

## 1 Préambule

- **Exercice 1** : Écrire une fonction `chaîne` qui prend en argument un tableau bidimensionnel d'entiers et renvoie une chaîne de caractères représentant son contenu.
- **Exercice 2** : Écrire une fonction `saisie` qui lit deux entiers  $n$  et  $m$ , puis  $n \times m$  entiers, et renvoie un tableau bidimensionnel d'entiers de  $n$  lignes et  $m$  colonnes contenant les entiers saisis.

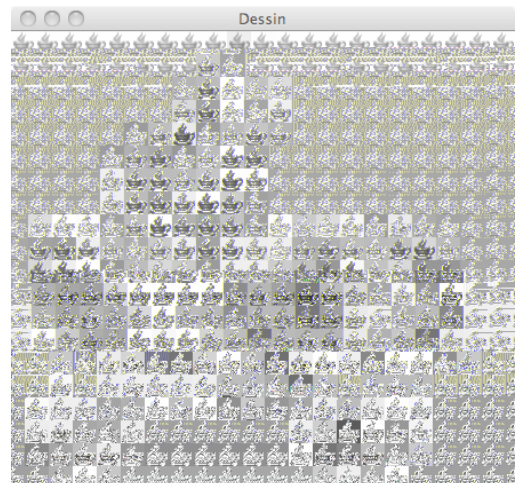
## 2 Recherche de motifs

- **Exercice 3** : Écrire une fonction `recherche` qui prend en argument un tableau bidimensionnel d'entiers et une valeur entière et teste si cette valeur apparaît dans une des cases du tableau.
- **Exercice 4** : Écrire une fonction `recherche` qui prend en arguments deux tableaux bidimensionnels d'entiers  $a$  et  $s$  et teste s'il existe un sous-tableau de  $a$  qui est égal à  $s$ . En d'autres termes, s'il existe des indices  $i$  et  $j$  tels que pour tous  $h < s.length$  et  $k < s[0].length$ , on a  $a[i+h][j+k] = s[h][k]$ .

## 3 Images en niveaux de gris

Une image en niveaux de gris est représentée par une matrice d'entiers courts `short[][]` contenant des valeurs comprises entre 0, représentant la couleur noire, et 255, représentant la couleur blanche.

- **Exercice 5** : Écrire une fonction `dessiner` qui dessine l'image en niveaux de gris passée en argument. En supposant que l'image contient des zones entières de couleur identique, comment peut-on minimiser le nombre d'appels à `setGray` ?
- **Exercice 6** : Écrire une fonction `antiAlias` qui implémente un *filtre d'anti-aliasage*. Elle renvoie une nouvelle image de la même taille que l'image en argument où les « créneaux » ont été adoucis en utilisant les niveaux de gris : l'intensité d'un pixel de la nouvelle image sera égale à la somme des trois quarts de la valeur du pixel correspondant de l'ancienne image et d'un quart de la moyenne des valeurs des huit pixels voisins.
- **Exercice 7** : Écrire une fonction `mosai` qui prend une image  $t$  en argument et renvoie une image mosaïque  $m$  de taille  $h^2 \times k^2$  (où  $h \times k$  est la taille  $t$ ) obtenue en remplaçant chaque pixel  $p$  de  $t$  par une copie de  $t$  à laquelle on applique un filtre dépendant du niveau de gris  $v$  du pixel  $p$  : chaque niveau de gris  $n$  de la copie est au choix remplacé par  $\frac{v \times n}{256}$ , par  $\sqrt{v \times n}$  ou par  $\frac{v+n}{2}$ .



## 4 Carrés magiques

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

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

► **Exercice 8 :** On se propose d'écrire une fonction `bachetDeMeziriac` qui prend en argument un entier  $n$  positif et renvoie un carré magique de dimension  $c \times c$  avec  $c=2*n+1$  selon la méthode de Bachet de Méziriac dont on peut trouver une explication sur le site <http://www.kandaki.com/CM-Construc11.htm>. Le schéma à l'adresse <http://i.imgur.com/rRBjL.png> peut aussi aider.

► **Exercice 9 :** On souhaite tester si un tableau bidimensionnel est un carré magique normal.

1. Écrire une fonction `estCarre` qui teste si un tableau bidimensionnel d'entiers donné en argument est une matrice carrée.
2. Écrire deux fonctions `sommeLigne` et `sommeColonne` qui prennent en arguments un tableau bidimensionnel d'entiers et un entier  $k$  et renvoient respectivement la somme des éléments de la ligne d'indice  $k$  et celle des éléments de la colonne d'indice  $k$  de ce tableau.
3. Écrire deux fonctions `sommeDiagonalNO` et `sommeDiagonalNE` qui prennent en argument un tableau bidimensionnel d'entiers et renvoient respectivement la somme de sa diagonale Nord-Ouest–Sud-Est et celle de sa diagonale Nord-Est–Sud-Ouest.
4. Écrire une fonction `estMagique` qui teste si le tableau en argument correspond à un carré magique.
5. Écrire une fonction `histogramme` qui prend en arguments un tableau `a` bidimensionnel d'entiers et un entier  $n$  et renvoie le tableau `h` de taille  $n$  dans lequel la valeur de `h[k]` pour  $0 \leq k < n$  est le nombre d'occurrences de la valeur  $k$  dans le tableau `a`.
6. Utiliser les deux fonctions précédentes pour écrire une fonction `estMagiqueNormal`.

► **Exercice 10 :** Pour tout entier positif  $n$ , on se propose de construire un carré magique de côté de longueur  $c=2*n+1$  selon la *méthode du losange* due à Conway (voir la construction pour  $n=3$  ci-dessous) :

- les entiers **impairs** entre 1 et  $c^2$  sont positionnés le long de diagonales d'un losange inscrit dans le carré ;
- les entiers **pairs** entre 1 et  $c^2$  sont positionnés en poursuivant les diagonales ;
- le tout est glissé dans le carré en identifiant les côtés gauche et droit et les côtés haut et bas.

	0	1	2	3	4	5	6
0				43			
1			29	37	45		
2		15	23	31	39	47	
3	1	9	17	25	33	41	49
4		3	11	19	27	35	
5			5	13	21		
6				7			

LosangeImp(3)

	0	1	2	3	4	5	6
0				43			
1			29	37	45		
2		15	23	31	39	47	
3	1	9	17	25	33	41	49
4		3	11	19	27	35	36
5			5	13	21	22	30
6				7	8	16	24

2 10 18 26 34 42  
4 12 20 28  
6 14

	0	1	2	3	4	5	6
0	26	34	42	43	2	10	18
1	20	28	29	37	45	4	12
2	14	15	23	31	39	47	6
3	1	9	17	25	33	41	49
4	44	3	11	19	27	35	36
5	38	46	5	13	21	22	30
6	32	40	48	7	8	16	24

Losange(3)

1. Construire à la main les carrés magiques selon la méthode de Conway pour  $n=1$  et pour  $n=2$ .
2. Écrire une fonction `LosangeImp` qui prend un entier positif  $n$  en argument et renvoie un carré de côté  $2n+1$  (sous forme de tableau bidimensionnel) dont le losange central est rempli selon la méthode de Conway.
3. Préciser comment modifier `LosangeImp` en une fonction `Losange` qui complète cette construction.