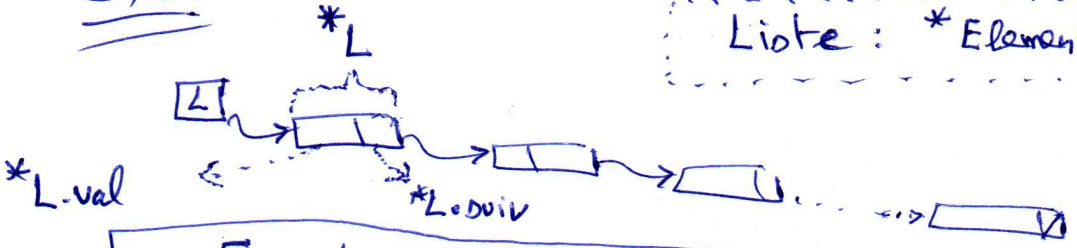
Correction – série 02

```
type Element = structure
     val : entier
     suiv : *Element

Liste : *Element
```

EXO1



1)

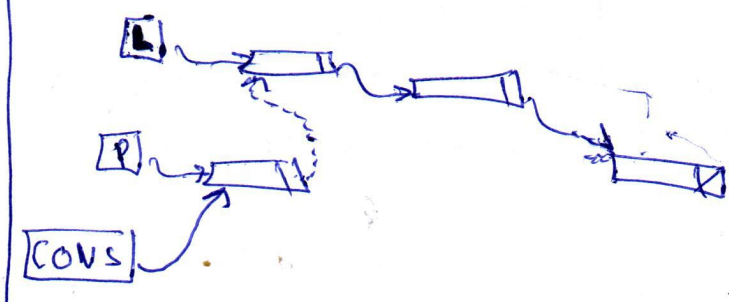
```
Fonction CAR (L : Liste) : entier
début
    si (L ≠ Nil) alors
        CAR ← *L.val        (ou bien CAR ← valeur(L))
    finsi
fin
```

2)

```
Fonction CDR (L : Liste) : Liste
début
    si (L ≠ Nil) alors
        CDR ← *L.suiv       (ou bien : CDR ← suivant(L))
    sinon
        CDR ← Nil
    fsi
fin
```

3)

```
Fonction CONS (x : entier, L : Liste) : Liste
 var  P : Liste
début
    P ← allouer( )
    *P.val ← x
    *P.suiv ← L
    CONS ← P
fin
```



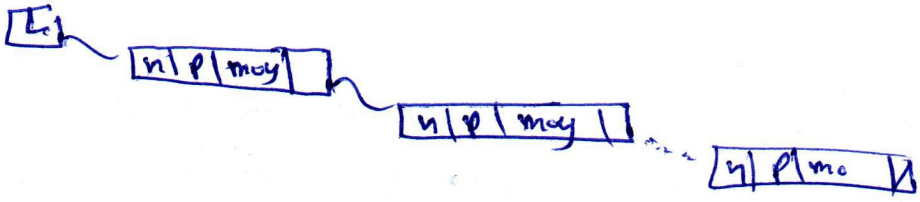
4/1

```
Fonction Triée (L : Liste) : Booleen
début
    si (L = Nil) alors                          : puisque la liste vide est toujours triée.
        Triée ← vrai
    sinon
        si (*L.suiv = Nil) alors                : puisque la liste d'un seul élément est toujours triée
            Triée ← vrai
        sinon
            si (*L.val ⩽ *(*L.suiv).val) alors
                Triée ← Triée (*L.suiv)
            sinon
                Triée ← faux
            fsi
        fsi
    fsi
fin
```

5/1

```
Fonction Fusion (L1 : Liste, L2 : Liste) : Liste
début
    si (L1 = Nil) alors
        Fusion ← L2
    sinon
        si (L2 = Nil) alors
            Fusion ← L1
        sinon
            si CAR(L1) < CAR(L2) alors
                Fusion ← CONS(CAR(L1), Fusion(CDR(L1), L2))
            sinon
                Fusion ← CONS(CAR(L2), Fusion(L1, CDR(L2)))
            fsi
        fsi
    fsi
fin
```

Exo1



1/

```
Type Etudiant = structure
        nom : char
        prenom : char
        moy : réel
        suiv : * Etudiant

     Liste : * Etudiant
```

2/

```
Fonction Afficher_liste (L : Liste)
début
    si (L ≠ Nil) alors
        écrire (*L.nom)
        écrire (*L.prenom)
        écrire (*L.moy)
        Afficher_liste (*L.suiv)
    fsi
fin
```

3/

```
procédure insérer_tete ( L : Liste)
var : nouv : Liste
début
    nouv = allouer()
    écrire ("saisir un nom")
      lire (*nouv.nom)
    écrire ("saisir un prénom")
      lire (*nouv.prenom)
    écrire ("saisir une moyenne")
      lire (*nouv.moy)
    *nouv.suiv ← L
     L ← nouv
fin
```

4/

```
Fonction Moy_Max (L : Liste) : réel
Var : max : réel
      P : Liste
Début
    Si (L ≠ Nil) alors
        max ← L*.moy
        P ← L*.suiv
        Tantque ( P ≠ Nil ) faire
            Si (max < *P.moy) alors
                max ← *P.moy
            fsi
            P ← *P.suiv
        fintq
        Return max
    finsi
fin
```

Exo4

1/

```
type Maillon = structure
        val : entier
        suiv : *Maillon
```

2/

```
Fonction Cardinal ( L : Liste ) : entier
Début
    Si (L = Nil) alors
        cardinal ← 0
    Sinon
        cardinal ← 1 + cardinal (L*.suiv)
    fsi
fin
```

3/ **Remarque** la liste L est passée par variable qui est initialisée à Nil

la variable L est une variable globale déclarée dans le programme principale

```
procédure interaction (L1 : Liste, L2 : Liste, L : Liste)
  var: nouv : Liste
début
  si (L1 ≠ Nil) et (L2 ≠ Nil) alors
    si val(L1) = val(L2) alors
      nouv ← allouer()
      *nouv.val ← val(L1)
      *nouv.suiv ← Nil
      si (L = Nil) alors
        L ← nouv
        P ← L
      sinon
        *P.suiv ← nouv
        P ← *P.suiv
      fsi
      interaction (*L1.suiv, *L2.suiv, L)
    sinon
      si (val(L1) < val(L2)) alors
        interaction (*L1.suiv, L2, L)
      sinon
        interaction (L1, *L2.suiv, L)
      fsi
    fsi
  fsi
fin
```

insertion à la fin de la liste L

ou bien

cette solution itérative, où P est une variable locale

```
Procédure intersection (L1 : Liste, L2 : Liste, L : Liste)
var : nouv : Liste
      P : Liste
début
   [illegible]
   tantque (L1 ≠ Nil) et (L2 ≠ Nil) faire
      si val(L1) = val(L2) alors
         nouv ← allouer()
         *nouv.val ← val(L1)
         *nouv.suiv ← Nil
         si (L = Nil) alors
            L ← nouv
            P ← L
         sinon
            *P.suiv ← nouv
            P ← *P.suiv
         fsi
         L1 ← *L1.suiv        L2 ← *L2.suiv
      sinon
         si (val(L1) < val(L2)) alors
            L1 ← *L1.suiv
         sinon
            L2 ← *L2.suiv
         fsi
      fsi
   ftq
fin
```

4)

Fonction Inclu(L_1 : Liste, L_2 : Liste) : Booleen