Université Denis Diderot - DEUG MASS/MIAS - IF121
Examen du lundi 19 janvier 2004 - Durée : 2 heures 30 minutes
Téléphones portables éteints - Les ordinateurs ne sont pas autorisés
Seules les notes de cours distribuées et vos notes personnelles sont autorisées.

Exercice 1

Rappels : on appelle PGCD de deux nombres entiers, le plus grand nombre entier qui les divise tous les deux (il est donc inférieur ou égal au plus petit des deux nombres) et on dit que deux nombres sont premiers entre eux si leur PGCD est égal à `1`.

L'objectif de cet exercice est de calculer de manière naïve le PGCD de deux nombres entiers a et b. La seule contrainte est de ne pas utiliser de tableau.

1.1. Écrire une fonction `estDiviseur` qui, étant donnés deux nombres entiers x et y, renvoie la valeur booléenne `true` si x divise y et la valeur booléenne `false` sinon.

1.2. Écrire une fonction `valeurPgcd1` qui, étant donnés deux entiers a et b positifs, renvoie la valeur du PGCD des deux nombres en testant pour chaque nombre x, à partir de `1`, la divisibilité de a et b par x.

1.3. Écrire une fonction `valeurPgcd2` qui, étant donnés deux entiers a et b positifs, renvoie la valeur du PGCD des deux nombres en testant pour chaque nombre x à partir du plus grand nombre susceptible d'être le PGCD la divisibilité de a et b par x.

1.4. Écrire un programme complet regroupant la méthode `estDiviseur` et l'une au choix des méthodes de calcul du PGCD qui lit deux nombres entiers et affiche leur PGCD.

1.5. Écrire une fonction `nbEuler` qui, étant donné un nombre entier n strictement positif, renvoie le nombre (entier) d'entiers inférieurs à n qui sont premiers avec lui (par exemple `nbEuler(6)` renvoie `2` car, parmi les nombres qui lui sont inférieurs, `6` n'est premier qu'avec `1` et `5`).

Exercice 2

On s'intéresse ici à des tableaux de nombres entiers à une seule dimension, possédant un nombre impair d'éléments et dont tous les éléments sont différents.

On appelle **médiane** d'un tel tableau T l'élément m de T tel que T contienne autant d'éléments strictement inférieurs à m que d'éléments strictement supérieurs à m.

2.1. Écrire une fonction `nbInf` qui, étant donné un tableau d'entiers T et un entier v, renvoie le nombre d'éléments du tableau T strictement inférieurs à v.

2.2. Écrire une fonction `mediane` qui, étant donné un tableau T satisfaisant les conditions énoncées, renvoie la position de la médiane dans le tableau T.

2.3. Écrire une fonction `verifTableau` qui, étant donné un tableau T de nombres entiers, renvoie la valeur booléenne `true` si le tableau T satisfait effectivement les conditions énoncées et la valeur `false` si ce n'est pas le cas.

2.4. Écrire une fonction `tabInf` qui, étant donné un tableau T, retourne `null` si T ne satisfait pas les conditions énoncées et un tableau contenant tous les éléments de T inférieurs à la médiane de T sinon.

Exercice 3

Lorsqu'une fonction renvoie la position d'un élément dans un tableau à une dimension, elle se contente de renvoyer un numéro de case, c'est-à-dire une valeur de type `int`.

Si l'on souhaite définir une fonction renvoyant la position d'un élément dans un tableau à *deux* dimensions, cette fonction doit renvoyer simultanément deux informations : un numéro de ligne et un numéro de colonne. Ceci n'est possible qu'en réunissant ces deux informations dans une même entité renvoyée par la fonction, par exemple un tableau d'entiers à deux éléments.

