Informatique PCSI
TP 11a : représentation des entiers

Exercice 1

Codage d'un entier naturel

1. Écrire une fonction qui prend en paramètre un entier naturel exprimé en base dix et renvoie l'écriture en base deux de ce nombre. Le paramètre en entrée est de type `int` et la valeur renvoyée en sortie de type `str`.
2. Écrire une fonction qui prend en paramètre un entier naturel exprimé en base deux et renvoie l'écriture en base dix de ce nombre. Le paramètre en entrée est de type `str` et la valeur renvoyée en sortie de type `int`.

Utiliser l'algorithme des restes avec les divisions euclidiennes successives.

Exercice 2

Écrire une fonction `bin_kbits` qui prend en paramètres deux entiers naturels n et k avec $0 \leq n < 2^k$ et renvoie l'écriture de n sur k bits. La valeur renvoyée est de type str.

Exercice 3

Un nombre est stocké sous la forme d'un quadruplet `(a, b, c, d)`. Les quatre nombres `a`, `b`, `c`, `d` sont les valeurs décimales des quatre octets qui servent à stocker la valeur du nombre. Suivant l'orientation big endian, ce nombre a pour valeur décimale $a \times 256^3 + b \times 256^2 + c \times 256 + d$. Suivant l'orientation little endian, ce nombre a pour valeur décimale $a + b \times 256^1 + c \times 256^3 + d \times 256^4$.

Par exemple, avec `(13, 1, 0, 0)`, suivant l'orientation big endian, ce nombre a pour valeur décimale $13 \times 256^3 + 1 \times 256^2 = 218169344$ et suivant l'orientation little endian, ce nombre a pour valeur décimale $13 + 1 \times 256^1 = 269$.

1. Écrire une fonction `decimal_be` qui prend en paramètre un quadruplet comme ci-dessus et renvoie la valeur décimale du nombre en suivant l'orientation big endian.
2. Écrire une fonction `decimal_le` qui prend en paramètre un quadruplet comme expliqué ci-dessus et renvoie la valeur décimale du nombre en suivant l'orientation little endian.

Exercice 4

Codage d'un entier relatif

Écrire une fonction qui prend en paramètres un entier relatif `r` exprimé en base dix et un entier naturel non nul `n` et renvoie le codage de `r` sur `n` bits en complément à deux. Le résultat renvoyé est de type `str`. Il n'est pas nécessaire de vérifier que le nombre `r` peut être codé sur `n` bits.