Manipulation de Listes

On donne les définitions suivantes pour représenter des listes chainées.

```
typedef int T;
typedef struct maillon *Liste;
struct maillon {
   T donnee;
   Liste suivant;
};

Liste cons (Liste l, T nvelt){
  Liste p;
  p = malloc(sizeof(struct maillon));
  p->donnee = nvelt;
  p->suivant=l;
  return(p);
}

int appartient(T x, Liste l){
  if ( l == NULL ) return(0);
  if ( x == l->donnee ) return(1);
  return(appartient(x, l->suivant));
}
```

Exercice 1

1. Écrire une fonction `add` qui prend une liste l, un élément e de type T et ajoute e à l.
2. Écrire une fonction `affiche` qui affiche la liste qui lui est passée en argument.

Exercice 2

1. Écrire une fonction itérative `position` qui pour une liste l et un élément e de type T retourne la position de la première occurrence de e dans l.
2. Écrire une fonction itérative `nombre` qui pour une liste l et un élément e de type T retourne le nombre d'occurremces de e dans l.

Exercice 3

Écrire de manière récursive et itérative une fonction `somme` qui calcule la somme des éléments d'une liste.

Exercice 4

Question 4.1

Écrire une fonction `recopie` qui retourne une copie de la liste qui lui est passée en argument.

Question 4.2

Écrire une fonction `concatener` qui prend en paramètre une liste l_1 et une liste l_2 et retourne une liste qui contient les éléments de l_1 suivis de ceux de l_2. On pourra partager la liste l_2 mais on ne modifiera pas la liste l_1

Question 4.3
Récrire `concatener` pour qu'elle modifie en place la première liste qui lui est passée en argument et retourne une valeur de `void`.

Exercice 5

Question 5.1
En utilisant la fonction `appartient` écrire une fonction `contient` prenant deux listes l_1 et l_2 en paramètres et retourne 1 si tous les éléments de l_1 sont dans l_2 et 0 sinon.

Question 5.2

1. En vous inspirant de ce qui a été fait sur les chaines, écrire une fonction `facteur_gauche` qui pour deux listes l_1 et l_2 retourne 1 si la liste l_1 est un début de la liste l_2 et 0 sinon.
2. En déduire une fonction `sous_liste` qui pour deux listes l_1 et l_2 retourne 1 si la liste l_1 est une partie de la liste l_2 et 0 sinon.

Exercice 6
Une *liste d'association* est une liste dont les éléments sont des couples (c, v) où c est une *clé* et où v est une *valeur*. On pourra prendre le type `char` pour les clés et le type `int` pour les valeurs.

1. Donner les types nécéssaires pour représenter des listes d'association.
2. Écrire les fonctions `clef_de` et `valeur_de` qui pour une liste d'association non vide retournent sa première clé et sa première valeur.
3. Écrire une fonction `ajoute` prenant en paramètres une liste d'association l, une clé c ainsi qu'une valeur v et qui retourne une liste dans laquelle le couple c, v a été ajouté au début de l.
4. Écrire une fonction `valeur_de` qui prend en paramètres une liste d'association l ainsi qu'une clé c et retourne la valeur correspondant à la première occurrence de c dans l.
5. Ici les clés sont des `char` et les valeurs sont des `int`. Écrire une fonction `plus` qui prend en paramètres deux clés c_1 et c_2 ainsi qu'une liste d'association l et retourne la somme des valeurs associées à c_1 et à c_2 dans l.