Durée de l'épreuve : 2h

Modalités : tous les documents, exceptés les livres, sont autorisés. Bien faire figurer les numéros des questions auxquelles vous répondez.

Vocabulaire : afin de faciliter l'énoncé des questions, nous utiliserons parfois le terme de « tableau » pour désigner un pointeur qui pointe sur une zone mémoire allouée pour plusieurs éléments d'un même type.

Le sujet propose d'écrire quelques fonctions très simples pour faire du calcul matriciel. Les types suivants forment la base des fonctions à développer.

```
typedef struct matrice *matrice;
struct matrice {
  int l;     /* Represente le nombre de lignes de la matrice      */
  int c;    /* Represente le nombre de colonnes de la matrice */
  double *m;     /* Le tableau des coefficients de la matrice */
}
```

avec la convention que les coefficients d'une matrice M sont stockés dans le tableau m ligne par ligne. Ainsi, la matrice

$$M = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

verra ses coefficients stockés dans le tableau {1, 2, 3, 4} (et non pas {1, 3, 2, 4}).

1. Ecrire une fonction

   ```
   matrice cree_matrice(int l, int c)
   ```

   qui crée une matrice de l lignes et c colonnes et la renvoie. Cette fonction doit faire toutes les allocations mémoire nécessaires. Elle renvoie NULL si l'un des paramètres a une valeur négative.

2. Ecrire une fonction

   ```
   void detruit_matrice(matrice m)
   ```

   qui libère (toute) la mémoire utilisée par la matrice m passée en paramètre.

3. Ecrire une fonction

   ```
   double get_coefficient(matrice m, int i, int j)
   ```

   qui renvoie le coefficient (i,j) (i.e. le coefficient de la ième ligne et de la jème colonne) de la matrice m. Cette fonction renvoie NAN si l'un des paramètres i ou j est incorrect (NAN est une constante définie dans math.h).

4. Ecrire une fonction

   ```
   void set_coefficient(matrice m, int i, int j, double v)
   ```

   qui fait en sorte que le coefficient (i,j) de la matrice m prenne la valeur v.

5. Ecrire une fonction

   ```
   matrice recopie(matrice m)
   ```

   qui crée une matrice semblable à m. Si m vaut NULL alors cette fonction ne crée rien et renvoie NULL.

6. Ecrire une fonction

   ```
   matrice mult(double v, matrice m)
   ```

   qui crée et renvoie une matrice qui est la matrice m multipliée par le scalaire v. Si la matrice m vaut NULL, alors cette fonction ne crée rien et renvoie NULL.

7. Ecrire une fonction

```
matrice somme(matrice m1, matrice m2)
```

qui crée et renvoie une matrice qui est la somme des 2 matrices m1 et m2. Si les dimensions des matrices ne sont pas compatibles ou que l'une des matrices passées en paramètre vaut NULL, cette fonction en crée rien et renvoie NULL. (Pour rappel : la somme de deux matrices donne une matrice dont les coefficients sont les sommes des éléments qui se correspondent dans les 2 matrices.)

8. Ecrire une fonction

```
matrice produit(matrice m1, matrice m2)
```

qui crée et renvoie une matrice qui est le produit des 2 matrices m1 et m2. Si les dimensions des matrices ne sont pas compatibles ou que l'une des matrices passées en paramètre vaut NULL, cette fonction ne crée rien et renvoie NULL. (Attention : il s'agit bien ici du produit matriciel usuel.)

9. Ecrire une fonction

```
double norm2(matrice m)
```

qui renvoie la norme euclidienne de m si m est une matrice colonne ou une matrice ligne (i.e. m->l=1 ou m->c=1) et qui renvoie NAN dans tous les autres cas. (Pour rappel, la norme euclidienne du vecteur $\mathbf{v} = [v_1 \dots v_d]$ à d composantes est donnée par $\|v\| = \sqrt{\sum_{i=1}^{d} v_i^2}$.

10. Ecrire une fonction

```
void affiche(matrice m)
```

qui produit l'affichage de la matrice m.

11. Ecrire une fonction main qui fait les choses suivantes :
    (a) crée 2 matrices 5×5 et les remplit aléatoirement de nombres compris entre -1 et +1,
    (b) calcule la somme des 2 matrices et l'affiche,
    (c) calcule le produit des 2 matrices et l'affiche,
    (d) libère toute la mémoire allouée,
    (e) se termine.
12. Proposer une manière de regrouper les différentes fonctions écrites et donner un **makefile** permettant la compilation du code produit.
13. Une manière de calculer la plus grande valeur propre d'une matrice M est la suivante (on suppose, pour simplifier l'algorithme, que M n'a que des valeurs propres positives) :
    (a) choisir un vecteur $\mathbf{v}^{(0)}$ de manière aléatoire
    (b) $i = 0$, $s = 0$
    (c) Répéter
        i. $\mathbf{w} = M\mathbf{v}^{(i)}$
        ii. $s = \|\mathbf{w}\|$
        iii. $\mathbf{v}^{(i+1)} = \mathbf{w}/s$
        iv. $d = \|\mathbf{v}^{(i+1)} - \mathbf{v}^{(i)}\|/\|\mathbf{v}^{(i)}\|$
        v. $i = i + 1$

        jusqu'à ce que $d \leq \varepsilon$
    (d) renvoyer s (qui est une estimation de la plus grande valeur propre)

    où $\varepsilon > 0$ est une constante fixée et petite (par exemple $\varepsilon = 1e^{-5}$).
    Programmer une fonction

```
double vp(matrice m)
```

qui renvoie la plus grande valeur propre de m en utilisant l'algorithem décrit.