Exercice 3

Thèmes abordés : conversion décimal/binaire, table de vérité, codage des caractères

L'objectif de l'exercice est d'étudier une méthode de cryptage d'une chaîne de caractères à l'aide du codage ASCII et de la fonction logique XOR.

1. Le nombre 65, donné ici en écriture décimale, s'écrit 01000001 en notation binaire. En détaillant la méthode utilisée, **donner** l'écriture binaire du nombre 89.

2. La fonction logique OU EXCLUSIF, appelée XOR et représentée par le symbole $\oplus$, fournit une sortie égale à 1 si l'une ou l'autre des deux entrées vaut 1 mais pas les deux.

On donne ci-contre la table de vérité de la fonction XOR.

E_1	E_2	$E_1 \oplus E_2$
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Si on applique cette fonction à un nombre codé en binaire, elle opère bit à bit.

$$\begin{array}{ll} & 1100 \\ \oplus & 1010 \\ \hline = & 0110 \end{array}$$

Poser et calculer l'opération : 11001110 $\oplus$ 01101011

3. On donne, ci-dessous, un extrait de la table ASCII qui permet d'encoder les caractères de A à Z.

On peut alors considérer l'opération XOR entre deux caractères en effectuant le XOR entre les codes ASCII des deux caractères. Par exemple : 'F' XOR 'S' sera le résultat de 01000110 $\oplus$ 01010011.

Code ASCII Décimal	**Code ASCII Binaire**	**Caractère**
65	01000001	A
66	01000010	B
67	01000011	C
68	01000100	D
69	01000101	E
70	01000110	F
71	01000111	G
72	01001000	H
73	01001001	I
74	01001010	J
75	01001011	K
76	01001100	L
77	01001101	M

Code ASCII Décimal	**Code ASCII Binaire**	**Caractère**
78	01001110	N
79	01001111	O
80	01010000	P
81	01010001	Q
82	01010010	R
83	01010011	S
84	01010100	T
85	01010101	U
86	01010110	V
87	01010111	W
88	01011000	X
89	01011001	Y
90	01011010	Z

On souhaite mettre au point une méthode de cryptage à l'aide de la fonction XOR.

Pour cela, on dispose d'un message à crypter et d'une clé de cryptage de même longueur que ce message. Le message et la clé sont composés uniquement des caractères du tableau ci-dessus et on applique la fonction XOR caractère par caractère entre les lettres du message à crypter et les lettres de la clé de cryptage.

Par exemple, voici le cryptage du mot ALPHA à l'aide de la clé YAKYA :

Message à crypter	A	L	P	H	A
Clé de cryptage	Y	A	K	Y	A
	↓	↓	↓	↓	↓
Message crypté	'A' XOR 'Y'	'L' XOR 'A'	'P' XOR 'K'	…	…

Ecrire une fonction `xor_crypt(message, cle)` qui prend en paramètres deux chaînes de caractères et qui renvoie la liste des entiers correspondant au message crypté.

Aide :

- On pourra utiliser la fonction native du langage Python `ord(c)` qui prend en paramètre un caractère `c` et qui renvoie un nombre représentant le code ASCII du caractère `c`.
- On considère également que l'on dispose d'une fonction écrite `xor(n1,n2)` qui prend en paramètre deux nombres `n1` et `n2` et qui renvoie le résultat de `n1` $\oplus$ `n2`.

4. On souhaite maintenant générer une clé de la taille du message à partir d'un mot quelconque. On considère que le mot choisi est plus court que le message, il faut donc le reproduire un certain nombre de fois pour créer une clé de la même longueur que le message.

Par exemple, si le mot choisi est YAK pour crypter le message ALPHABET, la clé sera YAKYAKYA.

Créer une fonction `generer_cle(mot,n)` qui renvoie la clé de longueur `n` à partir de la chaîne de caractères `mot`.

5. Recopier et **compléter** la table de vérité de $(\boldsymbol{E_1} \oplus \boldsymbol{E_2}) \oplus \boldsymbol{E_2}$.

$\boldsymbol{E_1}$	$\boldsymbol{E_2}$	$\boldsymbol{E_1} \oplus \boldsymbol{E_2}$	$(\boldsymbol{E_1} \oplus \boldsymbol{E_2}) \oplus \boldsymbol{E_2}$
0	**0**	**0**	
0	**1**	**1**	
1	**0**	**1**	
1	**1**	**0**	

A l'aide de ce résultat, **proposer** une démarche pour décrypter un message crypté.