Nouvelle Calédonie – 2023 – sujet1 - Correction

Exercice 2 (4 points)

1

a. Bits de parité des entiers 9 et 16

- 9 en binaire : **1001**
  Somme des bits = 2 (pair) ⟶ bit de parité = **0**
- 16 en binaire : **10000**
  Somme des bits = 1 (impair) ⟶ bit de parité = **1**

👉 *Commentaire : Il fallait convertir en binaire puis compter le nombre de bits à 1. Attention aux erreurs de conversion ($16 = 2^4$).*

---

b. Deux exemples d'erreurs non détectables

Exemple 1 :
Message initial : 1001 (parité paire)
Si deux bits sont modifiés ⟶ 1100
La somme reste paire ⟶ erreur non détectée.

Exemple 2 :
Message initial : 1010
Si deux bits changent ⟶ 0101
La parité reste identique ⟶ erreur non détectée.

👉 *Commentaire : Le bit de parité détecte uniquement un nombre impair d'erreurs. Toute modification d'un nombre pair de bits passe inaperçue.*

---

2. Fonction Python calcul_parite

```python
def calcul_parite(Liste_bits):
    somme = sum(Liste_bits)
    if somme % 2 == 0:
        return 0
    else:
        return 1
```

👉 *Commentaire : L'utilisation de sum() et du modulo 2 était attendue. Une version plus concise avec* `return somme % 2` *était également valable.*

3

a. Justifier le codage de Hamming de [1, 0, 0, 1]

Données :
d1 = 1
d2 = 0
d3 = 0
d4 = 1

Calcul des bits de parité :

- p1 = parité de [d1, d2, d4] = [1, 0, 1] → somme = 2 → p1 = 0
- p2 = parité de [d1, d3, d4] = [1, 0, 1] → somme = 2 → p2 = 0
- p3 = parité de [d2, d3, d4] = [0, 0, 1] → somme = 1 → p3 = 1

Message encodé :
[p1, p2, d1, p3, d2, d3, d4]
→ **[0, 0, 1, 1, 0, 0, 1]**

👉 *Commentaire : Il fallait bien respecter l'ordre final des bits. Beaucoup d'élèves inversent les positions des bits de contrôle.*

---

b. Fonction Python codage_hamming

```python
def codage_hamming(donnees):
    d1, d2, d3, d4 = donnees
    p1 = calcul_parite([d1, d2, d4])
    p2 = calcul_parite([d1, d3, d4])
    p3 = calcul_parite([d2, d3, d4])
    return [p1, p2, d1, p3, d2, d3, d4]
```

👉 *Commentaire : L'utilisation de la fonction calcul_parite était exigée. Attention au respect exact de l'ordre des bits.*

4

a. Calcul des bits de contrôle après erreur

Message initial :
M = [0, 0, 1, 1, 0, 0, 1]

Erreur sur d1 → devient 0

Nouveau message :
[0, 0, 0, 1, 0, 0, 1]

Calcul des bits de contrôle :

- c1 = parité de [p3, d2, d3, d4] = [1, 0, 0, 1] → somme = 2 → c1 = 0
- c2 = parité de [p2, d1, d3, d4] = [0, 0, 0, 1] → somme = 1 → c2 = 1
- c3 = parité de [p1, d1, d2, d4] = [0, 0, 0, 1] → somme = 1 → c3 = 1

Donc :
[c1, c2, c3] = [0, 1, 1]

👉 *Commentaire : Il fallait recalculer avec la valeur modifiée de d1. Erreur fréquente : utiliser les anciennes valeurs.*

---

b. Triplets correspondant à d3 altéré et d4 altéré

En numérotation binaire des positions :

1 → p1
2 → p2
3 → d1
4 → p3
5 → d2
6 → d3
7 → d4

- d3 est en position 6 → 110
  donc [c1, c2, c3] = **[1, 1, 0]**

- d4 est en position 7 → 111
  donc [c1, c2, c3] = **[1, 1, 1]**

👉 *Commentaire : Les bits de contrôle forment le numéro binaire de la position erronée.*

---

c. Supériorité du code de Hamming

Le bit de parité simple permet uniquement de détecter certaines erreurs (nombre impair de bits modifiés).

Le code de Hamming :

- détecte les erreurs,
- localise précisément le bit erroné,
- permet de corriger automatiquement une erreur sur un bit.

👉 *Commentaire : Le mot clé attendu est "correction d'erreur". C'est la grande différence avec la simple parité.*
