TP 9 – LISTES CHAÎNÉES

Info1.Algo1 - 2022-2023 Semestre Impair

Exercice 1

Dans le fichier **ex01__lecture__modification.py** , compléter le corps des fonctions suivantes afin que chacune d'entre elles passent la fonction de test associée :

- La fonction `nombre_elements` qui accepte en paramètre une `liste` et retourne le nombre d'éléments de cette liste.
- La fonction `lire_element` qui accepte en paramètre une `liste` ainsi qu'un indice entier `i` et retourne la `valeur` de l'élément placé à l'indice `i` dans la liste.
- La fonction `ecrire_element` qui accepte en paramètre une `liste`, un indice entier `i` ainsi qu'une `valeur`, écrit cette `valeur` à l'indice `i` et retourne la liste ainsi constituée.

Exercice 2

Dans le fichier **ex02__insertion__suppression.py**, compléter le corps des fonctions suivantes afin que chacune d'entre elles passent la fonction de test associée :

- La fonction `ajouter_element_fin` qui accepte en paramètre une `liste` ainsi qu'une `valeur`, ajoute cette valeur en fin de liste et retourne la liste ainsi constituée.
- La fonction `inserer_element` qui accepte en paramètre une `liste`, un indice entier `i` ainsi qu'une `valeur`, insère cette `valeur` dans la liste à l'indice `i` et retourne la `liste` ainsi constituée.

  (si l'indice `i` est égal à la longueur de la liste, la `valeur` est ajoutée en fin de liste)
- La fonction `supprimer_element` qui accepte en paramètre une `liste` et un indice entier `i`, supprime l'élément placé à l'indice `i` dans la liste, et retourne la liste ainsi constituée.

Exercice 3

Dans le fichier **ex03__somme__maximum.py**, compléter le corps des fonctions suivantes afin que chacune d'entre elles passent la fonction de test associée :

- La fonction récursive `somme` qui accepte en paramètre une `liste` et retourne la somme de ses éléments.
- La fonction récursive `maximum` qui accepte en paramètre une `liste` non vide et retourne la plus grande de ses valeurs.

Exercice 4

Dans le fichier **ex04__comparaison.py**, compléter le corps de la fonction récursive `sont_egales` qui accepte en paramètres deux listes `liste1` et `liste2` et retourne `True` si les listes contiennent exactement la même séquence de valeurs, et `False` sinon.

Exercice 5

Dans le fichier **ex05__inversion.py**, compléter le corps de la fonction récursive `inverser_rec` qui accepte deux listes en paramètres :

- `liste` : une liste dont on souhaite obtenir une copie mais dans l'ordre inverse.
- `liste_inverse` : une liste vide lors du premier appel et qui permet de constituer progressivement la liste inverse au fil des appels récursifs.

Indication : La fonction récursive `inverser_rec` est appelée par la fonction **(non récursive)** `inverser` qui ne nécessite pas le paramètre `liste_inverse`.

Exercice 6

Dans le fichier **ex06__recherche.py**, compléter le corps de la fonction récursive `rechercher_rec` qui accepte trois paramètres :

- `liste` : une liste.
- `valeur` : une valeur dont on cherche les indices dans la liste.
- `indice` : un entier égal à `0` lors du premier appel et qui permet de savoir à quel indice on se trouve dans la liste de départ.

La fonction retourne la liste des indices dans `liste` où `valeur` à été trouvée.

Indication : La fonction récursive `rechercher_rec` est appelée par la fonction **(non récursive)** `rechercher` qui ne nécessite pas le paramètre `indice`.

Exercice 7

Dans le fichier **ex07__extraire__negatifs__positifs.py**, compléter le corps de la fonction `extraire_negatifs_positifs` qui accepte en paramètres une liste chaînée contenant des entiers et retourne un tuple constitué de 2 listes chaînées. La première liste chaînée est constituée des éléments **strictement négatifs** de la liste donnée en paramètre, et la seconde liste chaînée est constituée des éléments **positifs ou nuls**. L'ordre des éléments dans chacune des listes retournées doit être conservé.

Exemple :

Si la liste chaînée donnée en entrée est `(16,(-6,(2,(-7,(0,(9,None))))))`, la fonction `extraire_negatifs_positifs` doit retourner un tuple constitué des deux listes suivantes :

- `(-6,(-7,None))`
- `(16,(2,(0,(9,None))))`

Exercice 8

Dans le fichier **ex08__convertir__en__chaine.py**, écrire la fonction récursive `convertir_en_chaine` qui accepte en paramètre une liste chaînée `liste` et retourne la chaîne de caractère de ses éléments, chaque élément étant suivi d'un espace (même le dernier).

