DS 1 - NSI Première (Correction)

Exercice 1

1)

Conversion de $(-25)_{10}$ en binaire sur 7 bits

On divise 25 par 2, en notant le reste à chaque étape jusqu'à ce que le quotient soit 0 :
25 = 2 x 12 + 1
12 = 2 x 6 + 0
6 = 2 x 3 + 0
3 = 2 x 1 + 1
1 = 2 x 0 + 1
Les restes, lus du bas vers le haut, donnent le nombre binaire 11001.
Pour représenter 25 sur 7 bits, on ajoute des zéros à gauche : 0011001_2.
donc $25_{10} = 0011001_2$ sur 7 bits

On utilse le complément à 2 sur 7 bits.
$(-25)_{10} = \overline{0011001}_2 + 1 = 1100110 + 0000001 = 1100111_2$ sur 7 bits

Conversion de $(-37)_{10}$ en binaire sur 6 bits

On divise 37 par 2, à chaque fois en notant le reste, jusqu'à obtenir un quotient de 0.
37 = 2 x 18 + 1
18 = 2 x 9 + 0
9 = 2 x 4 + 1
4 = 2 x 2 + 0
2 = 2 x 1 + 0
1 = 2 x 0 + 1
Les restes lus du bas vers le haut donnent le nombre binaire 100101.
donc $37_{10} = 0100101_2$ sur 7 bits
On utilse le complément à 2 sur 7 bits.
$(-17)_{10} = \overline{0100101}_2 + 1 = 1011010 + 0000001 = 1011011_2$ sur 7 bits

2)

$$\begin{array}{rl} & 1100111_2 \\ + & 1011011_2 \\ \hline = & 11000010_2 \end{array}$$ sur 8 bits

Mais, nous sommes sur 7 bits, donc :
$1100111_2 + 1011011_2 = 1000010_2$ sur 7 bits

3) Conversion de 1001110_2 en binaire sur 7 bits

On utilse encore le complément à 2.
$\overline{1000010}_2 + 1 = 0111101 + 000001 = 0111110 = 2^5 + 2^4 + 2^3 + 2^2 + 2^1 = 32 + 16 + 8 + 4 +$
donc $1000010_2 = (-62)_{10}$

On pouvais sans douter : $(-25) + (-37) = -62$

Exercice 2

1)

Entrée[6]:
```
def decimal_to_binary(n):
    result = ""
    while n > 0:
        result = str(n % 2) + result
        n = n // 2
    return result

# Exemple
decimal_to_binary(25)
```

Sortie[6]:
```
'1010'
```

Entrée[5]:
```
# Exemple
decimal_to_binary(37)
```

Sortie[5]:
```
'100101'
```

2)

Entrée[2]:
```
def binaire_vers_decimal(binaire):
    decimal = 0
    puissance = len(binaire) - 1
    for chiffre in binaire:
        decimal += int(chiffre) * (2 ** puissance)
        puissance -= 1
    return decimal

# Exemple d'utilisation
binaire_vers_decimal("0111110")
```

Sortie[2]:
```
62
```

Exercice 3

1) Conversion de 10.3_{10} en binaire

On commence par la partie entière, soit par 10.
On divise 12 par 2, à chaque fois en notant le reste, jusqu'à obtenir un quotient de 0.
10 = 2 x 5 + 0
5 = 2 x 2 + 1
2 = 2 x 1 + 0
1 = 2 x 0 + 1
Les restes lus du bas vers le haut donnent le nombre binaire 1100.
donc $10_{10} = 1010_2$

On prends maintenant la partie décimale 0,3.
0,3 x 2 = 0,6 donc 0
0,6 x 2 = 1,2 donc 1
0,2 x 2 = 0,4 donc 0
0,4 x 2 = 0,8 donc 0
0,8 x 2 = 1,6 donc 1
0,6 x 2 = 1,2 Nous l'avons déjà eu. Les résultats sont lus du haut vers le bas.
donc $0,3_{10} = 0,0[1001]_2$

et donc $10,3_{10} = 1010_2 + 0,0[1001]_2 = 10,0[1001]_2$

2) Conversion de $10011,11_2$ en décimal

$10011,11_2$
$= 1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}$
$= 16 + 0 + 0 + 2 + 1 + 0,5 + 0,25$
$= 19 + 0,75$
$= 19,75_{10}$

Exercice 4 : Expressions booléennes

1) Table de vérité pour A AND (B OR NOT C) :

A	B	C	NOT C	B OR NOT C	A AND (B OR NOT C)
0	0	0	1	1	0
0	0	1	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	1

2) Table de vérité pour (A AND B) OR (A AND NOT C) :

A	B	C	NOT C	A AND B	A AND NOT C	(A AND B) OR (A AND NOT C)
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	1	0	1	1	1	1
1	1	1	0	1	0	1

3) Comparaison des deux expressions :

A AND (B OR NOT C) = (A AND B) OR (A AND NOT C)
Les deux expressions sont équivalentes, car on obtiens le même résultats.
Mais, on le savait déjà, car il y la distributivité du AND sur le OR.