Listes Chainées

Dans ce TD, nous allons voir comment manipuler des listes chainées stockant des éléments de façon efficace.

Chaque maillon d'une liste est représenté par la structure suivante :

```
typedef structure cellule{
entier valeur;
LienSur structure cellule suivant;
}Cellule;
```

▸ **Exercice 1.** ***Insertion***
Écrire une fonction `inserer(entier element,transforme LienSur Cellule liste)` *qui ajoute l'élément passé en paramètre à la liste. On suppose que l'on possède une fonction* `LienSur Cellule ReserverCellule()` *qui réserve de la place pour une cellule et retourne un lien dessus.*

▸ **Exercice 2.** ***Longueur d'une liste***

Écrire une fonction `entier longueur(LienSur Cellule liste)` *renvoyant la longueur d'une liste chainée d'entier.*

▸ **Exercice 3.** ***Affichage***
Écrire une fonction `afficherListe(LienSur Cellule liste)` *qui affiche la liste donnée en paramètre. Pour afficher un entier on pourra utiliser la fonction* `afficherEntier(entier n)`.

▸ **Exercice 4.** ***Suppression***
Écrire une fonction `booleen supprimer(entier element,transforme LienSur Cellule liste)` *qui supprime l'élément passé en paramètre si il est présent dans la liste. La fonction doit renvoyer* `VRAI` *si l'élément a été trouvé (et supprimé) ou* `FAUX` *sinon. Pour libérer l'espace mémoire d'une cellule qui a été alloué (avec* `reserverCellule()`*) on utilisera la fonction* `liberer(LienSur Cellule liste)`.

▸ **Exercice 5.** *Réécrire les fonctions* `inserer(entier element,transforme LienSur Cellule liste)` *et* `supprimer(entier element, transforme LienSur Cellule liste)` *en supposant que la liste chainée est triée par ordre croissant. La liste doit rester triée après appel de ces deux fonctions.*