Exemple :

Si la liste chaînée `liste` vaut `(8,(4,(7,(1,None))))`, alors la chaîne de caractères retournée est `"8 4 7 1 "`.

Exercice 9

Dans le fichier **ex09__premier__indice.py**, compléter le corps de la fonction `premier_indice` qui accepte en paramètre une `liste` chaînée ainsi qu'une `valeur`, et retourne l'indice de la première occurrence de `valeur` dans la liste. Si la `valeur` n'est pas présente dans la liste, la fonction retourne `-1`.

Exemple : Si la liste chaînée est `(3,(6,(5,(6,(5,(2,None))))))` et la valeur `5`, alors l'indice retourné est `2`.

Fonctions utilitaires des algorithmes de tri

Exercice 10 (tri par insertion)

Dans le fichier **ex10__inserer__dans__liste__triee.py**, écrire la fonction récursive `inserer_dans_liste_triee` qui accepte en paramètres :

- Une liste chaînée `liste` triée par **ordre croissant**.
- Une `valeur`.

La fonction insère cette nouvelle valeur dans la liste chaînée, de telle façon que la liste reste triée par ordre croissant.

Exemple : Si la liste chaînée est `(3,(4,(6,None)))` et la valeur est `5`, la liste retournée est `(3,(4,(5,(6,None))))`.

Exercice 11 (tri par sélection)

Dans le fichier **ex11_extraire_minimum.py**, écrire la fonction récursive `extraire_minimum` qui accepte en paramètre une **liste chaînée** non vide, **extrait** le minimum de cette liste chaînée et retourne un tuple constitué du minimum trouvé ainsi que de la liste chaînée ainsi **modifiée**.

> **Attention ! : On conserve l'ordre des éléments dans la liste privée du minimum**.

Exemple : Si la liste chaînée `liste` est `(7,(1,(6,(2,None))))`, alors le minimum extrait est `1` et la liste modifiée est `(7,(6,(2,None)))`.

Exercice 12 (tri fusion)

Dans le fichier **ex12_partager_liste.py**, écrire la fonction récursive `partager_liste` qui accepte en paramètre une variable `liste` et retourne deux listes chaînées `liste1` et `liste2` composées chacune d'un élément sur deux de `liste`.

Indications :

- Le partage en deux se fait selon un principe semblable à celui de la fermeture éclair.
- Le premier élément de `liste1` est identique au premier élément de `liste`.

Autrement dit : `tete(liste1)==tete(liste)`.

Exemple :

Si `liste` est `(1,(6,(8,(3,None))))`, alors les deux listes `liste1` et `liste2` sont respectivement `(1,(8,None))` et `(6,(3,None))`.

Exercice 13 (tri fusion)

Dans le fichier **ex13_fusionner_listes_triees.py**, écrire la fonction récursive `fusionner_listes_triees` qui accepte en paramètre deux listes chaînées `liste1` et `liste2` triées en ordre croissant et retourne la liste chaînée `liste` triée

elle aussi en ordre croissant et résultat de la fusion des listes chaînées `liste1` et `liste2` .

Exemple :

Si les deux listes chaînées `liste1` et `liste2` sont respectivement `(1,(6,None))` et `(3,(8,None))`, alors la liste chaînée `liste` est `(1,(3,(6,(8,None))))`.

Exercice 14 (tri rapide)

Dans le fichier **ex14_partitionner_pivot.py**, écrire la fonction récursive `partitionner_pivot` qui accepte en paramètre une variable `liste` ainsi qu'une valeur de `pivot` et retourne deux listes chaînées `liste_inf` et `liste_sup` composées respectivement des éléments de `liste` qui sont inférieurs ou égaux à `pivot` et de ceux qui sont strictement supérieurs à `pivot`.

Exemple :

Si la liste chaînée `liste` est `(8,(2,(7,(9,None))))` et que le `pivot` est `7` , alors `liste_inf` et `liste_sup` sont respectivement `(2,(7,None))` et `(8,(9,None))`.

Exercice 15 (tri rapide)

Dans le fichier **ex15_concatener_listes.py**, écrire la fonction récursive `concatener_listes` qui accepte en paramètre deux listes chaînées `liste1` et `liste2` et retourne la liste chaînée `liste`, résultat de la concaténation des listes chaînées `liste1` et `liste2` .

Exemple :

Si les deux listes chaînées `liste1` et `liste2` sont respectivement `(8,(4,None))` et `(7,(1,None))`, alors la liste chaînée `liste` est `(8,(4,(7,(1,None))))`.