Laboratoire

Classe : PRG1-E	**Labo no** : 05	Vecteur

But

- Utilisation de tableaux « vector »
- Utilisation intense de « algorithm » et « numercics »
- => Utiliser le moins de boucles possibles

A faire

Développer une librairie mettant à disposition les éléments nécessaires à une manipulation facilitées de vecteurs et de matrices d'entiers.

Vocabulaire	Vecteur	tableau (vector) à 1 dimension
	Matrice	tableau à 2 dimensions (vector de vector)

Celui-ci doit contenir les sous-programmes suivants :

Operateur <<	Affiche un Vecteur au format `(v1, v2, …, vn)` [1]
Operateur <<	Affiche une Matrice au format `[(..),(..),(..)]` [1]
estCarree	Retourne un booléen indiquant si la matrice est carrée [2]
estReguliere	Retourne un booléen indiquant si la matrice est régulière [2] (toutes les lignes de même taille)
minCol	Retourne la longueur min des vecteurs d'une matrice
sommeLigne	Retourne un vecteur contenant la somme des valeurs de chacune des lignes.
vectSommeMin	Retour le vecteur d'une matrice dont la somme des valeurs est la plus faible
shuffleMatrice	Mélanger les vecteurs d'une matrice sans altérer les vecteurs. La *seed* du générateur est basée sur l'heure [3]
sortMatrice	Trier dans l'ordre croissant une matrice en fonction de l'élément min d'un vecteur. `[(4, 4), (1, 3), (2)] => [(1, 3), (2), (4, 4)]`
sommeDiagDG	Rend par un paramètre la somme des valeurs de la diagonale « / ». Retourne *false* en cas d'erreur et *true* si ok.
sommeDiagGD	Rend dans un paramètre la somme des valeurs de la diagonale « \ ». *false* en cas d'erreur et *true* si ok.

Un programme principal de votre choix doit montrer le bon fonctionnement de votre librairie. Aucun saisie utilisateur n'est demandée.

Notes

1 En utilisant des itérateurs et sans utiliser « auto »
2 Une matrice vide est régulière
3 https://www.cplusplus.com/reference/random/default_random_engine

L a b o r a t o i r e

La surcharge des opérateurs de flux n'est pas encore vue.
Vous pouvez faire une fonction « afficher » et/ou lire **Stream extraction**

Stream extraction and insertion

The overloads of `operator>>` and `operator<<` that take a std::istream& or std::ostream& as the left hand argument are known as insertion and extraction operators. Since they take the user-defined type as the right argument (`b` in a@b), they must be implemented as non-members.

```
std::ostream& operator<<(std::ostream& os, const T& obj)
{
    // write obj to stream
    return os;
}
```

```
std::istream& operator>>(std::istream& is, T& obj)
{
    // read obj from stream
    if( /* T could not be constructed */ )
        is.setstate(std::ios::failbit);
    return is;
}
```