

DS 1 d'informatique

2 heures

Lisez attentivement les consignes ci-dessous

- *Les documents, calculatrices et autres appareils électroniques sont interdits.*
- *La qualité de la rédaction sera prise en compte dans l'évaluation.*
- *La présentation de la copie sera prise en compte dans l'évaluation :*
 - ▷ *Encadrez les résultats principaux.*
 - ▷ *Soignez votre écriture.*
 - ▷ *Maintenez une marge dans vos copies et aérez vos copies.*
 - ▷ *Numérotez vos copies.*
- ***Vos programmes doivent être clairs.***
 - ▷ *Choisissez judicieusement les noms de vos variables ainsi que les noms de vos fonctions auxiliaires.*
 - ▷ *Aérez vos programmes.*
 - ▷ *Faites des indentations suffisamment grandes.*
 - ▷ ***Indiquez les indentations par des traits verticaux.***
 - ▷ *L'introduction de **fonctions auxiliaires** est recommandée lorsqu'elle permet d'écrire des programmes plus clairs.*
 - ▷ *Si nécessaire, commentez vos programmes en utilisant une couleur secondaire.*
- ***Sauf instruction explicite du contraire, une fonction ne doit pas modifier ses arguments.***
- *Si vous constatez ce qui vous semble être une erreur d'énoncé, signalez-le sur votre copie en expliquant les initiatives que vous avez été amené à prendre.*

Nombres entiers en informatique

Codage et opérations

Le but de ce problème est d'étudier comment les nombres entiers sont codés dans la mémoire d'un ordinateur et comment les opérations sur ces nombres sont effectuées. On distinguera à ce titre deux types d'entiers : d'abord les entiers naturels, puis les entiers relatifs. On s'intéressera en particulier aux opérations suivantes : somme, produit et puissance.

Préliminaire

1. Écrire une fonction `copie(l)` qui prend en argument une liste `l` (composée de nombres) et qui renvoie une copie de cette liste.

Partie I – Codage des entiers naturels

2. Écrire une fonction `listeChiffresBase2(n)` qui prend en argument un entier n et qui renvoie la liste `l` des chiffres de son écriture en base 2, « écrite à l'envers ».
Le chiffre de l'écriture de n correspondant à 2^k sera ainsi dans `l[k]`.

► `listeChiffresBase2(16)` renverra `[0, 0, 0, 0, 1]`.
► `listeChiffresBase2(1)` renverra `[1]`.
► `listeChiffresBase2(0)` renverra `[]`.

3. Écrire une fonction `ecritureBase2(n)` qui prend en argument un entier n et qui renvoie une chaîne de caractères commençant par « 0b » et suivie des chiffres de l'écriture en base 2 de n (dans le bon sens).

Cette fonction utilisera la fonction `listeChiffresBase2(n)`.

► `ecritureBase2(16)` renverra `'0b10000'`.
► `ecritureBase2(1)` renverra `'0b1'`.
► `ecritureBase2(0)` renverra `'0b0'`.

4. Écrire une fonction `dechiffre(l)` qui prend en argument une liste `l` de zéros et de uns, « écrite à l'envers », et qui renvoie l'entier dont cette liste est l'écriture en base 2.

► `dechiffre([])` renverra 0.
► `dechiffre([1, 1, 1])` renverra 7.
► `dechiffre([0, 0, 0, 1])` renverra 8.

Définitions et conventions

- Dans ce problème, on utilisera les définitions suivantes :
 - ▷ un 2-entier est une liste de zéros et de uns ;
 - ▷ un 2-entier sera dit de taille N ssi sa longueur (en tant que liste) vaut N .
- Voici des exemples de 2-entiers :

[0, 0, 1], [0, 1, 1, 0], [0, 0, 0, 0], [].

- Les 2-entiers correspondent aux entiers naturels via les fonctions

`listeChiffresBase2(n)` et `dechiffre(l)`.

On prendra donc la convention de l'écriture « à l'envers ».

- Par exemple, le 2-entier [0, 0, 0, 1] correspond à 8 ; il est de taille 4.
- On notera les 2-entiers : `nn`, `mm`, `pp`, etc.

Partie II – Premières fonctions

5. Écrire une fonction `estNul(nn)` qui prend en argument un 2-entier `nn` et qui renvoie :

- `True` s'il est nul ;
- `False` sinon.

► `estNul([1, 0, 1])` renverra `False`.
► `estNul([0, 0, 0])` renverra `True`.
► `estNul([])` renverra `True`.

6. Écrire une fonction `entierAjuste(nn, N)` qui prend en argument un 2-entier `nn` et une taille $N \in \mathbb{N}^*$ et renvoie un 2-entier de taille N tel que

- si la taille de `nn` est $< N$, il est construit à partir de `nn` en y ajoutant des zéros.
- si la taille de `nn` est $\geq N$, il est construit à partir de `nn` en ne gardant que les termes d'indices les plus petits.

► `entierAjuste([1, 0, 1], 5)` renverra [1, 0, 1, 0, 0].
► `entierAjuste([1, 0, 1, 0, 1, 1], 4)` renverra [1, 0, 1, 0].

7. (a) Donner l'écriture en base 2 de 424.

- (b) On note `nn = listeChiffresBase2(424)`.

Quel entier sera renvoyé par `entierAjuste(nn, 5)` ?

On donnera la réponse sous la forme d'un entier écrit en base 10.

