TP Les sous-programmes et les tableaux

Exercice 1 — *Multiple de 3*

Dans cet exercice, nous allons écrire un algorithme qui détermine si un nombre est multiple de trois, en utilisant la propriété suivante :

"un nombre est multiple de 3 si la somme de ses chiffres est elle-même un multiple de 3".

1. Écrire une fonction `int sommeChiffre (int nombre)` qui prend un nombre entier en paramètre, en renvoie la somme des chiffres de ce nombre.

   Exemple : si le nombre est 978, la somme sera $9+7+8=24$. Voici l'algorithme que nous allons utiliser que nous illustrons sur le nombre 978 :

   (a) Pour extraire le nombre des unités, on fait d'abord la division entière de 978 par 10, et on obtient 97. On multiplie ensuite ce résultat par 10, et on obtient 970. Il ne reste plus qu'à soustraire au nombre initial 978 le résultat obtenu 970, et on obtient le chiffre des unités : $978-970=8$.

   (b) Pour extraire le chiffre des dizaines, on divise notre nombre par 10 (division entière), on obtient 97 et on recommence : on fait la division entière de 97 par 10, on obtient 9, que l'on multiplie par 10, on obtient 90, que l'on soustrait à 97, et on obtient : $97-90=7$

   (c) Pour extraire celui des centaines, on divise notre nombre par 10 (division entière), on obtient 9 et on recommence : on fait la division entière de 9 par 10, on obtient 0, que l'on multiplie par 10, on obtient 0, que l'on soustrait à 9, et on obtient : $9-0=9$

   Il faudra evidemment additionner à chaque itération chaque chiffre obtenu dans un compteur que l'on retournera à la fin de la méthode.

   Ainsi, vous écrirez d'abord la méthode `int extraireChiffreUnite(int nombre)` qui extrait le chiffre des unités du `nombre` passé en paramètre. Il suffira ensuite dans la méthode `int sommeChiffre (int nombre)` d'appeler la méthode `extraireChiffreUnite` itérativement comme décrit dans l'exemple.

2. Écrire une fonction `boolean multipleTrois(int nombre)` qui renvoie un booléen permettant de savoir si un nombre est un multiple de 3 ou non. Pour cela, on calculera la somme des chiffre du nombre tant que celle si est supérieure à 10. Il suffira ensuite de vérifier si cette somme vaut 3, 6 ou 9 pour déterminer si le nombre est bien un multiple de 3.

Exercice 2 — *Tableau à 2 dimensions*

1. Écrivez une fonction

   ```
   public static int[][] saisie()
   ```

   qui lit deux entiers `n` et `m`, puis $\texttt{n} \times \texttt{m}$ entiers, puis retourne un tableau de `n` lignes et `m` colonnes contenant les entiers saisis.

2. Écrivez une fonction

   ```
   public static void affichage(int[][] a)
   ```

qui affiche le tableau passé en paramètre.

Écrivez une fonction `main` qui saisit un tableau et l'affiche, et testez votre programme.

3. Écrivez une fonction

   ```
   public static boolean recherche(int[][] a, int v)
   ```

   qui retourne `true` s'il existe une paire d'indices i, j tels que `a[`i`][`j`]` $=$ `v`.

4. Écrivez une fonction `main` qui lit un tableau au clavier, une valeur, et affiche si la valeur se trouve dans le tableau. (Vous pouvez bien-sÃ»r vous servir des fonctions écrites dans la partie précédente.)

Exercice 3 — *carré magique*

Un *carré magique* est une matrice carrée de taille $n \times n$ telle que la somme de chaque rangée, de chaque colonne et de chaque diagonale ait la même valeur. Un carré magique est dit *normal* s'il contient chaque entier compris entre 1 et n^2 exactement une fois. Par exemple, le tableau suivant est un carré magique normal :

$$\begin{bmatrix} 6 & 7 & 2 \\ 1 & 5 & 9 \\ 8 & 3 & 4 \end{bmatrix}$$

1. Écrivez une fonction

   ```
   public static boolean carre(int[][] a)
   ```

   qui retourne `true` si le tableau `a` est une matrice carrée (qui a autant de lignes que de colonnes). Écrivez une fonction `main` et testez votre programme.

2. Écrivez deux fonctions

   ```
   public static int ligne(int[][] a, int i)
   public static int colonne(int[][] a, int j)
   ```

   qui retournent la somme de la `i`-ème ligne (resp. de la `j`-ème colonne) du tableau passé en paramètre. Écrivez une fonction `main` qui vous permette de tester ces fonctions.

3. Écrivez deux fonctions

   ```
   public static int diagonale1(int[][] a)
   public static int diagonale2(int[][] a)
   ```

   qui retournent la somme de la diagonale majeure (resp. de la diagonale mineure) du tableau passé en paramètre. Écrivez une fonction `main` qui vous permette de tester ces fonctions.

4. Écrivez une fonction

   ```
   public static int[] histogramme(int[][] a, int n)
   ```

   qui retourne l'histogramme du tableau `a`, c'est-à-dire le tableau `h` de taille `n` tel que `h[`i`-1]` contient le nombre d'occurrences de la valeur i dans le tableau `a`.

5. Écrivez une fonction

   ```
   public static boolean normal(int[][] a)
   ```

   qui retourne `true` si le tableau `a` est normal et `false` sinon.

6. Écrivez un programme qui demande à l'utilisateur de rentrer un tableau, et affiche s'il s'agit d'un carré magique normal.

Exercice 4 — *la courte paille - (Bonus)*

5 joueurs décident de choisir l'un d'entre eux en tirant à la courte paille. La paille est représentée par un entier compris entre `1` et `5`. Les joueurs sont représentés par les lettres `a`, `b`, `c`, `d`, `e`.

Compléter le programme suivant qui tire aléatoirement une paille pour chaque joueur et affiche le nom du gagnant. Il est possible qu'il y ait égalité de pailles gagnantes, dans ce cas, le joueur dont le nom est le plus petit en suivant l'ordre alphabétique gagne.

Rappel :
Pour effectuer un tirage pseudo-aléatoire, il faut créer un générateur d'une séquence pseudo-aléatoire de la classe `Random`. Il faut l'importer :
`import java.util.Random`
puis créer un objet initialisant la séquence
`Random alea = new Random();`
puis générer un entier à l'aide de la méthode `nextInt( int x )` qui retourne la valeur aléatoire dans l'intervalle `[0,x[` qui suit dans la séquence.
`int a = alea.nextInt(x);`

```
import java.util.Random;
public class CourtePaille{
  public static void main(String[] args ){
  // déclarer et initialiser le tableau représentant les 5 pailles
   ...
  // déclarer et initialiser le tableau représentant les 5 joueurs
   ...
  // création d'un objet initialisant la séquence
  Random alea = new Random();
  // tirage aléatoire des 5 pailles.
  for( int i=0;i<5;i++ )
   ...
  // détermination de l'indice de la plus courte et donc de l'indice du gagnant
  // il est possible qu'il y ait égalité.
  // l'indice de la paille dans le tableau paille est en correspondance avec l'indice du joueur
  // dans le tableau joueurs.
  int plusCourte = 0;
  for( int i=1;i<5;i++ )
          ...
  // affichage du joueur gagnant
  System.out.println("␣le␣gagnant␣est␣:␣"+ ... );
  }
}
```