université PARIS DIDEROT PARIS 7

Partiel — corrigé | PF1 | 21/10/2015 10^{45}–12^{45}

Aucun document ou support autre que le sujet ou les copies d'examen n'est autorisé.
(la copie ou les brouillons du voisin ne sont pas des supports autorisés).
Éteignez impérativement vos mobiles.

Lorsque des calculs sont nécessaires, il est impératif de les présenter sur la feuille d'examen. Il est aussi nécessaire de **justifier** ses réponses.

1 Exercice (4 points)

Pour toutes les bases plus grandes que 10, on utilisera comme chiffres complémentaires les lettres de l'alphabet, soit `0123456789ABCDEFG...`.

1. `(836A84345B3AA8A)`$_{12}$ est-il divisible par 2 ? par 3 ?
   `(836A84345B3AA8A)`$_{12}$ vaut `(836A84345B3AA8)`$_{12} \times 12 + 10$ (puisque `A`$_{12}$ c'est 10), c'est-à-dire $x \times 12 + 10$, pour un certain x (peut importe sa valeur). Par conséquent le nombre est clairement divisible par 2 (12 et 10 le sont), mais pas par 3 car 12 l'est mais pas 10.
2. `(39AB4924BB86)`$_{12}$ est-il divisible par 2 ? par 3 ?
   `(39AB4924BB86)`$_{12}$ vaut `(39AB4924BB8)`$_{12} \times 12 + 6$, c'est-à-dire $x \times 12 + 6$ pour un certain x (peut importe sa valeur). Par conséquent le nombre est clairement divisible par 2 et par 3.
3. Sans les convertir, indiquer parmi les nombres suivants, lesquels ont une écriture binaire se terminant `00` : `(4444444444)`$_{10}$, `(40604042)`$_{10}$, `(40678020)`$_{10}$, `(13278909810)`$_{10}$.
   Un nombre dont l'écriture binaire termine par `00` est divisible par 4 (4 s'écrit `(100)`$_2$) (et réciproquement). `(4444444444)`$_{10}$ est divisible par 4 son écriture en base 2 se terminera par `00`. `(40604042)`$_{10}$ est divisible par 2 mais pas par 4 (sa division par 2 produit un nombre impair), par conséquent son écriture binaire ne termine pas par `00`. `(40678020)`$_{10}$ est divisible par 4, son écriture binaire termine par `00`. `(13278909810)`$_{10}$ n'est pas divisible par 4, son écriture binaire ne termine pas par `00`.
4. Convertir en base 2, le nombre `(827)`$_{10}$.
   $827 = 2 \times 413 + 1$, $413 = 2 \times 206 + 1$, $206 = 2 \times 103 + 0$, $103 = 2 \times 51 + 1$, $51 = 2 \times 25 + 1$, $25 = 2 \times 12 + 1$, $12 = 2 \times 6 + 0$, $6 = 2 \times 3 + 0$, $3 = 2 \times 1 + 1$, $1 = 2 \times 0 + 1$. `(827)`$_{10}$ = `(1100111011)`$_2$
5. Sans effectuer le calcul arithmétique, comment s'écrit en base 2 le résultat du calcul (en base 10) suivant : $8 \times 827 + 3$?
   $8 = 2^3$, s'écrit donc `1000`$_2$. par conséquent il suffit de prendre l'écriture de 827 en base 2 et lui rajouter 3 zéros en fin, donc 8×827 s'écrit `(1100111011000)`$_2$. On doit lui ajouter 3 qui s'écrit `(11)`$_2$, donc $8 \times 827 + 3$ s'écrit `(1100111011011)`$_2$
6. Calculer en base 2 la somme de `101011` et de `1011110`, poser le calcul avec les retenues.
7. Calculer en base 2 le produit de `10001110101` par `1011`, poser le calcul avec les retenues.

2 Exercice (4 points)

En fouillant les archives d'un vieux scientifique, on a retrouvé les traces de ses calculs ; malheureusement il semble y manquer quelques informations, la seule chose certaine est qu'il utilisait la base 10, et que pour chaque trace deux lettres ne peuvent correspondre à un même chiffre décimal. Voici deux des traces retrouvées :

```
    A B
+   B A
  C B C
```

```
    M A N G E R
+   M A N G E R
  G R O S S I R
```

Retrouver pour chacune la correspondance avec les chiffres de la base 10 de sorte que les additions soient numériquement correctes (aucun des nombres n'est écrit avec un `0` en tête). Préciser quels calculs/déductions vous ont mené à la solution. Pour chaque énigme la correspondance est différente...

