Structures de données
TD - 1

1. Exo. 1 - Bibliothèque pour manipuler des matrices (Tableaux de `float` à deux dimensions).
   L'implémentation d'une matrice peut se faire avec deux approches :
   — en utilisant un tableau de tableaux ;
   — en utilisant un unique tableau de $N \times M$ éléments.
   (a) Déterminez comment peut-on implémenter un tableau à deux dimensions (avec N lignes et M colonnes) à partir d'un unique tableau de $N \times M$ éléments ? Quel est l'avantage de cette implémentation par rapport a une implémentation avec des tableau de tableaux ?
   (b) Pour les deux implémentations possibles, écrivez la structure de donnée pour stocker une matrice de N lignes et M colonnes.
   (c) Pour les deux implémentations possibles, écrivez la fonction pour allouer une matrice de N lignes et M colonnes.
   (d) Pour les deux implémentations possibles, écrivez la fonction pour libérer une matrice.
   (e) Pour les deux implémentations possibles, écrivez les fonctions pour lire et écrire dans la matrice.
   (f) À partir des accesseur de la question (e), écrivez la fonction pour calculer le produit matriciel entre deux matrices. Pour rappel, la multiplication entre deux matrices $\mathbf{A} \in \mathcal{M}_{M,N}$ et $\mathbf{B} \in \mathcal{M}_{N,P}$ s'écrit :

$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} \times b_{k,j} \tag{1}$$

   avec $c_{i,j}$ la i-ème ligne et j-ème colonne de la matrice $\mathbf{C} \in \mathcal{M}_{M,P}$.
   (g) Quand l'on lit ou écrit dans la mémoire, il est important que les opérations successives soit faites sur des cases successives dans la mémoire. Il faut impérativement éviter les sautes dans la mémoire pour favoriser la mise en cache des données et diminué les temps d'accès aux données.
   Dans votre réponse de la question (f), regardez si vous avec des sauts dans la mémoire et proposez des modifications dans vos codes pour éviter ces sauts en mémoire.

2. Exo. 2 - Recherche des k plus petits éléments d'un tableau `T`. Nous considérons T un tableau non trié de n entiers.
   (a) Écrivez un algorithme qui retourne les indexes des k plus petits éléments du tableau `T` en procédant par recherches successives.
   (b) Écrivez un algorithme qui retourne les indexes des k plus petits éléments du tableau `T` en triant le tableau au préalable (la fonction de tri par ordre croissant est donnée `tri(T)` ; son coût est : $\mathcal{O}(n \log(n))$)
   (c) Comparez le temps mis par chacun des algorithmes pour $k = 1..n$.