TP 4-5 / C++ / CIR 2 (STL)

- **Exercice 1**

➔ Exécuter et commenter le programme C++ suivant :

```
#include <iostream>
#include <set>
using namespace std;

int main(){
        int a[] = {7, 4, 9, 1, 3, 4, 8, 2, 7, 5, 3, 6, 10, 4, 8, 10, 1, 2};
        multiset<int> s(&a[0], &a[17]);
        multiset<int>::iterator p = s.begin();
        while (p != s.end()) cout << *p++ << " ";
        return 0;
}
```

➔ Reprendre le même code avec un container de type set.

- **Exercice 2**

En vous basant sur la classe fraction et des algorithmes STL, écrire un programme qui permet de lire une liste de fractions à partir d'un fichier et de les afficher dans l'ordre croissant, ainsi que :

➔ d'afficher la somme totale des fractions.
➔ de supprimer les valeurs répétées,
➔ de supprimer les valeurs négatives.
➔ d'afficher la valeur minimale,
➔ d'afficher la valeur maximale.

- **Exercice 3**

➔ Implémenter la fonction suivante:

```
template<typename Iter>
typename Iter::value_type
partial_dot(Iter first1, Iter last1, Iter first2, Iter last2)
```

La fonction partial_dot est le produit scalaire entre deux conteneurs limités par first1 et last1 pour le premier et first2 et last2 pour le second.

- **Exercice 4**
  Considérant un ensemble de valeurs entières contenues dans un std::vector, utiliser les fonctions de la bibliothèque « algorithm »

(http://en.cppreference.com/w/cpp/algorithm) afin de résoudre de manière efficace les problèmes suivants (veillez à minimiser l'utilisation explicite de boucles) :

- ➔ Déterminer si toutes les valeurs sont positives ou nulles;
- ➔ Remplacer toutes les valeurs négatives par 0;
- ➔ Afficher toutes les valeurs positives (utiliser un std::ostream_iterator)
- ➔ Déterminer la position du premier et du dernier élément négatif.

- **Exercice 5**

Déclarer et remplir une « std::map » qui permet d'associer un numéro de mois à son nom :

- ➔ Imprimer cette liste dans l'ordre alphabétique.
- ➔ Ecrire une procédure de conversion de dates du format « 31 Décembre 2014 » au format « 31/12/2014 ».

Exercice 6

- ➔ Déclarer et remplir une structure « std::map<string, string> » qui permettra d'associer un acronyme (Junia, ISEN, ISEN, FIFA, SFR, CEDEX, …) à sa signification. Le programme demande à l'utilisateur de saisir un acronyme, affiche la signification trouvé, ou « inconnu » le cas échéant en demandant la nouvelle signification à ajouter à la liste. Le programme s'arrête lors de la saisie du mot « fin ».

La liste initiale des acronymes est stockée dans un fichier, ligne par ligne, sous la forme :

Acronyme=Définition

- **Exercice 7**

Ecrire un programme permettant de réaliser le processus d'authentification. Ce programme devra :

- ➔ récupérer la liste des utilisateurs à partir d'un fichier (username/password crypté) et la stocker dans une structure de type « std::map »,
- ➔ créer un utilisateur, s'il est lancé avec l'argument « create » (Le mot de passe est à confirmer par l'utilisateur)
- ➔ effectuer trois tentatives d'authentification avant de quitter, s'il est lancé avec l'argument « login ».

- **Exercice 8 (déjà vu en cours mais finalement assez intéressant à consulter à nouveau)**

Modifier le programme de l'exercice 3 du TP2 pour permettre de compter les occurrences des lettres et mots présents dans un texte (lu à partir d'un fichier) à l'aide de l'une des classes set, multiset, map, multimap :

- ➔ Les séparateurs et les ponctuations seront ignorés.
- ➔ Le programme ne différencie pas la casse (ne distingue pas les majuscules des minuscules).

➔ Affiche les listes des lettres et mots avec le nombre d'occurrences dans deux ordres : alphabétique et par ordre d'apparition.