# TD Listes doublement chainées

**Exercice 1**

En considérant la définition d'une Liste doublement chainée suivante

```
type
   Liste=Structure
          tete:^Cellule
          queue:^Cellule
          taille :entier
    fin structure

  Cellule=structure
         elem :<type>
         succ : ^Cellule
         pred : ^Cellule
     Fin structure
```

**Nb :** la propriété taille donne le nombre des éléments contenus dans la liste

Donner l'implémentation de chacune des procédures et fonctions suivantes :

1. **Procédure insérerOptimisée(var L : Liste, x :type, position: entier)** Cette procédure permet d'insérer l'élément x à une position donnée en minimisant le nombre d'itérations permettant l'accès à la position d'insertion. C'est-à-dire si la position est plus proche de la tête que de la queue on commence le parcours à partie de la tête, sinon on commence le parcours à partir de la queue.
2. **Procédure SupprimerOptimisée(var L : Liste, position: entier)** Cette procédure permet supprimer l'élément se trouvant à une position donnée en minimisant le nombre d'itérations permettant l'accès à la position de suppression. C'est-à-dire si la position est plus proche de la tête que de la queue on commence le parcours à partir de la tête, sinon on commence le parcours à partir de la queue.

3. **Fonction ListeSupprimerRepetition(var L :Liste) :entier** Cette fonction permet de supprimer toutes les répétitions dans la liste et renvoyer le nombre de suppressions.
4. **Procédure Append(Var L1 :Liste, L2 :Liste)** cette procédure ajoute les éléments de la liste L2 à la fin de la liste L1 (la liste L1 se retrouve alors modifiée)

**Problème Liste Doublement chainées**

On veut gérer à travers une liste doublement chaînée un ensemble de livres. Pour chaque livre on garde le **titre** et la **quantité** qui indique le nombre de copies disponibles. Faite la déclaration de la structure ***livre***, **noeud*, liste*** nécessaires pour notre liste puis répondre à ces questions:

1. Écrire une procédure **ajouterLivre(var L:liste, titre:chaine, nbCopies:entier)** qui ajoute un livre dans la liste (en précisant le nombre de copies à ajouter). Si le livre existe déjà, il y'aura augmentation du nombre de copies. Si le livre n'existe pas et que sa quantité est inférieure ou égale à 1 l'ajout sera en tête de liste. Sinon, l'ajout sera en queue de liste.

1. Ecrire une fonction ***taille(L:liste):entier*** qui retourne la taille de la liste.
2. Écrire une procédure ***afficher(L:liste)*** qui affiche tous les Livres dans la liste.
3. Ecrire une procédure **décrémenterLivre *(L:liste,* titre:chaine*)*** qui décrémente le nombre de livres disponibles pour un titre donné. Le parcours doit commencer à partir de la queue.
4. Écrire une fonction **supprimerVide (var L:liste):boolean** qui supprime le premier livre à partir du début dont la quantité est égale à zéro tout en retournant la valeur vrai. Sinon, aucun livre ne sera supprimé et la fonction retourne faux.
5. Écrire une precédure ***EditerLivre(liste L, pos:entier)*** qui affiche le livre se trouvant à la position **pos** de la liste, en optimisant la recherche de ce livre.
6. Écrire une procédure ***supprimerToutVide(var Liste L)*** qui supprime tous les livres dans la liste dont la quantité est égale à zéro.