

Exercice 2 : sur les multensembles (12 points)

Un multensemble est une collection dans laquelle l'ordre des éléments n'importe pas mais où une même valeur peut apparaître plusieurs fois. Par exemple $\{3, 2, 1\} = \{1, 3, 2\}$ mais $\{1, 2, 3, 3\} \neq \{1, 2, 2, 3\} \neq \{1, 2, 3\}$. Pour coder un multensemble, nous allons stocker les valeurs dans un tableau en ajoutant leur multiplicité. Par exemple le multensemble $\{15.5, 3.25, 15.5, 1.0\}$ sera stocké comme le tableau

| | | |
|-----------|-----------|----------|
| (15.5, 2) | (3.25, 1) | (1.0, 1) |
|-----------|-----------|----------|

; effectivement, la valeur 15.5 apparaît 2 fois, 3.25 et 1.0 n'apparaissent qu'une seule fois.

Le problème, c'est que les couples n'existent pas en C. Nous allons commencer par contruire des couples dont le premier élément est un flottant et le second un entier.

1. Créer un nouveau type `couple`. Cette structure aura deux champs : un champs `valeur` de type flottant et un champs `nombre` de type entier. Par exemple si `valeur` vaut 15.5 et `nombre` vaut 3, cela signifiera que le nombre 15.5 apparaît trois fois dans notre multensemble.

.....

.....

.....

.....

.....

.....

2. Écrire une fonction `nouveau_couple` qui prend en paramètre un flottant `x` et renvoie un `couple` de valeur `x`. Par défaut, le champs `nombre` sera initialisé à 1.

.....

.....

.....

.....

.....

.....

.....

3. Créer maintenant un nouveau type `multensemble`. Un `multensemble` sera défini à partir d'une structure contenant trois champs :

- `mem` : un tableau de `couple`. Le tableau sera alloué sur le tas.
- `taille` : un entier correspondant à la taille du tableau.
- `n` : un entier correspondant au nombre de cases du tableau effectivement utilisées.

.....

.....

.....

.....

.....

.....

7. Écrire une fonction `ajout` qui prend en paramètre un pointeur vers un `multensemble` et un flottant `x` et ajoute ce dernier au `multensemble`.

- Si un couple de valeur égale à `x` se trouve déjà dans le tableau, on incrémente le champs `nombre` correspondant.
- Sinon, on ajoute un nouveau couple correspondant à `x` à la fin du tableau. On pourra utiliser la fonction de la question 2. On fera bien attention d'avoir la place nécessaire pour ajouter le nouveau couple, quitte à utiliser la fonction `agrandir` (uniquement si nécessaire).

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

8. Écrire une fonction `nettoyage` qui prend en argument un pointeur vers un `multensemble` et libère toute la mémoire correspondante.

.....

.....

.....

.....

.....

