Conversions binaire <--> décimal (chaînes de caractères, niveau 1)

1. Conversion en binaire d'un nombre entier positif

Pour donner l'écriture binaire d'un nombre entier positif *n*, il suffit d'appliquer l'algorithme suivant :

- si *n* est nul, afficher 0 ;
- tant que *n* est strictement positif, afficher *n* modulo 2 (c.-à-d. le reste de la division de *n* par 2), puis diviser *n* par 2.

Le problème avec l'algorithme ci-dessus est qu'il affiche le nombre *à l'envers*. Par exemple pour 2, il affiche `01` au lieu de `10`.

Il suffit simplement pour palier ce problème d'utiliser une structure de données auxiliaire dans le langage de programmation choisi (par exemple une chaîne de caractères) et d'y ajouter les symboles 0 ou 1 non pas à la fin comme le ferait un affichage standard, mais **au début** de la structure de données choisie. Une fois celle-ci remplie, il ne reste plus qu'à l'afficher normalement (c.-à-d. dans l'ordre usuel).

On peut aussi palier ce problème d'ordre d'affichage des bits en recourant à la version récursive suivante :

- si *n* est supérieur ou égal à deux, afficher l'écriture binaire de n/2;
- (puis) si *n* est pair afficher 0, sinon afficher 1.

Notes :

1. Beaucoup de langages de programmation offrent directement le moyen d'afficher en binaire une valeur entière. Le but de cet exercice n'est bien sûr pas d'utiliser de tels moyens, mais bien de programmer *par vous-même* l'algorithme de conversion.
2. On suppose ici que le nombre entier à convertir est assez petit pour être correctement représenté sur un `int` du langage de programmation utilisé. Il n'est pas question dans cet exercice de gérer la représentation mémoire de nombres entiers plus grands que ce que permet le langage de programmation utilisé.

2. Conversion binaire --> décimal

La conversion réciproque (binaire vers décimal positif) se fait simplement en ajoutant la puissance de 2 correspondant à chaque '1' présent dans l'écriture binaire.

On peut par exemple utiliser l'algorithme suivant :

- initialiser *r* à 0 et *p* à 1
- En allant du dernier bit au premier (sens inverse de lecture) :
  - **Si** le bit est à 1, augmenter *r* de *p* (*r* <-- *r* + *p*)
  - multiplier *p* par 2

3. Conversion de nombres négatifs

On peut maintenant s'intéresser aux nombres négatifs, représenté en binaire en « *complément à 2* ». Pour simplifier ici, on ne fixera pas la taille de la représentation binaire, mais on ajoutera simplement le bit de signe au début (= à gauche) de l'écriture binaire minimale nécessaire pour représenter le nombre (**exemples ci-dessous**).

Commencez pour cela par écrire une fonction qui retourne le complément à 2 d'une écriture binaire.
Le plus simple pour cela est d'inverser tous les bits à gauche du 1 le plus à droite (et laisser le reste inchangé).

Par exemple, le complément à 2 de « 100110100 » est « 011001 100 » (notez que la partie soulignée est inchangée et le reste inversé).

Une fois une telle fonction à disposition, il suffit d'appliquer les conversions précédentes si le nombre est positif (bit de signe à 0) et d'appliquer le complément à 2 à la conversion de leur opposé si le nombre est négatif.

On prêtera cependant attention au fait de *rajouter* un bit supplémentaire (bit de signe) au début des écritures binaires

non signées.

Valeurs de test

Commencez par tester avec des valeurs simples (puissances de 2).

L'écriture binaire sans bit de signe de 42 est `101010`.

L'écriture binaire signée de 42 est `0101010`.

L'écriture binaire signée de -42 est `1010110`.