Semaines 10 et 11
Listes - Manipulations diverses

Finir feuille précédente si besoin.

Exercice 31 : Recherche et suppression du minimum (unique) d'une liste.

```
Action SupprimerMin(ES L : liste d'entiers)
var Adr : TAdresse
    Trouvé : booléen
début
rechercheMin(L, Trouvé, Adr)   // modif fonction recherche, c'est tout...
Si Trouvé    // indique liste non vide
Alors Si Adr = NULL
      Alors SupprimerEnTête(L)
      Sinon SupprimerAprès(L, Adr)
fin

Action rechercheMin( E L : TListe ;
                     S Trouvé : booléen, PrecMin : TAdresse )
var Adr, Prec : TAdresse
    Min, X : entier

début
Trouvé <- Faux
Adr <- AdressePremier(L)
Si Adr <> NULL  // si liste non vide
Alors début
      Trouvé <- Vrai
      Min <- ValeurElément(L, Adr)
      PrecMin <- NULL
      Prec <- Adr
      Adr <- AdresseSuivant(L, Adr)
      Tant Que Adr <> NULL
      Faire début
            X <- ValeurElément(L, Adr)
            Si X < Min
            Alors début
                  Min <- X
                  PrecMin <- Prec
                  fin
            Prec <- Adr
            Adr <- AdresseSuivant(L, Adr)
            fin
```

```
      fin
fin
```

Exercice 32 : Ecrire une action qui supprime d'une liste de `TEtudiant` tous ceux dont la note est inférieure à 10.

```
 Action SupprInf10( ES L : liste de TEtudiant)
var AdrPréc, Adr : TAdresse ;
      Etd : TEtudiant ;

début
AdrPréc <- NULL ;
Adr <- AdressePremier(L);
Tant Que Adr <> NULL
Faire début
      Etd <- ValeurElément (L, Adr);
      Si Etd.Moy < 10
      Alors début
            Adr <- AdresseSuivant(L, Adr);
            Si AdrPréc = NULL
            Alors SupprimerEnTête(L)
            Sinon SupprimerAprès(L, AdrPréc)
            fin
      Sinon début
            AdrPréc <- Adr ;
            Adr <- AdresseSuivant(L, Adr)
            fin
      fin
fin
```

Pour éviter d'itérer le test fastidieux (suppression en tête ou après telle adresse) on peut commencer le traitement de la liste à partir du deuxième élément (s'il existe). A la fin, on traite le premier élément.

```
Action SupprInf10Best (ES L : liste de TEtudiant) ;

var Adr, AdrPréc : TAdresse
      E : TEtudiant

début
AdrPréc <- AdressePremier(L)
Si AdrPréc <> NULL
Alors début
      Adr <- AdresseSuivant(L, AdrPréc)
      Tant Que Adr <> NULL
      Faire  début
```

```
            E <- ValeurElement(L, Adr)
            Si E.Moy < 10
            Alors début
                  Adr <- AdresseSuivant(L, Adr)
                  SupprimerAprès(L, AdrPréc)
                  fin
            Sinon début
                  AdrPréc <- Adr ;
                  Adr <- AdresseSuivant(L, Adr)
                  fin
            fin // TQ
      E <- ValeurElement(L, AdressePremier(L))
      Si E.Moy < 10
      Alors SupprimerEnTête (L)
      fin // SI
fin
```

Remarque : une autre solution consiste à faire 2 boucles successives :

- supprimer en tête tant qu'il le faut,

- puis supprimer après.

Au moins donner l'idée si on n'écrit pas l'algorithme.

