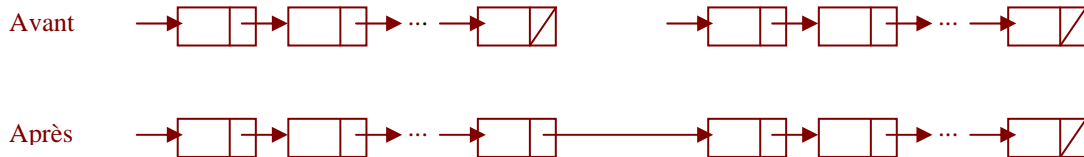


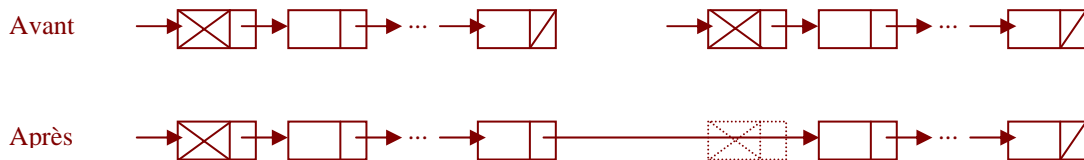
EXERCICE 2 : Concaténation de deux listes chaînées.

On demande d'écrire trois algorithmes de la **concaténation de deux listes chaînées** dans trois cas de représentation (liste "classique", liste avec élément fictif en tête, liste avec élément sentinelle en queue).

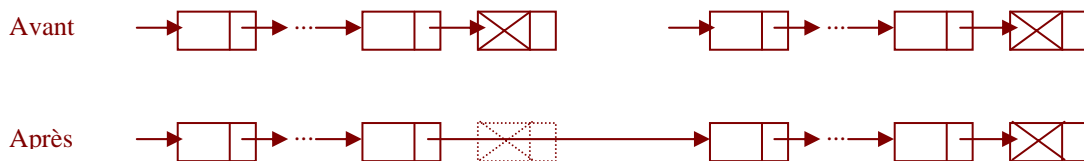
Question 2.1 : Les deux listes sont représentées de manière "classique"



Question 2.2 : Les deux listes sont représentées avec des éléments fictifs en tête.



Question 2.3 : Les deux listes sont représentées avec des éléments sentinelles en queue.



EXERCICE 3 : Suppression de toutes les occurrences du premier élément d'une séquence.

On donne la fonction suivante :

SansPremEl : une séquence d'éléments → une séquence d'éléments
 { SansPremEl(S) est la séquence S dont toutes les occurrences de son premier élément ont été supprimées }

SansPremEl ([]) = []
 SansPremEl (e ◦ S) = Sans (e, S) {suppression de toutes les occurrences de e dans S}

Sans : un élément, une séquence d'éléments → une séquence d'éléments
 {Sans (x, S) est la séquence S dont toutes les occurrences de x ont été supprimées}

Sans (x, []) = []
 Sans (x, S . e) = soit S' = Sans (x, S)
 dans selon x, e
 e = x : S'
 e ≠ x : S' . e

Réaliser une action **Supprimer_PremEI** spécifiée ci-dessous :

Supprimer_PremEI : **une action (la donnée-résultat Tête : une AdDoublet)**
{ état initial : Tête désigne une séquence d'éléments *S* représentée par chaînage ;
état final : Tête désigne la séquence d'éléments *SansPremEI(S)* représentée par chaînage }

Doublet : **le type** <info : un élément ; suc : une AdDoublet>
AdDoublet : **le type pointeur** de Doublet

Remarque : l'action **Supprimer_PremEI** doit libérer la mémoire inutilisée.

EXERCICE 4 : *Séquence des sommes des sous-séquences d'une séquence d'entiers donnée.*

Etant donnée une séquence d'entiers *S*, on veut construire la séquence **SommesSousSeq(S)** qui est la séquence des sommes des sous-séquences croissantes de *S* de longueurs maximales.

Exemple : **SommesSousSeq**([2, 4, 5, 1, 7, 4, 2, 9, 3, 5, 7, 12]) = [11, 8, 4, 11, 27]

Réaliser une action **Lister_Sommes** spécifiée ci-dessous :

Lister_Sommes : **une action (la donnée-résultat Tête : une AdDoublet)**
{ état initial : Tête désigne une séquence d'entiers *S* représentée par chaînage ;
état final : Tête désigne la séquence d'entiers *SommesSousSeq(S)* représentée par chaînage }

Doublet : **le type** <info : un entier ; suc : une AdDoublet>
AdDoublet : **le type pointeur** de Doublet

Remarque : l'action **Lister_Sommes** doit libérer la mémoire inutilisée.