Travaux dirigés et TP n° 9

Codes Correcteurs

Codes correcteurs

Exercice 1 (Codeur Correcteur de parité transversale et longitudinale(64,49))

1. Compléter les données suivantes

							Bits de parité
0	0	0	0	1	0	0	
1	0	0	1	0	0	1	
0	0	1	0	0	0	1	
0	1	0	1	0	1	0	
1	0	0	1	0	0	1	
0	0	1	0	0	0	1	
0	1	0	0	0	1	0	

							Bits de parité
0	0	0	0	1		0	1
1	0	0	1	0	0	1	1
0	0	1	0		0	1	0
0	1	0	1	0	1	0	1
1	0	0		0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	0	0	1	1	0	0	0

2. Corriger les tableaux suivants

							Bits de parité	
0	0	0	0	1	0	0	1	
1	0	0	1	0	0	1	1	
0	0	1	0	0	0	1	0	
0	1	0	0	0	1	0	1	!
1	0	0	1	0	0	1	1	
0	0	1	0	0	0	1	0	
0	1	0	0	0	1	0	0	
0	0	0	1	1	0	0	0	
			!					

							Bits de parité	
0	0	0	0	1	0	0	1	
1	0	0	1	0	0	1	1	
0	0	1	0	0	0	1	0	
0	1	0	1	0	1	0	0	!
1	0	0	1	0	0	1	1	
0	0	1	0	0	0	1	0	
0	1	0	0	0	1	0	0	
0	0	0	1	1	0	0	0	
							!	

							Bits de parité	
0	0	0	0	1	0	0	1	
1	0	0	1	0	0	1	1	
0	0	1	0	0	0	1	0	
0	1	0	1	0	1	0	1	
1	0	0	1	0	0	1	1	
0	0	1	0	0	0	1	0	
0	1	0	0	0	1	0	0	
0	0	0	1	1	0	0	1	!
							!	

							Bits de parité	
0	0	0	0	1	0	0	1	
1	0	0	1	0	0	1	1	
0	0	1	1	0	0	1	0	!
0	1	0	1	0	1	0	1	
1	1	0	1	0	0	1	1	!
0	0	1	0	0	0	1	0	
0	1	0	0	0	1	0	0	
0	0	0	1	1	0	0	0	
	!		!					

3. En différenciant les cas : le bit est dans les données", une somme de contrôle ou la somme de contrôle totale, calculer les distances de Hamming de codages proches.
4. Montrer que c'est un code 1-correcteur.
5. Donner une configuration comportant 2 erreurs non corrigibles.
6. Montrer que c'est un code 3-détecteur mais pas 4-détecteur.
   Tableaux à annoter selon les besoins :

							Bits de parité
0	0	0	0	1	0	0	1
1	0	0	1	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	1	0	1	0	1
1	0	0	1	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	0	0	1	1	0	0	0

							Bits de parité
0	0	0	0	1	0	0	1
1	0	0	1	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	1	0	1	0	1
1	0	0	1	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	0	0	1	1	0	0	0

							Bits de parité
0	0	0	0	1	0	0	1
1	0	0	1	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	1	0	1	0	1
1	0	0	1	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	0	0	1	1	0	0	0

							Bits de parité
0	0	0	0	1	0	0	1
1	0	0	1	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	1	0	1	0	1
1	0	0	1	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	0	0	1	1	0	0	0

7. Construire un CodeurParParite capable de rattraper une erreur.

Exercice 2 (Calculs sur $F_2(X)$ et class PolF2)
Soient les polynômes ($A(X) = X^4 + X^3 + X^2 + X$, ($B(X) = X^4 + X^2 + 1$ et $C(X) = X^2 + 1$ dans $F_2(X)$. Calculer A+B, A.B, B^2 et diviser A, puis B, par C (quotient et reste).

Exercice 3 (Classe PolF2)
Compléter le code des méthodes suivantes.

```
class PolF2(object):
    "Polynôme dans F2"
    def __init__(self,x):
        """
        Défini par une liste d' ElmntZnZ
        >>> PolF2([ElmtZnZ(1,2),0,1,0,1])
        PolF2([ElmtZnZ(1,2), ElmtZnZ(0,2), ElmtZnZ(1,2), ElmtZnZ(0,2), ElmtZnZ(1,2)])
        >>> PolF2(0b1000110010) #Entier -> Polynome dans F2
        PolF2([ElmtZnZ(0,2), ElmtZnZ(1,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(1,2), ElmtZnZ(1,2), ElmtZnZ
        >>> PolF2(0)
        PolF2([ElmtZnZ(0,2)])

    def degre(self):
        """
        >>> PolF2(0b100011).degre()
        5

    def distanceHamming(self,other):
        """
        >>> PolF2(0b100011).distanceHamming(PolF2(0b1100011))
        1

    def __add__(self,other):
        """
        >>> PolF2(0b100011)+PolF2(0b1100011)
        PolF2([ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ
        >>> PolF2(0b1100011)+ PolF2(0b100011)
        PolF2([ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ

    def __mul__(self,other):
        """
        >>> PolF2.monome(2)*PolF2.monome(1)
        PolF2([ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(1,2)])

    def __mod__(self,other):
        """
        >>> PolF2(0b11000101)%PolF2(0b11000)
        PolF2([ElmtZnZ(1,2), ElmtZnZ(0,2), ElmtZnZ(1,2)])

    def __floordiv__(self,other):
        """
        >>> PolF2(0b11000101)//PolF2(0b11000)
        PolF2([ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(0,2), ElmtZnZ(1,2)])

    def __int__(self):
        """
        >>> int(PolF2([ElmtZnZ(1,2), ElmtZnZ(0,2), ElmtZnZ(1,2)]))
        5
```

Exercice 4 (Class Code Correcteur CRC8)
Construire le code des méthodes suivantes autour d'un codeur CRC ajoutant 1 octets 'correcteur' au bloc de 32bits.

```
class CodeurCRC8(CodeurCA):
    """CodeurCRC codant en 40bits des blocs 32bits avec CRC sur 8bits"""
    def __init__(self,Pg=PolF2(0b110011011) ):

    def blocCode(self,M,verbose=False):
        '''Renvoie M codeé en CRC avec un octet de plus"
        >>> print(f"0x{CodeurCRC8().blocCode(0xab345678):x}")
        0xab34567821

    def estBlocValide(self,valc):
        """
        >>> CodeurCRC8().estBlocValide(0xab34567821)
        True
        >>> CodeurCRC8().estBlocValide(0xab34567820)

    def blocValideLePlusProche(self,valc,verbose=False):
        """
        >>> print(f"0x{CodeurCRC8().blocValideLePlusProche(0xab34567821):x}")
        0xab34567821
        >>> print(f"0x{CodeurCRC8().blocValideLePlusProche(0xab35567821):x}")
        0xab34567821

    def blocDecode(self,valc):
        """
        >>> print(f"0x{CodeurCRC8().blocDecode(0xab34567821):x}")
        0xab345678
        >>> print(f"0x{CodeurCRC8().blocDecode(0xbb34567821):x}")
        0xab345678

    def blocAvecErreur(val,nbBits=32,nbErreurs=1):
        """Renvoie le bloc val avec nbErreurs bits changés"""

    def binCode(self,monBinD,verbose=True,nbErreurs=0):

    def binDecode(self,monBinC:Binaire603)->Binaire603:

    def testDistance(self,nmax=0x101):
     """Affiche la distance minimales entre les codage des blocs 0 ) nmax"""

    def binDecode(self,monBinC:Binaire603)->Binaire603:
```