Exercice 33 : Fusion de deux listes triées de `TEtudiant` (avec élément fictif).

```
Action FusionListes( E L1, L2 : listes de TEtudiant ; S L3 : liste de TEtudiant)
var Adr1, Adr2, Adr3 : TAdresse
    Fictif, E1, E2 : TEtudiant

début
CréerListe (L3)
InsérerEnTête (L3, Fictif) // Elément fictif, de valeur indéterminée,
                           // et qui sera supprimé à la fin ;
                           // permet d'éviter de nombreux tests :
                           // L3 n'est pas vide
Adr1 <- AdressePremier (L1)
Adr2 <- AdressePremier (L2)
Adr3 <- AdressePremier (L3)
Tant Que Adr1 <> NULL et Adr2 <> NULL
Faire début
      E1 <- ValeurElément (L1, Adr1)
      E2 <- ValeurElément (L2, Adr2)
      Si E1.Nom < E2.Nom
      Alors début
            InsérerAprès (L3, Adr3, E1)
            Adr1 <- AdresseSuivant (L1, Adr1)
      Sinon début
            InsérerAprès (L3, Adr3, E2)
```

```
            Adr2 <- AdresseSuivant (L2, Adr2)
      Adr3 <- AdresseSuivant (L3, Adr3) // on a inséré un élément E1 ou E2
      fin
// On est arrivé au bout d'au moins une des deux listes L1 ou L2

Tant Que Adr1 <> NULL
Faire début
    E1 <- ValeurElément (L1, Adr1)
    InsérerAprès (L3, Adr3, E1)
    Adr1 <- AdresseSuivant (L1, Adr1)
    Adr3 <- AdresseSuivant (L3, Adr3)
    fin

Tant Que Adr2 <> NULL
Faire début
    E2 <- ValeurElément (L2, Adr2)
    InsérerAprès (L3, Adr3, E2)
    Adr2 <- AdresseSuivant (L2, Adr2)
    Adr3 <- AdresseSuivant (L3, Adr3)
    fin

SupprimerEnTête (L3) // suppression de l'élément fictif
FIN
```

Exercice 34 : Tri à bulle d'une liste de de `TEtudiant`

```
Action TriBulle( ES L : listes de TEtudiant)
var AdrPrecPrec, AdrPrec, Adr : TAdresse
    Fictif, E1, E2 : TEtudiant
    Triée : booléen

début
    Adr <- AdressePremier (L)
    Si Adr <> NULL et AdresseSuivant(L,Adr) <> NULL // L contient au
                                                    // moins deux éléments
    Alors début
        InsérerEnTête (L, Fictif) // Elément fictif, de valeur indéterminée,
                                  // et qui sera supprimé à la fin ;
                                  // permet d'éviter de nombreux tests :

        Triée <- faux
        Tant que non Triée
        faire début
            Triée <- vrai
            AdrPrecPrec <- AdressePremier(L)
            AdrPrec <- AdresseSuivant(L,AdrPrecPrec)
            Adr <- AdresseSuivant(L,AdrPrec)
```

```
            Tant que Adr <> NULL
            Faire début
                E1 <- valeurElément(L,AdrPrec)
                E2 <- valeurElément(L,Adr)
                Si E1 > E2
                Alors début
                    InsererAprès(L,Adr,E1)
                    SupprimerAprès(L,AdrPrecPrec)
                    Triée <- faux
                fin
                AdrPrecPrec <- AdresseSuivant(L,AdrPrecPrec)
                AdrPrec <- AdresseSuivant(L,AdrPrecPrec)
                Adr <- AdresseSuivant(L,AdrPrec)
            fin
        fin
        SupprimerEnTete(L)
    fin
FIN
```

S'il reste du temps ...

Lors du TP, les étudiants auront à implémenter l'algorithme de tri fusion d'une liste. Pour leur faire gagner du temps lors du TP, leur expliquer le principe sur un exemple : recherche du milieu de la liste et tri des deux sous listes de façon récusive.

Voici l'algo qui leur est donné sur le feuille de TP :

```
tri(ES l : Liste, E  a, b : TAdresse)
var m : TAdresse
début
  Si (l, a, b) possède au moins deux éléments
  Alors début
        m <-  milieu(l, a, b)
        tri(l,a,adresseSuivant(l,m))
        m <-  milieu(l, a, b) // le tri de la première moitié
                              // a pu changer l'adresse de l'objet médian
        tri(l, m,b)
        fusion(l,m,a,b)
        fin
fin
```