Programmation C L2.1
Listes chaînées ordonnées

Définition de la structure

On utilise des listes chainées d'entiers dans lesquelles les cellules sont ordonnées par valeurs croissantes.
Le type `Liste` est inchangé, c'est au programmeur d'assurer la bonne gestion de la liste ordonnée. Les fonctions de manipulation doivent veiller à maintenir ou utiliser l'ordre. On utilisera la structure suivante.

```
typedef struct cellule {
 int valeur;
 struct cellule *suivant;
}Cellule,*Liste;
```

Exercice 1. Manipulation

- *Écrire une fonction* `int insertionOrdonnee(Liste* lst, int val)` *qui ajoute la valeur* `val` *dans la liste en conservant l'aspect ordonné de celle-ci. Il faudra utiliser la fonction d'allocation de cellule. La fonction* `insertionOrdonnee` *renverra 0 en cas de problème et 1 sinon.*
- *Écrire une fonction* `int creeListeTriee(int n, Liste* lst)` *qui crée une liste triées de* `n` *entiers aléatoires en les insérant au fur et à mesure et qui renvoie le nombre de cellules crées.*
- *Écrire une fonction* `int nombreInferieur(Liste lst, int val)` *renvoyant le nombre d'entiers inférieurs à* `val` *dans la liste ordonnée* `lst`.
- *Écrire une fonction* `int supprimeDoublon(Liste lst)` *qui supprime les doublons de la liste* `lst`. *La fonction renvoie le nombre de valeurs supprimées. (On pourrait également écrire une fonction équivalente de prototype* `int supprimeDoublon(Liste *lst)` *suivant la cellule supprimée).*

Exercice 2. Fusion

On veut fusionner les cellules de 2 listes triées en une seule liste.

- *Écrire une fonction* `void fusion(Liste *un, Liste *deux)` *qui transforme* `*un` *en la fusion des 2 listes. La liste* `*deux` *sera vide après l'appel. On utilisera directement les cellules des listes sans allocation.*
- *Utiliser la fonction de fusion pour écrire la fonctions* `void triFusion(Liste *lst)` *effectuant le tri fusion de la liste non ordonnée* `lst`.

Exercice 3. *Dans cet exercice on souhaite comparer deux manières de créer une liste triée de taille* `n`.

- *Écrire une fonction* `int creeListe(int n, Liste* lst)` *qui crée une liste (non triée) de* `n` *entiers aléatoires par insertion en tête. La fonction renverra le nombre de cellules crées.*
- *Comparer le temps d'execution de la création d'une liste de 10000 nombres aléatoires triée en utilisant la fonction* `creeListeTriee` *ou en utilisant la fonction* `creeListe` *avec la même suite d'entiers alátoires (réinitialiser le fonction avec la même graine) puis en triant la liste obtenue avec le tri fusion. Votre* `main` *contiendra le code suivant :*

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

#define TAILLE 100000
[...]
time_t graine = time();
Liste lstTriee, lstFusion;
time_t t, tpsTriee, tpsFusion;
srand(graine); /* Initialisation de l'aleatoire avec 'graine'*/
t = time(NULL);
// On cree directement une liste triee
if (creerListeTriee(TAILLE, &lstTriee) != TAILLE) {
    fprintf(stderr,"Erreur d'allocation lors de la creation de lstTriee");
    return 1;
   }
tpsTriee = time(NULL) - t;

srand(graine); /* Re-initialisation de l'aleatoire avec 'graine' pour obtenir les memes nombres*/
t = time(NULL);
/* On commence par creer une liste non triee*/
if (creerListe(TAILLE, &lstFusion) != TAILLE){
    fprintf(stderr,"Erreur d'allocation lors de la creation de lstFusion");
    return 1;
    }
/* On trie la liste obtenue*/
triFusion(&lstFusion);
tpsFusion = time(NULL) - t;

printf("Le temps necessaire a la creation d'une liste de taille %d triee directement
       est %d secondes.\n", TAILLE, tpsTriee);
printf("Le temps necessaire a la creation d'une liste de taille %d puis d'efectuer un
       tri fusion est %d secondes.\n", TAILLE, tpsFusion);
```