Pour la **première énigme**, on sait qu'aucun zéro n'est présent en tête de l'écriture des nombres donc `A` $\neq 0$, `B` $\neq 0$ et `C` $\neq 0$. `C` est la retenue de l'addition des chiffres précédents (avec peut-être une retenue), mais une retenue ne peut-être autre chose que 0 ou 1 quelque soit la base, par conséquent `C` $= 1$. On a donc :

```
    A B
+   B A
  1 B 1
```

Ce qui nous donne les équations suivantes : (dernière colonne) `A` + `B` $= 11$ car ni `A` ni `B` ne sont nuls, il y a donc une retenue ; et (colonne

du milieu) A + B + 1 = 1B soit A + B + 1 = 10 + B, par conséquent A + 1 = 10 donc A = 9, et la première équation nous donne B = 2. En effet, 92 + 29 = 121.

Pour la **seconde énigme**, sait que G = 1, donc

		M	A	N	1	E	R
+		M	A	N	1	E	R
	1	R	O	S	S	I	R

. (dernière colonne) il n'y a qu'une possibilité R = 0, donc

	M	A	N	1	E	0
+	M	A	N	1	E	0
1	0	O	S	S	I	0

. (première colonne) 2×M = 10, il ne peut y avoir une retenue en provenance de la seconde colonne sinon le résultat serait impair, par conséquent M = 5, donc

	5	A	N	1	E	0
+	5	A	N	1	E	0
1	0	O	S	S	I	0

. On a les équations 2 × E = rI (on ne connait pas r), 2 + r = S (pas de retenue possible), 2 × N = r′S et 2 × A + 1 = O. L'équation 2 × N = r′S nous permet de déduire que S est pair, donc que $r = 0$ et donc S = 0 (on savait que 2+r = S), de plus on peut en déduire que $r' = 1$ et N = 6 (N ne peut être égal à 1 déjà pris...), donc

	5	A	6	1	E	0
+	5	A	6	1	E	0
1	0	O	2	2	I	0

et il reste 2 × E = I et 2 × A + 1 = O. I est donc pair, il ne peut être égal à 4 puisque 2 est déjà pris par S, ni 6 (déjà pris par S), reste donc I = 8 et donc E = 4. Reste à résoudre 2 × A + 1 = O. O est impair et ne peut être égal à 3 car 1 est déjà pris, ni 5 car 2 déjà pris, ni 9 car 4 déjà pris, reste donc 7. A = 3 et O = 7. On a bien 536140+536140=1072280.

3 Exercice (8 points)

Dans cet exercice on s'intéresse à la représentation des nombres en base 2 sur 7 bits.

