Types abstraits de données

Compétences

- Compléter l'implémentation du type abstrait de données **liste**
- Algorithmique sur le type liste

Introduction

Sur une liste nous n'avons pas accès directement à un élément du corps de la liste. Le seul accès possible est celui à la valeur de tête. Pour effectuer une opération il est donc nécessaire de partir de l'élément de tête et de parcourir le corps de la liste. On peut distinguer deux cas :

1. La liste est vide : fin du travail
2. La liste n'est pas vide et on fait un appel récursif sur le corps si le traitement n'est pas terminé.

Pour les exercices suivants nous utiliserons le fichier *liste.py* ou *liste_sentinelle.py*

Implémenter les fonctions suivantes :

1. Écrire une fonction récursive `inputListRec()` permettant de saisir une liste au clavier. On arrête la saisie quand l'utilisateur saisie une chaîne vide
2. Écrire une fonction récursive `printListRec(L:liste)` permettant d'afficher le contenu d'une liste.
3. Écrire une fonction récursive `strListRec(L:liste) -> str` qui renvoie une chaîne représentant la liste.
4. Écrire une fonction `search(L:liste, element) -> bool` qui recherche la présence d'un élément dans une liste.
5. Écrire une fonction `sortedSearch(L:liste, element) -> bool` qui recherche d'un élément dans une liste triée par odre croissant. Attention la recherche doit s'arrêter dès que l'on se rend compte que l'élément n'y est pas.
6. Écrire une fonction `deleteItem(L:liste, element) -> liste` qui permet de supprimer un élément de la liste.
7. Écrire une fonction `deletePosition(L:liste, position:int) -> liste` qui permet de supprimer l'élément occupant la position donnée.
8. Écrire une fonction `insertPosition(L:liste, position:int) -> liste` qui permet d'insérer un élément à une position donnée. Si la position est plus grande que la taille de la liste, alors l'insertion se fait à la fin de la liste.
9. Écrire une fonction `insertSorted(L:liste, element) -> liste` qui permet d'insérer un élément à sa position dans une liste triée
10. Écrire une fonction `changeElement(L:liste, old, new)->liste` qui recherche et supprime un élément puis insert le nouvel élément.
11. Écrire une fonction `changePosition(L:liste, position:int, new) -> liste` qui recherche et supprime un élément puis insert le nouvel élément.
12. Écrire une fonction `merge(L1:liste, L2:liste) -> liste` qui effectue la fusion de deux listes.

    ```
    L1:1->3->5
    L2:2->4->6->8
    résultat:1->2->3->4->5->6->8
    ```

13. Écrire une fonction `sortedMerge(L1:liste, L2:liste)->liste` qui effectue la fusion triée de deux listes par ordre croissant.

    ```
    L1:1->7->11
    L2:2->4->6->8->10
    résultat:1->2->4->6->7->8->10->11
    ```