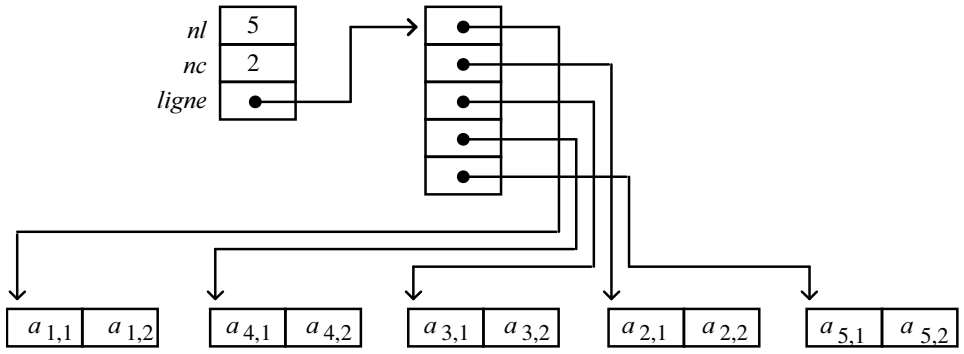
« Supermatrices »

L'objet de ce problème est l'écriture d'une *bibliothèque de fonctions* constituant une implantation améliorée des matrices de nombres flottants, par exemple en double précision. Une matrice codée comme nous le proposons ici est appelée une *supermatrice*. Elle est composée de trois éléments :

1° une structure, appelée le *descripteur* de la supermatrice, faite de trois champs :
- *nl*, le nombre de lignes de la matrice
- *nc*, le nombre de colonnes de la matrice
- *ligne*, l'adresse du tableau décrit ci-dessous

2° un tableau de *nl* pointeurs, qui sont les adresses des lignes de la matrice. Le premier élément pointe la première ligne, le second la deuxième, etc.

3° un espace où sont alloués les $nl \times nc$ éléments de la matrice (des **`double`**), rangés par lignes. Lorsque cela n'entraînera pas un travail supplémentaire, on fera en sorte que ces lignes occupent un espace contigu, dans lequel elles sont rangées en accord avec leurs indices, mais cette propriété ne pourra pas être toujours garantie. Bien entendu, les coefficients d'une même ligne seront rangés de manière contiguë.



Le travail demandé consiste à écrire

- dans un fichier *supermat.c*, les fonctions spécifiées ci-dessous. Seule la version compilée de ce fichier (le fichier objet) est destinée a être fournie aux utilisateurs de notre bibliothèque,
- dans un fichier *supermat.h*, les déclarations nécessaires pour utiliser cette bibliothèque,
- un programme « client » qui teste toutes les fonctions (même s'il n'a aucune autre fonction).

Nous supposons désormais que le type **`SUPERMAT`** a été défini comme « adresse d'un descripteur de supermatrice ».

1. La fonction :

```
SUPERMAT allouerSupermat(int nl, int nc);
```

alloue une supermatrice entièrement nouvelle, de taille *nl* lignes × *nc* colonnes. Les coefficients $a_{i,j}$ auront des valeurs quelconques, mais tous les autres éléments de la supermatrice auront été correctement initialisés.

2. La *macro* :

```
double acces(SUPERMAT a, int i, int j);
```

doit être telle que « **`acces(a, i, j)`** » soit une *lvalue* représentant $a_{i,j}$. Elle permet donc à un client d'accéder aux coefficients d'une supermatrice sans avoir à faire un usage explicite de la structure de cette dernière.

Question : pourquoi en faire une macro plutôt qu'une fonction ?

3. La fonction :

```
SUPERMAT superProduit(SUPERMAT a, SUPERMAT b);
```

prend en entrée deux supermatrices *a* et *b* et rend une nouvelle supermatrice, représentant le résultat du produit matriciel $a \times b$, ou NULL si ce produit n'a pas pu être effectué (mauvais nombre de lignes et colonnes, ou impossibilité d'allouer la matrice résultat).

4. La fonction :

```
void permuterLignes(SUPERMAT a, int i, int j);
```

permute les lignes *i* et *j* de la supermatrice *a*. Cette opération sera programmée en visant l'efficacité maximum.

5. La fonction

```
SUPERMAT sousMatrice(SUPERMAT a, int l1, int l2, int c1, int c2);
```

alloue et garnit la sous-matrice définie par les éléments $(a_{i,j})$ de la supermatrice *a* tels que $l_1 \leq i \leq l_2$ et $c_1 \leq j \leq c_2$. La sous-matrice créée utilisera les coefficients déjà alloués lors de la construction de *a* : on n'allouera un espace nouveau que pour le descripteur et la table de lignes.

6. La fonction

```
SUPERMAT matSupermat(double *m, int nld, int ncd, int nle, int nce);
```

effectue la transformation matrice → supermatrice, c'est à dire création d'une supermatrice initialisée avec les coefficients de la matrice ordinaire dont *m* représente l'*adresse du premier élément*. La fonction rend la supermatrice créée à titre de résultat. On allouera de l'espace uniquement pour le descripteur et le tableau des lignes (la supermatrice nouvelle partagera donc les coefficients avec la matrice initiale).

Les valeurs *nld* et *ncd* sont respectivement le nombre de lignes et de colonnes figurant dans la déclaration de la matrice pointée par *m*. Les valeurs *nle* et *nce* sont les nombres *effectifs* de lignes et de colonnes de cette matrice.

7. La fonction

```
void supermatMat(SUPERMAT sm, double *m, int nld, int ncd);
```

effectue la transformation supermatrice → matrice. La matrice ordinaire, dont *m* représente l'*adresse du premier élément* (dimension déclarée : $nld \times ncd$) est garnie avec les coefficients de la supermatrice *sm*.

8. La fonction

```
int contiguite(SUPERMAT a);
```

analyse la supermatrice *a* et rend :

- 2 si les lignes de *a* sont contiguës et placées dans l'ordre des indices : la deuxième ligne suit la première, la troisième suit la seconde, etc. ;
- 1 si les lignes de *a* sont contiguës mais dans le désordre ;
- 0 dans les autres cas.

9. Pensez-vous qu'il est possible d'écrire une fonction

```
void rendreSupermat(SUPERMAT sm);
```

qui serait la « réciproque » de **`allouerSupermat`**, c'est-à-dire qui libérerait l'espace occupé par la supermatrice indiquée.

☺☹☹