CORRECTION DES EXERCICES de la Leçon 04

a/ Quelle est la propriété nécessaire que doit posséder un code binaire pour qu'on puisse effectuer des opérations arithmétiques à l'aide de ses combinaisons?

Le code doit être pondéré, c'est à dire que le poids (la valeur) d'un bit dépend de la place qu'il a dans le nombre et uniquement de cela par exemple: le code binaire naturel est pondéré (poids des bits 1,2,4,8...) il peut être utilisé pour faire des opérations arithmétiques, mais le code binaire réfléchi ne peut pas être utilisé pour ces opérations.

b/ Si l'on travaille avec des nombres de 8 bits incluant un bit de parité avec parité impaire, ajouter le bit de parité aux nombres ci-dessous:

0011000, 1011101, 1111111, 0000000, 1000000

la parité impaire signifie que le nombre de 1 dans le nombre binaire doit être impair, si ce nombre est effectivement impair on ajoute un 0, s'il est pair on ajoute un 1

0011000	**1**
1011101	**0**
1111111	**0**
0000000	**1**
1000000	**0**

c/ Écrire en code « complément à 2 » 8 bits les nombres suivants:

+1, -1, +122, -55, - 128

Les nombres positifs s'écrivent comme en binaire naturel, valeur maximum +127, (0111 1111), le bit de poids fort doit rester à 0. Les nombres négatifs sont le complément à 2 des nombres positifs.(inversion des bits +1) valeur minimum -128

+	**1**	0	0	0	0	0	0	0	1
-	**1**	1	1	1	1	1	1	1	1
+	**122**	0	1	1	1	1	0	1	0
+	**55**	0	0	1	1	0	1	1	1
-	**55**	1	1	0	0	1	0	0	1
-	**128**	1	0	0	0	0	0	0	0

d/ Qu'est-ce que des combinaisons binaires adjacentes ?

Des combinaisons binaires adjacentes sont des combinaisons qui ne diffèrent que d'un seul bit.

Exemples: 0000 0001 et 0000 0011 sont adjacents

0000 0001 et 0000 0010 ne sont pas adjacents

e/ Construire le code binaire réfléchi 4 bits

0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

f/ Les informations suivantes nous parviennent en code de hamming 7 bits réalisé à partir du code binaire naturel les vérifier et éventuellement les corriger et en extraire l'information binaire naturelle.

0011110, 1101011, 1101001

1/ tests de **0011110**

• • 1 ⊕ 3 ⊕ 5 ⊕ 7
• • 2 ⊕ 3 ⊕ 6 ⊕ 7
• • 4 ⊕ 5 ⊕ 6 ⊕ 7

• • 0 ⊕ 1 ⊕ 1 ⊕ 0 = 0
• • 1 ⊕ 1 ⊕ 0 ⊕ 0 = 0
• • 1 ⊕ 1 ⊕ 0 ⊕ 0 = 0

Tous les tests donnent 0 le nombre est donc correct. Le nombre binaire utile est celui que l'on obtient lorsque les bits de parité ont été éliminés. Nous éliminerons donc les bits 1,2,4

0011110 • 001-1-- • **0011**

2/ tests de **1101011**

• • 1 ⊕ 0 ⊕ 0 ⊕ 1 = **0**
• • 1 ⊕ 0 ⊕ 1 ⊕ 1 = **1**
• • 1 ⊕ 0 ⊕ 1 ⊕ 1 = **1**

le test est non nul , la concaténation de • • • donne 110 = 6, il faut donc inverser le bit 6. Le nombre rétabli donne **1001011**

Le nombre binaire utile est:

1001011 • 100-0-- • **1000**

3/ test de **1101001**

• • 1 ⊕ 0 ⊕ 0 ⊕ 1 = **0**
• • 0 ⊕ 0 ⊕ 1 ⊕ 1 = **0**
• • 1 ⊕ 0 ⊕ 1 ⊕ 1 = **1**

le test est non nul , la concaténation de • • • donne 4, il faut donc inverser le bit 4. Le nombre rétabli donne **1100011**

Le nombre binaire utile est:
1100001 • 110-0-- • **1100**

g/ Réflexion sur les tests de Hamming. En supposant qu'il n'y a qu'une erreur dans un nombre, supposez qu'elle se trouve d'abord dans la colonne 1, puis dans la 2, puis dans la 3 etc et tirez en des conclusions quant au choix des bits testés dans chacun des tests.
Dans un code réalisé avec 4 bits de parité, combien de bits utiles pourrait-on avoir, où seraient-ils placés, quels seraient les tests à effectuer.

Rappel de la structure des tests

• • 1 ⊕ 3 ⊕ 5 ⊕ 7
• • 2 ⊕ 3 ⊕ 6 ⊕ 7
• • 4 ⊕ 5 ⊕ 6 ⊕ 7

En supposant qu'il n'y ait qu'une seule erreur, si celle-ci se trouve dans le colonne 1 qui n'est utilisée que dans le test • seul celui-ci donnera 1 et la concaténation de • • • donne 001
Si l'erreur est dans le colonne 2 qui n'est testée que dans • la concaténation donnera 010, si l'erreur est dans la colonne 3 qui est testée dans • et dans•• le test donnera 011 etc On voit que les tests doivent être choisis de telle sorte qu'ils n'englobent que les N° de colonne incluant leur poids. Exemple le test• • qui a le poids 2 teste les colonnes dont le N° contient 2

2 0010
3 0011
6 0110
7 0111

Le test • de poids 4 ne teste que les colonnes contenant 4

4 0100
5 0101
6 0110
7 0111

Dans un code réalisé avec 4 bits de parité, combien de bits utiles pourrait-on avoir,
où seraient-ils placés, quels seraient les tests à effectuer.
A l'aide de 4 bits de parité, il sera possible d'effectuer 4 tests que nous nommerons • •• •• •• et de former des nombres jusqu'à 15 bits le code utile aura donc 15 - 4 = 11 bits
.

Les tests à effectuer seront:

• • 1 ⊕ 3 ⊕ 5 ⊕ 7 ⊕ 9 ⊕ B ⊕ C ⊕ F
• •• 2 ⊕ 3 ⊕ 6 ⊕ 7 ⊕ A ⊕ B ⊕ E ⊕ F
• •• 4 ⊕ 5 ⊕ 6 ⊕ 7 ⊕ C ⊕ D ⊕ E ⊕ F
• • 8 ⊕ 9 ⊕ A ⊕ B ⊕ C ⊕ D ⊕ E ⊕ F

h/ Quels codes doit recevoir une imprimante qui laisse 3 espaces en début de ligne puis écrit IUT en lettres majuscules revient en début de ligne et passe à la ligne suivante.

Trois codes espace (SPACE)	20,20,20
Code du I majuscule	49
Code du U majuscule	55
Code du T majuscule	54
Code du retour en début de ligne (carriage return = retour chariot, CR survivance des machines à écrire et des télétypes)	0D
Code ligne suivante LF (line feed = passer une ligne).	0A

D'où la suite de codes:

20,20,20,49,55,54,0D,0A