1. Combien de nombres peut-on représenter au plus ?
   Il y a $2^7 = 128$ mots différents de longueur 7 sur un alphabet a deux lettres. Si chaque mot représente un nombre différent, on peut au plus représenter 128 nombres différents.
2. Si la representation est non signée, quels sont les nombres entiers représentés ? Donner votre réponse sous la forme d'un intervalle.
   Le mot `0000000` représentera 0 (le plus petit) et le mot `1111111` représentera 127 (le plus grand), et l'intervalle sera donc $[0, 128[$ (ou $[0, 127]$).
3. Dans la représentation non signée, quels sont les codages des nombres $(37)_{10}$ et $(126)_{10}$?
   $37 = 2 \times 18 + 1$, $18 = 2 \times 9 + 0$, $9 = 2 \times 4 + 1$, $4 = 2 \times 2 + 0$, $2 = 2 \times 1 + 0$, $1 = 2 \times 0 + 1$ ce qui donne `100101` que l'on doit compléter pour obtenir un mot de longueur 7 soit `0100101`.
   $126 = 2 \times 63 + 0$, $63 = 2 \times 31 + 1$, $31 = 2 \times 15 + 1$, $15 = 2 \times 7 + 1$, $7 = 2 \times 3 + 1$, $3 = 2 \times 1 + 1$, $1 = 2 \times 0 + 1$, ce qui donne `1111110`.
4. Dans la représentation non signée, le mot `0111000` correspond-il à un nombre divisible par 8 ? par 7 ?
   Il est divisible par 8 car $8 = 2^3$ et les trois derniers chiffres sont des zéros (on a à faire à une multiplication par la puissance de la base).
   Il est divisible par 7 car 7 s'écrit $(111)_2$ par conséquent le nombre est 7×2^p qui est divisible par 7 (on sait même que $p = 3$).
5. Dans la représentation non signée, le mot `1100110` correspond-il à un nombre divisible par 2 ? par 3 ?
   Il est divisible par 2 car son dernier chiffre est 0.
   Il est divisible par 3. 3 s'écrit 11_2. Il est facile de constater que `1100110`=`1100000`+`0000110`, or `1100000`$=3 \times 2^5$ et `0000110`$=3 \times 2$, par conséquent c'est divisible par 3.
6. Donner deux instructions Java différentes permettant d'obtenir le quotient de la division entière d'un nombre contenu dans une variable de type `int` et de nom `n` par le nombre 32.
   `n/32` et `n>>5` car $32=2^5$.
7. Si la représentation est signée en complément à 2, quels sont les nombres relatifs représentés ? Donner l'intervalle.
   En complément à deux le plus petit nombre (un nombre négatif) représenté correspond au mot `1000000`, pour connaître sa valeur absolue on calcule l'inversion chiffre à chiffre soit `0111111` et on ajoute 1 soit `1000000` ce qui correspond à $2^7 = 128$, Le plus petit nombre représenté est donc -128. Le plus grand nombre correspond au mot `0111111` qui vaut $2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127$ ou encore on remarque que `0111111`=`1000000`-`0000001` donc 128-1=127.
   L'intervalle est $[-128, 127]$ ou $[-128, 128[$.
8. Dans la représentation signée en complément à deux, quels sont les codages des nombres $(27)_{10}$ et $(-17)_{10}$?
   $27 = 2 \times 13 + 1$, $13 = 2 \times 6 + 1$, $6 = 2 \times 3 + 0$, $3 = 2 \times 1 + 1$, $1 = 2 \times 0 + 1$, $0 = 2 \times 0 + 0$, $0 = 2 \times 0 + 0$, donc 27 s'écrit `0011011`.
   Pour coder -17, il faut trouver le codage de 17. $17 = 2 \times 8 + 1$, $8 = 2 \times 4 + 0$, $4 = 2 \times 2 + 0$, $2 = 2 \times 1 + 0$, $1 = 2 \times 0 + 1$, $0 = 2 \times 0 + 0$, $0 = 2 \times 0 + 0$, donc 17 s'écrit `0010001`. Il faut prendre son inverse chiffre à chiffre soit `1101110` et lui ajouter 1 soit `1101111`.

9. Dans la représentation signée en complément à deux, à quel nombre en base 10 correspond le codage `1011110` ? C'est un nombre négatif car sa représentation commence par un 1, il faut donc calculer sa valeur absolue en prenant son inverse chiffre à chiffre, soit `0100001` et ajouter 1 ce qui donne `0100010` et calculer sa valeur en base 10 ce qui donne $2^5 + 2^1 = 32 + 2 = 34$. Le nombre est donc -34.
10. Effectuer dans la représentation signée en complément à deux les opérations suivantes (en précisant à chaque fois si le résultat peut être considéré comme correct dans l'arithmétique ordinaire) `0110010+0100101`, `0011101+0010010`, `0100111-0011011`, `0000111`$\times$`0001110`.

4 Exercice (4 points)

Soit le bout de programme Java suivant :

```
1  int i = 9;
2  short s = 22;
3  byte b = 4;
4
5  int résultat = i+s-b;
6  System.out.println(résultat);
7
8  int résultat2 = résultat*résultat;
9  System.out.println(résultat2);
10
11 byte résultat3 = (byte)(résultat*résultat);
12 System.out.println(résultat3);
```

Quels sont les affichages que produit son exécution ? Pourquoi ?
Le premier affichage sera 27 car le calcul bien qu'exprimé avec des variables de types entiers différents sera effectué en convertissant tous les types en `int`. On a donc 9+22-4 soit 27. Ce 27 étant stocké en complément à 2 sur 32 bits dans la variable `résultat`.
Le second affichage sera 729, aucun débordement ne se produit car 729 est largement inférieur au plus grand entier positif représentable en complément à deux sur 32 bits.
Le troisième affichage sera -39 car le produit calculé est bien 729 en complément à 2 sur 32 bits mais le résultat est convertit en complément à 2 sur 8 bits (type `byte`) ; cette conversion consiste en une simple tronquature, seuls les 8 derniers bits sont conservés. L'écriture de 729 en complément à deux sur 32 bits est `00000000000000000000001011011001` on aurait pu se contenter de dire `0...01011011001`. La tronquature ne conserve que les 8 derniers chiffres soit `11011001`. Le nombre étant en complément à deux il faut calculer sa valeur absolue : `00100110`+1, donc $32 + 4 + 2 + 1 = 39$. Le nombre est donc -39, d'où l'affichage.