Université Paris 7 - Denis Diderot
IF1 — Partiel — 25 Novembre 2006 — Durée : 2h 30mn
Documents, calculettes, ordinateurs et téléphones non autorisés

---

Exercice 1.

Le langage Java permet de réaliser des opérations bit à bit sur des valeurs entières : l'opération est appliquée à tous les couples de bits en même position dans les représentations binaires des deux valeurs. Les opérations permises sont :

- le ET (conjonction), opération notée `&`
- le OU (disjonction), opération notée `|`
- le OU exclusif, opération notée `^`

Pour ces opérations, la valeur `0` d'un bit est interprétée comme faux et la valeur `1` comme vrai.
Ainsi, pour deux octets (variables x et y de type `byte`) correspondant respectivement aux suites de bits 01101110 et 00111001,

- `x & y` correspond à la suite de bits 00101000
- `x | y` correspond à la suite de bits 01111111
- `x ^ y` correspond à la suite de bits 01010111

Exécutez à la main la séquence suivante en justifiant votre réponse.

```
  .......
byte b1 = 111; byte b2 = 19; byte b3 = -108;
Deug.println(b1 & b2); Deug.println(b1 | b2); Deug.println(b1 ^ b2);
Deug.println(b1 & b3); Deug.println(b1 | b3); Deug.println(b1 ^ b3);
  .......
```

Exercice 2.

Exécutez à la main la séquence Java suivante et en particulier indiquer ce qui est affiché à l'écran en justifiant vos résultats :

```
    .......
int a = 10, b = 4, c;
float x, y;
c = a + 2 * b; a = a + 5; b = b + 2;
Deug.println(a); Deug.println(b); Deug.println(c);
c = a / b; x = a / b;
Deug.println(c); Deug.println(x);
y = b; x = a / y;
Deug.println(x);
    .......
```

Exercice 3.

L'utilisation des tableaux n'est pas autorisée dans cet exercice.

On s'intéresse aux résultats d'une élection dans un scrutin de listes dans lequel il y a n sièges à pourvoir et trois listes candidates obtenant respectivement a, b et c voix.
Ecrire un programme qui :

- demande le nombre de sièges à pourvoir ;
- demande le nombre de voix pour chacune des trois listes ;
- vérifie que les données sont des entiers valides (au moins un siège et nombres de voix obtenues par les listes positifs ou nuls) ;
- calcule le coefficient électoral Q, c'est-à-dire le quotient entier de la division du nombre total de suffrages exprimés par le nombre de sièges à pourvoir ;
- calcule et affiche le nombre de sièges obtenus par chacune des listes selon la règle de la meilleure liste. Une liste obtient ainsi automatiquement un nombre de sièges égal au quotient de la division entière du nombre de voix qu'elle a obtenues par Q. Les sièges restants sont attribués à la liste ayant obtenu le plus de voix.

Exercice 4.
L'utilisation des tableaux n'est pas autorisée dans cet exercice.

Ecrire un programme qui permette à son utilisateur d'entrer au clavier des valeurs entières non nulles de son choix et calcule à la volée (sans mémoriser tous les entiers lus) d'une part la somme `sPos` et le produit `pPos` des valeurs positives lues et d'autre part la somme `sNeg` et le produit `pNeg` des valeurs négatives lues. Lorsque le programme lira la valeur 0 (cette valeur n'étant pas considérée comme faisant partie des données à traiter), la lecture se terminera, les quatre valeurs calculées seront affichées et le programme s'arrêtera. Si aucun nombre positif n'est lu, `sPos` et `pPos` seront nuls et s'il n'y en a pas de négatif lu, `sNeg` et `pNeg` seront nuls.
Exemple : pour les entiers saisis 4, 2, -8, 3, -1, -2, 5, 0, on calcule et affiche finalement sPos = 14 (4+2+3+5), pPos = 120 (4*2*3*5), sNeg = -11 (-8 + (-1) + (-2)) et pNeg = -16 (-8 * (-1) * (-2)).

Exercice 5.
Cet exercice utilise des tableaux.
Une question peut être traitée en réutilisant ce qui a été fait dans les précédentes

On dispose d'une suite de nombres entiers dans un tableau t. On se propose d'écrire une fonction réalisant le tri par ordre croissant des valeurs dans ce tableau sans jamais comparer deux éléments du tableau entre-eux. Pour cela on suppose que chaque élément est compris entre 0 et n, où n est un entier "pas trop grand", quitte à lisser aux extrémités en identifiant à 0 toutes les valeurs négatives à 0 et à n toutes les valeurs supérieures à n. Il est alors possible d'allouer un tableau d'entiers de taille $n+1$. La méthode utilisée consiste à compter dans ce tableau le nombre de 0, le nombre de 1, le nombre de 2, ... qui apparaissent dans le tableau initial. Cela conduit à ce que nous appelons la *représentation occurrentielle* du tableau.

5.1. Ecrire une méthode Java `repOcc1` qui prend en argument un tableau t d'entiers et un entier n supposé positif ou nul, et qui construit sa représentation occurrentielle, c'est-à-dire le tableau *occ* du nombre d'occurrences de chaque entier $\leq n$ qu'il contient et renvoie la référence de ce nouveau tableau.
Par exemple, si le tableau *t=[3 1 0 4 3 2 1 1 6 3 2 0 1 0 6 2]* et si $n = 6$, *occ* sera *[3 4 3 3 1 0 2]* alors que si $n = 5$ le tableau *occ* sera *[3 4 3 3 1 2]* (par lissage, les deux valeurs 6 sont assimilées à 5) et si $n = 7$ il sera *[3 4 3 3 1 0 2 0]* (le dernier élément est inutile).

5.2. Ecrire une méthode Java `repOcc2` qui, étant donné un tableau t d'entiers quelconques, recherche le plus petit entier n positif ou nul permettant sa représentation occurrentielle sans lissage de valeurs positives, construit le tableau correspondant à cette représentation et renvoie sa référence. Cette représentation est dite optimale.
Par exemple, pour le tableau précédent, la représentation occurrentielle sera calculée pour $n = 6$. Pour un tableau ne contenant que des valeurs négatives, n sera égal à 0 (tous les élements du tableau seront assimilés à 0).

5.3. Ecrire une méthode Java `tabTriSansRep` qui, étant donné un tableau $t1$ d'entiers quelconques, construit un tableau $t2$ correspondant à sa forme triée sans répétition : il contient, en ordre croissant un exemplaire unique de chacun des nombres appartenant au tableau $t1$, les nombres négatifs étant confondus avec 0. La fonction renverra la référence du tableau construit.
Par exemple, si le tableau $t1$ est *[3 1 0 4 -1 3 2 1 1 6 3 2 0 1 0 6 2]*, le tableau *[0 1 2 3 4 6]* sera construit par la fonction.

5.4. Ecrire une fonction `afficheTab` qui étant donné un tableau d'entiers reçu en paramètre affiche ses valeurs séparées par un espace à raison de 20 éléments par ligne.

5.5. Ecrire un programme Java qui
- demande et lit la taille d'un tableau t ;
- après avoir alloué le tableau, lit ses éléments ;
- lit un entier positif non nul et affiche la représentation occurrentielle du tableau t relativement à cet entier ;
- calcule et affiche sa représentation occurrentielle optimale (cf. 5.2) ;
- calcule et affiche sa forme triée sans répétition (cf. 5.3).