DM: Nombres négatifs et complément à 2.

1) a. Pour convertir un nombre négatif de la base 10 à la base 2, il faut d'abord coder la valeur absolue du nombre en binaire sur le nombre de bits indiqué. Le résultat peut-être sur 4, 8, 16, 32 ou 64 bits, si il ne l'est pas alors on lui ajoute les bits manquants devant. Ensuite on fait le complément à 1 du nombre précédent, puis on ajoute 1 au nombre binaire obtenu. On obtient ainsi un nombre binaire négatif.
   b. Les différentes fonctions qui seront nécessaires pour effectuer cette conversion sont :

   - dec2bin : cette fonction convertit les nombres décimaux à des nombres binaires.

   - fillBits: cette fonction remplie les bits manquant sur le nombre de bits voulu. Elle est nécessaire pour atteindre 4, 8, 16, 32 ou 64.

2) def osBits(n, maxBits=16):
"""
Cette fonction ajoute les bits manquants
Entrée : n représentation d'un nombre décimal sous forme d'une chaîne de caractère binaire.
Sortie : nombre décimaux sous forme de chaîne de caractère binaire avec le nombre de bits souhaité.
"""

```
        L = len(n)
        if L > maxBits:
                n = n[0:maxBits]
        elif L < maxBits:
                while L % maxBits != 0:
                        n =  '0' +n
                        L = len(n)
return n
```

   Voici la fonction osBits, qui ajoute des bits manquants si un nombre n'est pas codé sur 8 ou 16 bits.

3) a. Cette fonction additionne deux nombres binaires a et b. Pour cela je défini d'abord la fonction dec2bin pour convertir les nombres décimaux en nombres binaires.

Dec2bin:

```
def dec2bin( n: int ) -> str :
    """
Entrée: un nombre décimal
Sortie: un nombre binaire
    """
    # a = b x q+r
    reste = ''
    a = n
    while a != 0:
        reste = str(a%2) + reste
        a = a // 2
    if n == 0:
    reste = '0'
    return reste
```

AddBin:

```
def addBin(a, b):
    a = osBits(dec2bin(a), 8)          # passe a sur 8 bits
    b = osBits(dec2bin(b), 8)          # passe b sur 8 bits
    c = ''
    rem = 0
    for i in range(7, -1, -1):
        S = int(a[i]) + int(b[i]) + rem                    # additionne le
i-eme element des str a et b, convertit en entiers, avec l'entier rem
        if S < 2:
            c = str(S) + c
            rem = 0
        elif S == 2:
            c = '0' + c
            rem = 1
        elif S == 3:
            c = '1' + c
            rem = 1
    return c                    # renvoie un entier c sur 8 bits
```

b. Les valeurs prises par la variable i dans la boucle définie ligne jaune sont: 7, 6, 5, 4, 3, 2, 1, 0.

c. La variable S représente la somme des i-eme éléments d'a et b et le reste. La variables rem représente le reste de l'addition entre nombres binaires. C'est-à-dire que pour une addition tel que : 0 + 0, 1 + 0 ou 0 + 1 le reste est égal à 0 et si l'addition est 1 + 1 le reste est 1.

d. L'algorithme fonctionne ainsi : il définit d'abord les valeurs a, b et c. Ensuite il rentre dans une boucle „pour", S représente ici la somme du dernier élément de a et b et le reste. Le programme fait ensuite une comparaison, si S plus petit que 2 (par exemple : addition du type : 0 + 0 + 1), alors c vaut la somme de la chaîne de caractère de S plus la valeur initial de c (0). Si S vaut 2 (Addition du type : 0 + 1 + 1), alors c prend la valeur de la somme de '0' + c, le reste vaut alors 1. Ou alors, si S vaut 3, alors c est égal à la somme '1' plus c. A la fin l'algorithme renvoie c.

4) A l'aide des questions précédentes et en utilisant la fonction dec2bin, j'ai écris une fonction dec2signedBin qui convertit un nombre positif ou négatif en base 10 en binaire signé en base 2.
Pour cette fonction j'utilise d'autres fonctions que j'ai écrit, j'utilise complement2 comme fonction, c'est une fonction qui fait le complément à 2 d'un nombre binaire et j'utilise également osBits.

complement2 :

```
def complement2(binaryString):
    """
Cette fonction fait le complément à 2 d'une chaîne de caractère binaire, c'est-à-dire qu'elle converti un nombre décimal négatif en nombre binaire.
    """
    binaryString = osBits(binaryString, 8)
    print(binaryString)
    complement = ''
    for i in range(len(binaryString)):
        if binaryString[i] == '0':
            complement += '1'
        else:
            complement += '0'

    result = addBin(a=complement, b=osBits('1', 8))

    return result
```

dec2signedBin :

```
def dec2signedBin(n):
        absn = abs(n)
        signedBinaryString = dec2bin(absn)
        signedBinaryString = osBits(signedBinaryString, 8)
        if n < 0:
                signedBinaryString = complement2(signedBinaryString)

        return signedBinaryString
```

Voici donc la fonction dec2signedBin.