3.1. Écrire une fonction

```
static double distance(int x0, int y0, int x1, int y1)
```

qui renvoie la distance, au sens géométrique usuel, entre les points du plan de coordonnées (`x0`,`y0`) et (`x1`,`y1`).
On rappelle que la distance entre les points (x, y) et (x', y') est donnée par la formule :

$$\sqrt{(x - x')^2 + (y - y')^2}$$

On pourra utiliser la fonction prédéfinie

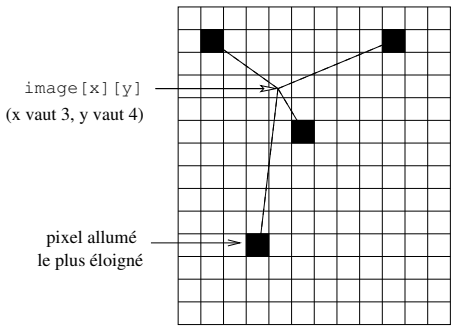
```
double Math.sqrt(double a)
```

qui renvoie la racine carrée du nombre transmis en paramètre.

3.2. À l'aide de la fonction `distance` précédente, écrire une fonction

```
static int[] plusEloigne(int x, int y, boolean[][] image)
```

répondant aux spécifications suivantes :
- `image` représente une image rectangulaire, sous la forme d'un tableau de booléens : une case de ce tableau de valeur `true` représente un pixel allumé, et une case de valeur `false` un pixel éteint ;
- la fonction doit créer et renvoyer un tableau à deux éléments contenant *la position du pixel allumé le plus éloigné de la position* `[x][y]`.



La première case du tableau renvoyé contiendra le numéro de ligne du pixel trouvé, et la seconde, son numéro de colonne. On supposera que, lorsque la fonction est appelée, l'image contient toujours au moins un pixel allumé. On pourra utiliser l'algorithme suivant :
- créer un tableau d'entiers `pos` à deux éléments ;
- stocker `x` et `y` dans `pos` ;

– pour chacune des positions `[i][j]` de l'image :

  si le pixel de position `[i][j]` est allumé, et si la distance entre les pixels de positions `[x][y]` et `[i][j]` est supérieure à la distance entre le pixel de position `[x][y]` et le pixel dont la position est stockée dans `pos`, alors stocker `i` et `j` dans `pos` ;

– renvoyer `pos`.

3.3. Écrire une fonction

```
static int[] plusProche(int x, int y, boolean[][] image)
```

renvoyant les coordonnées du pixel allumé le plus *proche* de la position `[x][y]`. Comme précédemment, on supposera que l'image contient toujours au moins un pixel allumé et vous pouvez, au choix (que vous préciserez) :

– supposer que la position `[x][y]` est toujours située dans l'image, et adapter l'algorithme précédent à ce nouveau problème, en initialisant le contenu de `pos` à des valeurs suffisamment grandes dépendantes de la taille de l'image ;
– trouver un meilleur algorithme, fonctionnant même lorsque la position `[x][y]` est située en dehors de l'image.

3.4. Plutôt que de faire renvoyer à `plusEloigne` et `plusProche` leur résultat dans un tableau à deux cases, on aurait pu également définir une nouvelle classe d'objets contenant les deux données à renvoyer.

a) Définir une classe `Position` permettant de représenter une position sous forme d'objet, une position correspondant donc à une seule entité regroupant deux variables entières `ligne` et `colonne` et ajouter à cette classe :

– un constructeur permettant de créer et d'initialiser un nouvel objet à partir de deux valeurs entières passées en argument ;
– un constructeur permettant de créer et d'initialiser un nouvel objet à partir d'une position donnée sous forme d'un tableau de deux entiers ;
– un constructeur permettant de créer et d'initialiser un nouvel objet à partir d'une position donnée sous forme d'une référence sur un objet existant de la classe `Position`.

b) Définir une fonction `positionPlusEloignee` recevant les mêmes paramètres que la fontion `plusEloigne` et qui renvoie un objet de type `Position` correspondant au pixel allumé le plus éloigné du point (on pourra appeler la fonction `plusEloigne`).