les additions : si on additionne deux nombres (positifs ou négatifs) codés en complément à 2^8, le résultat (s'il est dans la plage des nombres représentables) est aussi un nombre codé en complément à 2^8.

a) **Convertir** les deux nombres suivants en complément à 2^8 : 12 et -53.
b) **Effectuer** l'addition en binaire des deux nombres.
c) **Vérifier** que le résultat est correct.

<u>Exercice 6</u> : Coder sur 16 bits des nombres relatifs.

On utilise ici le codage en complément à 2^{16} (2 octets).
Quels sont les **nombres minimum** et **maximum** que l'on représenter ?

<u>Exercice 7</u> : Déterminer la taille des nombres entiers positifs en bases 2 et 16.

Si on utilise *n* chiffres binaires, on peut représenter tous les nombres de 0 à 2^{n-1}.

a) **Vérifier** cette affirmation pour quelques valeurs de n : 2, 3 et 8.
b) (*) Inversement, **combien** faut-il de chiffres binaires pour pouvoir représenter tous les nombres entiers de 0 à 19999 ?

<u>Exercice 8 (*)</u> : Des résultats d'additions étranges.

Expliquer pourquoi sur 8 bits on obtient les résultats suivants :

a) 127 – 1 = 126
b) 127 + 1 = -128
c) 127 + 2 = -127
d) 127 + 127 = -2

<u>Exercice 9 (*)</u> : Expliquer le bug de l'an 2038.

Les machines UNIX suivent la norme IEEE 1003 (ou norme POSIX) qui spécifie, entre autres, que le temps est compté en secondes à partir du 1er janvier 1970 à 00:00:00 temps universel. De nombreux systèmes de fichiers codent ce temps en un entier signé 32 bits (le signe permet de désigner les dates antérieures à 1970).
Pourquoi parle-t-on alors du bug de l'an 2038 ?