On se propose d'implémenter en Maple quelques variantes des algorithmes du pivot. Dans la première partie on forme quelques procédures de bases qui seront combinées dans la deuxième partie.
Dans tout le texte, la matrice à p lignes et q colonnes est stockée sous la forme d'un tableau `array(1..p,1..q)` de nom `A`.
Les noms `A`, `p`, `q` sont passés aux procédures comme des variables *globales*.
Cet usage d'une variable globale est commode pour l'implémentation informatique. Elle a le défaut de conforter une erreur très fréquente qui est d'écrire une suite de transformations élémentaires comme une suite d'égalités.

1. Procédures primaires

Toutes les procédures demandées doivent figurer dans la même zone d'exécution qui commence par une commande `restart;`.
Chacune doit être testée dans une zone particulière bien renseignée et les tests ne doivent pas être effacés. Les noms des procédures doivent être exactement respectés. Si on n'indique pas ce q'une procédure doit renvoyer, elle ne renverra rien ni n'affichera rien. Elles travaillent *en place*. Vous pouvez par exemple terminer par :

```
...
return();
end proc;
```

Écrire les procédures suivantes :
- `permutL(i,j)` : permute les lignes i et j du tableau `A`.
- `permutC(i,j)` : permute les colonnes i et j du tableau `A`.
- `ajoutL(i,j,lambda)` : ajoute λ-fois la ligne j à la ligne i.
- `nettoieB(i)` : nettoie le bas(à partir de la ligne $i+1$) de la colonne i comme dans l'algorithme I du cours. C'est à dire, pour j de $i+1$ à p, en ajoutant λ-fois (pour le bon λ) la ligne i à la ligne j.
- `nettoieT(i)` : nettoie toute la colonne i comme dans l'algorithme I' du cours.
- `pivotP(i)` : renvoie une liste $[,i]j$ avec j entre i et p tel que $a_{j,i} \neq 0$, renvoie 0 si un tel entier n'existe pas.
- `pivotT(i)` : renvoie une liste $[j,k]$ avec j entre i et p et k entre i et q tels que $a_{j,k} \neq 0$, renvoie 0 si un tel couple d'entiers n'existe pas.
- `pivotPM(i)` : renvoie une liste $[j,i]$ avec j entre i et p tel que

$$a_{j,i} = \max\{|a_{k,i}|, k \in \{i, \cdots, p\}\}$$

renvoie 0 si $a_{j,i} = 0$.

2. Procédures principales

1. Écrire des procédures `alI()`, `alIp()`,`alII()` implémentant les algorithmes I,I' et II du cours ne renvoyant rien et affichant le mot "fin" lorsqu'elles se terminent.
   Ces procédures ne seront pas directement appelées par d'autres mais serviront à fournir l'ossature du code des procédures suivantes.
2. Écrire une procédure `inversible()` qui renvoie `true` si la matrice `A` est inversible et `false` si elle ne l'est pas.
3. Écrire une procédure `rang()` qui renvoie le rang de la matrice `A`.
4. Écrire une procédure `inverse()` qui renvoie un tableau contenant la matrice inverse de A si celle ci est inversible et 0 si elle ne l'est pas.
5. Écrire une procédure `deter()` qui renvoie le déterminant d'un matrice carrée.
6. Écrire une procédure qui procède à une décomposition LU d'une matrice.

Rémy Nicolai Apivot