Partie III – Comparaison des entiers naturels

8. Écrire une fonction `entierNettoye(nn)` qui prend en argument un 2-entier `nn` et qui renvoie un 2-entier correspondant au même entier que `nn` mais qui n'a pas de zéros inutiles.

► `entierNettoye([1, 0, 1, 0, 0]) renverra [1, 0, 1].`

► `entierNettoye([0, 0]) renverra [].`

9. Écrire une fonction `comparaison(nn, pp)` qui prend en argument deux 2-entiers `nn` et `pp` et qui renvoie :

- 1 si $n > p$,
- 0 si $n = p$,
- -1 si $n < p$,

où n et p sont les entiers naturels représentés par `nn` et `pp`.

Votre fonction ne devra utiliser aucun des opérateurs « < », « > », « <= », « >= » de Python.

► `comparaison([1, 0, 0], [1, 1]) renverra -1.`

► `comparaison([0, 0, 1], [1, 1]) renverra 1.`

► `comparaison([0, 0, 1], [0, 0, 1, 0]) renverra 0.`

Partie IV – Codage des entiers relatifs

Complément à deux

On fixe $N \in \mathbb{N}^*$.

On souhaite coder les nombres entiers relatifs dans $\llbracket -2^{N-1}, 2^{N-1} - 1 \rrbracket$ à l'aide des 2-entiers de taille N .

Si $n \in \llbracket -2^{N-1}, 2^{N-1} - 1 \rrbracket$, on procède comme suit :

- si l'entier n est ≥ 0 , on le code à l'aide de `listeChiffresBase2(n)` qu'on complète éventuellement de zéros ;
- si l'entier n est < 0 , on part de l'écriture de $-n$ en tant que 2-entier de taille N ; on transforme les zéros en uns et vice versa dans cette écriture ; enfin, on ajoute 1 à ce 2-entier.

Le résultat est appelé écriture en base 2 de n , avec N bits, suivant la convention du complément à 2.

10. Donner l'écriture en base 2 de -1 avec 4 bits, suivant la convention du complément à 2.
11. Écrire une fonction `miroir(nn)` qui prend en argument un 2-entier `nn` et qui renvoie le 2-entier de même taille déduit de `nn` en échangeant les zéros et les uns.

► `miroir([1, 1, 1, 1]) renverra [0, 0, 0, 0].`

► `miroir([1, 0, 1, 0]) renverra [0, 1, 0, 1].`

12. Écrire une fonction `increment(nn)` qui prend en argument un 2-entier `nn` et qui renvoie le 2-entier déduit de `nn` en lui ajoutant 1.

Votre fonction ne devra pas utiliser l'opérateur « + » de Python.

► `increment([1, 1, 1, 1]) renverra [1, 0, 0, 0].`

► `increment([1, 0, 1, 0]) renverra [1, 0, 1, 0].`

13. Écrire une fonction `codage(n, N)`, qui prend en argument deux entiers `n` et `N`, qu'on supposera tels que $n \in \llbracket -2^{N-1}, 2^{N-1} - 1 \rrbracket$ et qui renvoie le 2-entier correspondant à l'écriture en base 2 de `n`, avec `N` bits, suivant la convention du complément à 2.
14. Écrire une fonction `signe(nn, N)`, qui prend en argument un 2-entier `nn` et un entier `N`, et qui détermine son signe, en supposant que `nn` a été obtenu à partir d'un entier `n` dans $\llbracket -2^{N-1}, 2^{N-1} - 1 \rrbracket$.
 Cette fonction renverra 1 si $n > 0$, 0 si $n = 0$ et -1 si $n < 0$.

Partie V – Somme des entiers naturels

Dans cette partie, on suppose que les 2-entiers codent des nombres entiers naturels.

15. Écrire une fonction `somme(nn, pp)`, qui prend en argument deux 2-entiers `nn` et `pp`, et qui renvoie leur somme (sous forme de 2-entier).
 Votre fonction ne devra pas utiliser l'opérateur « + » de Python.

► `somme([0, 1, 1, 0], [1])` renverra `[0, 1, 1, 1]`.
 ► `somme([0, 1, 1, 0], [1, 0])` renverra `[1, 0, 0, 0]`.

Partie VI – Produit des entiers naturels

Dans cette partie, on suppose que les 2-entiers codent des nombres entiers naturels.

16. Écrire une fonction `decremente(nn)` qui prend en argument un 2-entier `nn` et qui fait ce qui suit :
- si $n \geq 1$, elle soustrait 1 à `nn`, elle supprime les zéros finaux de `nn` et elle renvoie `True` ;
 - sinon, elle supprime les zéros finaux de `nn` et elle renvoie `False`.
- où n est l'entier correspondant à `nn`.
 Votre fonction ne devra pas utiliser l'opérateur « - » de Python.
17. Écrire une fonction `produit(nn, mm)`, qui prend en argument deux 2-entiers `nn` et `mm`, et qui renvoie leur produit (sous forme de 2-entier).
 Votre fonction ne devra pas utiliser l'opérateur « + » ni l'opérateur « * » de Python.

Partie VII – Puissance entre entiers naturels

Dans cette partie, on suppose que les 2-entiers codent des nombres entiers naturels.

18. Écrire une fonction `puissance(nn, pp)`, qui prend en argument deux 2-entiers `nn` et `pp`, et qui renvoie le 2-entier correspondant à `nnpp`.
 Votre fonction ne devra pas utiliser l'opérateur « ** » de Python.
 Votre fonction devra implémenter l'algorithme d'exponentiation rapide.

FIN DU SUJET.

